МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет нафти і газу

Кафедра інформаційно-телекомунікаційних технологій та систем

&

С. В. Зікратий

СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

КОНСПЕКТ ЛЕКЦІЙ

Івано-Франківськ
2018

УДК 004.89 (076.5)
З-60

Рецензент:
Николайчук М. Я. кандидат технічних наук, доцент кафедри інформаційно-телекомунікаційних технологій та систем Івано-Франківського національного технічного університету нафти і газу

Рекомендовано методичною радою університету
(протокол № 5 від 29 березня 2018 р.)

Зікратий С. В.

З-60 Системи штучного інтелекту: конспект лекцій – Івано-Франківськ: ІФНТУНГ, 2018. – 152 с.

МВ 02070855-11291-2018

Конспект лекцій містить основні теоретичні відомості з дисципліни “Системи штучного інтелекту”. Розроблений відповідно до робочої програми даної навчальної дисципліни. Може бути використаний студентами денної та заочної форм навчання.

Призначений для підготовки фахівців освітнього рівня “бакалавр” за спеціальністю 151 – Автоматизація та комп’ютерно-інтегровані технології.

МВ 02070855-11291-2018
УДК 004.89 (076.5)

© Зікратий С. В.
© ІФНТУНГ, 2018

ЗМІСТ

1 Вступ до дисципліни 7
1.1 Інтуїтивне розуміння поняття "інтелект" 7
1.2. Приклади інтелектуальних задач 8
1.3. Аналіз основних визначень поняття "інтелект" 10
1.4. Тест Тьюринга і фатичний діалог 11
2 Інтелект як високоорганізована кібернетична система 13
2.1 Алгоритмічний і декларативний підходи до керування 13
2.2. Поповнення первинних інструкцій 13
2.3. Формалізація понять алгоритмічносгі та декларативності 14
2.4. Квазіалгоритми та джерела квазіалгоритмічіності 15
2.5 Інтелектуальні системи із загальнокібернетичиих позицій 16
2.6. Типова схема функціонування інтелектуальної системи 18
2.7. Класифікація основних напрямів досліджень 19
2.8. Соціальні наслідки інтелектуалізації комп'ютерних технологій 20
3 Представлення знань в інтелектуальних системах 22
3.1 Знання і деякі підходи до їх подання 22
3.2. Вербально-дедуктивне визначення знань 23
3.3 Дані і знання. Основні визначення 24
3.4 Особливості знань 25
3.5 Неоднорідність знань. Області і рівні знань 27
3.6. База знань, як об'єднання простіших одиниць 27
3.7 Бінарні предикати і тріада “об'єкт–атрибут–значення” 28
3.8 Проблема неточних і неповних знань 29
3.9 Класифікація рівнів розуміння 29
4 Логічні моделі та метод резолюцій 34
4.1 Логічні побудови та логічні моделі 34
4.2 Основні положення числення предикатів 35
4.3 Приведення формул до нормальних форм 38
4.4 Метод резолюцій 40
5 Мережеве представлення знань. Семантичні мережі 46
5.1 Концептуальні графи 46
5.2 Семантичні мережі 48
5.2.1 Представлення контексту 48
5.2.2 Представлення «сукупність -посилання» 49
5.2.3 Канонічні графи 50
5.2.4 Правила побудови 51

5.2.5 Успадковані властивості 51
5.2.6 Решітки типів; ієрархії типів 52
5.2.7 Решітки множин і решітки типів 52
5.2.8 Схеми і схематичні кластери 54
5.3 Міркувань, що використовують семантичні мережі 55
6 Об'єктне представлення знань. Фреймові моделі 57
6.1 Зчеплення 57
6.2 Фрейми і слоти 58
6.2.1 Явні фрейми 58
6.2.2 Функціональні фрейми 58
6.2.3 ∀–квантифікація 59
6.3 Міркування, що використовують об'єктне уявлення 59
6.3.1 Паропоєднання 59
6.3.2 Функціональні атрибути 60
6.4 Автоматичні міркування, що використовують фрейми 61
6.4.1 Ієрархічні міркування, що використовують фрейми 62
6.4.2 Міркування з винятками 63
7 ПРОДУКЦІЙНІ МОДЕЛІ ПРЕДСТАВЛЕННЯ ЗНАНЬ 65
7.1 Класифікація ядер продукції 67
7.2 Пряме та зворотне виведення 68
7.3 Типові дисципліни виконання продукцій 69
7.4 Основні стратегії вирішення конфліктів у продукційних системах 70
8 ФОРМАЛІЗАЦІЯ НЕДОСТОВІРНИХ І НЕЧІТКИХ ЗНАНЬ 73
8.1 Модальна логіка предикатів 73
8.1.1 Модальні оператори 74
8.1.2 Приклади модальних операторів 74
8.1.3 Синтаксис модальної логіки предикатів 74
8.2 Тризначна семантика для модальної логіки предикатів 75
8.3 Семантика можливих світів 76
8.4 Ламбда-числення 78
9 ЛОГІЧНЕ ВИВЕДЕННЯ ЗА НЕДОСТОВІРНИХ ЗНАНЬ 80
9.1 Поняття про неточне логічне виведення 80
9.2 Деякі визначення з теорії ймовірностей 80
9.3 "Об'єктивна" та "суб'єктивна" невизначеність 82
9.4 Загальні принципи неточного виведення 84
9.4.1Точкові та інтервальні міри неточності 85
9.4.2 Проблема комбінування свідоцтв 86
9.4.3 Приклади застосування мір достовірності 86

9.5 Формалізації мір ризику за неточного логічного виведення 87
9.5.1 Деякі проблеми виведення 88
9.5.2 Схема EMYCIN 89
10 Нечітка логіка 91
10.1 Математичний апарат 91
10.1.1 Операції над нечіткими множинами 92
10.1.2 Наочне представлення операцій над нечіткими множинами 92
10.1.3 Властивості операцій ∩ і ∪. 93
10.1.4 Алгебраїчні операції над нечіткими множинами 94
10.2 Нечітка та лінгвістичні змінні 96
10.3 Нечіткий логічний вивід 99
11 ОСНОВНІ ПІДХОДИ ДО ПЛАНУВАННЯ ЦІЛЕСПРЯМОВАНИХ ДІЙ 103
11.1 Планування цілеспрямованих дій і прийняття рішень 103
11.2 Повний перебір 104
11.3 Методи "сліпого" пошуку 105
11.3.1 Випадковий пошук 105
11.3.2. Пошук "в глибину і ширину" 105
11.3.3 Алгоритм рівних цін 106
11.4 Евристичний пошук 107
11.4.1 Алгоритм "підйому на гору" 107
11.4.2 Глобальний облік відповідності мети 108
11.4.3 А-алгоритм 109
11.5 Простір задач і простір станів 112
11.6 Пошук з розповсюдженням обмежень 114
12 Вступ в експертні системи 116
12.1 Технології розробки експертних систем. 121
12.1.1 Особливості розробки ЕС. 121
12.1.2 Етапи розробки ЕС 123
12.2 Класифікаційні ознаки ЕС 127
12.2.1 Характеристики наочної області. 129
12.2.2 Характеристики типів вирішуваних в проблемній області завдань. 129
12.2.3 Характеристика інструментальних засобів розробки ІІС. 131
13 Склад і організація даних і знань в ЕС 136
13.1 Відмінності знань від даних. 136
13.2 Організація даних в робочій пам'яті ЕС. 138
13.3 Організація знань в БЗ 138
13.4 Моделі представлення знань в ЕС 140
14 Методи придбання знань. 142

14.1 Основні аспекти процесу витягання знань 142
14.2 Методи витягання знань 145
ПЕРЕЛІК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ 152

1 Вступ до дисципліни

1.1 Інтуїтивне розуміння поняття "інтелект"

З давніх-давен людині були необхідні помічники для полегшення виконання тих чи інших операцій. Тож людська думка винаходила різноманітні механізми, машини та інші витвори людини. Поява електронно-обчислювальних машин дала змогу автоматизувати трудомісткі розрахункові операції. Згодом стало зрозуміло, що ці машини можна використовувати не тільки для обчислень, а й для керування різними пристроями, складними автоматизованими виробництвами тощо. Поширення набули *роботи* — програмно керовані пристрої, здатні безпосередньо взаємодіяти з фізичним світом та виконувати в ньому певні задані дії. Такі роботи широко застосовуються на багатьох виробництвах. Вони можуть працювати і в умовах, яких людина не в змозі витримати, – на дні океану, всередині ядерного реактора та ін. Але такі помічники вимагають автоматизованого керування, тобто їм постійно потрібно "підказувати", що і як вони мають робити у визначеній ситуації.

Природна мова на сучасному етапі малопридатна для цього через свою складність і неоднозначність. Один з шляхів вирішення згаданої проблеми – це формулювання інструкцій мовою, зрозумілою виконавцеві, навіть якщо він – робот, тобто написання програм. Програмування полягає у перекладі інструкцій, написаних мовою, близькою до природної, мовою, яку здатна сприйняти обчислювальна система. Відомі складності сучасного програмування, пов'язані з необхідністю надмірної алгоритмізації, тобто детального ретельного розписування своєрідних програмних кроків, – інструкцій з урахуванням усіх можливих ситуацій. З цієї ситуації існує єдиний вихід – підвищення рівня "розумності", інтелектуальності сучасних комп'ютерів і роботів. Тож постає питання: що розуміється під такими поняттями, як "інтелектуалізація", *"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 існуватиме ситуацій, в яких штучні інтелектуальні системи зможуть замінити людей, тим інтелектуальнішими вважатимуться ці системи.

Навряд чи є сенс протиставляти штучний інтелект і інтелект людини. Слід спробувати визначити поняття "інтелект" взагалі, незалежно від його походження.

Людина вважається інтелектуальною від природи — її інтелект вироблявся протягом мільйонів років еволюції. Тож людина вміє вирішувати багато інтелектуальних задач. Кожна людина вважає, що розуміє значення слова "інтелект", але якщо попросити її навести тлумачення цього слова, в переважній більшості чіткої відповіді не дістанемо. І дійсно, дати визначення поняття "інтелект", яке б задовольняло всіх, очевидно, неможливо (далі будуть

проаналізовані деякі такі спроби). Але відсутність чіткого трактування не заважає оцінювати інтелектуальність на інтуїтивному рівні.

Можна навести як мінімум два методи такої оцінки: ***метод експертних оцінок*** **і** ***метод тестування***.

У разі застосування методу експертних оцінок рішення про ступінь інтелектуальності приймає досить велика група експертів (незалежно або у взаємодії, способів організації якої відомо досить багато).

У разі використання методу тестування пропонується розв'язати ті чи інші тестові завдання. Існує значна кількість інтелектуальних тестів, апробованих практикою, для оцінки рівня розумових здібностей людини, що знайшли застосування в психології та психіатрії.

Потрібно наголосити, що поняття "штучний інтелект" не можна зводити лише до створення пристроїв, які повністю або частково імітують діяльність людини. Не менш важливим є інше завдання: **виявити механізми, які лежать в основі діяльності людини, щоб застосувати їх при вирішенні конкретних науково-технічних завдань.** І це лише одна з можливих проблем.

1.2. Приклади інтелектуальних задач

Розглянемо і проаналізуємо в загальних рисах деякі проблеми, які доводиться постійно вирішувати людському розуму: ***розпізнавання образів, мислення*** **та** ***обчислювальні задачі.***

На інтуїтивному рівні можна сформулювати кілька типових завдань розпізнавання образів (або просто розпізнавання):

• завдання ***ідентифікації*** полягає в тому, що об'єкт, який спостерігається людиною, потрібно вирізнити з-поміж інших (наприклад, побачивши іншу людину, впізнати у ній свою дружину);

• ***проблема розпізнавання в класичній постановці:*** визначити належність об'єкта, що спостерігається, до одного із заздалегідь відомих класів об'єктів (наприклад, відрізнити легковий автомобіль від вантажного).

Людина класифікує просто. Наприклад, чоловік, повернувшись додому з роботи, відразу ж впізнає свою дружину, але більшість людей не зможе пояснити, як він це робить, у повному обсязі. Як правило, раціонального пояснення немає. Теорія розпізнавання, яка інтенсивно розвивається, необхідна для того, щоб навчити штучні інтелектуальні системи розв'язувати завдання розпізнавання на основі досвіду розпізнавання людиною. Зокрема, сформульовано такий ключовий принцип [264]: ***будь-який об'єкт у природі – унікальний; унікальні об'єкти – типізовані.*** Відповідно до цього принципу розпізнавання здійснюється на основі аналізу певних характерних **ознак.** Вважається, що в природі не існує двох об'єктів, для яких збігаються **абсолютно всі** ознаки, і це теоретично дозволяє здійснювати ідентифікацію. Якщо ж для якихось об'єктів **деякі** ознаки збігаються, ці об'єкти теоретично можна об'єднувати в групи або класи саме за цими ознаками.

Проблема полягає у тому, що різноманітних ознак існує незліченна

кількість. Незважаючи на легкість, з якою людина проводить розпізнавання, вона дуже рідко в змозі виокремити суттєві для цього ознаки. До того ж об'єкти, як правило, змінюються з часом. Тому далі ми спробуємо показати, що розпізнавання об'єктів і ситуацій має виняткове значення для орієнтації людини в навколишньому світі і прийняття нею вірних рішень. Розпізнавання, як правило, здійснюється людиною на інтуїтивному, підсвідомому рівні – адже людина вчилася цьому мільйони років.

Інша інтелектуальна задача – моделювання ***мислення,*** зокрема формулювання наслідків з фактів, які безпосередньо спостерігаються або вважаються відомими. Можна виокремити два типи процесів мислення:

• підсвідоме ***інтуїтивне*** мислення, механізми якого на сучасному етапі вивчені недостатньо і яке дуже важко формалізувати та автоматизувати;

• ***дедуктивні*** логічні побудови за формалізованими законами логіки.

Дедукцією *називається перехід від загального до часткового, виведення часткових наслідків із загальних правил.*

Тут пересічна людина рідко в змозі пояснити, за якими алгоритмами вона здійснює логічне виведення. Але методики та алгоритми, за якими **можна** автоматизувати виведення наслідків з фактів або логічну перевірку тих чи інших фактів, відомі. Насамперед це формальна логіка Арістотеля, сформована на основі конструкцій, які дістали назву ***силогізмів.*** Вони практично неподільно панували в логіці аж до початку XIX століття, тобто до появи булевої алгебри. Немає ніякої необхідності наводити якісь формальні визначення силогізмів. Наведемо лише один класичний приклад.

Перше твердження: *Усі люди смертні.*

Друге твердження: *Сократ – людина.*

Висновок: *Сократ смертний.*

Якщо перше і друге твердження у силогізмі істинні та задовольняють певним загальним формальним вимогам, тоді й висновок буде істинним, незалежно від змісту тверджень, що входять до силогізму. При порушенні цих формальних вимог легко припуститися логічних помилок, подібних до нижченаведених:

Усі 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.

Арістотелем було запропоновано кілька формальних конструкцій силогізмів, які він вважав достатньо універсальними. Лише у XIX столітті почала розвиватися сучасна математична логіка, яка розглядає силогізми Арістотеля як один з часткових випадків. Основою більшості сьогоднішніх систем, призначених для автоматизації логічних побудов, є ***метод резолюцій*** Дж. Робінсона (буде описаний далі). Але практична реалізація логічних побудов зіткнулася з серйозними проблемами. Головна з них – це феномен, який Р. Беллман назвав ***прокляттям розмірності.*** На перший погляд, описати знання про зовнішній світ можна було б, наприклад, таким чином: *"Об'єкт А має риси*

X, Y, Z. Об'єкт В відрізняється від А тим, що має рису Н, і т. д. ". Але зовнішній світ є винятково складним переплетінням різноманітних об'єктів і зв'язків між ними. Для того, щоб тільки ввести всю цю інформацію до пам'яті інтелектуального пристрою, може знадобитися не одна тисяча років. Ще триваліший період потрібен, щоб врахувати всі необхідні факти при логічному виведенні. Реальні програми, що здійснюють логічне виведення (вони часто називаються ***експертними системами),*** мають досить обмежене застосування, лімітований набір фактів і правил з певної, більш-менш окресленої предметної галузі, і можуть використовуватися лише у цій галузі.

Щодо людини, то якість її логічного мислення також часто буває далекою від бездоганної. Люди часто роблять логічні помилки, а інколи взагалі керуються принципами, невірними з точки зору нормальної логіки. 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 в основі поведінки людини лежали спроби проводити дедуктивні побудови від логічного початку до логічного кінця, людина була б практично нездатною до будь-якої діяльності – фізичної або розумової. Це вимагало б значного часу для аналізу.

Спільною рисою згаданих вище проблем є їх **погана формалізованість,** відсутність або незастосовність чітких алгоритмів розв'язку. Вирішення подібних задач і є основним предметом в теорії штучного інтелекту.

До зовсім іншого класу належать задачі, пов'язані з обчисленнями. 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. Тож були запропоновані ефективні принципи комп'ютерних обчислень, що радикально відрізняються від тих, які застосовуються людиною. Ці добре формалізовані алгоритмічні методики забезпечили такий рівень вирішення обчислювальних задач, який виявився абсолютно недосяжним для людського інтелекту. Водночас цей високий рівень алгоритмізації значною мірою зумовив слабкість традиційних комп'ютерних систем при розв'язанні тих інтелектуальних задач, з якими людина справляється непогано.

З появою таких обчислювальних потужностей мрійники 60—70-х років XX століття поставили завдання моделювати в повному обсязі роботу людського мозку. Теоретично цю проблему з певними обмеженнями можна було б вирішити. Але складність потрібних обчислень виявилася такою, що змусила більшість дослідників відійти від поставленого завдання і перейти до простіших задач: моделювання принципів роботи людського мозку при розв'язанні конкретно визначених типів задач.

1.3. Аналіз основних визначень поняття "інтелект"

Було зроблено чимало спроб дати формальне визначення поняття "інтелект", зокрема "штучний інтелект". Мабуть, відомішою з них є визначення предмета теорії штучного інтелекту, яке було введено в обіг видатним

дослідником у галузі штучного інтелекту Марвіном Мінським. яке відображає таку основну думку: *"Штучний інтелект є дисципліна, що вивчає можливість створення програм для вирішення задач, які при розв'язанні їх людиною потребують певних інтелектуальних зусиль "*. Але і це визначення має вади. Головна з них полягає у недосконалій формалізації поняття "певні інтелектуальні зусилля". Таких зусиль вимагає, наприклад, виконання простих арифметичних обчислень, але чи можна вважати інтелектуальною програму, яка здатна до здійснення тільки таких операцій? Відтак у ряді книг та енциклопедій до наведеного визначення додається поправка: *"Сюди не входять задачі, для яких відома процедура їх розв'язання"*. Важко вважати таке формулювання задовільним. Розвиваючи цю думку далі, можна було б продовжити: отже, якщо я не знаю, як виконувати якусь 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"*

Існують і конструктивніші визначення інтелекту. Наприклад, *"Інтелект є здатність правильно реагувати на нову ситуацію"*. Уявіть собі, наприклад, звичайний калькулятор. Цілком імовірно, що на ньому ніколи не обчислювали суму двох нулів. Тоді завдання "обчислити 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?

1.4. Тест Тьюринга і фатичний діалог

Відомий англійський учений Алан Тьюринг сформулював тезу, що визначає момент, з якого машину можна вважати інтелектуальною. Нехай експерт за допомогою телефону або подібного віддаленого пристрою спілкується з кимось (або чимось), що може бути як людиною, так і машиною. Експерт дає певні тести-завдання. За результатами відповідей він повинен визначити, з ким має справу – з людиною чи ЕОМ. Якщо він приймає комп'ютер за людину, комп'ютер може вважатися інтелектуальним. Така перевірка дістала назву ***тесту Тьюринга.*** Багато спеціалістів вважали, що подібного тесту цілком достатньо 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 наприкінці 60-х років американський кібернетик Дж. Вейценбаум створив дві програми – ЕЛІЗА і ДОКТОР – для використання в психіатрії. Виявилось, що вони можуть невірно зорієнтувати експерта в умовах тесту Тьюринга.

Тому при побудові цих програм в основу була покладена ідея ***фатичного діалогу.***

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

Так, на висловлення 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 жодного відношення ні до розуміння, ні до інтелекту, але багато людей, які проводили з цими програмами тест Тьюринга, вирішили, що мають справу з людиною.

Наприклад, діалог з цією програмою може бути таким:

Лікар (програма): *Я Вас слухаю.*

Пацієнт: *Мене бентежить погода.*

Лікар (програма): *Як давно Вас бентежить погода?*

Пацієнт: *Відтоді, як померла моя мати.*

Лікар (програма): *Розкажіть про Вашу сім 'ю.*

Пацієнт: *Усі вони захоплюються комп 'ютерами.*

Лікар (програма): *Машини Вас лякають.*

Пацієнт: *Ні.*

Лікар (програма): *Будь ласка, не будьте так небагатослівні...*

І так далі до нескінченності.

В основі методу, необхідного для реалізації такого діалогу, може лежати **механічне порівняння** речень, які вводяться людиною, зі зразками (шаблонами) речень, що зберігаються програмою. Якщо речення збіглося повною мірою з одним із зразків, видається одна з відповідей, заздалегідь заготовлених для цього зразка.

2 Інтелект як високоорганізована кібернетична система

2.1 Алгоритмічний і декларативний підходи до керування

Традиційно розрізняють два підходи до керування складними системами та до програмування роботів і комп'ютерів, які могли б розв'язувати ті чи інші задачі. У разі розв'язку задачі на сучасних комп'ютерах в основному реалізовано 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 послідовних інструкцій. Якщо таку послідовність вдається написати, комп'ютер зможе вирішувати поставлену задачу, якою б складною вона не була. Але, зрозуміло, він виконуватиме інструкції, абсолютно не розуміючи їх змісту.

Інший ***підхід – декларативний.*** Інтелектуальному виконавцеві (людині чи комп'ютеру) досить сказати, що треба робити, тобто лише сформулювати завдання, побудувавши всі взаємозв'язки між об'єктами предметної області. Як це завдання виконуватиметься — повинен визначити сам виконавець.

Наведемо для прикладу дві задачі. Різниця в складності їх розв'язання зумовлена тим, що в одному випадку вдається формалізувати постановку і запропонувати чіткі алгоритми розв'язку, в іншому – ні. Отже, потрібно запустити космічний корабель так, щоб він приземлився на Місяць, взяв зразки ґрунту та привіз їх назад. Задача дуже складна, але вона піддається точній алгоритмізації. Математичні методи дозволяють чітко розрахувати траєкторію руху корабля. Навіть якщо він випадково відхилиться від цієї траєкторії, існують методи автоматичного регулювання, які дозволять ліквідувати це відхилення.

А от послати слухняного робота до магазину за пляшкою молока – задача на кілька порядків складніша. 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, він повинен знайти інший магазин. Неможливо передбачити всі ситуації, які можуть виникнути, а відтак — алгоритмізувати розв'язання 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.

2.2. Поповнення первинних інструкцій

Інструкції, написані природною мовою, можуть бути, з одного боку, досить неоднозначними, а з іншого, – вони завжди спираються на те, що виконавець має певний апріорний досвід. Розглянемо такий приклад. Уявіть собі дві таблички. На першій, яка знаходиться при вході на будівництво, написано *"Обов 'язково одягніть на голову каску"*. На другій, біля входу до лондонського метро,— *"Обов'язково візьміть на руки собаку "*. 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 зазначається. 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 дійти 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 не обов'язково.

Людина має певну суму **знань про світ,** яка дозволяє їй орієнтуватися в життєвих ситуаціях та приймати вірні рішення. Крім того, людина вміє певним чином використовувати ці знання. Цих самих рис повинна набути система штучного інтелекту. Можна стверджувати, що здатність до поповнення тих первинних описів ситуацій, що сприймаються безпосередньо, є однією з ключових рис інтелектуальної системи.

Зовнішній світ настільки багатий, що неможливо попередньо описати життєві ситуації з тим ступенем деталізації й однозначності, які б дозволили закласти в системи, що проектуються, 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.

2.3. Формалізація понять алгоритмічносгі та декларативності

Спробуємо формалізувати деякі з впроваджених раніше понять. Введемо такі позначення:

q – первинні інструкції, записані без будь-яких змін у тому вигляді, в якому їх сформулював автор; аналогічно можна визначити *первинний опис ситуації* як результат її безпосереднього сприйняття;

S – множина факторів, які визначають поточний стан виконавців; розглядатимемо *S* як об'єднання двох множин: *S′* та *S″*. Тут *S* – множина *контрольованих факторів,* які відомі авторові процедури і на які він може мати вплив (в цілому, контрольованість і керованість – це не одне й те саме: фактор може бути контрольованим, проте некерованим, але для наших цілей це не має суттєвого значення); *S″* – множина *неконтрольованих факторів;*

Z – знання, які має виконавець;

r_a – робочий алгоритм, який формується та реалізується виконавцем при r_a алгоритмічному підході. При цьому, як правило, автоматично формується і програма $p_л$, що відповідає цьому алгоритму;

$r_д$ – робочий алгоритм, який формується та реалізується інтелектуальним $r_д$ виконавцем при декларативному підході. Нагадаємо, що згідно з фундаментальними тезами Тьюринга та Чорча (все, що може бути виконано будь-яким виконавцем, може бути промодельовано на машині Тьюринга) сам факт виконання завдання свідчить про існування цього алгоритму. Щоправда, цей алгоритм може і не усвідомлюватися виконавцем, а в даному разі не може бути явно сформульований у вигляді алгоритму чи програми. Якщо ж програма,

що відповідає алгоритму r_a повинна бути згенерована, йдеться про *задачу синтезу програм.*

Тоді можна записати такі співвідношення:

$r_a=f(q, S')$

$r_a=h(q, S', S'', Z)$

де f і h — певні функції.

Принциповим є наступне питання. Можна вважати, що алгоритм r_a формується на основі первинних інструкцій q однозначно. При цьому також можуть відбуватися зміна і поповнення первинних інструкцій, але цей процес має повністю контрольований і детермінований характер. *Як* приклад можна навести компіляцію програм, написаних мовами високого рівня. Тому автор процедури може бути впевнений у гарантованому результаті (якщо, звичайно, були дотримані певні формальні вимоги до первинних інструкцій і забезпечено належний стан виконавця).

На противагу цьому, за декларативного підходу такої впевненості немає. Інтелектуальний виконавець певним чином поповнює первинні інструкції, але їх авторові не завжди відомо, яким чином це поповнення відбувається. Тому не можна бути впевненим у результаті виконання інструкцій, і ця втрата гарантованості є неминучою платою за відмову від алгоритмічності. Отже, слід розглянути узагальнене поняття алгоритму, яке дістало назву *"квазіалгоритм "*.

2.4. Квазіалгоритми та джерела квазіалгоритмічіності

Ми пересвідчилися, що чисто алгоритмічний підхід виявив свою неспроможність у разі створення інтелектуальних систем, призначених для вирішення досить складних неформалізованих задач. Тому для теорії і практики штучного інтелекту важливе значення має узагальнене поняття *"квазіалгоритм",* або *"квазіалгоритмічна процедура"*.

Нагадаємо, що алгоритмом називається чітка недвозначна послідовність інструкцій, що зрозуміла виконавцеві і обов'язково приводить до гарантованого результату за скінченний час. На відміну від цього, інструкції квазіалгоритму можуть бути не зовсім чіткими, тож результат виконання квазіалгоритмічної процедури не обов'язково є гарантованим.

Можна окремити як мінімум чотири основні джерела квазіалгоритмічності. Вони можуть доповнювати один одного, а у деяких випадках – описані схожими математичними моделями. Проте ці джерела мають принципово різну природу, а саме:

1. ***Дія випадкових чинників,*** що не залежать від виконавця. Формально, з огляду на цей фактор, квазіалгоритмом слід вважати будь-який алгоритм, що розрахований на роботу за певних умов. Якщо ці умови зміняться, алгоритм може не привести до потрібного результату. Так, навіть програма на Паскалі, яка дає змогу обчислювати суму двох чисел, не буде завершена, якщо під час її виконання в комп'ютері випадково зникне напруга. Якщо ж не вдається чітко окреслити межі застосування алгоритму або забезпечити додержання необхідних умов для його роботи, ми маємо справу зі справжнім

квазіалгоритмом.

2. ***Недостатнє врахування автором алгоритмічної процедури особливостей виконавця.*** Це приводить до того, що виконавець неправильно розуміє, що від нього вимагається. Процедура може бути в принципі алгоритмічною, за нормальних умов приводити до гарантованого результату, але цей результат, можливо, виявиться передбачуваним для автора і навіть несподіваним. Наприклад, програміст бажає обчислити суму перших 50 натуральних чисел, але не знає, як працюють цикли. Він може вважати, що цикл виконується, поки лічильник **менший** за верхню межу (хоча насправді цикл виконується, поки лічильник менший за верхню межу або **дорівнює** їй).

3. ***Нечіткість формулювань,*** що фігурують в описі процедури. Розглянемо будь-який кулінарний рецепт. Наприклад: *"Розтерти тісто, змішати з дрібно порізаними яблуками, додати солі і перцю за смаком і смажити до появи рум'яної скоринки"*. Це є типовим прикладом нечіткості формулювань, що приводить до невизначеностей. До якої межі слід розтирати тісто? Що означає "дрібно порізані"? "за смаком"? Як формалізувати момент "появи рум'яної скоринки"? 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 й уточнити цей опис на основі наявних знань і досвіду. Сам по собі факт виконання завдання свідчитиме 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, такі внутрішні алгоритми для кожного виконавця будуть різними, що зумовить неоднакові результати – у даному разі різні смакові якості виробу.

4 ***"Свобода волі",*** яку можуть мати високоорганізовані, по-справжньому інтелектуальні системи. У них є власні цілі, і якщо дана процедура їм суперечить, ці системи можуть відмовитися її виконувати або виконати не так, як це від них вимагається. Тобто "свобода волі" полягає у тому, що інтелектуальна система самостійно приймає рішення про свою поведінку і залежно від цих рішень може виконувати або не виконувати зовнішні розпорядження, або ж виконувати так, як вона вважає за потрібне.

Слід зазначити, що справжньою квазіалгоритмічністю найчастіше вважають ту, що зумовлена третім і четвертим чинниками.

2.5 Інтелектуальні системи із загальнокібернетичиих позицій

Природно вважати інтелектуальну систему різновидом кібернетичної. Можна розглядати такі ситуації:

• чисто ***зовнішнє керування:*** система сприймає зовнішні інструкції у вигляді речень природної мови, програм, написаних алгоритмічними мовами високого рівня, та ін. і намагається їх виконати. Ці зовнішні інструкції раніше були ***первинними інструкціями.*** Тоді ці інструкції природно вважати зовнішніми чинниками для системи, і саме вони формують її мету;

• ***самокерування:*** система на основі аналізу зовнішньої ситуації приймає

рішення, орієнтовані на досягнення власної мети. Первинною при цьому є зовнішня ситуація у тому вигляді, в якому вона безпосередньо сприймається системою. Результат такого сприйняття раніше був названий ***первинним описом ситуації',***

• ***комбіноване керування,*** за яким система в принципі може бути самокерованою, тобто керуватися власною метою. Однак вона може припускати і зовнішнє керування. Тоді вона розглядає мету керуючої системи як свою власну. Дуже важливим при цьому є питання пріоритетів, а саме: система повинна визначити, яка мета є для неї важливішою: власна чи зовнішня. Зіставлення зовнішньої мети з власною і визначає "свободу волі", про яку йшлося вище.

Отже, ***принциповою рисою інтелектуальної системи*** *є те, що вона намагається планувати свої дії для досягнення певної мети на основі первинного опису ситуації.*

Розглянемо дане положення детальніше. Система має певну мету і прагне так планувати свої дії, щоб досягти її. Вхідними стимулами системи можна вважати поточну ситуацію, що сприймається та аналізується нею. Результатом реакції системи стає зміна зовнішньої ситуації, і поведінка системи коригується залежно від того, бажаною чи небажаною є ця зміна.

Саме таке коригування і є реалізацією одного з основних зворотних зв'язків інтелектуальної системи.

Необхідною компонентою інтелектуальної системи є ***знання про світ,*** на основі яких вона поповнює первинний опис зовнішньої ситуації для І глибшого розуміння цієї ситуації, аналізує можливі наслідки своїх дій і, можливо, формує або коригує свою мету (рис. 1.1).

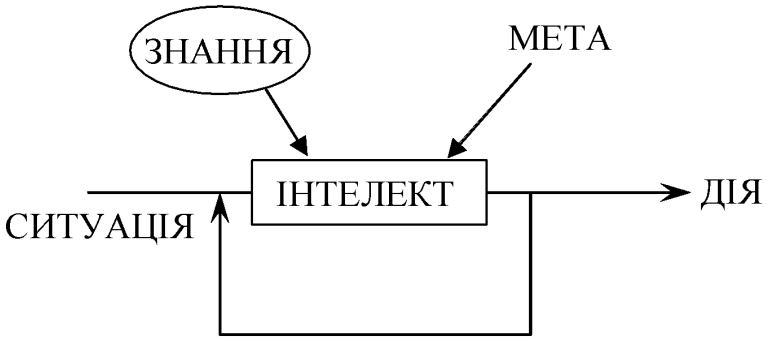


Рисунок 1.1 – Схема самокерування інтелектуальної системи

Усе це стосується будь-якої інтелектуальної системи, незалежно від І того, чи йдеться про природний розум людини або інопланетянина, чи про штучний інтелект. За докладнішого трактування можна вважати, що **І** чим **інтелектуальнішою є система, тим ефективніше вона може досягати мети.**

Людина не отримує всі свої знання і вміння відразу. Тож неможливо так само негайно передати всі необхідні знання і вміння системі штучного інтелекту, призначеній для вирішення досить складних задач. Тому винятково важливе значення має здатність інтелектуальної системи до ***навчання,*** тобто до

поповнення своїх знань і набуття нових можливостей. Таке і навчання може відбуватися шляхом спілкування інтелектуальної системи з людиною або іншою інтелектуальною системою. Якщо ж навчання відбувається самостійно, йдеться про ***самонавчання.***

Справді розвинені інтелектуальні системи повинні мати здатність безпосередньо сприймати оточуючий світ за допомогою ***спеціальних органів чуттів.***

Більше того, це сприйняття має бути пов'язаним з тими поняттями, які входять до знань системи або які формуються в ній. Без такого зв'язку слова залишаються лише словами. Людина ж у процесі мислення істотно спирається на чуттєві асоціації, пов'язані з цими словами. Наприклад, слово "пожежа" викликає у свідомості людини цілий пласт уявлень, пов'язаних з такими зовнішніми подразниками, як "яскраво", "гаряче" і т. п. На основі цих асоціацій виникають інші. Все це має рефлекторний характер і зумовлює активну участь підсвідомих інтуїтивних процесів у людському мисленні, при прийнятті людиною рішень і т. п. Крім того, без набуття таких рис важко говорити про те, що система штучного інтелекту може мати якусь власну мету.

Для людини характерний тісний зв'язок "лівопівкульних" понять, якими вона оперує в процесі логічного мислення, з "правопівкульними" образами, які підкріплюються сигналами, що надходять від органів сприйняття зовнішнього середовища. У сучасних системах штучного інтелекту такий зв'язок практично відсутній.

До переліку інтелектуальних задач можна віднести такі:

– аналіз ситуацій;

– розпізнавання образів;

– логічне мислення;

– розуміння нової інформації;

– навчання і самонавчання;

– планування цілеспрямованих дій.

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.*

Це визначення не може претендувати на повноту та універсальність. Воно фокусує увагу на основних завданнях, які повинна вміти вирішувати будь-яка інтелектуальна природна, технічна або програмна система.

2.6. Типова схема функціонування інтелектуальної системи

Зрозуміло, що функціонування інтелектуальної системи можна описати як постійне прийняття рішень на основі аналізу поточних ситуацій для досягнення певної мети. Тому природно виокремити стани, які утворюють *типову схему функціонування інтелектуальної системи:*

1. Безпосередньо сприймає зовнішню ситуацію; результатом є формування первинного опису ситуації.

2. Зіставляє первинний опис зі знаннями системи і поповнює цей опис; результатом є формування вторинного опису ситуації в термінах знань системи. Цей процес можна розглядати як процес *розуміння ситуації* або як процес перекладу первинного опису на внутрішню мову системи. При цьому можуть змінюватися внутрішній стан системи та її знання.

Вторинний опис може бути не єдиним, і система має змогу вибирати між різними вторинними описами. Крім того, система в процесі роботи може переходити від одного вторинного опису до іншого. Якщо ми можемо формально задати форми внутрішнього представлення описів ситуацій та операції над ними, то сподіватимемося на певний автоматизований аналіз цих описів.

3. Планує цілеспрямовані дії та приймає рішення, тобто аналізує можливі дії та їх наслідки і вибирає ті дії, що найкраще узгоджуються з метою системи. Таке рішення зазвичай формулюється певною внутрішньою мовою (свідомо або підсвідоме).

4. Зворотно інтерпретує прийняте рішення, тобто формує робочий алгоритм для реагування системи.

5. Реалізація реакції системи; наслідками є зміни зовнішньої ситуації і внутрішнього стану системи і т. д.

Проте особливо зауважимо: не слід вважати, що зазначені етапи є повністю відокремленими у тому розумінні, що наступний етап починається тільки після того, як повністю закінчиться попередній. Навпаки, для функціонування інтелектуальної системи характерним є взаємне проникнення цих етапів. Наприклад, ті чи інші рішення можуть прийматися вже на етапі безпосереднього сприйняття ситуації. Насамперед це рішення про те, на які зовнішні подразники слід звертати увагу, а на які не обов'язково. Зовнішніх подразників так багато, що їх сприйняття повинно бути вибірковим.

2.7. Класифікація основних напрямів досліджень

Однією з найтрадиційніших класифікацій напрямів досліджень у галузі штучного інтелекту є класифікація Е. Ханта.

Перший напрям – ***біологічний.*** В його основі лежать спроби вирішення інтелектуальних задач шляхом безпосереднього моделювання психофізіологічних особливостей мозку людини засобами електронно-обчислювальних машин. З часом стало зрозумілим, що розв'язати цю проблему в повному обсязі неможливо і недоцільно. Подібне моделювання особливостей людського розуму призведе до копіювання не тільки позитивних, але й негативних якостей. Стало очевидно, що таке моделювання, хоча й 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 напрям дослідження проблем штучного інтелекту ***прагматичний.*** У ньому майже не розглядається психофізіологічна діяльність людського мозку, а розвиваються підходи до вирішення інтелектуальних задач незалежно від

того, як співвідносяться ці підходи з тими процесами, що відбуваються в мозку людини.

Інша класифікація систем штучного інтелекту визначається моделями і методами, що використовуються для вирішення практичних задач. Йдеться про ***символьний*** та ***конекціоністський підходи,*** що детально розглядатимуться в ході вивчення даного курсу. Зараз же можна відмітити, що символьний підхід орієнтований на моделювання процесів свідомого логічного мислення, а конекціоністський — підсвідомих рефлекторних процесів. Ці підходи можуть застосовуватися як роздільно, так і в комбінації, проте найліпших успіхів можна досягти, лише комбінуючи їх. Зрозуміло, що без використання рефлекторних методів жодна серйозна інтелектуальна проблема не може бути вирішена через *прокляття розмірності.* З іншого боку, рефлекторні 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. Характерною особливістю сучасних досліджень є зближення та взаємопроникнення символьного та конекціоністського підходів.

І, нарешті, дослідження в галузі штучного інтелекту можна класифікувати ***за типами задач.*** Серед практичних інтелектуальних задач, у розв'язку яких можна відмітити реальні здобутки – розпізнавання образів, автоматизація логічних побудов, створення діалогових систем типу "людина-машина" та систем автоматизованого перекладу текстів, робототехніка.

Слід зазначити, що жодна з існуючих систем штучного інтелекту, які працюють у перелічених галузях, не може реалізовувати згадані вище функції інтелектуальної системи в достатньому обсязі. Тому про них краще говорити як про ***інтелектуалізовані системи,*** які мають певні риси, що наближають їх до дійсно інтелектуальних систем.

2.8. Соціальні наслідки інтелектуалізації комп'ютерних технологій

Нарешті можна коротко зупинитися на соціальних наслідках інтелектуалізації інформаційних технологій. Позитивні наслідки зрозумілі та загальновідомі. З наслідків, не дуже сприятливих для людини, найочевидніший – 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. Щодо майбутнього, то тут письменники-фантасти малюють похмурі картини повного захоплення владу комп'ютерами і навіть фізичного винищення людства.

Було запропоновано ряд принципів, орієнтованих на зменшення ризиків, пов'язаних з комп'ютеризацією та інтелектуалізацією. Найвідомішими з них є сформульовані американським фантастом Айзеком Азімовим. Ось ці принципи:

• *комп 'ютер або робот не повинен заподіяти шкоди людині;*

• *комп'ютер або робот має виконувати накази людини, якщо це не суперечить першому принципу;*

• *комп 'ютер або робот повинен прагнути до самозбереження, якщо це не суперечить першим двом принципам.*

Проблема полягає в іншому: як ці або подібні їм принципи втілювати в життя. Слід зауважити, що час, коли штучний інтелект розвинеться настільки, що зможе реально загрожувати людському, настане ще дуже не скоро. Цей висновок, зроблений у 60—70-ті роки XX століття, залишається незмінним і сьогодні (якщо, звичайно, не розглядати можливості, пов'язані І з "доступом до ядерної кнопки" і подібні їм).

Серед інших негативних аспектів інтелектуалізації можна відмітити те, що сучасні комп'ютерні технології полегшують здійснення не тільки корисних справ, але й реалізацію злих умислів, зловживань, злочинів тощо.

3 Представлення знань в інтелектуальних системах

3.1 Знання і деякі підходи до їх подання

У "Філософському словнику" знання визначаються як "відображення об'єктивних властивостей та зв'язків світу".

Знання *є інформаційною основою інтелектуальних систем, оскільки саме вони завжди зіставляють зовнішню ситуацію зі своїми знаннями і керуються ними при прийнятті рішень. Не менш важливим є те, що знання – це систематизована інформація, яка може певним чином поповнюватися і на основі якої можна отримувати нову інформацію, тобто нові знання.*

Існує багато підходів до визначення поняття "знання". На сучасному етапі домінуючою парадигмою, що лежить в основі найвідоміших моделей подання знань у системах штучного інтелекту, можна вважати парадигму (певну сукупність ключових принципів), характерну для ***символьного*** *підходу*[1]. Цю парадигму можна охарактеризувати як *вербально-дедуктивну***,** або ***словесно-логічну,*** через певні чинники:

• будь-яка інформаційна одиниця задається вербальне, тобто у формі, наближеній до словесної, у вигляді набору явно сформульованих тверджень або фактів;

• основним механізмом отримання нової інформації на базі існуючої є *дедукція,* тобто висновок від загального до часткового. Такий дедуктивний підхід є бездоганним з логічного погляду. В його основі лежать транзитивність імплікації (якщо з *a* випливає *b,* з *b* випливає *c,* то з *a* випливає *c*) і дія квантора узагальнення (якщо деяка властивість *P* виконується для будь-якого елемента множини *M,* а *x* є *M,* то *P* виконується для *x).*

Але вербально-дедуктивне подання знань не є повним, оскільки: дедуктивний висновок не виступає єдино можливим. Мислення людини багато в чому є рефлекторним, інтуїтивним. Воно, як правило, спирається на підсвідомі процеси. Людина часто робить висновки за аналогією, асоціацією. Ці висновки не завжди вірні, але вони істотно доповнюють процеси дедуктивного мислення. Провести дедуктивні логічні побудови з самого початку до кінця на практиці, як правило, неможливо. Без підсвідомого мислення стає неможливим будь-яке відкриття – як правило, підсвідоме породження гіпотези, яка потім перевіряється дедуктивним або експериментальним шляхом;

– далеко не всі знання є вербальними. Так, жодне твердження не зберігається в пам'яті людини явно. Відомо, що в основі діяльності мозку людини лежить передача сигналів між нервовими клітинами. Часто людина не може сформулювати свої знання. Наприклад, будь-хто знає, що таке "стіл". Але якщо попросити людину дати визначення цього поняття, можуть виникнути

[1] Символьний підхід до побудови та опису систем штучного інтелекту полягає у тому, що кожне поняття задається окремим словом (символом), зміст якого розкривається через зв'язок з іншими поняттями.

проблеми. Таким чином, поняттями можна оперувати і не знаючи чіткого визначення. Або типовими є слова: "Я не можу пояснити чому, але мені здається, що...".

Тому необхідно розвивати інші моделі знань, окрім вербально-дедуктивних. Наприклад, у рамках конекціоністського підходу[2] можна розглядати моделі на основі однорідного поля знань. ***Однорідним полем знань*** у даному разі називатимемо сукупність простих однорідних елементів, які обмінюються між собою інформацією: нові знання народжуються на основі певних процедур, визначених над полем знань.

Знання про відповідні факти або поняття ***зберігаються полем знань*** *вербальна,* якщо реалізація фіксованого набору процедур, заданих над полем знань, дозволяє за розумний час отримати вербальне формулювання того чи іншого факту або вербальний опис поняття. Знання ***зберігаються*** *полем* ***невербально,*** якщо застосування фіксованого набору процедур, заданих над полем знань, дає змогу використовувати факт або поняття без отримання їх вербального опису.

Людина у своїй діяльності комбінує дедуктивні висновки з рефлекторними рішеннями. Таке комбінування повинно бути притаманне і розвиненим інтелектуальним системам. Наприклад, можна організувати паралельне Функціонування механізмів дедуктивного виведення і конекціоністських механізмів.

3.2. Вербально-дедуктивне визначення знань

У рамках вербально-дедуктивної парадигми можна навести таке визначення знань.

Знаннями інтелектуальної системи називається трійка <F, R, P>, і *де F – сукупність* ***явних фактів,*** *які зберігаються в пам'яті системи в явному вигляді, R – сукупність* ***правил виведення,*** *які дозволяють на основі відомих знань набувати нових знань, P –* ***сукупність процедур,*** *які визначають, яким чином слід застосовувати правила виведення.*

Базою знань *інтелектуальної системи називатимемо сукупність усіх знань, що зберігаються в пам'яті системи.*

У базах знань прийнято розрізняти *екстенсіональну* та *інтенсіональну* частини.

Екстенсіональною частиною бази знань називається сукупність усіх явних фактів, інтенсіональною частиною – сукупність усіх правил і *виведення та процедур, за допомогою яких з існуючих фактів можна виводити нові твердження.*

Надалі ми називатимемо явні факти просто фактами, якщо це не викликатиме непорозумінь.

---

[2] Конекціоністським називається підхід, у рамках якого система штучного інтелекту Розглядається як однорідна сукупність порівняно простих елементів, що взаємодіють між собою.

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

3.3 Дані і знання. Основні визначення

Інформація, з яким мають справу ЕОМ, розділяється на *процедурну* і *декларативну*. Процедурна інформація упредметнена в *програмах*, що виконуються в процесі рішення задач, декларативна інформація – у *даних*, з якими ці програми працюють. Стандартною формою представлення інформації в ЕОМ є *машинне слово*, що складається з визначеного для даного типу ЕОМ числа двійкових розрядів – *бітів*. Машинне слово для представлення даних і машинне слово для представлення команд, що утворять програму, можуть мати однакове або різне число розрядів. Останнім часом для представлення даних і команд використовуються однакові по числу розрядів машинні слова. Однак у ряді випадків машинні слова розбиваються на групи по восім двійкових розрядів, що називаються *байтами*.

Однакове число розрядів у машинних словах для команд і даних дозволяє розглядати їх в ЕОМ у якості однакових інформаційних одиниць і виконувати операції над командами, як над даними. Уміст пам'яті утворить *інформаційну базу*.

У більшості існуючих ЕОМ можливий витяг інформації з будь-якої підмножини розрядів машинного слова аж до одного біта. У багатьох ЕОМ можна з'єднувати два або більш машинні слова в слово з більшою довжиною. Однак машинне слово є основною характеристикою інформаційної бази, тому що його довжина така, що кожне машинне слово зберігається в одній стандартній комірці пам'яті, постаченої індивідуальним ім'ям - адресою осередку. По цьому імені відбувається витяг інформаційних одиниць з пам'яті ЕОМ і запису їх у неї.

Паралельно з розвитком структури ЕОМ відбувався розвиток інформаційних структур для представлення даних. З'явилися способи опису даних у виді векторів і матриць, виникли облікові структури, ієрархічні структури. В даний час у мовах програмування високого рівня використовуються *абстрактні типи даних*, структура яких задається програмістом. Поява *баз даних* (БД) знаменувало собою ще один крок на шляху організації роботи з декларативною інформацією. У базах даних можуть одночасно зберігатися великі обсяги інформації, а спеціальні засоби, що утворять *систему керування базами даних* (СУБД), дозволяють ефективно маніпулювати з даними, при необхідності витягати них з бази даних і записувати них у потрібному порядку в базу.

В міру розвитку досліджень в області ІС виникла концепція знань, що об'єднали в собі багато рис процедурної і декларативної інформації.

В ЕОМ *знання* так само, як і *дані*, відображаються в знаковій формі - у

виді формул, тексту, файлів, інформаційних масивів і т.п. Тому можна сказати, що знання - це особливим образом організовані дані. Але це було б занадто вузьке розуміння. А тим часом, у системах ШІ знання є основним об'єктом формування, обробки і дослідження. *База знань*, нарівні з базою даних, - необхідна складового програмного комплексу ШІ. Машини, що реалізують алгоритми ШІ, називаються *машинами, заснованими на знаннях*, а підрозділ теорії ШІ, зв'язаний з побудовою експертних систем, - *інженерією знань.*

3.4 Особливості знань

Особливості знань:

1. Внутрішня інтерпретованість. Кожна інформаційна одиниця повинна мати унікальне ім'я, по якому ІС знаходить її, а також відповідає на запити, у яких це ім'я згадане. Коли дані, що зберігаються в пам'яті, були позбавлені імен, то була відсутня можливість їхньої ідентифікації системою. Дані могла ідентифікувати лише програма, що витягає їх з пам'яті за вказівкою програміста, що написав програму. Що ховається за тим або іншим двійковим кодом машинного слова, системі було невідоме.

Таблиця 3.1 – Приклад структури для збереження даних

Прізвище	Рік народження	Спеціальність	Стаж, число років

Якщо, наприклад, у пам'ять ЕОМ потрібно було записати зведення про співробітників установи, представлені в табл. 1, то без внутрішньої інтерпретації в пам'ять ЕОМ була б занесена сукупність з чотирьох машинних слів, що відповідають рядкам цієї таблиці. При цьому інформація про те, якими групами двійкових розрядів у цих машинних словах закодовані зведення про фахівців, у системи відсутні. Вони відомі лише програмістові, що використовує дані табл. 1 для рішення виникаючих у нього задач. Система не в змозі відповісти на питання типу "Що тобі відомо про Петрова?" або "Є чи серед фахівців сантехнік?".

При переході до знань у пам'ять ЕОМ вводиться інформація про деяку *протоструктуру інформаційних одиниць*. У розглянутому прикладі вона являє собою спеціальне машинне слово, у якому зазначене, у яких розрядах зберігаються зведення про прізвища, роки народження, спеціальностях і стажах. При цьому повинні бути задані спеціальні словники, у яких перераховані наявні в пам'яті системи прізвища, року народження, спеціальності і тривалості стажу. Усі ці *атрибути* можуть відігравати роль імен для тих машинних слів, що відповідають рядкам таблиці. По них можна здійснювати пошук потрібної інформації. Кожен рядок таблиці буде екземпляром протоструктури. В даний час СУБД забезпечують реалізацію внутрішньої інтерпретованості всіх інформаційних одиниць, що зберігаються в базі даних.

2. Структурованість. Інформаційні одиниці повинні мати гнучку структуру. Для них повинний виконуватися "принцип матрешки", тобто рекурсивна вкладуваність одних інформаційних одиниць в інші. Кожна

інформаційна одиниця може бути включена до складу будь-який інший, і з кожної інформаційної одиниці можна виділити деякі складові її інформаційні одиниці. Іншими словами, повинна існувати можливість довільного встановлення між окремими інформаційними одиницями відносин типу "частина - ціле", "рід - вид" або "елемент - клас".

3 Звязаність. В інформаційній базі між інформаційними одиницями повинна бути передбачена можливість установлення зв'язків різного типу. Насамперед ці зв'язки можуть характеризувати відносини між інформаційними одиницями. Семантика відносин може носити декларативний або процедурний характер. Наприклад, дві або більш інформаційні одиниці можуть бути зв'язані відношенням "одночасно", дві інформаційні одиниці – відношенням "причина –наслідок" або відношенням "бути поруч". Приведені відносини характеризують декларативні знання. Якщо між двома інформаційними одиницями установлене відношення "аргумент – функція", то воно характеризує процедурне знання, зв'язане з обчисленням визначених функцій. Далі будемо розрізняти *відносини структуризації, функціональні відносини, каузальні відносини* і *семантичні відносини*. За допомогою перших задаються ієрархії інформаційних одиниць, другі несуть процедурну інформацію, що дозволяє знаходити (обчислювати) одні інформаційні одиниці через інші, треті задають причинно – наслідкові зв'язки, четверті відповідають всім іншим відносинам.

Між інформаційними одиницями можуть встановлюватися й інші зв'язки, наприклад, що визначають порядок вибору інформаційних одиниць з пам'яті або вказують на те, що дві інформаційні одиниці несумісні один з одним в одному описі.

Перераховані три особливості знань дозволяють увести загальну модель представлення знань, яку можна назвати *семантичною мережею*, що представляє собою ієрархічну мережу, у вершинах якої знаходяться інформаційні одиниці. Ці одиниці володіють індивідуальними іменами. Дуги семантичної мережі відповідають різним зв'язкам між інформаційними одиницями. При цьому ієрархічні зв'язки визначаються відносинами структуризації, а неієрархічні зв'язки - відносинами інших типів.

4. Семантична метрика. На множині інформаційних одиниць у деяких випадках корисно задавати відношення, що характеризує ситуаційну близькість інформаційних одиниць, тобто силу асоціативного зв'язку між інформаційними одиницями. Його можна було б назвати *відношенням релевантності* для інформаційних одиниць. Таке відношення дає можливість виділяти в інформаційній базі деякі типові ситуації (наприклад, "покупка", "регулювання руху на перехресті"). Відношення релевантності при роботі з інформаційними одиницями дозволяє знаходити знання, близькі до вже знайденого.

5. Активність. З моменту появи ЕОМ і поділу використовуваних у ній інформаційних одиниць на дані і команди створилася ситуація, при якій дані пасивні, а команди активні. Усі процеси, що протікають в ЕОМ, ініціюються командами, а дані використовуються цими командами лише в разі потреби. Для ІС ця ситуація не прийнятна. Як і в людини, у ІС актуалізації тих або інших дій сприяють знання, що маються в системі. Таким чином, виконання програм у ІС

повинно ініціюватися поточним станом інформаційної бази. Поява в базі фактів або описів подій, установлення зв'язків може стати джерелом активності системи.

Перераховані п'ять особливостей інформаційних одиниць визначають ту грань, за якої дані перетворюються в знання, а бази даних переростають у *бази знань* (БЗ). Сукупність засобів, що забезпечують роботу зі знаннями, утворить *систему керування базою знань* (СУБЗ). В даний час не існує баз знань, у яких повною мірою були б реалізовані внутрішня інтерпретованість, структуризація, звязаність, уведена семантична міра і забезпечена активність знань.

3.5 Неоднорідність знань. Області і рівні знань

Для успішного здійснення операцій зі знаннями необхідно відокремлювати в базі знань певні фрагменти, які називаються *областями знань*. Ці області знань повинні бути відносно незалежними між собою, що означає наступне:

– зміни в одній області знань не повинні приводити до суттєвих змін в інших областях;

– вирішення складної задачі можна, як правило, звести до підзадач таким чином, що для вирішення кожної з цих підзадач достатньо знань з однієї області.

Такий розподіл є важливим для полегшення проектування і використання бази знань. Зокрема, різні області знань можуть проектуватися незалежно одна від одної.

Виходячи з постановки задачі можна по-різному розділяти знання на області. Так, для експертних систем, які ведуть діалог з користувачем мовою, наближеною до природної, можна виокремити такі області знань:

– *предметна область,* яка містить знання про конкретну предметну галузь, в якій працює експертна система;

– *область мови,* яка містить знання про мову, якою ведеться діалог;

– *область системи,* яка містить знання експертної системи про власні можливості;

– *область користувача,* яка містить знання про користувача. Наявність їх дозволяє враховувати індивідуальні особливості кожного користувача. Наприклад, пояснення користувачеві можуть надаватися залежно від рівня його підготовленості;

– *область діалогу,* яка містить знання про мету діалогу, а також про форми та методи його організації.

Знання можуть різнитися *за рівнем задання і рівнем детальності.* За *рівнями задання* розрізняють знання нульового рівня (конкретні і абстрактні) і знання вищих рівнів знання про знання *(метазнання).* Число рівнів може бути продовжено.

У разі класифікації знань за *рівнями детальності* до уваги береться ступінь деталізації знань. Кількість рівнів детальності залежить від специфіки задачі, обсягу знань і моделі їх задання.

3.6. База знань, як об'єднання простіших одиниць

База знань є кон'юнкцією більш простих тверджень. Можна вважати, що кожне з них відповідає окремому реченню природної мови. Подібні твердження часто називають *концептуальними одиницями.* Останні можуть бути або фактами, або правилами виведення.

Якщо концептуальна одиниця являє собою факт, вона може бути описана певним *предикатом,* тобто логічною функцією, що залежить від тієї чи іншої кількості змінних і може приймати одне з двох можливих значень: *істинність* або *хибність.* Конкретні твердження утворюються з відповідних предикатів шляхом підстановки конкретних значень аргументів.

При цьому для опису одного й того самого твердження можна використовувати різні предикати.

Так, три твердження:

7+5 = 12 8+9 = 17 3+0 = 7

Відповідають одному предикатові *Плюс(a, b, c).*

Перші дві підстановки конкретних значень замість змінних *a, b, c* породжують істинні твердження, третя — хибне. Формально ці твердження будуть мати запис: *Плюс(7, 5, 12), Плюс(8, 9, 17) Плюс(3, 0, 7).*

Але можна розглядати інший, загальніший предикат: *Операція(назва_операції, a, b, c)*, який можна використовувати для запису тверджень не тільки про додавання, але й про будь-яку іншу бінарну операцію: множення, ділення і т. п. Зокрема, можливим є запис: *Операція (Плюс, 7, 5, 12)*.

Який тип розглянутих предикатів слід обирати, залежить від проектувальника бази знань і специфіки конкретної задачі.

3.7 Бінарні предикати і тріада “об'єкт–атрибут–значення”

Концептуальна одиниця може бути достатньо складною. Тому концептуальну одиницю, що є фактом, часто зручно розглядати як кон'юнкцію елементарніших тверджень, а саме — ***бінарних фактів,*** кожний з яких описується ***бінарним предикатом,*** тобто предикатом, який залежить від двох змінних. Якщо формалізувати правила об'єднання бінарних предикатів у складніші одиниці, на основі бінарних предикатів можуть бути сконструйовані які завгодно складні твердження та системи знань. Розглянемо *приклад.*

Маємо твердження “*Студент Іванов отримав п 'ятірку на іспиті зі штучного інтелекту”.*

Якщо переписати цю фразу у вигляді кон’юнкції бінарних фактів, запис може мати вигляд: *Є(Іванов, Студент)*

Є (Ісп_Шт_Інт, Іспит)

Ісп_Шт_Інт(Іванов, 5)

Предикат *Ісп_Шт_Інт* був введений для задання зв'язку між різними інформаційними одиницями.

У вигляді бінарного можна переписати й *унарний предикат,* тобто предикат, який залежить від однієї змінної.

Наприклад, предикат *Птах (Х)***,** який означає, що *Х* є птахом, можна переписати в еквівалентному вигляді *Є(Х, Птах)***.**

Якщо певний унарний предикат описує властивість об'єкта, наприклад, *Літає(Х)*, її можна переписати у вигляді *Літає(Х, так)* або у вигляді *Є(Х, Кл_Літ)***,** де *Кл_Літ* **–** клас об'єктів, які літають.

Бінарним предикатам і бінарним фактам безпосередньо відповідає ***тріада "об'єкт–атрибут–значення".*** Об'єкт у цій тріаді – це, як правило, назва деякої інформаційної одиниці, атрибут – назва певної ознаки, а значення – конкретне значення цієї ознаки для даного об'єкта.

Тріада "об'єкт–атрибут–значення" є просто іншою формою запису бінарного факту. Так, якщо переписати наш приклад у вигляді сукупності згаданої тріади, запис матиме вигляд:

Іванов–Є–Студент
Ісп_Шт_Інт–Є–Іспит
Іванов–Ісп_Шт_Інт–5.

Остання тріада цього прикладу задає зв'язок між різними об'єктами.

3.8 Проблема неточних і неповних знань

Раніше ми розглядали проблеми, які необхідно було вирішувати при проектуванні та розробці баз знань. Серед інших проблем можна відмітити такі.

Знання можуть бути **неповними.** Це означає, що для доведення або спростування певного твердження може не вистачати інформації. У багатьох системах логічного виведення прийнято ***постулат замкненості світу***. Це означає, що на запит про істинність деякого твердження система відповідає "*так"* тоді і тільки тоді, коли його можна довести; якщо ж довести, твердження неможливо, система відповідає "*ні"*. Водночас "*неможливо вести через нестачу інформації"* і *"доведено, що ні "* — це зовсім не одне й те саме. З огляду на це бажано, щоб експертна система запитувала користувача про факти, яких не вистачає.

Знання можуть бути **недостовірними.** Наприклад, на результат виконання продукції можуть впливати випадкові чинники (об'єктивна невизначеність) або ж експерт може бути не зовсім упевненим у деякому факті чи правилі виведення (суб'єктивна невизначеність).

Ненадійність знань і недостовірність наявних фактів обов'язково повинні враховуватися в процесі логічних побудов. Звичайно, можна було б просто відкидати факти та правила виведення, які викликають сумнів, але тоді довелося б відмовитися від цінної інформації. Тому необхідно розвивати процедури, які дозволяють здійснювати логічні побудови при недостовірних даних, і використовувати ці процедури в експертних системах. Необхідно враховувати **модальності,** а саме: необхідність або можливість того чи іншого факту, ставлення суб'єкта до деякого твердження і т. п. Крім того, в таких системах часто доводиться мати справу з неточно визначеними, нечіткими поняттями, такими, як *"великий ", "маленький "* тощо.

3.9 Класифікація рівнів розуміння

Важливою задачею, що пов’язана з представленням знань і інтелектуальних системах, є розуміння системою здобутих знань. В існуючих ІС можна виділити п'ять основних рівнів розуміння і два рівні метарозуміння.

Перший рівень характеризується схемою*, що* показує, що будь-які відповіді на питання система формує тільки на основі прямого змісту, уведеного з тексту. Якщо, наприклад, у систему введений текст: "О восьмій ранку, після сніданку, Петров пішов в університет. О другій годині він повернувся додому. Після обіду він пішов гуляти", то на першому рівні розуміння система зобов'язана уміти відповідати правильно на питання типу: "Коли Петров пішов на заняття?" або "Що зробив Петров після обіду?". У лінгвістичному процесорі відбувається морфологічний, синтаксичний і семантичний аналіз тексту і питань, що відносяться до нього. На виході лінгвістичного процесора виходить внутрішнє представлення тексту і питань, з якими може працювати блок висновку. Використовуючи спеціальні процедури, цей блок формує відповіді. Іншими словами, уже розуміння на першому рівні вимагає від ІС визначених засобів представлення даних і висновку на цих даних.

Другий рівень: На другому рівні додаються засоби логічного висновку, засновані на інформації, що міститься в тексті. Це різноманітні логіки тексту (часова, просторова, каузальна і т.п.), що здатні породити інформацію, що явно відсутня у тексті. Для нашого приклада на другому рівні можливе формування правильних відповідей на питання типу: "Що було раніше: заняття в університеті або обід?" або "Гуляв Петров після повернення з університету?" Тільки побудувавши часову структуру тексту, ІС зможе відповісти на подібні питання.

Схема ІС, за допомогою якої може бути реалізований другий рівень розуміння, має ще одну базу знань. У ній зберігаються закономірності, що відносяться до часової структури подій, можливої їхньої просторової організації, каузальної залежності і т.п., а логічний блок володіє всіма необхідними засобами для роботи з псевдофізичними логіками.

Третій рівень: До засобів другого рівня додаються правила поповнення тексту знаннями системи про середовище. Ці знання в ІС, як правило, носять логічний характер і фіксуються у виді сценаріїв або процедур іншого типу. На третьому рівні розуміння ІС повинна дати правильні відповіді на питання типу: "Де був Петров о десятій ранку?" або "Звідки Петров повернувся о другій годині дня?" Для цього треба знати, що означає процес "перебування в університеті" і, зокрема, що цей процес є безперервним і що суб'єкт, що бере участь у ньому, увесь час знаходиться "в університеті".

Схема ІС, у якій реалізується розуміння третього рівня, зовні не відрізняється від схеми другого рівня. Однак у логічному блоці повинні бути передбачені засоби не тільки для чисто дедуктивного висновку, але і для висновку по сценаріях.

Три перерахованих рівні розуміння реалізовані у всіх практично працюючих ІС. Перший рівень і частково другий входять у різноманітні

системи спілкування природною мовою.

Наступні два рівні розуміння реалізовані в існуючих ІС лише частково.

Четвертий рівень: Замість тексту *в* ній використовується розширений текст, що породжується лише при наявності двох каналів одержання інформації. По одному в систему передається текст, по іншому – додаткова інформація, відсутня в тексті. При людській комунікації роль другого каналу, як правило, грає зір. Більш одного каналу комунікації мають інтелектуальні роботи, що володіють зором.

Зоровий канал комунікації дозволяє фіксувати стан середовища "тут і зараз" і вводити в текст інформацію, що спостерігається. Система стає здатної до розуміння текстів, у які введені слова, прямо зв'язані з тією ситуацією, у якій породжується текст. На більш низьких рівнях розуміння не можна зрозуміти, наприклад, текст: "Подивіться, що зробив Петров! Він не повинний був брати це!" При наявності зорового каналу процес розуміння стає можливим.

При наявності четвертого рівня розуміння ІС здатна відповідати на питання типу: "Чому Петров не повинний був брати це?" або "Що зробив Петров?" Якщо питання, що надійшло в систему, відповідає третьому рівневі, то система видає потрібну відповідь. Якщо для відповіді необхідно залучити додаткову інформацію, то внутрішнє представлення тексту і питання передається в блок, що здійснює співвіднесення тексту з тією реальною ситуацією його породження, що доступна ІС по зоровому або якому-небудь іншому каналі фіксації ситуації зовнішнього світу.

П'ятий рівень: Для відповіді на цьому рівні ІС крім тексту використовує інформацію про конкретного суб'єкта, що є джерелом тексту, і загальну інформацію, що зберігається в пам'яті системи, що відноситься до комунікації (знання про організації спілкування, про цілях учасників спілкування, про норми участі в спілкуванні). Теорія, що відповідає п'ятому рівневі, – це так називана теорія мовних актів.

Була звернена увага на те, що будь-яка фраза не тільки позначає якесь явище дійсності, але і поєднує в собі три дії: *локуцию, ілокуцию і перлокуцію.* Локуція – це розмова, сама по собі, тобто ті дії, які слід зробити, щоб висловити свою думку. Ілокуція – це дія за допомогою розмови: питання, спонукання (наказ або прохання) і твердження. Перлокуція - це дія, якою намагаються здійснити деякий вплив на слухаючого: "лестити", "дивувати", "умовляти" і т.д. *Мовний акт* можна визначити як мінімальну осмислену (або доцільну) одиницю мовного поводження. Кожен мовний акт складається з локутивного, ілокутивного і перлокутивного акта.

Для четвертого і п'ятого рівнів розуміння цікаві результати по невербальним (несловесним) компонентах спілкування і психологічних принципів, що лежать в основі спілкування. Крім того, у правила поповнення тексту входять правила висновку, що спираються на знання про даного конкретного суб'єкта спілкування, якщо такі знання в системи є. Наприклад, система може довіряти даному суб'єктові, вважаючи, що породжуваний їм текст щирий. Але може не довіряти йому і розуміти текст, коректуючи його у відповідності зі своїми знаннями про суб'єкта, що породив текст. Знання такого

типу повинні спиратися на психологічні теорії спілкування, що поки розвиті недостатньо.

Наприклад, на вхід системи надходить текст: "Ніна обіцяла незабаром прийти". Якщо про Ніну в системи немає ніякої інформації, вона може звернутися до бази знань і використовувати для оцінки часового покажчика "незабаром деяку нормативну інформацію. З цієї інформації можна довідатися, що з великою часткою впевненості "незабаром" не перевищує півгодини. Але в системи може матися спеціальна інформація про ту Ніну, про яку мова йде у вхідному тексті. У цьому випадку система, одержавши потрібну інформацію з бази знань, може приготуватися, наприклад, до того, що Ніна швидше за все прийде не раніше чим через годину.

Перший метарівень: На цьому рівні відбувається зміна вмісту бази знань. Вона поповнюється фактами, відомими системі й містяться в тих текстах, що у систему введені. Різні ІС відрізняються друг від друга характером правил породження фактів зі знань. Наприклад, у системах, призначених для експертизи в області фармакології, ці правила спираються на методи індуктивного висновку і розпізнавання образів. Правила можуть бути засновані на принципах імовірностей, розмитих висновків і т.п. Але у всіх випадках база знань виявляється апріорно неповної й у таких ІС виникають складнощі з пошуком відповідей на запити. Зокрема, у базах знань стає необхідним немонотонний висновок.

Другий метарівень: На цьому рівні відбувається породження метафоричного знання. Правила породження знань метафоричного рівня, використовувані для цих цілей, являють собою спеціальні процедури, що спираються на висновок за аналогією й асоціації. Відомі в даний час схеми висновку за аналогією використовують, як правило, діаграму Лейбница, що відбиває лише окремий випадок міркувань за аналогією. Ще більш бідні схеми асоціативних міркувань.

Якщо розглядати рівні і метарівні розуміння з погляду архітектури ІС, то можна спостерігати послідовне нарощування нових блоків і ускладнення реалізованих ними процедур. На першому рівні досить лінгвістичного процесора з базою знань, що відносяться тільки до самого тексту. На другому рівні в цьому процесорі виникає процедура логічного висновку. На третьому рівні необхідна база знань. Поява нового каналу інформації, що працює незалежно від вихідного, характеризує четвертий рівень. Крім процедур, зв'язаних з роботою цього каналу, з'являються процедури, що погоджують між собою результати роботи двох каналів, що інтегрують інформацію, одержувану по кожному з них. На п'ятому рівні розвиток одержують різноманітні способи висновку на знаннях і даних. На цьому рівні стають важливими моделі індивідуального і групового поводження. На метарівнях виникають нові процедури для маніпулювання знаннями, яких не було на більш низьких рівнях розуміння. І цей процес носить відкритий характер. Розуміння в повному обсязі – якась, очевидно, недосяжна мрія. Але розуміння на рівні "побутового розуміння" людей у ІС цілком досяжно.

Існують і інші інтерпретації феномена розуміння. Можна, наприклад,

оцінювати рівень розуміння по здатності системи до пояснення отриманого результату. Тут можливий не тільки рівень *пояснення,* коли система пояснює, що вона зробила, наприклад, на основі введеного в неї тексту, але і рівень *обґрунтування (аргументації),* коли система обґрунтовує свій результат, показуючи, що він не суперечить тій системі знань і даних, якими вона розташовує. На відміну від пояснення обґрунтування завжди зв'язане із сумою фактів і знань, що визначаються сучасний момент існування системи. І текст, що вводиться для розуміння, в одних станах може бути сприйнятий системою як щирий, а в як помилковий. Крім пояснення й обґрунтування можлива ще одна функція, зв'язана з розумінням текстів, – *виправдання.* Виправдати щось означає, що виведені твердження не суперечать тій системі норм і цінностей, що закладені в ІС. Існуючі ІС типу експертних систем, як правило, здатні давати пояснення і лише частково обґрунтування. У повному обсязі процедури обґрунтування і виправдання ще не реалізовані.

4 Логічні моделі та метод резолюцій

4.1 Логічні побудови та логічні моделі

Поняття "логіка" має глибокі філософські корені. Для теорії і практики штучного інтелекту особливе значення має таке питання: яким чином можна формалізувати логічні побудови так, щоб вони могли здійснюватися автоматично, без участі людини.

Як уже зазначалося, логічне виведення за дедукцією (від загальних правил до часткових наслідків) є одним з основних механізмів мислення, до того ж цілком бездоганним з погляду логіки.

Можна розглядати, як мінімум, дві задачі пов'язаних з автоматизацією дедуктивних побудов:

– ***задача виведення наслідків,*** яка формулюється так: знайти всі вірно побудовані формули, які можна логічним шляхом вивести з аксіом на основі правил виведення; або, у практичнішій інтерпретації - знайти всі наслідки із заданих фактів;

– задача перевірки справедливості даного твердження. Часто вона трактується як задача ***автоматичного доведення теорем*** і формулюється таким чином. Дана деяка теорія (логічна модель); потрібно встановити, чи є певне твердження теоремою, тобто чи можна його вивести з аксіом даної теорії.

Автоматизація дедуктивних побудов знаходить широке практичне застосування, наприклад, в експертних системах. Як ще один приклад можна згадати програмні системи, які аналізують математичні тексти, написані спрощеною математичною мовою, та перевіряють правильність доведень, що містяться у цих текстах.

Механізм автоматичного доведення теорем було покладено в основу однієї з основних парадигм сучасного програмування – ***логічного програмування.*** Програма розглядається як набір логічних формул разом з теоремою, що має бути доведена. Найвідомішим представником логічного програмування є мова Пролог.

Логічні моделі є одним з основних засобів завдання знань. Позитивною характеристикою логічних моделей виступає однозначність теоретичного обґрунтування і можливість реалізації формально точних логічних побудов; недоліком − формальний процедурний стиль мислення. Людська логіка − інтелектуальна модель з нечіткою структурою. У цьому полягає її відмінність від формальної логіки.

Логічною моделлю *називається формальна система, що задається четвіркою <T, P, A, B>. Тут T – множина базових елементів (алфавіт), P – множина синтаксичних правил, на основі яких конструюються правильно побудовані формули; A – множина аксіом, тобто* ті *формул, які приймаються за істинні, B – множина правил виведення.*

У рамках логічної моделі істинному твердженню відповідає ***теорема,*** тобто правильно побудована формула, яка може бути виведена з аксіом шляхом скінченого числа застосувань правил виведення.

Логічні моделі та Продукційні системи основний акцент роблять саме на дедуктивному логічному виведенні.

У найтиповішому випадку логічна модель завдання знань базується на формалізмах ***логіки предикатів першого порядку.***

4.2 Основні положення числення предикатів

Предикатом називається деяка логічна функція від довільного числа аргументів, яка приймає одне з двох можливих значень − *"істина "* або *"хибність ".* Предикат можна розуміти як певне твердження, істинність якого залежить від змінних — об'єктів, про які йдеться в цьому твердженні. Як приклад можна навести фразу *"X більше за* 2 ". Цей предикат є функцією від аргументу *X* і набуває значення *"істина",* наприклад, при *X*= 3, і *"хибність"* при *X*= 1.

Логіку предикатів деякою мірою можна вважати спеціальним математичним апаратом формалізації людського мислення. Тому визнається, що мови програмування логічного типу є найзручнішими для роботи з базами знань.

Числення предикатів використовує такі основні елементи:

1) константи (константні терми) c_1, c_2, ...;
2) змінні (змінні терми) x_1, x_2, ...;
3) функціональні літери f_1, f_2, ...;
4) предикатні літери p_1, p_2, ...;
5) логічні символи $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\sim$
6) спеціальний символ •.

Елементарне твердження складається з предиката і зв'язаних з ним термів. Складні твердження будуються з елементарних за допомогою логічних зв'язок. Серед них можна виділити логічні зв'язки: *"і" (and, ∧), "або " (or, ∨), "ні" (not, ~)* та імплікацію (→). Імплікація посідає особливе місце, оскільки вона використовується для побудови правил виведення і читається "якщо ...,тоді ...".

Наступний перелік містить опис зв'язків, які використовуються в логіці, та їх змістовних інтерпретацій. Тут *a* та *b* означають будь-які твердження.

Таблиця 4.1 – Основні логічні операції

Зв'язок	Запис	Інтерпретація
Заперечення	$\sim a$	"не є *a*"
Кон'юнкція	$a\&b$	"*a* та *b"*
Диз'юнкція	$a \vee b$	*"a* або *b"*
Імплікація	$a \rightarrow b$	"з *a* випливає *b*"
Тотожність (еквівалентність, рівність)	$a \leftrightarrow b$	„*a* еквівалентне *b*”

Наприклад, запис:

чоловік (женя) $\vee$ *жінка (женя)*

є однією з формалізацій запису твердження, що людина Женя може бути

або чоловіком, або жінкою. Аналогічно,

чоловік (*X*) → *людина* (*X*) означає, що якщо дехто на ім'я *X* є чоловіком, то він є і людиною.

Для імплікації та тотожності справедливі такі твердження:

a → *b* дорівнює (~ a) ∨ *b,*

a ↔ *b* дорівнює *(a* ∧ *b)* ∨ (~ *a* ∧ ~ *b*),

a ↔ *b* дорівнює *(a* → b*)* ∧ (*b* → *a*).

З кожним предикатом може бути пов'язаний *квантор* – елемент, який визначає, за яких умов предикат перетворюється на істинне висловлювання. Розрізняють *квантор узагальнення* (∀) і *квантор існування* (∃).

Якщо *u* позначає змінну, а *r* – твердження, то запис ∀u *r* означає, що *"r справджується для всіх u ",* а запис ∃u *r* означає, що *"існує u, для якої r справджується".* Перший квантор називають квантором узагальнення, оскільки кажуть про будь-що у всесвіті *("для будь-якого u, ... ").* Для зручності запису позначатимемо його *all*, другий називається квантором існування, оскільки вказує на існування деяких об'єктів *("існує u таке, що ... ").* Для зручності запису його позначатимемо exists.

Ось приклади використання цих кванторів:

All (X, чоловік (X)→ людина (X))

Означає, що для будь-якої змінної *X,* якщо вона означає чоловіка, то означатиме і людину також, або ж для будь-якого *X,* якщо *X* – чоловік, то *X* – людина; або ж: всі чоловіки –люди.

За тим самим принципом:

exists (Z, батько (Іван, Z) ∧ *жінка* (Z))

означає, що існує певний об'єкт, що може бути позначений змінною Z. При цьому Іван є батьком Z і Z – жінка. Ми можемо прочитати це таким чином: *існує Z таке, що Іван є батьком Z і Z – жінка;* або: *Іван має дочку.*

У логіці предикатів елементарним об'єктом, який має значення "істина", є *атомарна формула (літерал).* Атомарна формула містить символьні позначення предиката і термів, які відіграють роль аргументів цього предиката. Узагальнено позначення предиката є ім'ям відношення, яке існує між аргументами.

Атомарна формула записується як позначення предиката і має такий вигляд: $P(t_1, t_2, ..., t_n)$

де *P* – позначення предиката, а $t_1, t_2, ..., t_n$ – терми.

Число термів для кожного предиката фіксоване і називається його *арністю.* Терми визначаються так:

– константний терм – терм;

– змінний терм – терм;

– якщо арність функціональної літери *f* є *n,* а $t_1, t_2, ..., t_n$ – терми, тоді $f(t_1, t_2, ..., t_n)$ також терм.

Вірно побудована формула отримується в результаті комбінування атомарних формул за допомогою логічних зв'язок.

Символ • позначає хибну замкнуту формулу і визначає поняття "протиріччя". Так, формула *A*→ • означає хибність *A,* вона еквівалентна

формулі ~ A.

Серед формул можна виокремити спеціальний клас формул: тотожньо істинні формули, які називають тавтологіями. Приклад тавтології:

$\sim\sim A \leftrightarrow A$.

Поняття формули вводиться рекурсивно:

– атом є формула;

– якщо A – формула то і $\sim A$ – фомула;

– якщо A формула то $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \equiv B$ теж формули;

– якщо A – формула і x – змінна, то $\forall x A$, $\exists x A$ – формули.

Формули логіки предикатів можна будувати виходячи з трьох множин: F – символи (імена функцій; P – символи (імена) предикатів; V – змінні. Об'єднання $A \cup P$ називається сигнатурою, вона визначається змістом вирішуваної задачі.

Змінна в формулі можуть бути зв'язаними та вільними. Входження змінної у формулу є зв'язаним, якщо змінна знаходиться за квантором або входить в область дії квантора по цій змінній. В іншому випадку входження – вільне.

Наприклад, $t(x) \wedge \forall y[s(x,y) \rightarrow \exists x(r(x,y) \vee t(y))]$

Змінна є вільною, якщо є хоча б одне вільне входження змінної в формулу.

Співставлення формулі логіки предикатів предикату є інтерпретацією. Інтерпретацією формули на не порожній множині M називається функція, яка задана на сигнатурі , що ставить у відповідність:

1) константі – елемент з M;

2) символу n–містної функції – деяку n–містну функцію, що визначається на M;

3) символу n–містного предикату – n–містний предикат, заданий на M;

Наприклад, задана сигнатура (P,D), яка визначає одномісний предикат $P(x)$ та двомісний предикат $D(x,y)$ та формулу $F=P(x)\& \forall y(P(y) \rightarrow D(x,y))$ на множині $M=\{2,3,6,9,12,15\}$

Нехай – «непарне число» та – «x поділяє y». В цьому випадку F отримає у відповідність x=3.

Якщо ϕ – інтерпретація, то предикат, що відповідає формулі F позначається, як $\phi(F)$.

Формули $F(x_1,\ldots,x_n)$ та $G(x_1,\ldots,x_n)$ рівносильні, якщо для любої інтерпретації ϕ з областю M вирази $(\phi F)(a_1,\ldots,a_n)$ та $(\phi G)(a_1,\ldots,a_n)$ при будь-яких $a_1,\ldots,a_n$ з M одночасно істинні або хибні.

Формула $F(x_1,\ldots,x_n)$ тотожно істинна, якщо для любої інтерпретації ϕ з областю M вирази $(\phi F)(a_1,\ldots,a_n)$ при будь-яких $a_1,\ldots,a_n$ з M є істинні.

Поняття рівносильності та тотожної істинності є рівними. Так, формули $F(x_1,\ldots,x_n)$ та $G(x_1,\ldots,x_n)$ рівносильні тоді і тільки тоді коли формули $F(x_1,\ldots,x_n) \leftrightarrow G(x_1,\ldots,x_n)$ тотожно істинні.

4.2.2 Поняття логічного наслідку

Формула $G(x_1,\ldots,x_n)$ називається логічним наслідком формул $F_1(x_1,\ldots,x_n)$, …, $F_k(x_1,\ldots,x_n)$, якщо для любої інтерпретації ϕ з областю M та при будь-яких $a_1,\ldots,a_n \in M$, із істинності виразів $(\phi F_1)(a_1,\ldots,a_n)$, …, $(\phi F_k)(a_1,\ldots,a_n)$ слідує

істинність виразу $(\phi G)(a_1,\ldots,a_n)$.

Поняття логічного наслідку пов'язано з поняттям не виконуваності. Так, формула $G(x_1,\ldots,x_n)$ називається логічним наслідком формул $F_1(x_1,\ldots,x_n)$, …, $F_k(x_1,\ldots,x_n)$ тоді і тільки тоді, токи множина формул $\{F_1(x_1,\ldots,x_n)$, …, $F_k(x_1,\ldots,x_n)$, $\sim G(x_1,\ldots,x_n)\}$ не виконувана.

4.3 Приведення формул до нормальних форм

Як зазначалося раніше, логічна формула, записана за допомогою імплікацій і тотожностей, може бути переписана в термінах кон'юнкцій, диз'юнкцій і заперечень. Розглянемо, як твердження числення предикатів перетворюється на спеціальну уніфіковану фразову форму.

Дамо неформальний опис такого переведення, що має шість основних дій.

Дія 1. Вилучення імплікацій і тотожностей.

Вилучення імплікацій "→" і тотожностей "↔" можна провести на основі формул:

a $\rightarrow b$ дорівнює ($\sim$ a) $\vee$ b,

$a \leftrightarrow b$ дорівнює $(a \wedge b) \vee (\sim a \wedge \sim b)$,

$a \leftrightarrow b$ дорівнює $(a \rightarrow b) \wedge (b \rightarrow a)$.

Так, формула $\forall x(P(x) \rightarrow F(x))$ перетвориться на формулу $\forall x(\sim P(x) \vee F(x))$.

Дія 2. Втягнення заперечення всередину.

Ця дія застосовується для випадків, коли "~" вживана у формулі, що не є частковою. Тоді робиться така заміна на основі правил:

$\sim (a \wedge b)$ дорівнює $(\sim a) \vee (\sim b)$,

$\sim \exists u, r$ дорівнює $\forall u, \sim r$,

$\sim \forall u, r$ дорівнює $\exists u, \sim r$.

Наприклад, $\sim(P(x)\& F(x))$ переводиться у $\sim P(x) \wedge \sim F(x)$.

Після другої дії у наших формулах запишаться лише заперечення, прив'язані до конкретних формул. Часткові твердження з або без "~" попереду ми називаємо ***точними.*** У наступних кількох діях ми будемо поводитися з точними твердженнями як з простими одиницями.

Дія 3. Заміна змінних.

Наступна дія включає вилучення існуючих кванторів. Це робиться шляхом вставлення нових константних символів на місце, де були змінні, що являють собою існуючі квантори. Тобто, замість того, щоб говорити, що існує об'єкт з певним набором характеристик, можна створити ім'я для такого об'єкта і просто сказати, що в нього є характеристики. В цьому і полягає основна суть введення сколемівських констант. Ця дія є не повністю еквівалентною, але вона має одну важливу властивість. Існує інтерпретація символів формули, яка справджує формулу тільки тоді, коли існує інтерпретація для сколемівської версії формули. Для наших потреб цієї форми відповідності достатньо.

Тому, $\exists x(P(x) \wedge F(x,y))$ перетворюється за допомогою сколемізації на $P(c_1) \wedge F(c_1, y)$,

де c_1, − деяка нова константа, що ніде раніше не використовувалась. Тут

важливим є застосування символу, який раніше ніде не вживався, оскільки твердження $\exists$ не каже, що є певна відповідність, а лише стверджує, що є така відповідність існує. Може виявитись, що "c_1" пов'язана з якимось іншим константним символом, але ця інформація не надається даним твердженням.

Коли у формулі використовуються квантори узагальнення, сколемізація стає не такою простою. Наприклад $\forall y \exists x(P(x, y))$ переходимо до $\forall y\, P(h(y), y)$.

Дія 4. Переміщення квантора узагальнення назовні.

Ця дія дуже проста. Квантор узагальнення переміщується назовні, при цьому формула зберігає свій попередній зміст. Наприклад,

$\forall x(P(x) \rightarrow \forall y(G(y) \rightarrow F(x,y)))$ перетворюється на $\forall x \forall y(P(x) \rightarrow (G(y) \rightarrow F(x,y)))$.

Оскільки кожна змінна тепер зв'язана квантором узагальнення ззовні формули, то квантори узагальнення більше не несуть додаткової інформації. Отже, можна скоротити формули, опустивши ці квантори. Ми просто мусимо пам'ятати, що кожна змінна зв'язана кванторами узагальнення, які ми опустили.

Дія 5. Перенесення "$\wedge$" через "$\vee$".

Логічна формула повинна бути зведена до особливої форми – нормальної кон'юнктивної формули, в якій кон'юнкції не з'являються посеред диз'юнкцій. Для цього існують два правила:

$(a \wedge b) \vee c$ дорівнює $(a \vee c) \wedge (b \vee c)$,

$a \vee (b \wedge c)$ дорівнює $(a \vee b) \wedge (a \vee c)$.

Наприклад, фразу $P(x) \vee (G(y,x) \wedge (F(x) \vee D(x))$ можна перетворити на фразу $(P(x) \vee (G(y, x)) \wedge (P(x) \vee F(x) \vee D(x))$.

Дія 6. Перехід до фраз.

Тепер наша формула є набором атомарних формул, розділених "$\vee$" або пов'язаних "$\wedge$". Якщо не брати до уваги "$\vee$", в нас може вийти наступне:

$(A \wedge B) \wedge (C \wedge (D \wedge E))$,

де змінні можуть заміщати складні твердження, але без "$\wedge$" всередині. По-перше, тепер дужки не потрібні, оскільки для "$\wedge$" вони не мають значення, і наш вираз еквівалентний до

$A \wedge B \wedge C \wedge D \wedge E$.

По-друге, порядок теж не має значення. По-третє, нам не потрібні і "$\wedge$", оскільки ми знаємо, що наша формула – це послідовність тверджень, з'єднаних "$\wedge$", і останні можна опустити, сказавши, що це є множина $\{A, B, C, D, E\}$. Кажучи, що це множина, ми наголошуємо на тому, що порядок не суттєвий. Елементи цієї множини називаються фразами. Отже, будь-яка формула предикатного числення еквівалентна (в певному розумінні) до множини фраз.

Повернемося до самої фрази. Оскільки елементами фрази є або точні, або розділені точно "$\vee$" твердження, то їх загальний вигляд може набувати кого вигляду:

$((F \vee W) \vee X) \vee (Y \vee Z)$,

змінні є точними. Тому можна знову перейти до множини розділених фраз $\{F, W, X, Y, Z\}$.

Тепер формула остаточно досягла фразової форми. Підсумуємо дещо.

Отже, фразова форма логіки предикатів задає такий спосіб запису формул, який використовують тільки з'єднувачі типу $\wedge$, $\vee$, $\sim$.

Негативна або позитивна атомарна формула називається літералом

Літерал називається *позитивним,* якщо він не стоїть під знаком заперечення.

Літерал називається *негативним,* якщо він стоїть під знаком заперечення.

Диз'юнктом (або фразою теорії) називається диз'юнкція деякої кількості атомарних формул.

Кожна формула – множина літералів, які з'єднані символом $\vee$. Негативні і позитивні літерали відповідно групуються. Схематично фраза має такий вигляд:

$P_1 \vee P_2 \vee \ldots P_n \vee N_t \vee N_2 \vee \ldots N_m$,

де $P_1, \ldots, P_n$ – позитивні літерали, а $N_1, \ldots, N_m$ – негативні. З іншого боку, фразу можна розглядати як узагальнення поняття імплікації. Дійсно, якщо A і B атомарні формули, тоді формулу $A \rightarrow B$ можна переписати і в такому еквівалентному вигляді: $\sim A \vee B$. Звідси отримаємо фразову формулу $B \vee \sim A$.

Фраза має таку семантику. Всі позитивні атомарні формули виступають у ролі альтернативних висновків, а негативні є необхідними умовами. Наприклад, $A \vee B \vee \sim C \vee \sim D$ зазначає, що A і B будуть істинними тоді і тільки тоді, коли є істинними C і D. Елементарна фраза має тільки один літерал.

У теорії і практиці автоматичного доведення теорем найвживанішими є фрази спеціального типу, які називаються *фразами Хорна* (або *хорківськими диз'юнктами)*

Фразою Хорна *називається диз'юнкція довільної кількості атомарних формул, з яких позитивною є не більше ніж: одна (тобто у фразі Хорна лише одна атомарна формула може не стояти під знаком заперечення).* Фрази Хорна по суті є імплікаціями. Справді, розглянемо фразу $\sim A \vee \sim B \vee \sim C \vee D$ (кожний з літералів *A, B, C, D* може залежати від довільної кількості констант і змінних, тут це несуттєво). Усі літерали, крім *D,* є негативними, отже, дана фраза являє собою фразу Хорна. Але відповідно до правил де Моргана вона еквівалентна фразі $\sim (ABC) \vee D$. А це, в свою чергу, є іншою формою запису імплікації (твердження *D* випливає з кон'юнкції тверджень *A, B, C*).

Розглянемо ще один приклад. Нехай маємо формулу

(особа (петро) $\wedge$ *особа (Іван))* $\wedge$ *((особа (X)* $\vee$ *мати (X, Y))* $\vee$ *~особа (Y)),* яка після відповідних дій перетворюється на:

особа (петро):-.

особа (Іван):-.

особа (X) :- мати (X, Y), *особа* (*Y*).

Останній запис уже нагадує текст прологівської програми.

4.4 Метод резолюцій

Тепер, коли ми знаємо, як зводити формули числення предикатів до певної стандартної форми, ми повинні дізнатися, що з ними робити, як і де їх можна застосувати. Цілком очевидно, що, маючи набір тверджень, варто

дослідити, які наслідки з них випливають. Нагадаємо, що твердження, які ми вважаємо істинними, називають аксіомами та гіпотезами, а ті твердження, що з них випливають, – теоремами.

Як уже зазначалося раніше, звістка про те, що цифрові комп'ютери можуть автоматично доводити теореми, у 60-ті роки спричинила бурхливий сплеск ентузіазму. Перші спроби знайти загальний алгоритм для вирішення задачі автоматизації дедуктивних побудов простежуються в роботах Лейбніца, Пеано, Гільберта. Центральне місце серед сучасних методів автоматичного доведення теорем займає ***метод резолюцій,*** запропонований Дж. Робінсоном у 1965 р. Використання цього методу виявилося ефективнішим, ніж класичне виведення на основі modus ponens.

Оскільки числення предикатів є алгоритмічне нерозв'язним, метод резолюцій дозволяє встановити, що деяка формула є теоремою, якщо вона насправді є теоремою. Для формул, що не є теоремами, метод резолюцій може працювати нескінченно довго. Саме на цій основі з'явилися ідеї, на яких базуються мови логічного програмування. Одним з фундаментальних проривів того часу було відкриття Дж. Робінсоном принципу аналізу та його застосування до автоматичного доведення теорем. Аналіз – це правило висновків, як одне твердження може випливати з іншого. Використовуючи аналітичний принцип, ми можемо на основі аксіом довести теореми стандартним шляхом з наших аксіом. Ми тільки маємо вирішити, до яких тверджень їх застосувати, і відповідний висновок дістанемо автоматично.

Аналіз розроблено і для роботи з формулами у фразовій формі. Маючи ці фрази, що пов'язані відповідним чином, він створить нову фразу, що буде наслідком перших двох. Основна ідея – це те, що якщо одна й та сама часткова формула зустрічається у правій частині однієї структури і в лівій частині іншої, то фраза отримується з'єднанням двох фраз з вилученням дубльованих формул, які з них випливають.

Метод резолюцій базується на правилі резолюцій.

Метод резолюцій – метод доведення того, що формула G є логічним наслідком формул F_1, F_2,…, F_n. Задача зводиться до перевірки виконуаності. Тобто формула G є логічним наслідком формул F_1, F_2,…, F_n тоді і тільки тоді коли множина формул $\{F_1, F_2,\ldots, F_n, \sim G\}$ невиконувана. Таким чином, можна відмітити, що метод резолюцій володіє такими особливостями:

– встановлює невиконуваність;

– оперує з дизюнктами.

Нагадаємо, що літералом ми називаємо атомарну формулу або її заперечення, дизюнктом − дизюнкцію літералів. Диз’юнкт може складатися і із одного літералу. В принципі можна дивитися , як на множину літералів, тобто не розрізняємо дизюнкти, які отримуються один з одного шляхом комутативності, асоціативності.

$X \vee \sim Y \vee X \cong X \vee \sim Y$

Крім того неохідності ввести поняття порожнього дизюнкта $\in$ (немістить літералів). Порожній дизюнкт − завжди хибний при будь-якій інтепритації,

тобто

$F \wedge \in \cong \in$, $F \vee \in \cong F$

Літерали L та $\sim L$ є протилежними. Кажуть, що дві фрази містять ***контрарну пару*** атомарних формул (або просто контрарну пару), якщо одна з них включає деяку атомарну формулу без заперечення, а інша – ту саму формулу під знаком заперечення.

На основі цього правило резолюцій можна подати, як:

Дві фрази, що містять контрарну пару, можуть бути резольвовані одна з іншою, і результатом резолюції (резольвентою) є диз'юнкція літералів, які залишаються в обох фразах після викреслення контрарної пари. Тобто із диз'юнктів $X \vee F$ та $\sim X \vee G$ виводимо диз'юнкт $F \vee G$.

Слід відмітити, що правило резолюцій завжди зберігає істинність. Це означає, що якщо $\phi(X \vee F)=1$ та $\phi(\sim X \vee G)=1$ для деякої інтерпретації ϕ, то і $\phi(F \vee G)=1$.

Правило резолюцій необов'язково застосовувати до крайніх лівих диз'юнктів. Оскільки, ми не розрізняємо диз'юнкти, що відрізняються порядком запису літералів.

Нехай S – множина диз'юнктів. Виводом з S називається послідовність диз'юнктів $D_1, D_2, \ldots, D_n$ така, що кожний диз'юнкт цієї послідовності належить S або слідує їх попередніх за правилом резолюцій. Диз'юнкт D виводиться з S, якщо існує вивід S, останнім диз'юнктом якого є D.

Наприклад, S=$\{\sim X \vee Y \vee Z, \sim Y \vee U, X\}$ D_1=…., $D_5 = Z \vee U$. Диз'юнкт $Z \vee U$ виводиться з S.

Застосування методу резолюцій базується на наступному твердженні: Множина диз'юнктів логіки висловлювань S невиконувана тоді і тільки тоді, коли з виводимо пустий диз'юнкт.

Для доведення того, що формула G є логічним наслідком формул $F_1,\ldots, F_n$ метод резолюцій застосовується наступним чином. Спочатку складається множина формул $T = \{F_1,\ldots,F_n, \sim G\}$. Потім кожна з цих формул приводиться до кон'юктивної нормальної форми і в отриманих формулах викреслюються знаки кон'юнкції. Отримується множина диз'юнктів S, та здійснюється вивід пустого диз'юнкту з S. Якщо пустий диз'юнкт виводиться з S, то формула G є логічним наслідком формул $F_1,\ldots,F_n$. Якщо вивести $\in$ з S неможливо, то формула G не є логічним наслідком формул $F_1,\ldots,F_n$.

Нехай з S виводимо $\in$. Припустимо протилежне − множина S виконувана, тобто існує інтепритація ϕ, при якій всі дизюнкти істинні. Виводимість $\in$ із S означає, що існує послідовність $D_1, D_2, \ldots, D_n = \in$, кожний дизюнкт якої належить S або отримується з попередніх за методом резолюцій. Якщо D_j належить та за припущенням $\phi(D_j)=1$, якщо отримується з попередніх то також $\phi(D_j)=1$ (правило зберігає істинність) при $j = n$ $\phi(\in)=1$ − що протиречить.

Наприклад, довести, що формула G=Z, є логічним наслідком формул (приклад 1)

$F_1 = \sim X \vee Y \rightarrow X \wedge Z$,

$F_2 = \sim Y \rightarrow Z$

Оскільки в логіці предикатів, в диз'юнктах можуть бути змінні, то попереднє правило резолюцій вже не підходить. Дійсно, множина диз'юнктів $S=\{P(x), \sim P(x)\}$ не виконувана (передбачається що змінна зв'язана квантором узагальнення). В той же час за попереднім правило отримати пустий диз'юнкт із S неможливо. Оскільки диз'юнкт $P(x)$ можна розуміти так, що $P(x)$ істинно для будь-якого x, то $P(x)$ буде істинним і для $x = a$. Отже зробивши підстановку, отримаємо множину диз'юнктів $S'=\{P(a), \sim P(a)\}$. Множини S та S' одночасно не виконувані, але з S' можна вивести пустий диз'юнкт. Таким чином, в логіці предикатів правило резолюцій необхідно доповнити можливістю робити підстановки.

Підстановкою називається множина рівностей $\sigma = \{x_1 = t_1, \ x_2 = t_2,\ldots, \ x_n = t_n\}$, де x_1, x_1, x_n – різні змінні, $t_1, t_2,\ldots, t_n$ – терми, причому терм t_i не містить змінної x_i.

Якщо $\sigma=\{x_1=t_1, x_2=t_2,\ldots, x_n=t_n\}$ – підстановка, а F – диз'юнкт, то через $\sigma(F)$ будемо позначати диз'юнкт, що отримується з M, одночасною заміною x_1 на t_1, x_2 на $t_2,\ldots, x_n$ на t_n. Причому терми можуть підставлятися лише замість змінних, а не констант.

Підстановка σ називається ***уніфікатором*** для множини виразів $\{F_1, \ldots, F_n\}$, якщо $\sigma(F_1)= \sigma(F_2) = \ldots = \sigma(F_n)$. *Якщо множина уніфікована, то існує як правило не один уніфікатор множини, а декілька, серед яких виділяється найбільш загальний уніфікатор.*

Уніфікатор σ для множини виразів $\{F_1,\ldots, F_n\}$ називається ***найбільш загальним уніфікатором,*** якщо для будь-якого уніфікатора θ цієї множини існує така підстановка λ, що θ являтиме собою композицію підстановок σ і λ ($\theta= \sigma\circ\lambda$).

Композицією підстановок $\sigma = \{x_1 = t_1, \ x_2 = t_2,\ldots, \ x_n = t_n\}$ і $\lambda = \{y_1 = u_1, y_2 = u_2,\ldots, \ y_m = u_m\}$ називається підстановка $\theta =\{x_1 = \lambda(t_1), \ x_2 = \lambda(t_2),\ldots, \ x_n = \lambda(t_n), y_1=u_1, y_2=u_2,\ldots, y_m=u_m\}$, з якої викреслені всі рівності виду $x_i=x_i$ та $y_i=u_i, y_i \in \{x_1, \ldots, x_n\}$.

Наприклад, (Приклад 2)

$\sigma = \{x = f(y), z = y, u = g(d)\}$

$\lambda = \{x = c, y = z\}$

$\lambda\circ\sigma = \{ x = f(y), \mathbf{z = z}, u = g(d), \mathbf{x = c}, y = z \} = \{ x = f(y), u = g(d), y = z \}$

Тепер можна сформулювати загальне правило резолюцій.

Дві фрази, що містять контрарну пару, можуть бути резольвовані одна з іншою, якщо літерали, що входять до контрарної пари, можуть бути уніфіковані, тобто якщо для них існує найбільш загальний уніфікатор σ. Результатом резолюції є **диз'юнкція літералів,** які залишаються в обох фразах після викреслення контрарної пари, причому до цих літералів повинен бути застосований уніфікатор σ.

В логіці предикатів також вводиться правило зклейки. Із дизюнкту $\Diamond P(t_1,\ldots, \ t_n) \vee \ldots \vee \ \Diamond P(s_1,\ldots, \ s_n) \vee F$ виводимо дизюнкт $\sigma(\Diamond P(t_1,\ldots, \ t_n)) \vee \sigma(F)$ (склейка першого дизюнкта).

◊ − знак заперечення або його відсутність.

Наприклад,

$\sim P(x, y) \vee \sim P(y, x) \vee \sim P(a, a) \vee Q(x, y, v) \rightarrow \sim P(a, a) \vee Q(a, a, v)$

Тоді виводом із множини дизюнктів S називається послідовність диз'юнктів D_1, D_2, …,D_n така, що кожний диз'юнкт D_j цієї послідовності належить S або виводиться з попередніх за правилом резолюцій або з попереднього за правилом зклейки. Відповідно D виводиться із S, якщо існує вивід із S послідніим дизюнктом якого є D.

Наприклад, (приклад 5)

$S=\{\sim B(x) \vee \sim C(x) \vee T(f(x)), C(y) \vee T(f(z)), B(a)\}$

$D_1=\sim B(x) \vee \sim C(x) \vee T(f(x))$

$D_2= C(y) \vee T(f(z))$

$D_3=\sim B(x) \vee \sim T(f(x)) \vee T(f(z))$ резолюція D_1 та D_2 з підстановкою y=x

$D_4=\sim B(x) \vee \sim T(f(x))$ із D_3 за правилом зклейки

$D_5= B(a)$

$D_6=\sim T(f(a))$ резолюція D_4 та D_5 з підстановкою $x=a$.

Існують формальні алгоритми, які дозволяють отримувати найбільш загальний уніфікатор. Але у більшості випадків достатньо деяких неформальних простих правил.

1. Будь-який терм зіставляється сам з собою.

Наприклад, дві фрази $P(\alpha) \vee \sim Q(b, c)$ та $Q(b, c) \vee \sim R(b, c)$, є резольвовані, і резольвента записується як $P(\alpha) \vee \sim R(b, c)$.

2. Різні константи не зіставляються одна з одною.

Тому фрази $P(\alpha) \vee \sim Q(b, c)$ та $Q(c, c) \vee \sim R(b, c)$ не є резольвованими.

3. Змінна може бути замінена константою або іншим термом.

Так, фрази $P(\alpha) \vee \sim Q(a, b)$ та $Q(x, y) \vee \sim R(x, y)$, резольвується, при цьому змінна x зіставляється з константою a, змінна y – з константою b. У резольвенті $P(\alpha) \vee \sim R(a, b)$ змінні замінені на константи, з якими вони зіставлялися.

Якщо в процесі виведення утворилися фрази $P(a)$ та $\sim P(a)$, їх резолюцією **є пустий** диз'юнкт (тотожна хибність, позначається ∈). **Саме виведення пустого диз'юнкта і означає суперечливість теорії.**

Наприклад,

$F_1 = \exists x[P(x) \wedge \forall y(C(y) \rightarrow Z(x, y))]$

$F_2 = \forall x \forall y(P(x) \wedge L(y) \rightarrow \sim Z(x, y))$

$G = \exists x(C(x) \rightarrow \sim L(x))$

Тепер можна сформулювати алгоритм **автоматичного доведення теорем на основі методу резолюцій.** Нагадаємо, що йдеться про перевірку істинності будь-якого твердження в рамках існуючої системи знань.

Застосовується метод доведення "від супротивного". При перевірці утворюється деяка пробна теорія, що утворюється **шляхом додавання заперечення нового твердження до вже 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.

Фундаментальною властивістю методу резолюцій є його ***повнота.*** Як уже

зазначалося, логіка предикатів першого порядку не є розв'язною. Тому для множини диз'юнктів, яка не є суперечливою, процедура, що ґрунтується на методі резолюцій, може працювати нескінченно довго.

Приклад. Нехай дано такий набір фраз:

– вовк їсть м 'ясо,

– тварини, які їдять м'ясо, є хижаками.

Потрібно довести, що:

– вовк є хижаком.

Введемо предикати $G(x, y)$, який означає *"x їсть y"* і $Q(x)$, який означає *"x є хижаком"*. Далі введемо предметні константи: W – вовк; M – м'ясо.

Тоді маємо базу знань:

$G(W, M)$

$\sim G(x, M) \vee Q(x)$.

Потрібно довести формулу $Q(W)$.

Крок 1. Утворюємо пробну теорію та додаємо до неї заперечення:

$G(W, M)$ (4.1)

$\sim G(x, M) \vee Q(x)$ (4.2)

$\sim Q(W)$ (4.3)

Крок 2. Застосовуємо резолюцію до (4.2) та (4.3); при цьому константа W зіставляється зі змінною x:

$G(W, M)$ (4.4)

$\sim G(x, M) \vee Q(x)$ (4.5)

$\sim Q(W)$ (4.6)

$\sim G(W, M)$ (4.7)

Крок 3. Внаслідок резолюції (4.4) і (4.7) утворюється пуста фраза. Це означає суперечливість пробної теорії та істинність твердження, яке перевіряється.

5 Мережеве представлення знань. Семантичні мережі

Раніше розглянути було використання числення предикатів для представлення знань. Числення предикатів можна комбінувати з такими порівняно ефективними механізмами висновку, як резолюція. Таким чином, логічні формалізми представлення знань охоплюють проблему міркувань достатньо витончено.

Недолік логічного формалізму – його неструктурованість: наприклад, для збору всієї інформації по одному об'єкту (конкретизації) приходиться переглянути всю множину логічних формул якоїсь бази даних (БД). Графічні уявлення, служать глобалізації і структуризації інформації. Граф збирає навколо одного вузла всю інформацію по деякому об'єкту. Графічні уявлення, такі, як концептуальні графи і семантичні мережі, дозволяють візуалізувати модель світу, якому належить вирішувана проблема. Такий тип представлення більше відображає об’єкти предметної області, чим правила, що відносяться до самого цьому світу.

Концептуальний граф представляє логічну формулу. Імена і аргументи предикатів представлені в ньому відповідно двома типами вузлів. Дуги графа сполучають імена предикатів з їх аргументами. Ці графи близькі до моделей, використовуваних фахівцями з БД. Одна з проблем при створенні БД – облік структури області експертизи. Візьмемо, наприклад, структури, що існують в надрах якогось підприємства. Різні відділи, секції, бюро і суб'єкти, їх що заповнюють, утворюють ієрархічну структуру. Її можна представити набором концептуальних графів.

Семантичні мережі представляють складніші структури. Така мережа складається з безлічі концептуальних графів, що представляють логічні формули. Вона дозволяє візуалізувати множину відносин між концептуальними графами, її складовими. Нею описується також універсум, в який занурені концептуальні графи (що є глобальною областю експертизи).

Концептуальні графи і семантичні мережі складають графічну версію числення предикатів.

5.1 Концептуальні графи

Логіка предикатів – це мова, яку можна інтерпретувати в термінах області міркувань (експертизи): логічні формули представляють фрази метамови. Аргументи предикатів і логічних Функцій істотно використовуються для представлення атрибутів, подій і станів. Імена предикатів указують спосіб пов'язання цих понять. Зокрема, ці імена представляють правила з'єднання, правила граматики і процедури.

Концептуальні графи містять прямокутники для представлення аргументів і круги для імен предикатів. Круг з'єднується стрілкою з прямокутником, якщо вони представляють відповідно ім'я і аргумент одного і того ж предиката.

Предикати можуть мати декілька аргументів. Отже, круги можуть мати декілька вхідних і/або вихідних стрілок. Більшість предикатів, використовуваних для представлення знань, володіють двома аргументами (бінарні предикати). Тоді круги сполучені з прямокутниками двома стрілками: що входить і виходить. Проілюструємо ці поняття на прикладі.

Побудуємо концептуальний граф предиката Посилка(„Павло”, „Марія”, „Книжка”), що формалізує фразу «Павло надсилає книгу Марії» (рис.5.1, а). На рис.5.1, б приведений концептуальний граф логічного представлення тієї ж фрази бінарними предикатами, тобто граф, відповідний формулі

Відправник(„Посилка”, „Павло”)∧

Одержувач(„Посилка”, „Марія”) ∧

Об'єкт(„Посилка”, „Книжка”) ∧

Є(„Посилка”, посилки).

При графічному представленні бінарних предикатів круги іноді опускають, а імена предикатів указують на стрілках. Загальне правило перетворення m-арного предиката (m≥2) вигляду:

Предикатне_ім'я(значення_1, значение_2,...,.., значение_m) в добуток m бінарних предикатів

∧Функція_j(предикатне_ім’я, значення_ j).

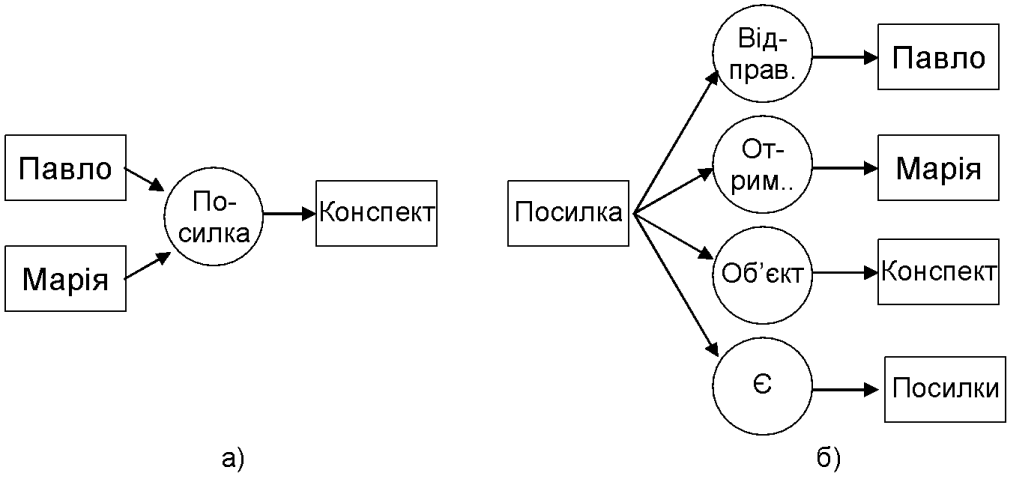


Рисунок 5.1 – Концептуальний граф (а), концептуальний граф логічного представлення бінарними предикатами

Пов'яжемо з кожним бінарним предикатом концептуальний граф, що відповідає тріаді „об’єкт – атрибут – значення” (рис.5.2).

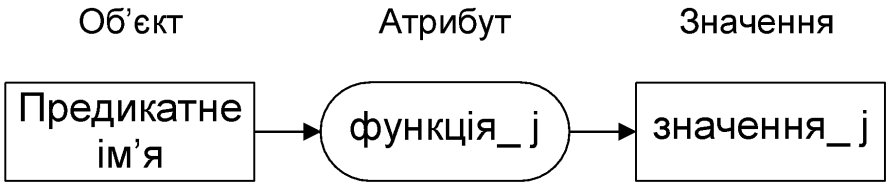


Рисунок 5.2 – Концептуальний граф бінарного предикату

Графічне представлення *m*-арних предикатів засноване на наступній угоді. Стрілка, що відповідає m-му аргументу, направлена до прямокутника, що представляє цей аргумент. Інші стрілки направлена від прямокутників до круга, що представляє ім'я предиката. Стрілки, як правило, нумерують від 1 до m, щоб явно показати відповідність між стрілкою і аргументом (рис.5.1).

Взагалі ім'я бінарного предиката представляє якусь функцію. Вузол концептуального графа, що вказує на це ім'я, називається зв'язуючим вузлом, бо він зв'язує два концепти, представлені двома аргументами бінарного предиката. Функція, представлена цим зв'язуючим вузлом, носить по тій же причині назва концептуального відношення. Терміни «ім'я бінарного предиката» і «аргумент», що представляє відповідно функцію і концепт, часто замінюють на «зв'язуючий вузол» і «вузол-концепт» в графічному уявленні.

5.2 Семантичні мережі

Кожен концептуальний граф представляє одну логічну формулу. Семантична набагато складніша. Вона представляє не тільки набір (з'єднання) формул, але також описує їх взаємозв'язки і відображення їх в контекст області міркувань. Семантичні мережі утворюються з концептуальних графів за правилами з'єднання, які будуть введені за допомогою прикладів.

Правила кон'юнкції і спрощення

Розглянемо набір з трьох фраз:

Фраза 1: Павло пише конспект,

Фраза 2: Павло посилає цей конспект Марії,

Фраза 3: Марія читає конспект (який їй передав Павло).

Кожну з цих фраз можна представити або формулою числення предикатів, або концептуальним графом. Перший етап побудови семантичної мережі заснований на використанні двох наступних формальних правил отримання концептуального графа g з двох графів g1 і g2.

Правило кон'юнкції. Якщо вузол-концепт c1 в g1 ідентичний вузлу-концепту c2 в g2, то g утворюється шляхом видаленням c2 і з'єднанням із c1, всіх зв'язуючих вузлів, які були пов'язані з c2 в g2.

Правило спрощення. Якщо концептуальний граф g містить два ідентичні (сполучених з одними і тими ж вузлами-концептами) зв'язуючі вузли, то можна видалити один з них разом з пов'язаними з ним стрілками.

5.2.1 Представлення контексту

Сам по собі концептуальний граф несе небагато інформації, тоді як включення в семантичну мережу дозволяє пов'язати його концепти і функціональні зв'язки з областю міркувань (експертизи). Наприклад, „Конспект з ШІ” – конкретизація слова „конспект”, яке представляє абстрактне поняття (або, інакше, тип). Відношення приналежності до типу представляється зв'язуючим вузлом (ім'ям предиката) *Конкр* (від слова «конкретизація»).

Окрім факту, що „Конспект з ШІ” відноситься до типу «конспект», ми хочемо показати, що вона належить деякій «множині конспектів» (кожен

студент володіє конспектами з різних дисциплін). Відношення приналежності множині представлене зв'язуючим вузлом (ім'ям предиката) *Елем* (від слова «елемент»). Відзначимо, що «тип» уживається в абстрактному сенсі (абстрактний тип). Твердження про типи – аналітичні, наприклад: середній вік студентів університету двадцять років. Навпаки, властивості множин – синтетичні: число студентів університету рівне шести тисячам.

Використовуємо зв'язуючий вузол, що представляє ім'я предиката *Підмн* для представлення відношення між двома множинами, з яких перше — підмножина другого. Як *Елем* оголошує про приналежність елементу множині, *Підмн* означає включення однієї множини в інше. *Елем* і *Підмн* відображають приналежність множині.

Зв'язуючий вузол *Це* використовується для представлення відношення між першим типом і більш загальним другим. Нагадаємо, що предикат *Конкр* колись використовувався нами для представлення приналежності індивіда типу. *Це* і *Конкр* відображають приналежність типу.

Концепти в семантичних мережах володіють різними властивостями, що діляться на аналітичні (властивості «типу») і синтетичні (властивості «множини»). Важливо враховувати цю відмінність при побудові мереж, щоб уникнути некоректних висновків.

5.2.2 Представлення «сукупність -посилання»

Існує графічне представлення з вузлами, що представляють типи, що називаються «вузлами-сукупностями», або «родовими вузлами», і вузлами, що представляють конкретизації типів і називаються «вузлами-індивідами» або «вузлами-посиланнями».

Семантична мережа утворюється послідовностями з трьох вузлів, сполучених так: за вузлом-посиланням (наприклад, що представляє конкретизацію „Конспект з ШІ”) слідує зв'язуючий вузол Конкр, що супроводжується вузлом-сукупністю (наприклад, що представляє тип «конспект»).

Ці три вузли можна згрупувати і представити одним «вузлом-прямокутником», що складається з двох полів: з «поля сукупності», що містить деякий тип, і наступного за цим полем «поля посилання», наприклад, з конкретизацією типу з першого поля. Поле посилання (праворуч від двокрапки) представляє конкретизацію „з ШІ”. Крім конкретизації його вмістом може бути змінна. Поле сукупності (зліва від двокрапки) представляє тип. Представлення [конспект: „Конспект з ШІ”) (або, коротше [конспект: з ШІ]) означає цілком певний об'єкт типу «конспект». Представлення [конспект: x] означає просто якийсь об'єкт типу «конспект». Відзначимо, що наявність змінної в правому полі взагалі необов'язково. Наприклад, можна замінити представлення [конспект: x] еквівалентним йому представленням [конспект].

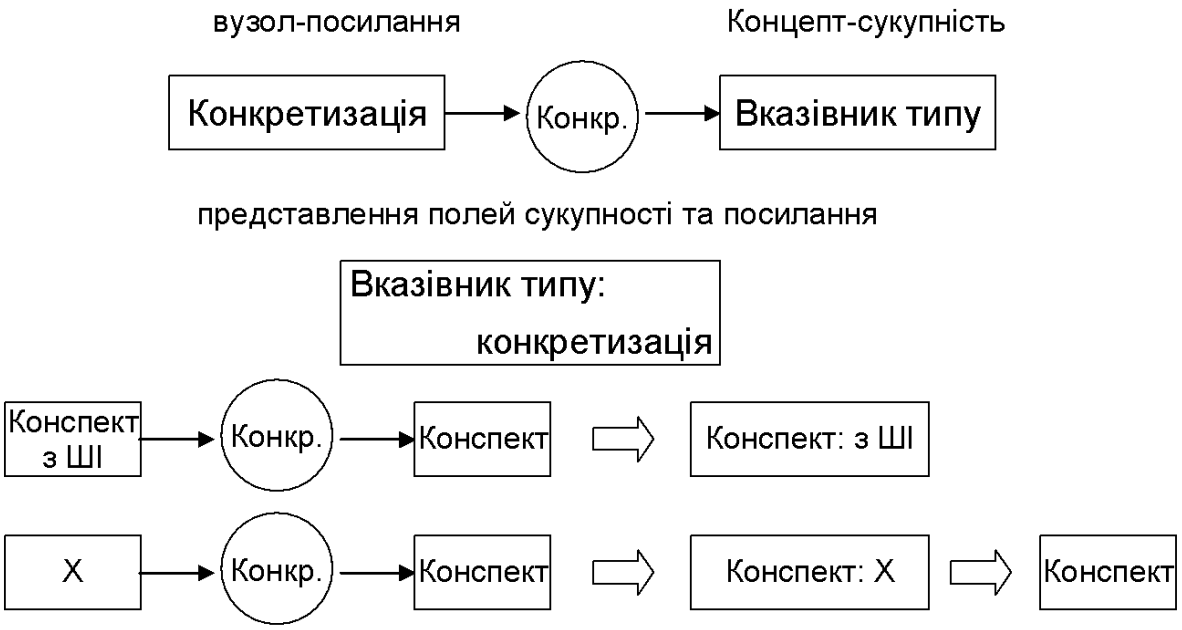


Рисунок 5.3 – Поля посилання та сукупності

Наприклад, фрази:

Павло надсилає (цілком визначений) конспект Марії

Павло посилає (якийсь) конспект Марія

представляються двома графами, які можна виділити на рис.5.4: першої фрази – графом без елементів з (2), друга фраза – графом без елементів з (1).

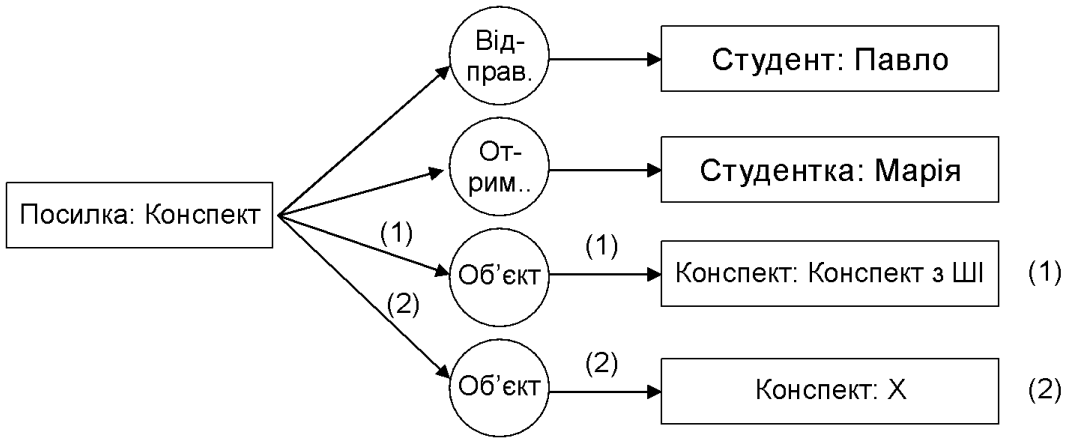


Рисунок 5.4 – Приклад концептуального графу

5.2.3 Канонічні графи

У концептуальному графі є вузли-концепти і вузли-зв'язки (що зв'язують вузли). Кожна вхідна в зв'язуючий вузол стрілка (або що виходить з нього) сполучена з вузлом-концептом. Проте деякі комбінації вузлів безглузді. Наприклад, можна зобразити графом фразу «видалите останнє слово наступного рядка» (якогось тексту). З позицій граматики ця фраза синтаксично і семантично коректна. Навпаки, фраза «видалити останній рядок наступного

слова» синтаксично коректна, але абсурдна (семантично некоректна). Щоб виключити графи нереальних (неможливих) ситуацій області міркувань, визначено канонічні (семантично коректні) графи дозволених комбінацій слів. Кожен складає уявлення про світ, що виражається канонічними графами, виходячи з особистого досвіду.

Таким чином, можливий шлях побудови таких графів наступний: мозок виробляє систему концептів, базуючись на відчуттях, що поступають, і так розставляє сформовані концепти, щоб вони відображали реальну ситуацію. Отже, можна вважати, що мозок будує канонічні графи, що представляють, взагалі кажучи, очевидності (іншими словами, тавтології числення предикатів). Нові канонічні графи будуються з тих, що є за певними правилами побудови. Ці правила формують граматику, для концептуальних графів точно так, як і це робиться за допомогою правил переписування Хомського у разі граматик, для синтаксичних структур природних мов.

5.2.4 Правила побудови

Існує чотири правила побудови для конструювання нового канонічного графа g з наявних графів g1, і g2 (які можуть співпадати) :

Правило кон'юнкції визначено раніше.

Правило спрощення: визначено раніше.

Правило копіювання: граф g є копія графа g1.

Правило обмеження: для будь-якого концепту з концептуального графа g тип (c) можна замінити якимось підтипом. Якщо c – концепт сукупності, то його посилання можна замінити індивідуальним покажчиком (якоюсь конкретизацією). Такі заміни дозволені лише тоді, коли посилання для c відповідає типу (c).

Наприклад у фразі „Павло посилає комусь свій конспект”, для „комусь” (позначення на графі X) можна було б застосувати правило обмеження, замінюючи позначену X сукупність меншою сукупністю «студентка». Проте раніше треба переконатися у властивості «Марія – студентка».

5.2.5 Успадковані властивості

Форма різних концептів, типів і множин в багатьох додатках ієрархічна. Класифікація тварин по родах, видах, сімействах, загонах і т.д. – класичний приклад ієрархії. Інший приклад – множина слів в словнику. Книги, транспортні засоби і споруди утворюють ієрархічно впорядковані множини. Можна ієрархічно класифікувати і абстрактні концепти дій, станів, властивостей, переконань. Використовувані в буденних міркуваннях класифікації, як правило, складніші за біологічні: індивід належить, взагалі кажучи, не одному типу і не одній множині. Проте в більшості випадків для ШІ використовуються порівняно прості ієрархії.

Спочатку проілюструємо ієрархію типів на прикладі, а потім дамо формальне визначення. Якщо відомо, що Павло – студент університету, то можна зробити висновок про наявність у нього атестату про середню освіту і про те, що він навчається в університеті. Інформація «Павло – студент

університету» надала нам цілий ланцюжок відомостей про Павла. Успадкованими називаються такі властивості, які можна вивести так само. Ясно, що деякі характеристики студентів університету можна описати, виходячи з приналежності до студентства; деякі характеристики студентів виводяться з того факту, що вона міститься в сукупності всіх людей. Міркування ефективності спонукають нас не пов'язувати з Павлом всі загальні властивості студентів університету, взагалі студентів і, нарешті, всіх людей. Вельми загальна властивість, як, наприклад, «У Павла дві руки і дві ноги», зіставляється вузлу, що характеризує всіх людей. Це дає можливість не проголошувати наявність вказаної властивості окремо для кожного індивіда.

5.2.6 Решітки типів; ієрархії типів

Ієрархія типів визначає частковий порядок на множині міток (ярликів) типів. Позначимо його символом ≤. Нехай t1 і t2 – мітки типів. Якщо t1≤ t2, то t1 – підтип типу t2, а t2 – надтип типу t1. Типи «викладач» і «студент» мають багато загальних надтипів: «працівник розумової праці», «людина», «вищий ссавець». Типи «професор» і «співробітник університету» мають загальний підтип «професор університету».

Для перетворення ієрархічної форми решітку треба ввести дві особливі мітки типів відповідно у найвищій і найнижчій точкак ієрархії – універсальний тип U (надтип всіх типів) і абсурдний тип A (підтип всіх типів). Ієрархія типів дасть решітку типів зі всіма властивостями решітки:

– будь-яка пара міток t1 і t2 має найменший загальний надтип;

– будь-яка пара міток t1 і t2 має найбільший загальний підтип;

– для будь-якого типу t виконується співвідношення A ≤ t ≤ U.

Дане відношення порядку представляється функціональним відношенням Це. Конкр означає приналежність індивіда типу.

5.2.7 Решітки множин і решітки типів

Можна також ввести відношення порядку для множин (підмножина множини) за допомогою предиката Підмн або зв'язуючого вузла з тим же ім'ям. При одноелементній підмножині Підмн замінюється на Елем. Ієрархія типів дозволяє впорядковано класифікувати властивості типів, а ієрархія множин – властивості множин. Введемо на концептуальних графах декілька функцій і операторів, що діють над множиною типів. Перш за все – функція тип. Вона відображає множину концептів на множині Т, елементи якого називаються мітками (ярликами) типу. Концепти с1 і с2 мають один тип, якщо тип(с1) = тип(с2). Денотатом типу t називається множина всіх тих сутностей, яка є конкретизаціями якогось концепту типу t. Оператор, що зіставляє типу t його денотат, позначається через Денот. Тип „студента університету” є підтип студента. Отже, денотат Денот(студент_університету) є підмножиною Денот(студент).

Кожній конкретизації (наприклад, Павло) зіставляється два види вузлів: вузол опису типу конкретизації і вузол опису множини, якій належить ця конкретизація. Таким чином, маємо три види вузлів: вузол, що представляє

індивіда (конкретизацію), який через проміжні зв'язуючі вузли сполучений з вузлами типу (Конкр) і множини (Елем). Природно ввести четвертий вид: {x|тип} для довільного індивіда (x) певного типа (тип).

Студент *x* (про якого більше нічого не відомо) буде позначений {*x*|студ}. Цей вузол сполучений з вузлом, що представляє множину студентів, через зв'язуючий вузол *Елем*. З іншого боку, він сполучений з вузлом, що представляє тип професора, через зв'язуючий вузол *Абст*. Оператор *Абст* визначає тип (абстрактний концепт) по опису якого-небудь елементу цього типа. Нарешті, можна визначити оператор «прототип_для», який можна рахувати приблизно зворотним оператору *Абст*. Оператор «прототип_для» (*Протот*) представляється зв'язуючим вузлом, який сполучає тип з описом цього типа, що задається «прототипом». Останній дозволяє стисло описати світ, розділений на типи об'єктів. Опис, що задається прототипом, можна розглядати як опис деякого типового представника множини. Отже, прототип – різновид міфічної константи. Вузол прототип відрізняється від вузла {*x*|*тип*}, який представляє множину загальних властивостей об'єктів одного типа або однієї множини. У цьому сенсі оператори *Абст* і *Протот* не є взаємно зворотними.

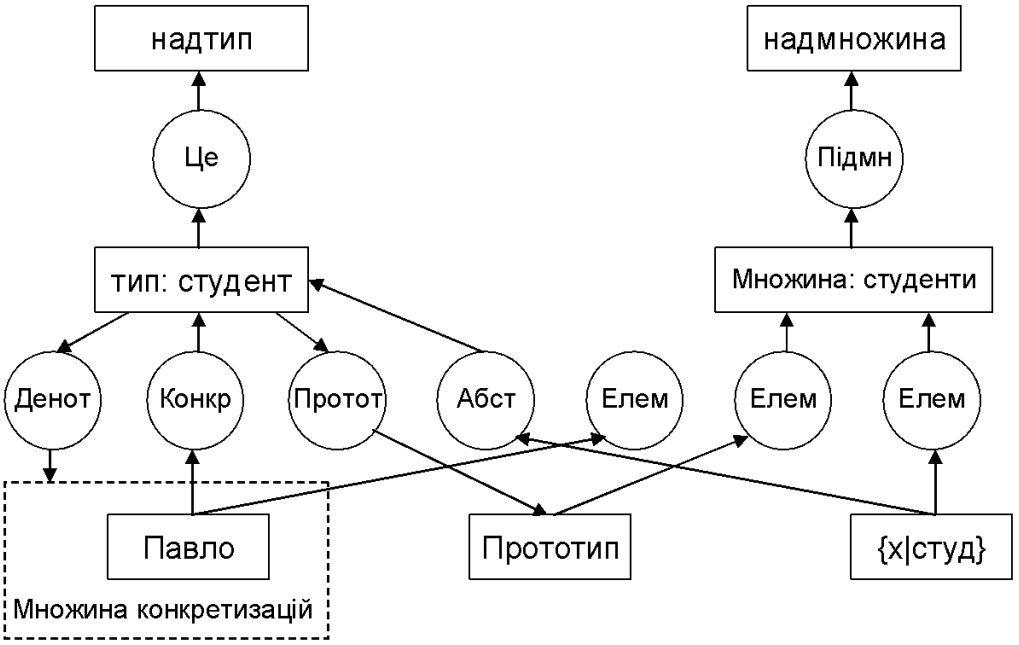


Рисунок 5.5 – Зв’язуючі вузли і вузли-концепти

Доведено, що решітки типів не ізоморфні решіткам множин. З абстрактним концептом «професора університету» пов'язана властивість володіння докторським ступенем. З іншого боку, цілком імовірно, що не кожен індивід з множини «професорів університету» має докторський ступінь. Отже, відповідні ієрархії типів і множин, не ізоморфні.

Можна визначити мітки типів, використовуючи так званий арістотелів підхід – через рід (genus) та відмінність (differentia). Тип визначається початковим типом рід і висловом, що називається відмінність і що відокремлює новий тип від початкового. Наприклад, Посилка – «подія (рід), яка відбувається, коли дві людини, відправник і одержувач, забезпечують переміщення предмету

поштою (відмінність)». Формально:

тип *Посилка*(*x*) є *Конкр* (*x*, *подія*) ⋀ (*рід*)

Відправник(*X*, Людина_1) ⋀ *Одержувач*(*X*, Людина_2) ⋀ *Об'єкт*(*X*, Лист) ⋀ *Спосіб* (*X*, Пошта) ⋀ *Конкр*(Людина_1, *Людина*) ⋀ *Конкр*(Людина_2, *Людина*) ⋀ ~*Рівно*(Людина_1, Людина__2) (*відмінність*)

Визначити який-небудь тип можна ще шляхом демонстрації декількох прикладів індивідів «якогось типа» і стверджуючи, що все, чим схожі індивіди, відноситься до обговорюваного типа. Прототип – це конкретизація типу або, інакше, типова конкретизація. Метод прототипів описує швидше типового індивіда якогось класу, чим довільного представника цього класу з набором характерних властивостей.

Прототип указує властивості, істинні в типовому, але не обов'язково у кожному окремому випадку. Він дозволяє гнучко визначати клас об'єктів. Прототип містить фіксовану частину (відповідну тому загальному, що є у всіх конкретизаціях цього класу) і змінну частину (обов'язкову чи ні, свою для кожної конкретизації).

5.2.8 Схеми і схематичні кластери

Концептуальний граф служить для представлення знань. Його можна також використовувати для міркувань і обчислень. Для цього введемо поняття схеми, пов'язавши його з правдоподібними міркуваннями та міркуваннями, що мають сенс. Схему можна визначити, вводячи послідовно обмеження на концептуальний граф.

– У довільному концептуальному графі не накладаються ніякі обмеження на розстановку вузлів.

– Канонічні графи мають семантичні обмеження, представляючи щось «семантично коректне» або «зрозуміле».

– Схеми включають «специфічні знання» про область міркувань (експертизи), представляючи все правдоподібне.

Поняття схеми передбачає не тільки визначення, але і спосіб застосування. Для ілюстрації цього аспекту спочатку порівняємо визначення схеми і типу. Формальне визначення схеми дамо потім. Кожен концепт має рівно одне визначення його типа, що дає необхідні і достатні умови приналежності конкретизації певному типу.

Кожному типу можна зіставити декілька схем, кожна з яких представлятиме один із способів застосування концепту даного типа. Це приводить до поняття «множини схем» як можливого джерела інформації, еквівалентної визначенню типу. А сам концепт можна визначити «множиною його можливих застосувань».

Для формалізації опису концепту можна ввести поняття кластера або набору схем. Кожна з них указує спосіб застосування даного концепту. Набір всіх можливостей застосування типу називається його схематичним кластером.

Схеми показують типові способи використання концептів. Вони не описують типових конкретизації для цих концептів. Навпаки, прототип – це типова конкретизація. Поняття схеми і схематичного кластера дозволяють більш

формалізований визначити прототип.

5.3 Міркувань, що використовують семантичні мережі

Одна з ключових проблем організації пам'яті – ефективне управління даними і запитами. При ієрархічній організації індивідуальні властивості пов'язані з іменами індивідів. Наприклад, індивідуальні властивості людини (вік, адреса, сімейний стан і т. д.) можна пов’язати з ім'ям Степан. Крім того, спадкова властивість пов'язує Степана з (абстрактним) концептом професор університету. Стрілки, що виходять з вузла Степан можна представити набором (кон'юнкцією) бінарних предикатів (рис. 5.6):

Вік(*Степан*, 60) ∧
Адреса (*Степан*, *Південний_бульвар_3*) ∧
Конкр(*Степан*, *проф_унів*).

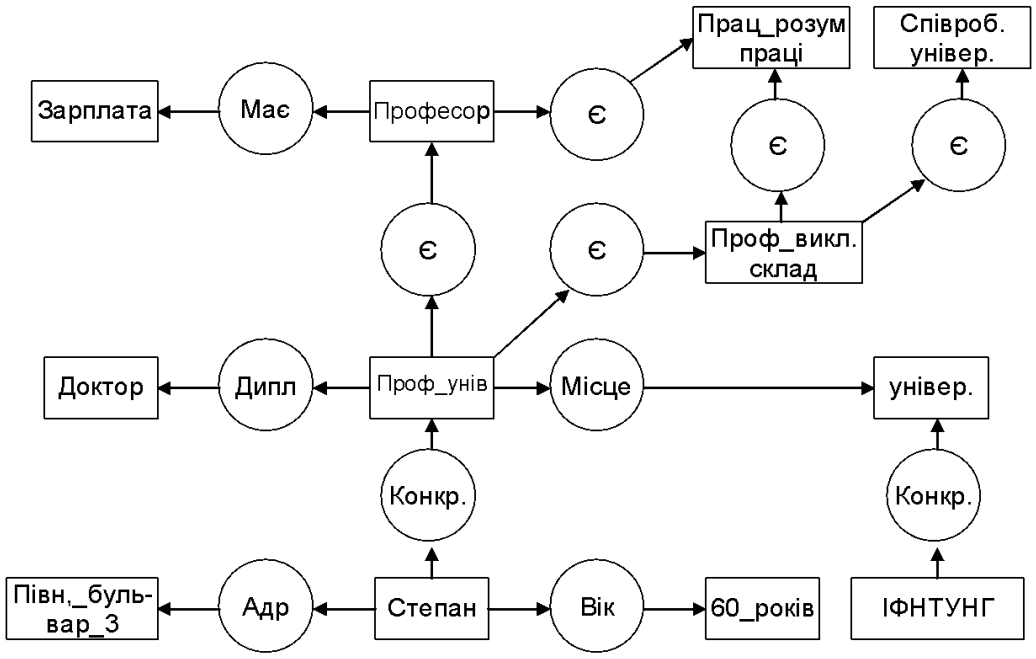


Рисунок 5.6 – Фрагмент семантичної мережі

Ці предикати указують, що Степанові 60 років, що він проживає на Південному бульварі будинок 3, неодружений і належить типу професорів університету.

Різні імена предикатів, пов'язані з вузлами-індивідами, утворюють першу множину гіпотез (деякої теореми). Наприклад, питання про адресу Павла виражається предикатом *Адреса*(*Степан*, *x*). Цей предикат складе мету (висновок) теореми, достовірні гіпотези якої – названі вище властивості Степана. Доказ зводиться до пошуку конкретизації для *x*, що дозволяє отримати названий висновок з гіпотез. Перевірка показує, що таким є *Південний_бульвар_3*.

Інакше піде справа з питанням: «Який диплом у Степана?», бо ми не знаходимо гіпотези вигляду

Диплом(*Степан, …*).

Але якщо скористатися зв'язком *Конкр*, можна одержати відповідну гіпотезу. Пов'язана з вузлом *проф_унів* частина графа представима логічною формулою:

Конкр(*x*, *проф_унів*) ⊃ (*Диплом*(*x*, *доктор*) ∧ *Місце* (*x*, *унів*) ∧ *Це*(*x*, *проф*) ∧ *Це*(*x*, *співроб_унів*))

Якщо додати цю логічну формулу до множини (індивідуальних) гіпотез, то можна, використовуючи звичайну техніку доведення теорем, вивести, що Степан має докторський ступінь. Таким чином, нова множин гіпотез і поставлене питання складають формулювання логічної теореми. Виходячи з цього формулювання і застосовуючи звичайний механізм доказу, одержуємо відповідь на питання.

Механізм доведення теорем в ієрархічному середовищі можна змоделювати таким чином. Питання (що становить висновок або мету доводжуваної теореми) має наступну загальну форму.

Вербально: Яке значення якогось індивіда по відношенню до деякої властивості?

Логічно: *Властивість_і*(*індивід_j*, *x*).

Очікувана відповідь – конкретизація, що представляє деяке значення для *x*.

Спочатку спробуємо довести цю теорему, користуючись лише гіпотезами, прямо пов'язаними з вузлом *індивід_j* у відповідному графі. Ці гіпотези мають наступну форму.

Вербально: Значення якогось індивіда по відношенню до деякої властивості є ...

Логічно: *Властивість_і*(*індивід_j, значення_і*).

Якщо так довести не вдається, використовуємо зв'язок *Конкр*, що виходить вузла, який представляє індивіда і вказує на нову множину гіпотез, яка має наступний загальний вигляд.

Вербально: Якщо індивід належить типу *t*, то він має додаткову множину значень по відношенню до додаткової множини властивостей.

Логічно: ∀*x*[*Конкр*(*x*, *тип_t*) ⊃ ∧ *Властивість_і*(*x*, *значення_і*)].

З цієї логічної формули і предиката *Конкр*(*індивід_ j*, *тип_t*) виводимо нову множину властивостей щодо даного індивіда. Якщо ці нові гіпотези дозволяють довести нашу теорему, то доказ закінчений. Якщо ж ні, то зв'язок *Це* вкаже на тип вище, який в свою чергу дасть нову множину гіпотез. Ця процедура з послідовною множиною гіпотез свідомо ефективніша за загальний метод доведення теорем, що не використовує ієрархічної структури даних.

Семантичні мережі належать до моделей, використовуваних фахівцями з БД. Вони відображають структуру якоїсь області і реалізацію цієї структури на ЕОМ. У наш час семантичні мережі використовуються в системах розпізнавання природних мов. Вони вмонтовані також в складніші формалізми, де служать для підтримки опису і класифікації суті (експертні системи). Переваги концептуальних графів – в можливості прямого узагальнення на модальні системи і інші формалізми, які важко представити логікою першого

порядку.

6 Об'єктне представлення знань. Фреймові моделі

Відомо, що клас об'єктів може визначатися одним типовим об'єктом. Об'єктне представлення можна одержати як з логічного, так і з мережевого. При об'єктному представленні збирають з логічних формул, що містять одні і ті ж конкретизації, крупніші структури, звані фреймами. Останні будуються навколо конкретизації області міркування. Якщо потрібен доступ до інформації по одному з цих об'єктів, то звертаються до відповідного фрейма і вже усередині блоку знаходять властивості і факти щодо даного об'єкту.

Бінарна версія числення предикатів привела до графічних позначень через семантичні мережі. Бінарний предикат можна представити трійкою вигляду

(*Об'єкт*, *атрибут_і*, *значення_і*).

Зібравши всі трійки з даним об'єктом, одержимо об'єктне представлення області міркувань щодо цього об'єкту. Загальна форма такого представлення наступна:

Об'єкт (*атрибут_і*, *значення_і*) (*і* = 1,…, *n*).

Тобто замість побудови різних незалежних формул будуємо якусь структуру з повною інформацією про об'єкт.

Ці об'єктні позначення прийняті в новому сімействі мов програмування, особливо добре пристосованих до виразу знань в ШІ – об'єктно-орієнтованих мовах.

6.1 Зчеплення

Припустимо, що ми хочемо представити фрази «Петро пише книгу, посилає її Марії, яка її читає». У БД з цими фразами використовувалися конкретизації Петро_2, Марія_4, Посилка_8 і Книга_22 для посилань в об'єктній мові на імена концептів метамови, згадані в цих фразах. Якщо розширити БД, то додадуться нові концепти і додаткова інформація про них.

Для використання знань корисно зібрати всі факти про даний концепт в одну множину, що називається зчепленням (по-англійськи unit). У нашому елементарному прикладі зчепленням *Петро_2*, *Марія_4* і *Книга_22* відповідатимуть логічні формули:

Петро_2
Пише (*Петро_2, Книга_22*)
Посилає (*Петро_2, Марія_4, Книга_22*)
Марія_4
Посилає (*Петро_2, Марія_4, Книга_22*)
Читає (*Марія_4, Книга_22*)
Книга_22
Пише (*Петро_2, Книга_22*)
Посилає (*Петро_2, Марія_4, Книга_22*)
Читає (*Марія_4, Книга_22*)

6.2 Фрейми і слоти

Якщо виразити ці фрази бінарними предикатами то зчеплення називатимуться фреймами. Предикат з трьома аргументами

Посилає(*Петро_2, Марія_4, Книга_22*)

перетвориться в добуток бінарних предикатів

Відправник (*Посилає, Петро_2*) ∧

Одержувач (*Посилає, Марія_4*) ∧

Об'єкт (*Посилає, Книга_22*).

Концепту «посилає» відповідає наступний фрейм:

ФРЕЙМ *Посилає* (об'єкт)

(*Відправник*, *Петро_2*) (слот-1)

(*Одержувач*, *Марія_4*) (слот-2)

(*Об'єкт*, *Книга_22*) (слот-3)

Кожна пара (*атрибут*, *значення*) фрейма називається слотом або (*ім'я_слота*, *значення_слота*). Сам фрейм по-англійськи — slot-and-filler notation. У цих позначеннях різні слоти згруповані навколо об'єкту, охарактеризованого фреймом.

У позначеннях slot-assertion відповідні бінарним предикатам слоти використовуються як ізольована сутність без угрупування в охоплюючому їх фреймі.

6.2.1 Явні фрейми

Раніше було показано, що часто корисне представлення знань з явною вказівкою всіх посилань. Саме тому ми постулювали в нашому прикладі існування цілком певної Посилки_8. Фраза «Петро посилає книгу Марії» бінарними предикатами представляється так:

Елем(*Посилка_8, посилки*) ∧

Відправник(Посилка_8, Петро_2) ∧

Одержувач(*Посилка_8, Марія_4*) ∧

Об'єкт(*Посилка_8, Книга_22*)

Отже, в процесі виявлення посилань можна не тільки дати явні значення аргументів і імена предикатів, але також імена представляючих вислови логічних формул. Наприклад, *Посилка_8* – ім'я вислову *Посилає*(*Петро_2*, *Марія_4*, *Книга_22*). Такий формалізм називається явним фреймом (по-англійськи case-frame).

6.2.2 Функціональні фрейми

Представлення бінарними предикатами легко можна виразити у функціональній формі. Відносини між *Посилка_8* і первинними аргументами для *Посилає* можна виразити функціями на множині посилок. Введемо наступні функціональні позначення для фреймів:

Посилка_8

елем : *(елем_із посилок)*

відправник : *Петро_2*

одержувач : *Марія_4*
об'єкт: *Книга_22*

Форма «елемент-із» в слоті має ім'я «елем» для вказівки того, що описаний фреймом об'єкт є елементом деякої множини (у нашому прикладі це множина посилок). Визначений у такий спосіб фрейм називається функціональним.

6.2.3 ∀–квантифікація

Подивимося, як можна здійснювати ∀-квантифікацію змінних при операції з фреймами. Фраза «Петро посилає книгу кожній жінці» записується бінарними предикатами в наступному вигляді:

$\forall x \exists y \exists z$ [*Відправник* (z, *Петро_2*) $\wedge$ *Одержувач*(x, y) $\wedge$ *Об'єкт*(z, y) $\wedge$ *Елем*(z, *посилки*) $\wedge$ *Конкр* (y, *книга*) $\wedge$ *Конкр* (x, *жінка*)].

Множина представлених змінною z посилок описано функціональним фреймом:

z(x)
елем: (*элем_із посилок*)
відправник: Петро_2
одержувач: x
об'єкт: *y(x)*

Цей фрейм цілком утворює область дії квантора узагальнення. Важливо відзначити, що всі представлені фреймами логічні формули записані передвареній сколемівській формі: заперечення виражені явно, змінні стандартизовані окремо, змінні, які були зв'язані кванторами існування, заміщені і квантори узагальнення застосовуються до всієї формули.

6.3 Міркування, що використовують об'єктне уявлення

Об'єктні представлення були введені фактично з прагматичних причин. Ці представлення часто пов’язані з цілком певними класами проблем. Об'єктно-орієнт*овані мови* (відомий з них Смолток) характеризуються взаємопроникненням структур даних і процедур. Пов'язані з цими представленнями закони висновку, взагалі кажучи, позбавлені формальній строгості правил виведення логіки предикатів. Концепція об'єктних уявлень більше прямувала турботою про ефективність числення, ніж прагненням до його повноти. Очевидно, можна було б розвинути для них системи висновків, еквівалентні використаним в логіці предикатів. Такий синтаксис був би вельми громіздкий, і його застосування позбавило б об'єктне представлення його прагматичних достоїнств. Замість того щоб базуватися на повній множині законів дедукції, різні використовувані в ШІ системи використовують скорочені версії класичних законів висновку. Пізніше ми дамо декілька прикладів.

6.3.1 Паропоєднання

Визначити (аналогічну уніфікації в логіці предикатів) операцію паропоєднання (по-англійськи matching operation) настійно необхідно, якщо ми

хочемо використовувати об'єктне уявлення як БД для запитальної системи (query system). До завдання властивостей операції парооб’єднання нагадаємо, що об'єктне представлення – альтернатива логічному формалізму.

Два об'єктні представлення паропоєднані тоді і тільки тоді, коли логічну формулу для першого можна уніфікувати з логічною формулою для другого. Слово «уніфікувати» означає тут, що існують підстановки для змінних, що роблять логічні формули ідентичними.

Якщо наша мета – побудувати «мову запитів» (query language) для витягання інформації з БД, то потрібне більш обмежене визначення парооб’єднання. Операція парооб’єднання взагалі не повинна бути симетричною в тому сенсі, що вона пов'язує висновок (об'єкт-мета) з гіпотезами (об'єктами-фактами). Об'єкт-мета «паропоєднується» з об'єктом-фактом, якщо логічна формула, що відповідає об'єкт-меті, уніфікується з одним із співмножників об'єкту-факту (представленого кон'юнкцією гіпотез і аксіом).

В термінах доведення теорем це означає, що операція паропоєднання може здійснюватися тоді і тільки тоді, коли мету можна довести з гіпотез і аксіом (фактів).

Розглянемо фрейм-факт (гіпотезу):

Посилка_8
елем: (елем_із посилок)
відправник: Петро_2
одержувач: Марія_4
об'єкт: Книга_22

Можна встановити паропоєднання цього фрейма-факту з фреймом-метою:

z(x)
елем: (елем_из посилок)
відправник: Петро_2
одержувач: x
об'єкт: y(x)

Фрейм-мета інтерпретується як питання: «Кому Петро послав книгу?» Паропоєднання фрейма-мети і фрейма-факту приводить до підстановки {(*z*, *Посилка_8*), (*x*, *Марія_4*), (*y*, *Книга-22*)}, яку можна інтерпретувати як відповідь на питання.

6.3.2 Функціональні атрибути

Атрибути описують множину характеристик, пов'язаних з функціями функціонального фрейма. Атрибут не обов'язково є конкретизацією деяких даних типу *Павло_2*; з тим же успіхом він може бути функціональним виразом. Розглянемо фразу «Павло посилає книгу Марії, а Петро одержує лист від людини, якій Павло послав книгу». Її перша частина виражається фреймом:

Посилка_8
елем: (елем_із посилок)
відправник : Петро_2
одержувач: Марія_4
об'єкт: Книга_22

Друга частина представляється фреймом, для якого деякий функціональний вираз є атрибутом «відправника»:

Посилка_9
елем: (елем_із посилок)
відправник: одержувач (Посилка_8)
одержувач: Павло_6
об'єкт: Лист_3

Марія_4 і *одержувач(Посилка_8)* – два різних способу представлення однієї і тієї ж особи.

Якщо намагаємося встановити паропоєднання між фреймами, що містять функціональні вирази як атрибути, то слід припустити, що ці вирази набудуть значення відразу, як тільки це буде можливо. Привласнення значень здійснюється за допомогою посилання на об'єкт, вказаний аргументом цієї функції.

Припустимо, що поставлене питання: «Чи посилала Марія що-небудь кому-небудь?», що представляється фреймом-метою:

z(x)
елем: (елем_із посилок)
відправник: Марія_4
одержувач: x
об'єкт: y

Множина фреймів-фактів містить *Посилку_8* і *Посилку_9*. Оскільки одержувач (*Посилка_8*) може набути значення *Марія_4*, то фрейм *Посилка_9* одержить *Марія_4* як значення функції відправник. Відповідь на питання дасть паропоєднання фрейма *Посилка_9* з фреймом *z*.

З допомогою підстановка {(*x*, *Павло_6*), (*y*, *Лист_3*)} можна одержати відповідь «так, Марія відправила лист Павлу».

6.4 Автоматичні міркування, що використовують фрейми

Як відомо, атрибути функцій у фреймах це не обов'язково константи або конкретизації, але можуть також бути функціональними виразами. Зробимо ще крок і розглянемо ще одну можливість для атрибутів (представляючих множину пов'язаних з об'єктами характеристик) – бути процедурами або підпрограмами. Це приводить нас до поняття об'єктно-орієнтованих мов, предназначених для автоматичних міркувань на основі об'єктного представлення.

Мова KRL – об'єктно-орієнтована, заснований на представленні знань фреймами. У ній всі знання описані в термінах UNITS-KRL. Обмежимося двома прикладами:

[*Подорож* UNIT *Абстрактна*
(SELF (*Подія*)) (*спосіб* (АБО *Авіа Авто Потяг*))
(*призначення* (*Місто*))]
[*Подорож* UNIT *Абстрактна*
(SELF (*Взаємодія*)) (*візитер* (*Людина*))
(*візит* (*Множина* з (*Людина*)))]

Першу частину можна інтерпретувати таким чином: «Подорож – абстрактне поняття, що пов'язане з літаком, машиною або поїздом і має якесь місто пунктом призначення». Другу частину можна інтерпретувати аналогічно. KRL матеріалізує деякий процес міркувань, керований правилом паропоєднання.

6.4.1 Ієрархічні міркування, що використовують фрейми

Операція паропоєднання аналогічна уніфікації з логіки предикатів. Можна також пов'язати об'єктне представлення з могутнішими правилами висновку, такими як теорема дедукції. Нільсон і Сова запропонували загальні операції дедукції, що використовують об'єктний формалізм і що базуються на логічній імплікації і правилі modus ponens. Розглянемо лише окремий випадок одного нерідко використовуваного закону дедукції, відповідно до якого за допомогою імплікації приписуються властивості кожному елементу певної множини. Йдеться про властивість спадкоємства. Повернемося до ієрархії, приведеної на рис.5.6, де типи «професор університету» і «професор» замінені відповідними множинами. Концептуальний граф, що оточує вузол *Степан_2*, представляється об'єктно і логічно таким чином:

Об'єктні позначення	**Логічні позначення**
Степан_2 *елем : (елем_із проф_унів)* *вік: 60* *адр: Північний_бульвар_3*	*Елем (Степан_2, проф_унів) ∧* *Вік (Степан_2, 60_років) ∧* *Адр(Степан_2, Північний_бульвар_3)*

Ми бачили, що питання на зразок «Скільки років Степану?» або «Хто проживає на Північному бульварі будинок, 3?» представляються фреймом-метою або логічним висновком:

Об'єктні позначення	**Логічні позначення**
Степан_2 *вік: x* *y* *адр*: *Північний_бульвар_3*	(Мета) *Вік(Степан_2, x)* *Адр(y, Північний_бульвар_3)*

Раніше було показано, як одержати відповіді на ці питання, використовуючи операцію паропоєднання (для об'єктних позначень) або уніфікацію (для логічних).

Далі розглянемо питання «Який диплом у Степана?», формалізований наступним чином:

Об'єктні позначення	**Логічні позначення**	
Степан_2 *дипл: y*	(Мета) *Дипл(Степан_2, y)*	(6.1)

Не можна одержати негайної відповіді на це питання. Доведеться звернутися до об'єктних і логічних позначень, що описують властивості індивіда *x* з множини викладачів університету (рис.5.6):

Об'єктні позначення	**Логічні позначення**
{ x | проф__унів}	(Правило)

дипл: доктор *місце : унів*	*Елем (х, проф_унів) ⊃* *(Дипл (х, доктор) ∧* *Місце (х, унів))*	(6.2)

Здійснимо паропоєднання і уніфікацію цілей (6.1) з правилами (6.2) для отримання нової мети (6.3):

Об'єктні позначення	**Логічні позначення**
Степан_2	(Мета)
елем: (елем_із проф_унів)	*Елем(Степан_2, проф_унів)*

Ця нова мета точно відповідає фразі «Степан – професор університету», що є фактом. Відповідь «Степан має ступінь доктора» також знайдено. Такий вид доведення відповідає системі прямої дедукції

Якщо не можна відповісти на питання, розглядаючи властивості індивіда *(Степан_2)* або множини *(проф_унів)*, що містить його, то звернемося до нової множини *(проф),* підмножиною якої є попередня множина. Підмножина професорів університету пов'язана множиною професорів наступним чином:

Об'єктні позначення	**Логічні позначення**
проф_унів	
елем: (підмнож в проф)	*Подмн (проф_унів, проф)*

6.4.2 Міркування з винятками

Представлення, що вивчалися дотепер, уразливі для наступного докору. Вони припускають строгі міркування, тоді як не завжди можна спертися на формально точні знання. Використовувані експертами в більшості областей правила лише приблизно точні і не завжди застосовні. Численні описи вигляду «всі *х* мають властивість *Р»* треба вважати лише приблизно істинними. Наприклад, можна говорити, що «всі птахи літають (за винятком страусів, пінгвінів і т. д.)».

Взагалі цікаво використовувати знання, ніби «всі птахи літають», і піддавати міркування сумніву лише в деяких наперед відомих виняткових випадках. Це називається *міркуваннями з винятками.* Запропоновані численні формалізації для виразу «всі *х* мають' властивість *Р,* за винятком тих випадків, коли явно вказано протилежне для деяких конкретизації змінної *х».*

Викладемо формалізм, що заснований на об'єктному представленні і допускає побудову міркування з винятком. Підхід, що розглядається нами, зберігає певну простоту міркувань. Він полягає в дозволі здійснювати виключення в міркуваннях, проведених за правилом ∀-квантифікації, що діє на формули з імплікацією. Твердження вигляду «всі викладачі університету мають ступінь доктора» можна спочатку зробити без жодних виключень. З нього виводимо, що «Степан має ступінь доктора, якщо Степан викладає в університеті». Якщо надалі з'ясується, що Степан не має ступеня доктора, то ми анулюємо нашу дедукцію і так змінимо універсальне твердження про викладачів університету, щоб Степан виявився винятком.

Спосіб, яким операція паропоєднання використовує спадкову властивість, може дати автоматичний механізм обробки винятків такого роду. Керуючий

операцією паропоєднання механізм може використовувати спадкову властивість для виведення якоїсь властивості об'єкту, тільки якщо специфічна інформація про цю властивість не з'являється у відповідному цьому об'єкту фреймі. Наприклад, ми цікавимося, який диплом у Степана. Для відповіді на питання спочатку спробуємо встановити паропоєднання фрейма-мети з фреймом-фактом. Якщо фрейм-факт по Степану має вигляд

Степан_2
елем: (елем_із проф_унів)
дипл: кандидат

то паропоєднання залучає уніфікацію *(у, кандидат)* і відповідь буде наступною: *кандидат* (Степан має диплом *кандидата)*. Навпаки, якщо фрейм-факт по Степану говорить лише, що він професор університету:

Степан_2
елем: *(елем_із проф_унів)*

то механізм паропоєднання використовує фрейм:

{ х | проф_унів}
дипл: доктор

і відповідь – *доктор.*

7 ПРОДУКЦІЙНІ МОДЕЛІ ПРЕДСТАВЛЕННЯ ЗНАНЬ

Продукції на ряду з фреймами є найбільш популярними засобами представлення знань в ІС (інтелектуальні системи). Продукції, з одного боку, близькі до логічних моделей. Це дозволяє організовувати на них ефективні процедури висновку. З іншого боку, продукції наочніше відображають знання, чим класичні наочні моделі.

У загальному вигляді під продукцією розуміється вираз наступного вигляду

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

де ***i* – ім'я продукції**, за допомогою якого дана продукція виділяється зі всієї множини продукцій. Як ім'я використовуватися ідентифікатор або порядковий номер продукції в їх множині, що зберігається в пам'яті системи.

Q характеризує **сферу застосування** продукції або клас ситуації, в якій дана продукція використовується. Розділення знань на окремі сфери дозволяє економити час на пошук потрібних знань.

A ⇒ B – ядро продукції, основна частина продукції. Інтерпретація ядра продукцій може бути різною і залежить від того, що стоїть зліва і праворуч від знаку секвенції '⇒'.

Звичайне прочитання ядра продукції виглядає так: ЯКЩО А, ТО В. Складніші конструкції ядра допускають в правій частині альтернативний вибір, наприклад, ЯКЩО А, ТО В1, ІНАКШЕ В2.

Секвенція може тлумачитися в звичайному логічному сенсі як знак логічного проходження В з істинного А. Якщо А не є дійсним виразом, то про В нічого сказати не можна.

Частина А ядра продукції називається **антецедентом**, або посилкою. Вона може складаються з елементарних пропозицій (умова, твердження), що називається зразками, сполучених логічними зв'язками АБО, І.

Частина В ядра продукції називається **консеквентом**, або висновком. Вона складається з одного або декількох пропозицій, які утворюють рішення правила, або указує на дію підмет виконанню.

Приклад. ЯКЩО двигун автомобіля не заводиться І стартер двигуна автомобіля не працює – антецендент (посилка)

ТО неполадки в системі електроживлення стартера. – Консеквент (висновок)

Елемент *P* – предикат умови застосування продукції. Звичайно P є логічним виразом (як правило, предикат). Коли P приймає значення "істина", ядро продукції активізується. Коли Р помилково, те ядро продукції не може бути використано.

Приклад. НАЯВНІСТЬ ГРОШЕЙ; ЯКЩО бажаєте купити річ Х, ТО заплатіть у касу її вартість І пред'явите чек продавцю;

Якщо в цій продукції умова застосовності ядра продукції помилкова тобто грошей немає, то застосувати ядро продукції неможливо

Елемент *N* – постумови продукції. Вони актуалізуються тільки тому випадку, якщо ядро продукції реалізувалося. Постумови продукції описують дії

і процедури, які необхідно виконати після реалізації В (правої частини ядра).

Приклад. Після покупки деякої речі в магазині необхідно в описі товарів, наявних в цьому магазині, зменшити кількість речей такого типу на 1.

Постумови продукції можуть бути також вказівками на зміни, які треба внести дану продукцію (в умову Р, в А або В) або зміни, які треба внести в інші продукції системи після реалізації даної продукції.

Якщо в пам'яті системи зберігається деякий набір продукцій, то вони утворюють систему продукцій.

У системі продукцій повинні бути задані спеціальні процедури управління продукціями, за допомогою яких відбувається актуалізація продукцій і вибір для виконання тієї або іншої продукції з числа актуалізованих.

У ряді систем штучного інтелекту використовуються комбінації продукційних моделей і інших, наприклад мережевих. У таких моделях декларативні знання описуються в мережевому компоненті, а процедурні знання в продукційних. В цьому випадку говорять про роботу продукційної системи над семантичною мережею.

Приклад: Хай продукція $A \Rightarrow B$. A і B є фрагментами інформації, що зберігається в базі знань. При мережевому представленні це можуть бути фрагменти семантичної мережі, при логічних моделях – формули того або іншого числення. Тоді сенс продукції $A \Rightarrow B$ полягає в заміні одного фрагмента бази знань іншим. Для актуалізації цієї продукції необхідно, щоб в базі знань існував фрагмент, що співпадає з A.

При пошуку в БЗ A грає роль зразка, а процедура такого пошуку називається ПОШУКОМ ПО ЗРАЗКУ.

Проілюструємо пошук за зразком на прикладі (рис.7.1, 7.2). Припустимо, що в БЗ для представлення знань використовуються наступні семантична мережа і продукція .

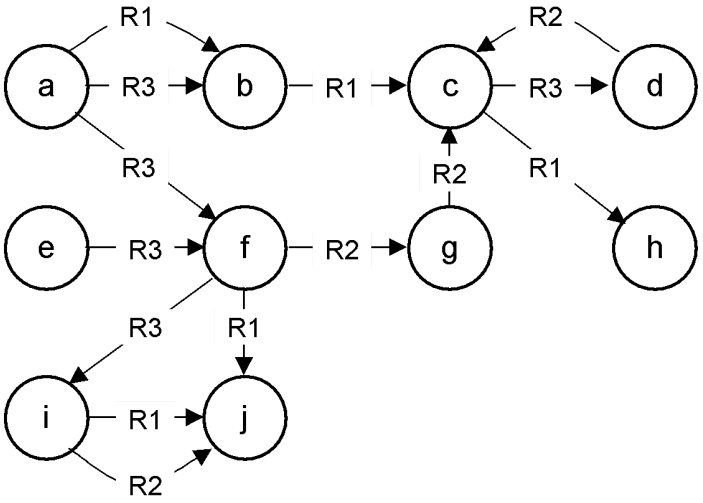


Рисунок 7.1 – Семантична мережа

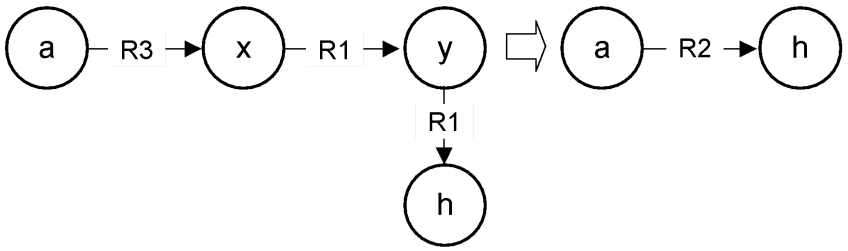


Рисунок 7.2 – Правило продукції,

На рис. 7.2 *x* і *y* – вільні змінні, яким можуть привласнюватися будь-які значення. Щоб застосувати продукцію до семантичної мережі, здійснюється пошук А в БЗ. Пошук можна організувати різними способами.

Можна, наприклад, спочатку шукати вершину *a*. Якщо в БЗ такої вершини немає, то пошук за зразком закінчується невдачею.

Якщо вершина *a* знайдена, то шукаються всі дуги, що виходять з неї, помічені відношенням *R*3, оскільки в зразку праворуч від цієї дуги стоїть вершина *x*, на місці вершини *x* в базі знань може знаходитися будь-яка вершина.

Якщо з *a* не виходитиме жодної дуги, поміченої відношенням *R*3, то пошук за зразком закінчується невдачею. Якщо такі дуги є, то відбувається перехід у всі вершини, з якими вершину а зв'язує відношення *R*3. У нашому прикладі відбувається перехід від вершини *a* до вершин *b* і *f*, з *b* починається пошук дуг, що виходять з них, помічених відношенням *R*1, що ведуть в будь-яку вершину, оскільки в зразку далі стоїть вершина, якій відповідає вільна змінна *y*.

Далі процес продовжується аналогічним чином.

У прикладі пошук за зразком є успішним.

Після знаходження *A* в семантичній мережі відбувається заміна, яка визначається правою частиною зразка *B*.

Виходить трансформована мережа (рис. 7.3)

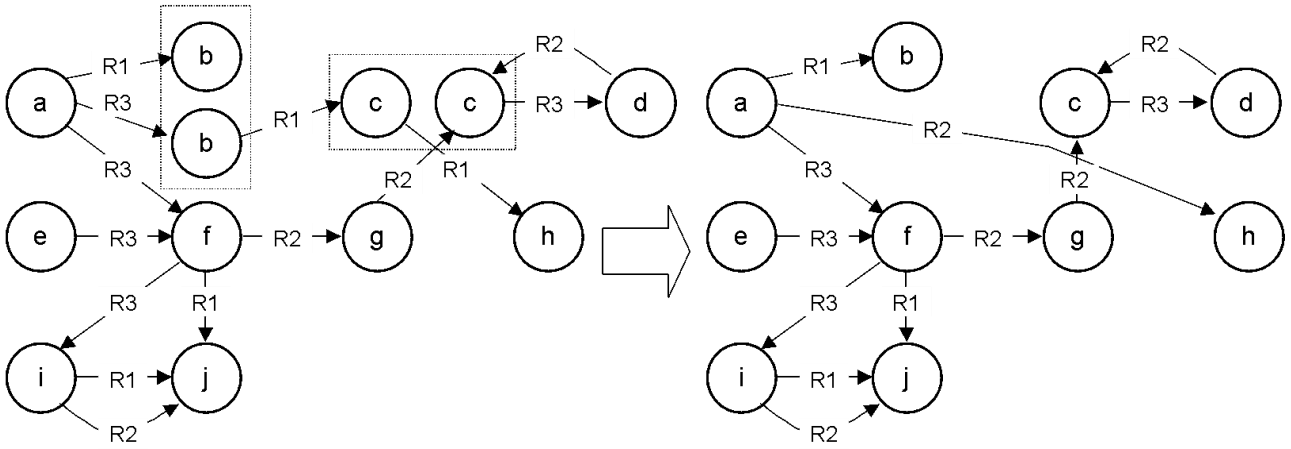


Рисунок 7.3 – Трансформована мережа

7.1 Класифікація ядер продукції

Ядра продукції можна класифікувати по різних підставах. Ядра можуть бути розділені на два типи:

1. Детерміновані

2. Недетерміновані

У детермінованих ядрах при актуалізації ядра і при здійснимості лівої частини права частина ядра виконується обов'язково. У недетермінованих ядрах права частина може виконуватися і не виконуватися. Таким чином секвенції в детермінованих ядрах реалізуються з необхідністю, а в недетермінованих - з можливістю.

Приклад: Інтерпретація ядра в цьому випадку може, наприклад, виглядати так: ЯКЩО А, ТО МОЖЛИВО В.

Можливість може визначатися деякими оцінками реалізації ядра.

Приклад: Якщо задана вірогідність виконання В при актуалізації А, то продукція може бути такою: ЯКЩО А, ТО З ВІРОГІДНІСТЮ р РЕАЛІЗУВАТИ В.

У простому випадку продукція складається лише з ядра. Оцінка реалізації ядра може бути лінгвістичною, пов'язаною з поняттям множини терма лінгвістичною змінною.

Приклад: ЯКЩО А, ТО З СИЛЬНОЮ ЧАСТКОЮ ВІРОГІДНІСТІ РЕАЛІЗУВАТИ В

Можливі інші способи завдання оцінки реалізації ядра.

Детерміновані продукції можуть бути однозначними і альтернативними. у другому випадку в правій частині ядра указуються альтернативні можливості вибору, які оцінюються спеціальними вагами вибору**.**

Як ваги вибору можуть використовуватися імовірнісні, лінгвістичні, експертні оцінки і т.п.

Приклад: ЯКЩО А, ТО НАЙЧАСТІШЕ ПОТРІБНО РОБИТИ В1, РІДШЕ В2.

Особливим типом є прогнозуюча продукція, в яких описуються наслідки, очікувані при актуалізації А.

Приклад: ЯКЩО А, ТО З ВІРОГІДНІСТЮ р МОЖНА ЧЕКАТИ В.

Системи продукцій широко поширені в ЕС.

7.2 Пряме та зворотне виведення

Існують дві основні стратегії логічного виведення в продукційних системах:

– пряме;

– зворотне.

Пряме виведення на основі наявних правил передбачає аналіз: 1) усіх наслідків з фактів, 2) наслідків з наслідків і т. д. Процес продовжується поки не буде встановлено істинність або хибність запиту користувача.

Наприклад, нехай є продукції *A =>B* та *B=>C* користувач оголосив про істинність факту *A* і спитав про істинність *C*. Ланцюжок прямого виведення матиме вигляд: з *A* виводиться *B, з B* виводиться С, отже *C* — істинне.

Пряме виведення не вважається ефективним через те, що може перебиратися чимало безперспективних продукцій, і, відповідно, породжується дуже багато зайвих проміжних результатів. Експертні системи на основі

прямого виведення використовуються, як правило, при плануванні і прогнозуванні.

При *зворотному виведенні* логічний блок починає роботу із запиту користувача, тобто з твердження, яке перевіряється. Розглядається одне з правил, на основі яких можна вивести це твердження, після чого перевіряється істинність лівої частини цього правила. Процес повторюється, доки не дійде до фактів, які вважаються істинними.

У нашому прикладі ланцюжок зворотного виведення матиме вигляд: "*C* виводиться *з B, B* виводиться *з A, A* істинне, отже, і *C* істинне".

Як і у випадку прямого виведення, зворотне виведення може вимагати повторень у разі невдалої спроби, і тому є стратегією перебору.

Зворотне виведення тісно пов'язано з резолюцією, зокрема лінійною, і з прологівським механізмом виконання програм.

7.3 Типові дисципліни виконання продукцій

Передусім ядра продукцій прийнято поділяти на *детерміновані* і *недетерміновані.* У детермінованих продукціях при виконанні лівої частини завжди виконується і права. У недетермінованих продукціях права частина при виконанні лівої виконується необов'язково. Можливі, наприклад, такі інтерпретації недетермінованих продукцій: *"Якщо A, то можливе B";* або *"Якщо A, то B з певною вірогідністю ".*

Існують різні режими керування виконанням готових продукцій, тобто продукцій, ліві частини яких справедливі, і, відповідно, ці продукції можуть бути виконані негайно.

Можна виокремити такі режими:

- *режим негайного виконання;*
- *режим формування конфліктного набору.*

У режимі негайного виконання, якщо знайдена перша-ліпша продукція, саме вона виконується невідкладно.

У режимі формування конфліктного набору знайдені готові продукції не виконуються відразу, а включаються до *конфліктного набору* — списку продукцій, готових до виконання. Лише після завершення формування конфліктного набору відбувається вирішення конфлікту, тобто вибір зі списку готових продукцій якоїсь однієї. Інша назва конфліктного набору *"фронт готових продукцій ".*

Розглянемо деякі проблеми, пов'язані з конфліктними наборами.

Нехай маємо продукції:

1) *A => B*

2) *A => C,*

і *A* істинне.

Тоді обидві продукції готові до виконання і утворюють конфліктний набір.

Якщо ядра продукцій інтерпретуються як імплікації, проблема є менш гострою. Продукційна система є чисто декларативною. Після застосування 1) тобто виведення *B, A* залишається істинним, і ми завжди можемо використати

продукцію 2). Таким чином, порядок застосування продукційних правил може впливати лише на час роботи системи, але не на істинність висновків.

Зовсім інша ситуація виникає, якщо ядра продукцій інтерпретуються як "виконати таку-то дію". Тоді після виконання 1) ситуація може змінитися: *A* може перестати бути істинним, і тоді продукція 2) може стати недосяжною. Тому, якщо насправді треба було вибирати *C,* вибір *B* стає помилкою, яку не завжди можна виправити.

Існують різні стратегії вирішення конфліктів, а також обмеження при включенні готових продукцій до конфліктного набору.

Розрізняють також *централізоване* і *децентралізоване керування* про-дукціями.

За централізованого керування продукційна система має центр керування, який вирішує, коли має спрацювати та чи інша продукція.

За децентралізованого керування кожна продукція може самостійно прийняти рішення про свій запуск. Один з типових механізмів, який застосовується при цьому, є механізм *"класної дошки"*. Таку назву має спеціальна область пам'яті, доступна для різних продукцій. На ній продукції, . що працюють паралельно, знаходять інформацію, що може ініціювати їх запуск. Туди ж вони пишуть інформацію, що може виявитись корисною для інших продукцій.

Зрозуміло, що "класних дошок" може бути кілька; можуть бути введені певні ієрархії "класних дошок" і т. п.

7.4 Основні стратегії вирішення конфліктів у продукційних системах

Очевидно, що найзагальнішим способом керування системою продукцій є комплексне планування її роботи, тобто прийняття рішень на основі аналізу всіх можливих дій та їх наслідків з погляду мети, що стоїть перед системою. Але зрозуміло, що вказаний принцип рідко може бути застосований у повному обсязі, насамперед через експоненційне зростання можливих варіантів розвитку подій.

Існують стратегії керування вирішенням конфліктів, мета яких — намагатися уникнути експоненційного вибуху.

Принцип «стосу книг». Ґрунтується на ідеї, що найкориснішою є та продукція, яка використовується найчастіше (так само, як у стосі книг ті, які найчастіше користуються попитом, як правило, опиняються зверху). Відповідно, з фронту готових продукцій вибирається продукція з максимальною частотою використання. Керування за принципом "стосу книг" доцільно застосовувати, якщо продукції відносно незалежні одна від одної, наприклад, коли кожна з них являє собою правило «**ситуація *A* ⇒ дія *B*».**

Принцип найдовшої умови. З фронту готових продукцій вибирається та, для якої стала справедливою найдовша умова виконання. Ідеологічною основою цього принципу є таке міркування, що часткові правила, застосовані до вузького класу ситуацій, є важливішими, ніж загальні правила для широкого класу ситуацій. Наприклад, нехай є дві продукції:

(1) *Якщо А – птах, то А літає.*

(2) *Якщо А – птах і А – пінгвін, то А не літає.*

Якщо відомо, що *А – пінгвін,* то за принципом найдовшої умови слід вибирати (2).

Принцип найдовшої умови доцільно застосувати, якщо продукції прив'язані до типових ситуацій, зв'язаних відношенням **«часткове–загальне».** До цього можна додати, що цей принцип заслуговує на особливу увагу при аналізі винятків. Так, у нашому прикладі правило (2) є винятком з правила (1); якби ми були впевнені, що (1) не має винятків, ми могли б застосувати це загальне правило у будь-якій ситуації.

Принцип метапродукцій. Ґрунтується на введенні до системи знань *метапродукцій,* тобто правил використання продукцій. Типовими мета-продукціями можуть бути правила, які визначають, що слід робити, якщо до фронту готових продукцій ввійшли або не ввійшли ті чи інші конкретні продукції.

Принцип «класної дошки». Конфлікти вирішуються на основі обміну інформацією з використанням «класної дошки». Застосування «класної дошки» часто комбінується з застосуванням метапродукцій.

Принцип вибору за пріоритетом. Кожній продукції надається пріоритет, що відображає її важливість. Ці пріоритети можуть бути різними для кожного типу ситуацій, а також статичними і динамічними. Динамічні пріоритети змінюються з часом і можуть відображати, наприклад, ефективність використання продукцій у минулому або час перебування продукції У фронті готових продукцій. Аналіз часу знаходження продукції у фронті готових продукцій цікавий тим, що на його основі можна дійти діаметрально протилежних висновків. З одного боку, система керування продукціями може з часом підвищувати динамічний пріоритет, щоб будь-яка продукція колись була обов'язково виконана. З іншого боку, система керування може дійти висновку, що, якщо продукція досі не виконувалась, то вона взагалі не потрібна, і, відповідно, не підвищити її пріоритет, а, навпаки, знизити.

Керування за іменами. Базується на заданні для імен продукцій формальної граматики або іншої процедури, що забезпечує звуження фронту готових продукцій і вибір з нього чергової продукції для виконання.

Переваги продукційних моделей:

1) Бульшу частину знань людини може бути записана у вигляді продукций.

2) Системи продукцій є модульними, звідси простота доповнення, модифікації і анулювання знань. За невеликим винятком видалення або додавання продукцій не приводить до змін в решті продукцій.

3) Висока однорідність організації знань, описаних по одному синтаксису.

4) При необхідності системи продукцій можуть реалізувати будь-які алгоритми і, отже, здатні відображати будь-яке процедурне знання доступне ЕОМ

5) Наявність в продукціях покажчиків на сферу застосування

продукцій дозволяє ефективно організувати пам'ять, скоротивши час пошуку в ній необхідної інформації.

6) Природний паралелізм в системі продукцій, асинхронність їх реалізації (продукції реалізуються відразу ж після реалізації попередньої) роблять продукційні системи зручною моделлю обчислень для ЕОМ нової архітектури.

Недоліки продукційних моделей:

1) При великому числі продукцій стає складною перевірка суперечності системи продукцій. Це примушує при додаванні нових продукцій витрачати багато часу на перевірку несуперечності нової системи.

2) Із-за властивій системі недетермінованості (тобто неоднозначного вибору продукції з фронту готових продукцій) виникають принципові труднощі при перевірці коректності роботи системи.

3) Вважається, що якщо в ІС число продукцій досягає 1000, то мало шансів, що система продукцій у всіх випадках правильно функціонуватиме. Тому число продукцій, з яким як правило, працюють сучасні ІС, не перевищує 1000.

8 ФОРМАЛІЗАЦІЯ НЕДОСТОВІРНИХ І НЕЧІТКИХ ЗНАНЬ

8.1 Модальна логіка предикатів

У буденній мові часто говорять про допустимість чого-небудь, про гіпотетичні події, цілі, які можна намагатися досягти, припущення про майбутнє. Велика частина фраз мови може бути то істинною те помилкової залежно від обставин, теперішнього моменту, точки зору кожного з нас. У природних мовах модальності «можливий», «необхідний» і «допустимий», виражаються допоміжними дієсловами, такими як «повинен» і «можу».

Можливість і необхідність називаються *алетичними модальностями* або *модальностями можливості.* Так само, як квантори «для всіх» і «існує» вводилися в синтаксисі логіки першого порядку, можна побудувати формальну мову, використовуючи пару понять «можливо»/«необхідно» як квантори, що діють на формули. Логічна система, що базується на операторах «можливо, щоб», і «необхідно, щоб», називається *логікою можливого,* або *алетичною логікою.*

Для позначення модальності «необхідно» використовується символ •. Формула •*F* читається «необхідно, щоб *F*» або «*F* необхідне». Подвійний • оператор позначається ◊. Формула ◊*F* читається «можливо, що *F*» або «*F* можливо». Один з цих операторів приймається за основний, а інший визначається через нього і заперечення (еквівалентність $\Diamond F \equiv \neg \Diamond \neg F$ можна встановити, застосовуючи доведення, подібні тим, які можна використовувати при доказі співвідношень $\forall x F \equiv \neg \exists x \neg F$).

У буденній мові уживаються і інші модальні форми, які важливо перенести в логіку. *Деонтіческая логіка* вводить модальності «дозволено» «обов'язково», реалізовуючі модальні мовні конструкції «вирішується» «треба, щоб». *Епістемічна логіка,* або *логіка знання* досліджує модальності «знання» і «віри», тоді як *часова логіка* вводить модальності «іноді» і «завжди» (в «майбутньому» і «минулому») разом з їх запереченнями «часто» і «ніколи».

Зважаючи на формальну схожість між цими логічними системами їх часто вивчають всі разом або хоч би по групах. Часто використовують термін *модальна логіка* для позначення сукупності цих логік. Найстарша серед них – логіка можливого. Часто саме її називають модальною.

Назва *модальна логіка* походить від того, що модальні логічні системи вводять такі оператори над логічними формулами, які дозволяють модифікувати їх інтерпретацію. Наприклад, в твердженнях «Можливо, що *F*», «Василь думає, що *F*», «Часто правда, що *F*», «В майбутньому, можливо, буде правда, що *F*», вирази, що передують логічній формулі *F* є модальними операторами. Вони можуть відноситися до якої завгодно логічної формули *F.* Значення істинності цих формул залежить не тільки від значення істинності формули *F,* яку вони містять, але і від моменту проголошення модальної формули (часові логіки), від людини, яка думає, що *F,* або вірить, що *F* (логіки віри), або від необхідного, можливого або випадкового характеру деякого факту (логіки можливого).

8.1.1 Модальні оператори

Є схожість між визначеннями кожної з пар операторів (на зразок «можливо»/«напевно», «іноді»/«завжди» і т.д.) і визначенням пари кванторів існування/узагальнення в логіці першого порядку. Це підказує наступне загальне визначення модальних операторів.

Модальний оператор узагальнення Δ *є:*

– або іменем модального оператора

– або виразом, що складається з імені модального оператора, за яким слідує список $(t_1, ..., t_n)$ *термів* t_i $(i = 1,...,n)$.

Модальний оператор існування ∇ *визначається через заперечення модального оператора узагальнення:*

∇F=def $\lceil \Delta \rceil F$. (8.1)

8.1.2 Приклади модальних операторів

Алетичні оператори:

– •: необхідно; •*A* – істинно тоді і тільки тоді, коли *A* необхідно істинне або *A* абсолютно істинно або *A* істинно у всіх можливих світах.

– ◊: можливо; ◊*A* – істинно, якщо *A* може виявитися істинним або якщо *A* умовно істинно або якщо *A* істинно в деякому можливому світі.

Часові оператори:

– *G*: завжди (в майбутньому); *GA* – істинно, якщо *A* залишиться істинним назавжди.

– *H*: завжди (у минулому); *HA* – істинно, якщо *A* завжди було істинним.

– *F* : іноді (в майбутньому); *FA* – істинно, якщо *A* іноді буде істинним.

– *P :* іноді (у минулому); *PA* – істинно, якщо *A* іноді виявлялося істинним.

– *U :* до тих пір, поки; *U*(*A, B*) – істинно, якщо *A* істинно (починаючи з теперішнього моменту) до тих пір, поки *B* не стане істинним в деякий момент в майбутньому. (*G, F* і *H*, *P* подвійні оператори в сенсі $FA \equiv \lceil G \rceil A$ і $PA \equiv \lceil H \rceil A$. Деякі часові логіки враховують тільки майбутнє. В цьому випадку часто вживають позначення • і ◊ відповідно для операторів *G* і *F*).

Епістемічні оператори:

– *вірить*(*x*): *вірить*(*x*) *A* – істинно, якщо індивід *x* вірить у формулу *A*.

Очевидно, що модальні формули незліченні, філософи запропонували і використали їх у величезній кількості. Всі ці формули можуть бути представлені за допомогою модальних операторів.

8.1.3 Синтаксис модальної логіки предикатів

Хай $M_1, M_2, ..., M_n$ – модальні оператори. Правила утворення модальних формул такі:

– Всі правила побудови з логіки предикатів (першого порядку) є також правилами побудови в модальній логіці предикатів.

– Якщо *F* – формула і M_j – модальний оператор, то M_jF – формула.

І знову основним завданням представлення знань є переклад фраз або описів, що відносяться до області експертизи, у формули модальної логіки

предикатів. Семантичне значення цих модальних формул повинне відповідати істинам області експертизи.

Вербально: Можливо, що Петро посилає книгу Марії

Логічно: ◊*Посилка* (*Петро_2, Марія_4, Книга_22*).

Вербально: Не було можливим, щоб Петро посилав що-небудь кожному,

Логічно: *P*(⌉ (◊ (∀ x ∃ *y* ∃ *z Відправник* (*z, Петро_2*) ∧ *Одержувач* (*z, x*) ∧ *Об'єкт* (*z, y*) ∧ *Елем* (*z, посилки*)))).

(Словесне прочитання цієї формули таке: у минулому (*P*) не є (⌉) можливим (◊), щоб Петро посилав що-небудь кожному).

Вербально: Якщо Петро вірить даному вислову, то він вірить, що вірить йому (аксіома позитивної інтроспекції)

Логічно: *вірить* (*Петро_2*) *p* ⊃ *вірить* (*Петро_2*) (*вірить* (*Жак_2*) *p*).

Вербально: Якщо Петро не вірить даному вислову, то він вірить, що не вірить йому (аксіома негативної інтроспекції)

Логічно: ⌉ *вірить* (*Петро_2*) *p* ⊃ *вірить* (*Петро_2*) (⌉ *вірить* (*Петро_2*) *p*).

Вербально: Якщо факт того, що Петро вірить даному вислову, значить, що воно істинно, то Петро знає цей вислів

Логічно: (*вірить* (*Петро_2*) *p* ⊃ *p*) ⊃ (*знає* (*Петро_2*)*p*). (У багатьох випадках бажано, щоб переконання людей були істинними. Це приводить до поняття швидше знання*, чим* віра.)

Вербально: Петро вірить, що він послав книгу (а не що-небудь інше) Марії.

Логічно: *вірить*(*Петро_2*)(*P* ∃ *x* [*Посилка*(*Петро_2, Марія_4, x*) ∧ *Елем*(*x, книги*)]).

Вербально: Петро вірить, що він послав книгу (цілком визначену) Марії.

Логічно: ∃ *x* [*вірить*(*Петро_2*)(*P посилка*(*Петро_2, Марія_4, x*)) ∧ *Елем* (*x, книги*)].

(Останній вираз показує, що в системі переконань Петра є формула типу [*P Посилка* (*Петро_2, Марія_4, x*) ∧ *Елем* (*x, книги*)]. Але її остаточний вигляд залежить від конкретного об'єкту, до якого відноситься змінна *x*. Передостанній вираз показує, що в системі переконань Петра міститься формула (P ∃ *x Посилка* ∧ *Елем*).)

8.2 Тризначна семантика для модальної логіки предикатів

Семантику для модальної логіки предикатів можна визначити, як для класичної. Простоти ради тимчасово проігноруємо складніші мови, засновані на модальній логіці. Проілюструємо модальну семантику, ввівши апроксимацію деяких форм логіки можливого за допомогою тризначної логіки. Бінарна логіка з двома значеннями {Н або 0, І або 1} сама елементарна. Істина і неправда – це дві множини висловів, і закони (класичної) логіки стверджують, що будь-який вислів – є елементом хоч би однієї з цих множин (закон виключеного третього) і що ніякий вислів не є елементом відразу двох цих множин (закон суперечності).

Які зміни треба внести в цю теорію, якщо вводяться модальності

«можливо» і «необхідно»? Треба розглянути декілька класів висловів. Позначимо через *N* клас «необхідних» висловів, через *P* – «можливих», через *I* – «неможливих» (або «абсурдних») і через *C* – «нейтральних» (або «можливо (випадково) помилкових») Ніякий вислів не належить одночасно *N* і *C* або *I* і *P.* Це закон суперечності. Далі, клас *N* міститься в *P,* а клас *I* – в *C*. Це відбито в законах походження можливого з необхідного і нейтрального з абсурдного:

будь-який необхідний вислів можливий,

будь-який абсурдний вислів не є необхідним.

Існують вислови, які є можливими і нейтральними одночасно. Їх називають «проблематичними». Множину таких висловів позначимо через *U.* Має місце закон виключеного четвертого:

будь-який вислів належить або H або U або I.

Подивимося, як цю теорію модальності можна перевести у форму, алгебри. Кожному з класів

H, U, I відповідає своя інтерпретація: «необхідно», «проблематично», «неможливо». Візьмемо три символи 2, 1, 0. Це логічні значення, що зіставляються вказаним інтерпретаціям. Кожному вислову можна приписати логічне значення. Ця тризначна логіка запропонована Лукасевичем.

Таблиця 8.1 – Таблиця тризначних функцій (зв'язки Лукасевича) та модальних операторів

F	⎤*F*	*F* ∧ *G*			*F* ∨ *G*			*F* ⊃ *G*			*F* ≡ *G*			◊*F*	€*F*
		G			*G*			*G*			*G*				
		0	1	2	0	1	2	0	1	2	0	1	2		
0	2	0	0	0	0	1	2	2	2	2	2	1	0	0	0
1	1	0	1	1	1	1	2	1	2	2	1	2	1	2	0
2	0	0	1	2	2	2	2	0	1	2	0	1	2	2	2

У логіці Лукасевича кожен вислів володіє одним із значень 0, 1 або 2 (що інтерпретуються як семантичне значення вислову). Семантичне значення модальної формули можна знаходити, використовуючи таблиці, приведені табл. 8.1. Вони задають семантику для апроксимації модальної логіки можливого за допомогою тризначної логіки.

8.3 Семантика можливих світів

Множину дійсних подій можна розділити на «випадково істинні» (які істинні, але могли б виявитися помилковими, якби розвиток світу йшов інакше; такими є історичні події) і «необхідно істинні» (які не можна заперечувати, не ставлячи під сумнів нормальний сенс слів мови, такими є математичні і логічні істини). Так само хибні події ділять на «випадково помилкові» і «необхідно помилкові». Будемо вважати, що «необхідна істина» (у існуючому світі) пов'язана з подією, що підтверджується не тільки в існуючому, але і у всіх можливих світах. Таким же чином «можлива істина» буде пов'язана з подією, що підтвердилася в снуючому світі, але не у всіх можливих світах. «Необхідна

помилкова» подія не підтверджується ні в якому з можливих світів, тоді як «випадково помилкова» помилкова в існуючому світі, але не у всіх можливих світах. Це ділення подій можна зобразити у вигляді діаграми (див. рис. 8.1).

Інтерпретаціям *необхідно істинно, нейтрально істинно, нейтрально помилково, необхідно помилково* можна зіставити відповідно логічні значення 3, 2, 1, 0. Отже, модальна логіка, пов'язана з операторами необхідно і випадкове, виявилася зануреною в чотиризначну логіку.

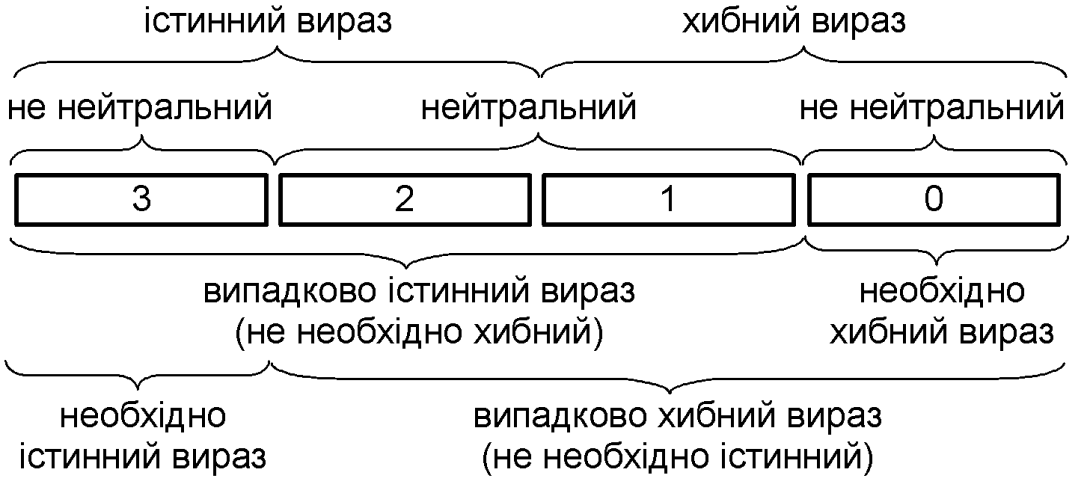


Рисунок 8.1 – Семантики можливих світів

Припустимо, що семантика можливих світів зводиться до семантики з двома можливостями: існуючий світ *X* і можливий *У*. Кожному вислову припишемо одне з логічних значень 0, 1, 2 або 3 відповідно до того, якій з умов, що приводяться нижче, воно задовольняє:

– помилково в *X* і в У (необхідно помилково),

– помилково в *X* і істинно в У (випадково, але не необхідно помилково)

– істинно в *X* і помилково в *У* (випадково, але не необхідно істинно)

– істинно в *X* і в У (необхідно істинно).

Вказана інтерпретація логічних значень приводить до таблиць (див. табл. 8.2) для заперечення, кон'юнкції і операторів • і ◊. Це задає деяку модальну семантику для чотиризначної логіки.

Таблиця 8.2 – Чотиризначні логічні операції і модальні оператори

F	⎤*F*	*F* ∧ *G*				◊*F*	€*F*
		G					
		0	1	2	3		
0	3	0	0	0	0	0	0
1	2	0	1	0	1	3	0
2	1	0	0	2	2	3	0
3	0	0	1	2	3	3	3

Відмітимо, що оператори заперечення і кон'юнкції, описані в табл.8.2 є

прямим добутком двох екземплярів аналогічних операторів з бінарної системи {0, 1}. Це легко перевіряється з використанням наступного двійкового кодування логічних значень 0, 1, 2, 3:

0=(0,0), 1=(0, 1), 2=(1,0), 3=(1, 1).

8.4 Ламбда-числення

Збагатимо модальне числення предикатів новим оператором над логічними змінними – *оператором ламбди* (або, коротше, λ-оператором). Існує такий спосіб представлення множин

{*x*| *x* є викладачем університету}

{*x*| –1 < *x* < 1, *x* – дійсне число},

При такому визначенні конкретної множини двозначності немає; потрібно тільки явно вказати ту змінну, на якій засноване визначення множини. Можна пошукати заміну неформального визначення формально логічним. Це приводить до позначень λ-*числення,* в якому множина описується таким чином:

λ*x* (логічна формула, що містить змінну *x*).

Наприклад, розглянемо фразу «Хтось посилає щось Марії» і її переклад на мову логіки предикатів *Посилка*(*x, Марія_4, y*).

Вираз (називається λ-*виразом або* λ-абстракцією)

λ*y* [*Посилка*(Петро*_2, Марія_4, y*)] (8.1)

визначає характеристичну функцію множини об'єктів, що послані Петром Марії, тоді як вираз

λ*x* [*Посилка* (*x, Марія_4, Книга_22*)] (8.2)

визначає характеристичну функцію множини індивідів, що посилають вказану книгу Марії. Так само вираз

λ*x, y* [*Посилка*(*x, Марія_4, y*)] (8.3)

визначає характеристичну функцію множини пар (відправник, посланий об'єкт), для яких Марія є, одержувачем.

Якщо *F* – формула, то λ*xF* вказує на описану за допомогою *F* множину, на елементах якої (що є конкретизаціями для *x*) *F* істинна. Вирази (8.1) і (8.2) можна інтерпретувати і як унарні предикати з аргументами *y* і *x* відповідно. Оператор λ, як і квантори узагальнення і існування, служить для скріплення входжень змінної.

Дамо більш формалізоване визначення λ-виразам.

n-арний λ-*вираз* λx_1, ..., x_n*F складається з логічної формули F, що називається тілом виразу, і множини змінних* x_1, ..., x_n *формули F, що є формальними параметрами виразу.*

Наступний за λ *список параметрів служить для того, щоб відрізнити формальні параметри від інших аргументів* (*або концептів*) *формули F.*

Тілом λ-виразу є формула логіки предикатів. Як тільки будуть задані значення *n* формальних параметрів, λ-вираз стане *n-арним* предикатом, що приймає значення «істина» або «помилка» залежно від того, які індивідуальні значення (конкретизації) привласнені формальним параметрам.

Синтаксис і семантика λ-виразу виникає з його визначення.

Синтаксис *Якщо F – логічна формула і x – змінна, то* λ*xF –* λ*-вираз.* Семантика *Значення* λ*-виразу* $\lambda x_1,...,x_n F$ *є значення n-арного предиката, одержаного привласненням конкретизації формальним параметрам.*

Вираз (8.3) стає бінарним предикатом, який можна позначити, наприклад, *Ламбда-посилка*(*x, y*). Він прийме значення І або Х залежно від здійсненого привласнення конкретизації змінним *x* і *y.* У деяких системах баз даних λ-вирази використовуються для постановки питань. Вираз (8.3) відповідав би питанню (запиту): «Назвіть послані до Марії об'єкти і відповідних відправників».

9 ЛОГІЧНЕ ВИВЕДЕННЯ ЗА НЕДОСТОВІРНИХ ЗНАНЬ

9.1 Поняття про неточне логічне виведення

Проблема недостовірних знань була в загальних рисах охарактеризована в попередньому розділі. Часто буває так, що експерт не зовсім упевнений в тому чи іншому факті, але, незважаючи на це, інформація залишається цінною і повинна бути включена до бази знань.

Висловлення називається ***неточним,*** якщо його істинність або хибність не можуть бути встановлені однозначно, тобто твердження не є ні абсолютно достовірним, ні абсолютно хибним.

Неточним виведенням *називається логічне виведення в умовах неточності* (*недостовірності*) *знань.*

Неточне виведення традиційно розглядається як самостійний напрям, хоча неточне твердження можна було б інтерпретувати як частковий випадок модального твердження.

Неточне виведення слід відрізняти від роботи з нечіткими знаннями, [x]оча останні також формально підпадають під сформульоване вище визначення.

9.2 Деякі визначення з теорії ймовірностей

Багато методик недостовірного логічного виведення тісно пов'язано з апаратом теорії ймовірностей. Тому для розуміння цього розділу потрібно навести основні необхідні факти з теорії ймовірностей.

Ймовірність події A (позначається $P(A)$) слід розуміти як міру достовірності деякої події. Завжди виконується властивість $0<P(A)<1$. При цьому *неможлива подія* (подія, яка ніколи не може відбутися) має ймовірність 0, а *достовірна подія* – ймовірність 1. Дамо формалізоване визначення.

Розглянемо *множину елементарних наслідків Q*. Елементарні наслідки розглядаються як можливі наслідки деякого експерименту, при цьому ці наслідки вважаються *рівноможливими та взаємовиключними***.** Тоді будь-яка подія A розглядається як підмножина множини елементарних наслідків. Подія A відбувається, якщо має місце будь-який елементарний наслідок, що входить до множини A. Якщо елементарний наслідок входить до множини A, кажуть, що він є *сприятливим* для події A.

Ймовірність події A визначається як відношення кількості сприятливих наслідків до загальної кількості елементарних наслідків.

На основі даного визначення в ряді випадків можна безпосередньо розрахувати ймовірності. Наведемо кілька прикладів.

Приклад. Підкидається гральний кубик, на кожній грані якого міститься певна кількість очок від 1 до 6. Яка ймовірність того, що випаде парна кількість очок?

Оскільки в результаті експерименту може випасти будь-яка кількість очок від 1 до 6, множина елементарних наслідків $Q = \{1, 2, 3, 4, 5, 6\}$.

Сприятливими для події A є наслідки 2, 4, 6, отже, $A = \{2, 4, 6\}$.

Маємо $P(A) = 3/6 = 0.5$.

Приклад. Підкидаються два гральних кубики. Яка ймовірність того, що сума очок, які випадуть на обох гранях, дорівнюватиме 5?

Експеримент полягає в підкиданні двох кубиків, і елементарний наслідок є парою, кожний елемент якої задає кількість очок, що випала на відповідному кубику. Таким чином,

Q = {11, 12, 13, 14, 15, 16, 21, ..., 61, 62, 63, 65, 66}, кількість елементарних наслідків дорівнює 36.

A = {14, 41,23, 32}.

Тоді $P(A) = 4/36 = 1/9 = 0.11$.

Подією, *протилежною* до події A (позначається $\overline{A}$), називається та, яка полягає в тому, що подія A не відбувається. На основі класичного визначення ймовірностей легко переконатися в тому, що

$P(\overline{A}) = 1 - P(A)$

Ймовірність події, по суті, є частотою появи цієї події. Це підкреслюється в такому визначенні.

Частотне визначення ймовірності. Нехай проводиться серія однотипних експериментів і нехай $K_A(n)$ – кількість випадків, в яких відбулася подія A, якщо було проведено n експериментів. Тоді

$$P(A) = \lim_{n\to\infty} \frac{K_A(n)}{n}.$$

Наприклад, якщо проводиться 1000 експериментів, а ймовірність події A дорівнює 0.3, то подія A настане приблизно в 300 випадках. При цьому зі зростанням кількості експериментів частина випадків, в яких відбувається дана подія, все більше наближається до ймовірності.

Введемо такі позначення:

$P(A \cup B)$ – ймовірність того, що настане або подія A, або подія B, або *і* A, *і* B;

$P(A \cap B)$ – ймовірність того, що настануть і A, *і* B.

Тоді $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

Зокрема, якщо A і B – несумісні (взаємовиключні) події, маємо $P(A \cup B) = P(A) + P(B)$.

Дуже важливим є поняття *умовної ймовірності.* Умовною ймовірністю події A за умови B (позначається $P(A\backslash B)$ називається ймовірність появи A за умови, що подія B уже відбулася.

Справедливе співвідношення

$P(A \cap B) = P(B) \cdot P(A\backslash B)$

Події A і B називаються *незалежними,* якщо поява A жодним чином не залежить від появи B. Інакше кажучи, A і B незалежні, якщо $P(A\backslash B) = P(A)$.

Кажуть, що події $H_1,..., H_n$ утворюють *повну групу подій,* якщо вони є взаємовиключними і сума їх ймовірностей дорівнює 1 (це означає, що з цих подій настане одна і тільки одна).

Якщо $H_1,...,H_n$ – повна група подій, то для будь-якої події A справедливі співвідношення

$$P(A) = \sum_{i=1}^{n} P(H_i) \cdot P(A \setminus H_i)$$

(формула повної ймовірності) та

$$P(H_i \setminus A) = \frac{P(H_i)P(A \setminus H_i)}{\sum_{i=1}^{n} P(H_i) \cdot P(A \setminus H_i)}$$

Остання формула називається *формулою Байєса* і має дуже велике значення, оскільки дозволяє обчислювати *апостеріорні* ймовірності $P(H_i \setminus A)$ гіпотез H_i за умови, що подія A відбулася, через апріорні ймовірності $P(A \setminus H_i)$. Більшість методик неточного логічного виведення так чи інакше пов'язані з формулою повної ймовірності та формулою Байєса.

При неточному логічному виведенні з кожним твердженням (фактом або правилом виведення) пов'язується число, яке характеризує міру його надійності. Ця характеристика називається *коефіцієнтом упевненості,* або *мірою достовірності.* Розрахунок коефіцієнтів упевненості тісно пов'язаний з ймовірнісними методами, хоча ненадійність інформації далеко не завжди носить імовірнісний характер.

Коефіцієнти упевненості часто визначаються на основі експертних оцінок. Проте в ряді випадків коефіцієнт упевненості можна отримати шляхом статистичних досліджень. Якщо ж здійснюється логічне виведення, то коефіцієнт упевненості нерідко можна підрахувати, якщо відомі міри достовірності умови та правила виведення.

Чим надійнішою є інформація, тим вищі відповідні коефіцієнти упевненості. Часто використовується шкала, за якої коефіцієнт 1 відповідає достовірно істинній події, 0 – достовірно хибній, а 0.5 – повній невизначеності.

Але згадана шкала не є єдино можливою. Так, експертні системи MYCIN та EMYCIN, що є розвитком MYCIN (ці системи призначені для виявлення мікроорганізмів у крові) використовують шкалу, в якій достовірній істинності відповідає значення 1, достовірній хибності – значення “–1”, а повній невизначеності – 0.

9.3 "Об'єктивна" та "суб'єктивна" невизначеність

Перелічимо деякі з можливих типів невизначеності.

«Об'єктивна» невизначеність. Цей тип пов'язаний з принципово випадковим характером процесів, які прогнозуються. Наприклад, передбачається результат перебігу певної хвороби, а статистичний аналіз свідчить, що досі ефективність лікування становила 70 %. У такому разі робиться відповідний висновок: пацієнт буде вилікований з імовірністю 70 %. Важливо розуміти, що «об'єктивна» невизначеність, як правило, пов'язана з впливом певних невідомих або неконтрольованих факторів і за своєю суттю являє собою ймовірність відповідної події.

Можна виокремити дві основні властивості "об'єктивної" невизначеності:

– подія, для якої визначається коефіцієнт упевненості, ще не відбулася,

але обов'язково настане момент, коли вона або відбудеться, або не відбудеться; після цього міра достовірності стане дорівнювати відповідно 1 або 0;

– якби було можливим провести серію експериментів за однакових умов, то частина експериментів, в яких подія відбулася б, приблизно дорівнювала б *a*; тут *a* – міра достовірності (фактично, ймовірність події).

«Суб'єктивна» невизначеність. Ми говоримо про цей тип невизначеності у разі, якщо подія, про яку йдеться, або вже відбулася, або не відбулася, і істинна міра достовірності дорівнює відповідно або 1, або 0. Але, якщо ця істинна міра достовірності експертові невідома (він не знає, чи відбулася подія, чи ні) або відома нечітко, то він дає власну оцінку міри надійності, і ця власна оцінка неминуче є суб'єктивною. Зрозуміло, що доцільність застосування ймовірнісних методів у такому разі є досить сумнівною.

Варто розрізняти два типи «суб'єктивної» невизначеності.

«Суб'єктивна» невизначеність при авторитетності джерела знань. Авторитетність джерела знань означає, що експерт є людиною серйозною і відповідальною та його власній оцінці можна довіряти. Тому власна оцінка істинності твердження приймається як робоча і використовується при логічному виведенні. Авторитетними, як правило, є серйозні наукові праці, енциклопедії тощо.

«Суб 'єктивна» невизначеність при неавторитетності джерела знань. У даному разі, незважаючи на ступінь упевненості «експерта», його оцінці довіряти не можна. Тут можна розглядати різноманітні випадки, наприклад:

– твердження є об'єктивно істинним, але «експерт» у ньому невпевнений і ступінь його кваліфікації не дозволяє йому адекватно сформулювати міру невпевненості;

– твердження є об'єктивно хибним, але «експерт» вважає його істинним і формулює як достовірне;

– твердження є об'єктивно хибним, «експерт» усвідомлює його хибність, але повідомляє про істинність твердження з метою ввести в оману.

Часто доводиться мати справу з *невизначеністю комбінованого типу,* тобто з невизначеністю, що поєднує риси як «об'єктивної», так і «суб'єктивної» невизначеності. Один з найпростіших випадків подібного поєднання полягає в такому. Розглядається твердження *«Завтра відбудеться подія А».* Існує істинна, «об'єктивна» ймовірність цієї події. Але експертові ця істинна оцінка ймовірності може бути або невідомою, або відомою неточно. Тоді він дає власну оцінку вказаної ймовірності, і ця власна оцінка має суб'єктивний характер.

Слід також звернути увагу на таке явище, як *конфлікт між різними джерелами знань.* Різні джерела знань можуть суперечити одне одному; наприклад, те, що людина прочитала в книзі, суперечить її знанням, або різні наукові школи приходять до протилежних висновків стосовно одного й того самого явища.

Принцип індиферентності є одним з найпростіших правил, яке допомагає приймати рішення в умовах невизначеності. Він формулюється так: якщо є *n* гіпотез і немає жодних свідчень на користь будь-якої з цих гіпотез, то міра

достовірності кожної гіпотези приймається за *1/n.*

Бездумне застосування принципу індиферентності може привести як до нерозумних висновків, так і до логічних помилок. Розглянемо простий приклад. Нехай є дві гіпотези: *«Під даним деревом є скарб»* і *«Під даним деревом немає скарбу».* Згідно з принципом індиферентності міру достовірності кожної з цих гіпотез слід прийняти за 50 %. Але 50 шансів зі ста – це дуже серйозна підстава для того, щоб почати шукати скарб під деревом. Зрозуміло, що такий висновок суперечить здоровому глузду.

9.4 Загальні принципи неточного виведення

Є два типи механізмів роботи з неточними твердженнями.

Неточне виведення «приєднаного» типу характеризується тим, що з кожним твердженням x пов'язується міра його достовірності $\gamma(x)$. Логічне виведення здійснюється за принципами, характерними для точних знань, але при цьому висновкам також приписується певна міра достовірності. При цьому необхідно задати:

– *функцію* $\gamma(x) = f(\gamma(x_1), ..., \gamma(x_n))$, яка задає міру неточності складного твердження x, якщо задані міри неточності його складових частин $x_1,...,x_n$. Наприклад, є два висловлення: x_1= *«завтра буде дощ»* з мірою достовірності $\gamma(x_1) = 0.3$ і x_2= *«завтра буде сніг»* з мірою достовірності $\gamma(x_2)=0.6$. З цих двох тверджень можна утворити ряд складених тверджень, наприклад: $x = x_1 \vee x_2$ (*«завтра буде або дощ, або сніг»*). Функція розрахунків неточності для складених тверджень може задаватись по-різному, наприклад, типовим є використання функцій $\gamma(x) = \max(\gamma(x_1), ..., \gamma(x_n))$ для диз'юнкції та $\gamma(x) = \min(\gamma(x_1), ..., \gamma(x_n))$ – для кон'юнкції. Тоді у нашому випадку $\gamma(x) = \max(\gamma(x_1), \gamma(x_2)) = \max(0.3, 0.6) = 0.6$;

– *функцію* $\gamma(y) = g(\gamma(x), \gamma(r))$; ця функція задає міру неточності висновку y, якщо задані міри неточності умови x та правила виведення r. Наприклад, маємо правило *r: «якщо завтра будуть опади, людина бере парасольку».* Нехай міра достовірності $\gamma(r)$ цього правила дорівнює 0.8, змістовно це можна інтерпретувати так: якщо очікуються опади, люди беруть парасольку у 80 випадках зі 100. Нехай міра достовірності умови (*«завтра будуть опади»*) $\gamma(x) = 0.6$. Для розрахунку міри достовірності висновку (*«певна людина візьме парасольку»*) можна вводити різні функції; типовим є використання добутку: $g(\gamma(x), \gamma(r)) = \gamma(x) \cdot \gamma(r)$. Тоді у нашому випадку $\gamma(y) = 0.6 \cdot 0.8 = 0.48$;

– *функцію комбінування свідоцтв* $\gamma(y) = h(\gamma_1(y), ..., \gamma_n(y))$, де $\gamma_i(y) = g(\gamma(x), \gamma(g_i))$, що означає: якщо існує кілька свідоцтв на користь (або проти) певного твердження, і кожне з цих свідоцтв приводить до певної міри істинності висновку, то на підставі цих мір потрібно побудувати деяку узагальнену міру. Формальніше, якщо для твердження y існує n правил виведення r_i типу "Якщо x_i то y“, і кожному з цих правил виведення приписані міри достовірності $\gamma(r_i)$, і кожна умова має свою міру достовірності $\gamma(x_i)$, то ми отримуємо n мір достовірності для y: $\gamma_i(y) = g(\gamma(x_i), \gamma(r_i))$. Функція

комбінування свідоцтв дозволяє отримати одну об'єднану міру достовірності. Комбінування свідоцтв є центральною проблемою неточного виведення.

Для кожної конкретної методики неточного логічного виведення при недостовірних знаннях потрібно визначити конкретний вигляд функцій *f*, *g* та *h*.

Таких методик запропоновано досить багато; чимало з них створювалося для використання в конкретних експертних системах.

Як зазначалося раніше, в основі більшої частини цих методик лежить апарат теорії ймовірностей. Усі вони мають багато спільних рис. Необхідно зауважити, що жодну з цих методик не можна вважати загальною та універсальною. Багато проблем виникає навіть у разі «об'єктивної» невизначеності, для якої застосування ймовірнісних методів дає найкращі результати (деякі з цих проблем ми детально розглянемо нижче).

У разі «суб'єктивної» невизначеності ситуація виявляється ще гіршою. Тому переважна більшість методів неточного логічного виведення не має належної теоретичної бази і носить відверто евристичний характер.

Теорія неточного логічного виведення на сучасному етапі інтенсивно розвивається, і цілком вірогідною є поява нових, досконаліших методик.

Механізми неточного виведення другого типу передбачають наявність спеціальних схем виведення, орієнтованих на схему представлення неточності. Надалі ми розглядатимемо лише виведення «приєднаного» типу, характерне для продукційних систем.

9.4.1Точкові та інтервальні міри неточності

Міри неточності тверджень можуть бути *точковими* та *інтервальними.* У разі точкового оцінювання з кожним твердженням пов'язується єдине число, яке задає міру його достовірності. Натомість за інтервального оцінювання міра достовірності задається певним інтервалом.

Інтервальні міри достовірності в цілому є надійнішими. Наприклад, дехто каже: «*Міра достовірності того, що команда А виграє баскетбольний матчу команди В, дорівнює 0.5».* Вважаємо, що 0 відповідає достовірній хибності, а 1 – достовірній істинності. Можливі як мінімум дві інтерпретації такої відповіді.

У першому випадку «експерт2 може не розумітися на баскетболі. У такому разі його відповідь фактично означає: *«Не маю жодного уявлення, але згідно з принципом індиферентності – п 'ятдесят на п 'ятдесят».* Ясно, Що з такої «оцінки» немає ніякої практичної користі.

Зовсім інша ситуація виникає, якщо експерт детально проаналізував ситуацію і дійшов висновку, що команди абсолютно рівні за силою. Відповідь «*0.5»* у такому разі означає: «*Команди мають рівні шанси на перемогу. Результат залежить від випадку, тобто є об'єктивно невизначеним. Якби провести не один матч, а цілу серію, то у половині матчів виграла б команда А, а у половині — команда В».*

Застосування інтервальних оцінок дозволило б розділити ці дві ситуації. У першому випадку інтервал невизначеності дорівнював би [0, 1], а в другому – був би значно вужчим, наприклад [0.45, 0.55].

Звичайно, це стосується лише випадків, коли люди об'єктивно оцінюють

ситуацію та адекватно висловлюють міру своєї невпевненості. Ясно, що у першому випадку ніщо не заважає «експертові» оцінити інтервал невизначеності, наприклад, як [0.9, 1.0], але це вже зовсім інша тема.

9.4.2 Проблема комбінування свідоцтв

Як уже зазначалося, проблема комбінування свідоцтв є центральною проблемою неточного логічного виведення. Вона пов'язана з тим, що існує кілька свідоцтв, які говорять або на користь певного висновку, або проти нього. Як за таких умов оцінити міру достовірності висновку?

Приклад. Вважатимемо, що значення 1 відповідає достовірній істинності, а 0 – достовірній хибності.

Нехай маємо два правила:

Правило 1. *Якщо очікується дощ, Іванов візьме з собою парасольку.*

Правило 2*. Якщо у Іванова буде багато речей, він не візьме з собою парасольку.*

Нехай дощ очікується з імовірністю 0.8, тоді за правилом 1 можна дійти висновку, що Іванов візьме парасольку, з мірою достовірності 0.8.

Нехай про те, що у Іванова буде багато речей, відомо з мірою достовірності 0.95. Тоді за правилом 2 міра достовірності того, що Іванов не візьме парасольку, оцінюється як 0.95, а міра достовірності протилежного прогнозу (*«Іванов візьме парасольку»*) – лише як 0.05.

Отже, ми маємо суперечливі свідоцтва, які дають майже протилежні прогнози. Як їх комбінувати? У найпростішому випадку можна взяти середнє арифметичне від обох свідоцтв і оцінити міру достовірності як (0.8 + 0.05)/2 = 0.425. Але можна брати і складніші функції комбінування свідоцтв. Зокрема, ми можемо враховувати міру надійності джерел інформації, а також те, чого Іванов більше не любить: мокнути під дощем чи носити з собою багато речей.

Якщо необхідно комбінувати різні свідоцтва, зменшується роль точкових оцінок мір достовірності і зростає значення інтервальних.

9.4.3 Приклади застосування мір достовірності

Може виникнути запитання: а що дає введення міри достовірності? Яка, наприклад, різниця, як ми оцінимо міру достовірності деякої події: як 0.3 чи як 0.7?

Можна навести як мінімум три ситуації, в яких більш-менш адекватна оцінка міри достовірності має велике практичне значення.

1. «Об'єктивна» невизначеність; статистичний характер явищ, що досліджуються. Нехай ми збираємося провести серію експериментів і оцінюємо успішність окремого експерименту з певною мірою достовірності. Тоді, якщо ця оцінка адекватна, ми можемо відразу спрогнозувати процент успіхів у серії експериментів. Наприклад, якщо міра достовірності успіху дорівнює 0.85, а проводиться 1000 експериментів, то ми можемо сказати, що приблизно 850 з них завершаться успішно.

2. «Об'єктивна» невизначеність; чітка структурованість явищ, що

досліджуються. Нехай ми прогнозуємо деяке явище, яке залежить від певної кількості відносно контрольованих факторів (подібна ситуація виникає, наприклад, при прогнозі погоди, прогнозі соціально-економічних явищ і т. п.). Тоді, якщо ми знатимемо межі зміни кожного фактора, ми можемо більш-менш точно спрогнозувати і явище, яке нас цікавить.

3. Прийняття рішень в умовах ризику і невизначеності. Невизначеність при цьому може носити як об'єктивний, так і суб'єктивний характер.

Розглянемо типовий приклад, який пояснює ситуацію. Гравець на кінних перегонах (з точки зору теорії – особа, яка приймає рішення) може поставити 100 гривень на певного коня. Якщо кінь приходить до фінішу першим, гравець повертає свої 100 гривень і отримує додатковий виграш у 100 гривень. Якщо ж ні – гравець втрачає свої 100 гривень.

Яке ж рішення повинен прийняти гравець? Воно насамперед визначається мірою достовірності події. Якщо міра достовірності виграшу (об'єктивна чи суб'єктивна) оцінюється як 0.7, то гравець оцінює свої шанси на виграш як значні і має всі підстави зробити ставку. Якщо ж міра достовірності дорівнює 0.3, приймається протилежне рішення.

Точніше кажучи, такі міркування є повністю справедливими, якщо суб'єктивна оцінка гравця є адекватною, він має достатній капітал, і ситуація повторювана, тобто гравець може зробити повторну ставку ("повторити експеримент") достатню кількість разів. Нехай, наприклад, суб'єктивна міра достовірності виграшу (власна оцінка гравця) дорівнює 0.7. Якщо ця суб'єктивна оцінка адекватна, її можна вважати об'єктивною, і вона фактично дорівнює ймовірності виграшу. Тоді можна провести такий розрахунок. Якщо проводиться 1000 забігів, гравець виграє у середньому у 700 випадках і програє у середньому в 300 випадках. Тоді його сумарний виграш у середньому становитиме $(700 – 300) \cdot 1000 = 400\,000$ гривень.

Якщо ж ставку можна зробити лише один раз, тоді навіть за умови адекватності міри достовірності висновок стає менш однозначним. Якщо, наприклад, капітал гравця становить усього 125 гривень, а ввечері йому конче потрібно мати при собі 100 гривень, ставку не слід робити навіть за дуже високих шансів на виграш. Навпаки, якщо ввечері необхідно мати 200 гривень, є сенс ризикнути і зробити ставку навіть за дуже низьких шансів.

9.5 Формалізації мір ризику за неточного логічного виведення

З попереднього параграфа випливає, що практичним критерієм вірності оцінки мір достовірності з точки зору *особи, що приймає рішення,* повинен стати виграш у разі, якщо ці міри оцінюються вірно, та програш – у разі, якщо вони оцінюються невірно. Необхідна якась кількісна оцінка *функції виграшу,* яка визначає виграш при прийнятті вірного рішення та програш – при прийнятті невірних рішень. Можна, крім того, сказати, що задана тим чи іншим чином функція виграшу є також мірою ризику, який зумовлений невірними оцінками.

Одна з класичних мір ризику пов'язана з прийняттям рішень про те, відбудеться чи не відбудеться деяка подія. Можливі два типи рішень:

β_0 – приймається рішення про те, що подія відбудеться; β_1 – приймається рішення про те, що подія не відбудеться; Можливі дві ситуації:

α_0 – подія насправді відбувається;

α_1 – подія насправді не відбувається.

Розглядається *матриця виграшів* (або *матриця ризиків*)*:*

	α_0	α_1
β_0	c_{00}	c_{01}
β_1	c_{10}	c_{11}

Тут c_{ij} – виграш у разі, якщо приймається рішення β_i а фактично має місце ситуація α_j.

Коефіцієнти c_{ij} можуть бути додатними, нульовими або від'ємними. Додатному коефіцієнту відповідає фактичний виграш, а від'ємному – фактичний програш. Слід розрізняти абсолютні та відносні виграші та програші. Так, виграш за невірного рішення може бути додатним, але меншим порівняно з тим, яким він міг би бути при вірному рішенні, і тоді йтиметься про відносний програш.

Слід зазначити, що коефіцієнти c_{ij} також можуть бути відомими неточно.

Очевидно, в переважній $c_{00} \geq c_{10}$, $c_{01} \leq c_{11}$ більшості випадків виконується властивість (правильне рішення повинно збільшити виграш порівняно з неправильним).

Далі, якщо міра об'єктивної невизначеності (ймовірність) події дорівнює λ то ймовірність того, що подія не відбудеться, дорівнює $1 - \lambda$. Тоді очікувані виграші R_0 та R_1, при прийнятті рішень відповідно β_0 та β_1, можна оцінити як

$R_0 = \lambda\, c_{00} + (1 - \lambda)\, c_{01}$;

$R_1 = \lambda\, c_{10} + (1 - \lambda)\, c_{11}$;

Тоді слід прийняти те рішення β_i для якого відповідний виграш R_i набуває більшого значення. Інша можлива міра ризику пов'язана з оцінкою міри достовірності деякої події A (ця оцінка може носити суб'єктивний характер). Дану міру ризику можна задати функцією $g_A(\rho, \tau)$ – виграш у разі, якщо міра достовірності події A оцінюється як τ у той час, коли вона дорівнює ρ.

Видається доцільним висунути щодо введеної таким чином функції виграшу такі вимоги:

1) *Домінування вірних рішень:* для будь-яких ρ та τ виконується співвідношення

$g_A(\rho,\tau) \leq g_A(\rho,\rho)$.

2) *Монотонність:* для будь-яких ρ, τ_1 та τ_2 справедливим є твердження: якщо $\mu(\rho,\tau_1) < \mu(\rho,\tau_2)$, то $g_A(\rho,\tau_1) > g_A(\rho,\tau_2)$. Тут $\mu(\rho,\tau)$ – міра близькості (відстань) між ρ та τ. Змістовно це означає, що чим точніше ми оцінили істинне значення міри достовірності, тим на більший виграш ми можемо розраховувати.

Ясно, що з властивості 2) випливає властивість 1); зворотне невірне.

В цілому вимоги 1) та 2) є досить розумними та реалістичними, хоча можна навести багато прикладів, коли вони обидві не виконуються.

9.5.1 Деякі проблеми виведення

Нехай ми маємо продукційне правило $A \Rightarrow B$ (*якщо A, то B*) при цьому коефіцієнт упевненості цього правила дорівнює γ. З погляду теорії ймовірностей цей коефіцієнт упевненості можна проінтерпретувати як $P(B\backslash A)$ – умовну ймовірність B за умови A.

Нехай коефіцієнт упевненості твердження A дорівнює α. Тут ми розглядаємо коефіцієнти упевненості окремих тверджень як їх ймовірності.

Чому дорівнює $\beta = P(B)$ – коефіцієнт упевненості висновку B. Зразу ж необхідно сказати, *що ми не можемо обчислити* $P(B)$ *точно* – для цього не вистачає інформації. Натомість ми можемо обчислити інтервал, до якого потрапить ця ймовірність.

Очевидно, події A та $\overline{A}$ складають повну групу подій. Тоді відповідно до формули повної ймовірності маємо:

$$\beta = P(B) = P(A)\,P(B\backslash A) + P(\overline{A})\,P(B\backslash \overline{A}) = \alpha\gamma + (1-\alpha)\,P(B\backslash \overline{A})$$

У цій формулі фігурує невідоме значення $P(B\backslash \overline{A})$, і саме тому точне обчислення β не є можливим. Але, оскільки $0 < P(B\backslash \overline{A}) < 1$, маємо

$$\alpha\gamma \le P(B) \le \alpha\gamma + (1-\alpha)$$

$$\beta \in [\alpha\gamma,\ \alpha\gamma + (1-\alpha)]$$

Отже, інтервал невизначеності для висновку B є тим меншим, чим більшим є коефіцієнт упевненості умови A. Якщо $\alpha = 1$, β визначається точно.

Якщо ж $\alpha = 0$, інтервал невизначеності для β становить [0, 1], а це еквівалентно повній відсутності будь-якої корисної інформації.

Ми бачимо, що навіть у найпростіших випадках пряме застосування теоретико-ймовірнісних співвідношень спричиняє проблеми. Ситуація ще більше ускладнюється, якщо невизначеність носить «суб'єктивний2 характер. Тому необхідно мати наближені, але простіші методики обчислення коефіцієнтів упевненості, які у більшості випадків давали б прийнятний результат.

9.5.2 Схема EMYCIN

Однією з найпростіших і найдавніших схем неточного логічного виведення була схема MYCIN, розроблена для однойменної експертної системи, що визначала наявність мікроорганізмів у крові. Згодом вона була модифікована, і ця модифікація дістала назву EMYCIN. Схема EMYCIN добре себе зарекомендувала та широко використовується і сьогодні.

У схемі EMYCIN з кожним твердженням A пов'язується коефіцієнт впевненості p (A), який змінюється від –1 до 1. Значення –1 відповідає достовірній хибності, 0 – повній невизначеності, 1 – достовірній істинності. Хоча механізм EMYCIN тісно пов'язаний з ймовірнісними методами, зрозуміло, що введені таким чином коефіцієнти упевненості не є ймовірностями.

Якщо ми маємо правило $A \Rightarrow B$, коефіцієнт упевненості якого дорівнює $\rho(A \Rightarrow B)$, то *коефіцієнт упевненості висновку* дорівнює:

$$\rho(B) = \rho(A) \cdot \rho(A \Rightarrow B)$$

Коефіцієнт упевненості *заперечення* обчислюється за формулою:

$\rho(\bar{A}) = -\rho(A)$

Коефіцієнт упевненості *кон'юнкції* тверджень A та B обчислюється як мінімум коефіцієнтів упевненості окремих складових:

$\rho(A \wedge B) = \min(\rho(A), \rho(B))$.

Коефіцієнт упевненості *диз'юнкції* тверджень A та B можна обчислювати як максимум:

$\rho(A \vee B) = \max(\rho(A), \rho(B))$.

Але частіше правило $(A \vee B) \Rightarrow C$ замінюють двома правилами $A \Rightarrow C$, $B \Rightarrow C$ і далі застосовують формули комбінування свідоцтв, описані нижче.

Комбінування свідоцтв. Нехай ми маємо два правила зі спільною правою частиною: $A \Rightarrow C$, $B \Rightarrow C$. З використанням раніше наведених формул ми можемо розрахувати два коефіцієнти упевненості для висновку C. Нехай за першим правилом ми отримали коефіцієнт упевненості τ_1, а за другим – коефіцієнт упевненості τ_2. Як поєднати ці два коефіцієнти упевненості та отримати комбінований коефіцієнт τ?

Схема EMYCIN розрізняє три випадки:

Випадок 1: $\tau_1 \geq 0$; $\tau_2 \geq 0$. Це означає, що обидва правила свідчать в одному напрямі на користь висновку, і, таким чином, підсилюють одне одного. Тоді комбінований коефіцієнт розраховується за формулою:

$\tau = \tau_1 + \tau_2 - \tau_1\tau_2$

Випадок 2: $\tau_1 \leq 0$; $\tau_2 \leq 0$. Обидва правила свідчать проти висновку. Комбінований коефіцієнт розраховується за формулою

$\tau = \tau_1 + \tau_2 + \tau_1\tau_2$

Випадок 3: коефіцієнти різного знака. Маємо суперечність: одне правило свідчить на користь висновку, а інше – проти нього. Тоді комбінований коефіцієнт обчислюється за формулою:

$$\tau = \frac{\tau_1 + \tau_2}{1 - \min(|\tau_1|, |\tau_2|)}$$

(результат визначається вагомішим свідченням, але його вплив дещо послаблюється).

Сьогодні не існує абсолютно досконалого і загальноприйнятого механізму логічного виведення при недостовірних даних.

10 Нечітка логіка

Математична теорія нечітких множин (fuzzy sets) і нечітка логіка (fuzzy logic) є узагальненнями класичної теорії множин і класичної формальної логіки. Дані поняття були вперше запропоновані американським ученим Лотфі Заде (Lotfi Zadeh) в 1965 р. Основною причиною появи нової теорії стала наявність нечітких і наближених міркувань при описі людиною процесів, систем, об'єктів.

Перш ніж нечіткий підхід до моделювання складних систем отримав визнання у всьому світі, пройшло не одне десятиліття з моменту зародження теорії нечітких множин. І на цьому шляху розвитку нечітких систем прийнято виділяти три періоди.

Перший період (кінець 60-х – початок 70 рр.) характеризується розвитком теоретичного апарату нечітких множин (Л. Заді, Э. Мамдані, Беллман). У другому періоді (70–80-і роки) з'являються перші практичні результати в області нечіткого управління складними технічними системами (парогенератор з нечітким управлінням). Одночасно почала приділятися увага питанням побудови експертних систем, побудованих на нечіткій логіці, розробці нечітких контролерів. Нечіткі експертні системи для підтримки ухвалення рішень знаходять широке застосування в медицині і економіці. Нарешті, в третьому періоді, який триває з кінця 80-х років і продовжується в даний час, з'являються пакети програм для побудови нечітких експертних систем, а сфери застосування нечіткої логіки помітно розширюються. Вона застосовується в автомобільній, аерокосмічній і транспортній промисловості, в області виробів побутової техніки, у сфері фінансів, аналізу і ухвалення управлінських рішень і багато інших.

10.1 Математичний апарат

Характеристикою нечіткої множини виступає функція приналежності (Membership Function). Позначимо через $\mu_c(x)$ – ступінь приналежності до нечіткої множини C, що є узагальненням поняття характеристичної функції звичайної множини. Тоді нечіткою множиною C називається множина впорядкованих пар виду $C=\{\mu_c(x)/x\}$, $\mu_c(x) \in [0,1]$. Значення $\mu_c(x) = 0$ означає відсутність приналежності до множини, 1 – повну приналежність.

Проілюструємо це на простому прикладі. Формалізуємо неточне визначення «гарячий чай». Як x (область міркувань) виступатиме шкала температури в градусах Цельсія. Очевидно, що вона буде змінюється від 0 до 100 градусів. Нечітка множина для поняття «Гарячий чай» може виглядати таким чином:

C={0/0; 0/10; 0/20; 0,15/30; 0,30/40; 0,60/50; 0,80/60; 0,90/70; 1/80; 1/90; 1/100}.

Так, чай з температурою 60 °С належить до множини «Гарячий» із ступенем приналежності 0,80. Для однієї людини чай при температурі 60 °С може опинитися гарячим, для іншого – не дуже гарячим. Саме у цьому і виявляється нечіткість завдання відповідної множини.

Для нечітких множин, як і для звичайних, визначені основні логічні операції.

10.1.1 Операції над нечіткими множинами

Включення. Нехай A і B – нечіткі множини на універсальній множині E. Кажуть, що A міститься в B, якщо $\forall x \in E\ \mu_A(x) \cdot \mu_B(x)$.

Позначення: $A \subset B$.

Іноді використовують термін «домінування», тобто у разі коли $A \subset B$, кажуть, що B домінує A.

Рівність. A і B рівні, якщо $\forall x \in E\ \mu_A(x) = \mu_B(x)$.

Позначення: $A = B$.

Доповнення. Нехай $M = [0,1]$, A і B – нечіткі множини, задані на E. A і B доповнюють один одного, якщо

$\forall x \in E\ \mu_A(x) = 1 - \mu_B(x)$.

Позначення: $B = \overline{A}$ або $A = \overline{B}$.

Очевидно, що $A = \overline{\overline{A}}$. (Доповнення визначено для M = [0,1], але очевидно, що його можна визначити для будь-якого впорядкованого M).

Перетин. $A \cap B$ – найбільша нечітка підмножина, що міститься одночасно в A і B. $\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$.

Об'єднання. A ∪ B – найменше нечітка підмножина, що включає як A, так і B, з функцією приналежності: $\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$.

Різниця. $A - B = A \cap \overline{B}$ з функцією приналежності: $\mu_{A-B}(x) = \mu_{A\cap\overline{B}}(x) = \min(\mu_A(x), 1 - \mu_B(x))$

Диз'юнктивна сума. $A \oplus B = (A - B) \cup (B - A) = (A \cap \overline{B}) \cup (\overline{A} \cap B)$ з функцією приналежності: $\mu_{A\oplus B}(x) = \max[\min(\mu_A(x), 1 - \mu_B(x)), \min(1 - \mu_A(x), \mu_B(x))]$

Приклади.

Нехай:

$A = 0{,}4/x_1 + 0{,}2/x_2 + 0/x_3 + 1/x_4$;

$B = 0{,}7/x_1 + 0{,}9/x_2 + 0{,}1/x_3 + 1/x_4$;

$C = 0{,}1/x_1 + 1/x_2 + 0{,}2/x_3 + 0{,}9/x_4$.

Де, $A \subset B$, тобто A міститься в B або B домінує A, C непорівняно ні з A, ні з B, тобто пари $\{A, C\}$ і $\{B, C\}$ – пари недомінованих нечітких множин. $A \neq B \neq C$.

$\overline{A} = 0{,}6/x_1 + 0{,}8/x_2 + 1/x_3 + 0/x_4$;

$\overline{B} = 0{,}3/x_1 + 0{,}1/x_2 + 0{,}9/x_3 + 0/x_4$.

$A \cap B = 0{,}4/x_1 + 0{,}2/x_2 + 0/x_3 + 1/x_4$.

$A \cup B = 0{,}7/x_1 + 0{,}9/x_2 + 0{,}1/x_3 + 1/x_4$.

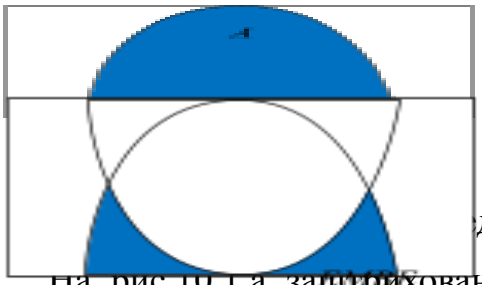
$A - B = A \cap \overline{B} = 0{,}3/x_1 + 0{,}1/x_2 + 0/x_3 + 0/x_4$;

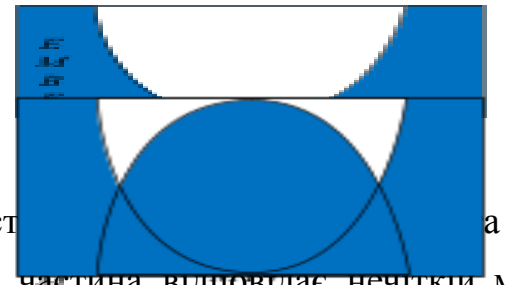
$B - A = \overline{A} \cap B = 0{,}6/x_1 + 0{,}8/x_2 + 0{,}1/x_3 + 0/x_4$.

$A \oplus B = 0{,}6/x_1 + 0{,}8/x_2 + 0{,}1/x_3 + 0/x_4$.

10.1.2 Наочне представлення операцій над нечіткими множинами

Для нечітких множин можна будувати візуальне подання. Розглянемо прямокутну систему координат, на осі ординат якої відкладаються значення $\mu_A(x)$, на осі абсцис в довільному порядку розташовані елементи E (ми вже використали таке подання в прикладах нечітких множин). Якщо E за своєю природою впорядковано, то цей порядок бажано зберегти в розташуванні елементів на осі абсцис. Таке уявлення робить наочними прості операції над нечіткими множинами.





дст а її доповнення

На рис.10.1,а заштрихована частина відповідає нечіткій множині A, б) відповідає $\overline{A}$, в) та г) відповідно $A \cap \overline{A}$ та $A \cup \overline{A}$.

10.1.3 Властивості операцій ⋂ і ⋃.

Нехай A, B, C – нечіткі множини, тоді виконуються наступні властивості:

$A \cap B = B \cap A$ та $A \cup B = B \cup A$ – комутативність

$(A \cap B) \cap C = A \cap (B \cap C)$ та $(A \cup B) \cup C = A \cup (B \cup C)$ – асоціативність;

$A \cap A = A$ та $A \cup A = A$ – ідемпотентність;

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ – дистрибутивність;

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ – дистрибутивність;

$A \cup \varnothing = A$, де $\varnothing$ – порожня множина, тобто $\mu_\varnothing(x) = 0 \ \forall x \in E$;

$A \cap \varnothing = \varnothing$;

$A \cap E = A$, де E – універсальна множина;

$A \cup E = E$;

$\overline{A \cup B} = \overline{A} \cap \overline{B}$, $\overline{A \cap B} = \overline{A} \cup \overline{B}$ – теореми де Моргана.

На відміну від чітких множин, для нечітких множин в загальному випадку: $A \cap \overline{A} \neq \varnothing$ та $A \cup \overline{A} \neq E$, (що, зокрема, проілюстровано вище в прикладі наочного представлення нечітких множин).

Зауваження. Введені вище операції над нечіткими множинами засновані на використанні операцій *max* та *min*. В теорії нечітких множин розробляються питання побудови узагальнених, параметрезованих операторів перетину, об'єднання і доповнення, що дозволяють врахувати різноманітні змістовні відтінки відповідних їм зв'язок «і», «або», «не».

Один з підходів до операторів перетину та об'єднання полягає у їх визначенні в класі трикутних норм і конорм.

Трикутної нормою (*t*-нормою) називається двомісна дійсна функція T, яка є бінарним відображенням: $[0,1] \times [0,1] \to [0,1]$, яка задовольняє таким умовам:

$T(0, 0) = 0$; $T(\mu_A, 1) = \mu_A$; $T(1, \mu_A) = \mu_A$ – обмеженість;

$T(\mu_A, \mu_B) \le T(\mu_C, \mu_D)$, якщо $\mu_A \le \mu_C$, $\mu_B \le \mu_D$ – монотонність;

$T(\mu_A, \mu_B) = T(\mu_B, \mu_A)$ – комутативність;

$T(\mu_A, T(\mu_B, \mu_C)) = T(T(\mu_A, \mu_B), \mu_C)$ – асоціативність;

Простим випадком трикутних норм є:

$\min(\mu_A, \mu_B)$

добуток $\mu_A \cdot \mu_B$

$\max(0, \mu_A + \mu_B - 1)$.

Трикутної конормой (*t*-конормой) називається двомісна дійсна функція *S* яка також є бінарним відображенням: $[0,1] \times [0,1] \to [0,1]$, з властивостями:

$S(1,1) = 1$; $S(\mu_A, 0) = \mu_A$; $S(0, \mu_A) = \mu_A$ – обмеженість;

$S(\mu_A, \mu_B) \ge S(\mu_C, \mu_D)$, якщо $\mu_A \ge \mu_C$, $\mu_B \ge \mu_D$ – монотонність;

$S(\mu_A, \mu_B) = S(\mu_B, \mu_A)$ – комутативність;

$S(\mu_A, S(\mu_B, \mu_C)) = S(S(\mu_A, \mu_B), \mu_C)$ – асоціативність;

Приклади t-конорм:

$\max(\mu_A, \mu_B)$

$\mu_A + \mu_B - \mu_A \cdot \mu_B$

$\min(1, \mu_A + \mu_B)$.

T-норма може розглядатися як узагальнений оператор AND, а *S*-норма – як узагальнений оператор OR.

10.1.4 Алгебраїчні операції над нечіткими множинами

Алгебраїчний добуток *A* і *B* позначається $A \cdot B$ і визначається так:

$\forall x \in E \; \mu_{A \cdot B}(x) = \mu_A(x)\, \mu_B(x)$.

Алгебраїчна сума цих множин позначається $A+B$ і визначається так:

$\forall x \in E \; \mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\, \mu_B(x)$.

Для операцій $\{\cdot, +\}$ виконуються властивості:

$A \cdot B = B \cdot A$, $A + B = B + A$ – комутативність;

$(A \cdot B) \cdot C = A \cdot (B \cdot C)$, $(A + B) + C = A + (B + C)$ – асоціативність;

$A \cdot \varnothing = \varnothing$, $A + \varnothing = A$, $A \cdot E = A$, $A + E = E$

$\overline{A \cdot B} = \overline{A} \cdot \overline{B}$, $\overline{A + B} = \overline{A} + \overline{B}$ – теореми де Моргана.

Не виконуються:

$A \cdot A = A$, $A + A = A$ – ідемпотентність;

$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$, $A + (B \cdot C) = (A + B) \cdot (A + C)$ – дистрибутивність;

а також $A \cdot \overline{A} = \varnothing, A + \overline{A} = E$.

Для прикладу доведемо, що властивість дистрибутивності не виконується, тобто $A \cdot (B + C) \ne (A \cdot B) + (A \cdot C)$. Позначимо $\mu_A(x)$ через a, $\mu_B(x)$ через b. Тоді в лівій частині маємо: а $(b + c - bc) = ab + ac - abc$; а в правій: $ab + ac - (ab)(ac) = ab + ac + a^2bc$. Це означає, що дистрибутивність не виконується оскільки $ab + ac - abc \ne ab + ac + a^2bc$.

Зауваження. При спільному використанні операцій $\{\cup, \cap, +, \cdot\}$ виконуються властивості:

$A \cdot (B \cup C) = (A \cdot B) \cup (A \cdot C)$;

$A \cdot (B \cap C) = (A \cdot B) \cap (A \cdot C)$;

$A + (B \cup C) = (A + B) \cup (A + C)$;

$A + (B \cap C) = (A + B) \cap (A + C)$.

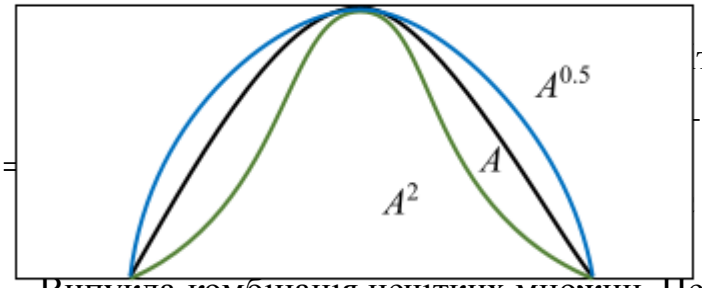
Продовжимо огляд основних операцій над нечіткими множинами.

На основі операції алгебраїчного добутку (принаймні для цілих α ця операція очевидна) визначається операція піднесення в степінь α нечіткої множини A, де α – позитивне число. Нечітка множина $A\alpha$ визначається функцією приналежності $\mu^{\alpha}_{A} = \mu^{\alpha}_{A}(x)$. Окремим випадком піднесення до степіню є:

$\mathrm{CON}(A) = A^2$ – операція концентрування,

$\mathrm{DIL}(A) = A^{0,5}$ – операція розтягування,

які використовуються при роботі з лінгвістичними невизначеностями.



ткої множини до степеня

α = позитивне число, таке, що

функцію приналежності:

Випукла комбінація нечітких множин. Нехай $A_1, A_2, .., A_n$ – нечіткі множини універсальної множини E, а $\omega_1, \omega_2, ..., \omega_n$ – невід'ємні числа, сума яких дорівнює 1.

Випуклою комбінацією $A_1, A_2, .., A_n$ називається нечітка множина A з функцією приналежності:

$$\forall x \in E\, \mu_A(x_1, x_2, ..., x_n) = \omega_1 \mu_{A_1}(x) + \omega_2 \mu_{A_2}(x) + ... + \omega_n \mu_{A_n}(x)$$

Декартовий добуток нечітких множин. Нехай $A_1, A_2, .., A_n$ – нечіткі підмножини універсальних множин $E_1, E_2, ..., E_n$ відповідно. Декартовий добуток $A = A_1 \times A_2 \times ... \times A_n$ є нечіткою підмножиною множини $E = E_1 \times E_2 \times ... \times E_n$ з функцією приналежності:

$$\mu_A(x_1, x_2, ..., x_n) = \min\{\mu_{A_1}(x), \mu_{A_2}(x), ..., \mu_{A_n}(x)\}$$

Оператор збільшення нечіткості використовується для перетворення чітких множин в нечіткі і для збільшення нечіткості нечіткої множини.

Нехай A – нечітка множина, E – універсальна множина і для всіх $x \in E$ визначені нечіткі множини $K(x)$. Сукупність усіх $K(x)$ називається ядром оператора збільшення нечіткості Φ. Результатом дії оператора Φ на нечітку множина A є нечітка множина виду:

$$\Phi(A, K) = \boxtimes_{x \in E} \mu_A(x) K(x),$$

де $\mu_A(x)K(x)$ – добуток числа на нечітку множину.

Приклад:

$E = \{1,2,3,4\}$;

$A = 0{,}8/1 + 0{,}6/2 + 0/3 + 0/4$;

$K(1) = 1/1 + 0{,}4/2$;

$K(2) = 1/2 + 0{,}4/1 + 0{,}4/3;$

$K(3) = 1/3 + 0{,}5/4;$

$K(4) = 1/4.$

Тоді $\Phi(A, K) = \mu_A(1)K(1) \cup \mu_A(2)K(2) \cup \mu_A(3)K(3) \cup \mu_A(4)K(4) =$

$= 0{,}8\ (1/1 + 0{,}4/2) \cup 0{,}6\ (1/2 + 0{,}4/1 + 0{,}4/3) = 0{,}8/1 + 0{,}6/2 + 0{,}24/3.$

Чітка множина α-рівня (або рівня α). Множиною α-рівня нечіткої множини A універсальної множини E називається чітке підмножина $A\alpha$ універсальної множини E, яка визначається наступним чином:

$A\alpha = \{x/\mu_A(x) \geq \alpha\}$, де $\alpha \leq 1$.

Приклад: $A = 0{,}2/x_1 + 0/x_2 + 0{,}5/x_3 + 1/x_4$,

тоді $A_{0.3} = \{x_3, x_4\}$, $A_{0.7} = \{x_4\}$.

Досить очевидне властивість: якщо $\alpha_1 \geq \alpha_2$, то $A_{\alpha 1} \leq A_{\alpha 2}$.

Теорема про декомпозиції. Будь-яка нечітка множина A може бути розкладена по його множинам рівня α у вигляді $A = \bigcup_{x \in M} \alpha A_\alpha$, де αA_α - добуток числа α на множину A, і α "пробігає" область значень M функції належності нечіткої множини A.

Приклад: $A = 0{,}1/x_1 + 0/x_2 + 0{,}7/x_3 + 1/x_4$ можна представити у вигляді:

$A = 0{,}1\ (1{,}0{,}1{,}1) \cup 0{,}7\ (0{,}0{,}1{,}1{,}) \cup 1\ (0{,}0{,}0{,}1) =$

$= (0{,}1/x_1 + 0/x_2 + 0{,}1/x_3 + 0{,}1/x_4) \cup (0/x_1 + 0/x_2 + 0{,}7/x_3 + 0{,}7/x_4) \cup$

$\cup\ (0/x_1 + 0/x_2 + 0/x_3 + 1/x_4) = 0{,}1/x_1 + 0/x_2 + 0{,}7/x_3 + 1/x_4.$

Якщо область значень функції приналежності складається з n градацій $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \ldots \leq \alpha_n$, то A (при фіксованих значеннях градацій) може бути представлена у вигляді:

$$A = \bigcup_{i=1}^{n} \alpha_i A_{\alpha_i},$$

тобто визначається сукупністю звичайних множин $\{A_{\alpha 1}, A_{\alpha 2}, \ldots, A_{\alpha i}\}$, де $A_{\alpha 1} \geq A_{\alpha 2} \geq, \ldots, \geq A_{\alpha i}$.

10.2 Нечітка та лінгвістичні змінні

Для опису нечітких множин вводяться поняття нечіткою і лінгвістичною змінних.

Кожна лінгвістична змінна описується наступним набором:

(N, T, E, G, P)

де N – назва лінгвістичної змінної;

T – множина значень, яка також називається базовою терм-множиною T. Елементами базової терм-множини є назви нечітких змінних;

X – універсальна множина (область міркувань);

G – синтаксичне правило, за яким генеруються нові терми із застосуванням слів природної або формальної мови;

P –·семантичне правило, яке кожному значенню лінгвістичної змінної ставить у відповідність нечітку підмножину множини X.

В свою чергу, нечітка змінна описується набором

(N, X, A),

де N – це назва змінною,

X – універсальна множина (область міркувань),

A – нечітка множина на X.

Розглянемо таке нечітке поняття як «Температура води на виході крану». Це і є назва лінгвістичною змінної. Сформуємо для неї базову терм-множина, яке складатиметься з п’яти нечітких змінних: {«дуже холодна», «холодна», «тепла», «гаряча», «дуже гаряча»} і задамо область міркувань у вигляді t=[0; 60] (градусів). Останнє, що залишилося зробити – побудувати функції приналежності для кожного лінгвістичного терма з базової терм-множини Т.

Існує понад десяток типових форм кривих для завдання функцій приналежності. Найбільшого поширення набули: трикутна, трапецеїдальна і гауссова функції приналежності.

Трикутна функція приналежності визначається трійкою чисел (a, b, c), і її значення в точці x обчислюється згідно виразу:

$$\mu(x) = \begin{cases} 1 - \dfrac{b-x}{b-a}, a \le x \le b \\ 1 - \dfrac{x-c}{c-b}, b \le x \le c \\ 0, \textit{в інших випадках} \end{cases}$$

При $(b-a) = (c-b)$ маємо випадок симетричної трикутної функції приналежності, яка може бути однозначно задана двома параметрами з трійки (a, b, c).

Аналогічно для завдання трапецеїдальній функції приналежності необхідна четвірка чисел (a, b, c, d):

$$\mu(x) = \begin{cases} 1 - \dfrac{b-x}{b-a}, a \le x \le b \\ 1, b \le x \le c \\ 1 - \dfrac{x-c}{d-c}, c \le x \le d \\ 0, \textit{в інших випадках} \end{cases}$$

При $(b-a) = (d-c)$ трапецеїдальна функція приналежності набирає симетричного вигляду.



а) б)

Рисунок 10.3 – Типові кусочно-лінійні функції приналежності.

Функція приналежності гаусова типу описується формулою

$$\mu(x) = \exp\left[-\left(\frac{x-c}{\sigma}\right)^2\right]$$

і оперує двома параметрами. Параметр c позначає центр нечіткої множини, а параметр σ відповідає за крутизну функції.

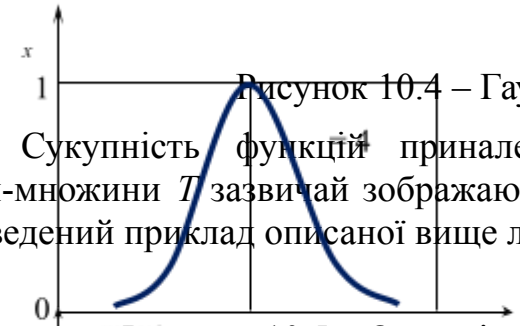


Рисунок 10.4 – Гаусова функція приналежності.

Сукупність функцій приналежності для кожного терма з базової терм-множини T зазвичай зображаються разом на одному графіку. На рис.10.5 приведений приклад описаної вище лінгвістичної змінної «Температура води».



Рисунок 10.5 – Опис лінгвістичної змінної «Температура води»

Кількість термів в лінгвістичній змінній рідко перевищує 7.

10.3 Нечіткий логічний вивід

Основою для проведення операції нечіткого логічного виводу є база правил, що містить нечіткі висловлювання у формі «ЯКЩО-ТО» і функції приналежності для відповідних лінгвістичних термів. При цьому повинні дотримуватися наступні умови (рис.10.6):

1. Існує хоч би одне правило для кожного лінгвістичного терма вихідний змінної.
2. Для будь-якого терма вхідної змінної є хоч би одне правило, в якому цей терм використовується як передумова (ліва частина правила).

Інакше має місце неповна база нечітких правил.

Хай в базі правил є m правил вигляду:

R_1: ЯКЩО x_1 це A_{11} . І . x_n це A_{1n}, ТО y це B_1

R_i: ЯКЩО x_1 це A_{i1} . І . x_n це A_{in}, ТО y це B_i

R_m: ЯКЩО x_1 це A_{i1} . І . x_n це A_{mn}, ТО y це B_m,

де x_k, k=1..n – вхідні змінні; y – вихідна змінна; A_{ik} – задані нечіткі множини з функціями приналежності.

Результатом нечіткого виводу є чітке значення змінної y* на основі заданих чітких значень x_k, $k = 1…n$.

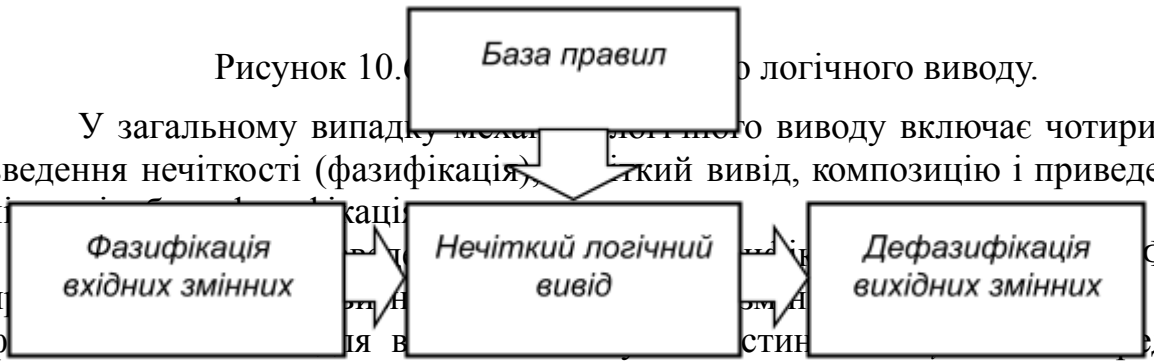


Рисунок 10.6 – Структура нечіткого логічного виводу.

У загальному випадку м… логічного виводу включає чотири етапи: введення нечіткості (фазифікація), …ткий вивід, композицію і приведення до чі… …каці… …Функції п… …до їх ф… …я в… …стин… …едумови кожного правила.

2. Логічний висновок. Обчислене значення істинності для передумов кожного правила застосовується до висновків кожного правила. Це призводить до однієї нечіткої підмножини, яка буде призначена кожній змінній висновку для кожного правила. В якості правил логічного висновку зазвичай використовуються тільки операції min (МІНІМУМ) або prod (Множення). У логічному висновку МІНІМУМУ функція приналежності виведення «відсікається» по висоті, що відповідає обчисленої ступеня істинності передумови правила (нечітка логіка «І»). У логічному висновку МНОЖЕННЯ функція приналежності виведення масштабується за допомогою обчисленої ступеня істинності передумови правила.

– min-активізація

$$\mu'(y) = \min(c_i, \mu(y))$$

– prod-активізація

$$\mu'(y) = c_i \cdot \mu(y)$$

– average-активізація

$$\mu'(y) = 0.5(c_i + \mu(y))$$

3. Композиція. Всі нечіткі підмножини, що відповідають кожній змінній висновку (у всіх правилах), об'єднуються разом, щоб формувати одну нечітку підмножина для кожної змінної висновку. При подібному об'єднанні зазвичай використовуються операції max (МАКСИМУМ) або sum (СУМА). При композиції МАКСИМУМУ комбінований висновок нечіткого підмножини конструюється як поточковий максимум за всіма нечітким підмножини (нечітке логічне «АБО»). При композиції СУМИ комбінований висновок нечіткого підмножини конструюється як поточкова сума за всіма нечіткими підмножинами, що відповідають змінним висновку, які визначається правилами логічного висновку.

– max-композиція

$$\mu(y) = \max_i(\mu'(y))$$

– sum-композиція

$$\mu(y) = \sum_i \mu'(y)$$

4. На закінчення (додатково) – приведення до чіткості (дефазифікації, defuzziflcation), якт використовується, коли слід перетворити нечіткий набір висновків в чітке число. Є велика кількість методів приведення до чіткості, тому розглянемо найбільш часто використовувані методи.

– центр ваги

$$y' = \frac{\int_{\min}^{\max} y\mu(y)dy}{\int_{\min}^{\max} \mu(y)dy}$$

– цент площ

$$\int_{\min}^{y'} \mu(y)dy = \int_{y'}^{\max} \mu(y)dy$$

– ліве модельне значення

$y' = \min\{y_i\}$,

де y_i – мода (висота нечіткої і-тої підмножини)

– праве модальне значення

$y' = \max\{y_i\}$

Алгоритми нечіткого виводу розрізняються головним чином видом використовуваних правил, логічних операцій і різновидом методу дефазифікації. Розроблені моделі нечіткого виведення Мамдані, Сугено, Ларсена, Цукамото.

Розглянемо докладніше нечіткий вивід на прикладі механізму Мамдані (Mamdani) (рис.10.7). Це найбільш поширений спосіб логічного виводу в нечітких системах. У ньому використовується мінімаксна композиція нечітких множин. Даний механізм включає наступну послідовність дій.

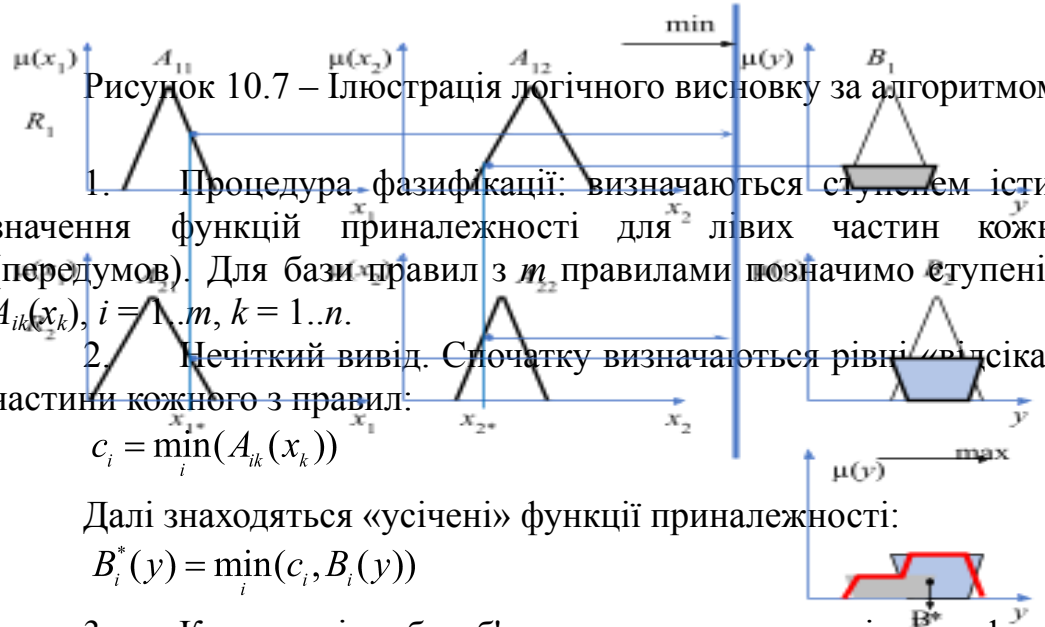


Рисунок 10.7 – Ілюстрація логічного висновку за алгоритмом Мамдані

1. Процедура фазифікації: визначаються ступенем істинності, тобто значення функцій приналежності для лівих частин кожного правила (передумов). Для бази правил з m правилами позначимо ступені істинності як $A_{ik}(x_k)$, $i = 1..m$, $k = 1..n$.

2. Нечіткий вивід. Спочатку визначаються рівні «відсікання» для лівої частини кожного з правил:

$$c_i = \min_i(A_{ik}(x_k))$$

Далі знаходяться «усічені» функції приналежності:

$$B_i^*(y) = \min_i(c_i, B_i(y))$$

3. Композиція, або об'єднання отриманих усічених функцій, для чого використовується максимальна композиція нечітких множин:

$$\mu(y) = \max_i(B_i^*(y))$$

де MF(y) – функція приналежності підсумкової нечіткої множини.

4. Дефазифікація, або приведення до чіткості. Може бути використаний будь-який наведений вище метод.

Рис. 10.7 графічно показує процес нечіткого виводу по Мамдані для двох вхідних змінних і двох нечітких правил $R1$ і $R2$.

11 ОСНОВНІ ПІДХОДИ ДО ПЛАНУВАННЯ ЦІЛЕСПРЯМОВАНИХ ДІЙ

11.1 Планування цілеспрямованих дій і прийняття рішень

Як ми вже зазначали, функціонування будь-якої інтелектуальної системи полягає в тому, що вона сприймає зовнішню ситуацію і певним чином реагує на неї. Проаналізувавши ситуацію, вона повинна прийняти рішення про вибір певної дії, виходячи з власної мети. Але таке рішення — це заключний етап процесу планування. *Планування цілеспрямованих дій* полягає в аналізі всіх можливих дій системи та наслідків цих дій. Аналізуючи можливі наслідки, система оцінює один з них як найсприятливіший для себе і вибирає ту дію, яка, на думку системи, повинна привести до очікуваного результату.

Широкий клас завдань у рамках прийняття рішень може бути сформульовано у вигляді класичної оптимізаційної задачі: знайти рішення, за яким деяка цільова функція досягає свого максимуму при заданих обмеженнях. Так, 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 є предметом науки, яка називається *дослідженням операцій*. Формальніше оптимізаційну задачу можна сформулювати в такому вигляді:

знайти $x = (x_1, \ldots, x_n)$*, за якого функція* $f(x)$ *досягає максимуму і задовольняються обмеження* $g_i(x) \geq 0$.

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

Приклад Мандрівник збирається в дорогу. Він може взяти в рюкзак певну кількість предметів різних типів. Нехай є n типів предметів; наявна кількість i-го предмета становить r_i. Кожний предмет має власну цінність c_i та вагу q_i. Потрібно зібрати рюкзак таким чином, щоб сумарна цінність узятих предметів була максимальною, але щоб сумарна вага не перевищувала заданої межі u. Формалізуємо цю постановку. Позначимо через x_i кількість предметів i-го типу, що беруться в дорогу. Тоді очевидним чином отримуємо математичну постановку задачі:

Знайти $x = (x_1, \ldots, x_n)$, *для якого* $\max \sum_{i=1}^{n} c_i x_i$ *та задовольняються обмеження* $\sum_{i=1}^{n} q_i x_i \leq u$, x_i – цілі, $x_i \geq 0$, $x_i \leq r_i$.

Будь-який елемент x, *який* задовольняє обмеженням $g_i(x) \geq 0$, називається *допустимим рішенням* задачі. Якщо умова максимізації не висувається, йдеться про *задачу пошуку допустимих рішень.* 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 $h(x) = – g(x)$.

Залежно від вигляду цільової функції та функцій-обмежень розрізняють неоднакові часткові випадки оптимізаційних задач, для кожного з яких існують свої методи вирішення. Так, оптимізаційна задача з лінійною цільовою функцією та лінійними обмеженнями називається *задачею лінійного програмування. Нелінійне програмування* є складнішим розділом дослідження операцій. Якщо всі x_i цілі, йдеться про *дискретне програмування.* Зокрема, саме до задач дискретного програмування належить згадана задача про рюкзак.

Серед багатьох ідей, які застосовуються для знаходження максимумів функцій і функціоналів, можна, мабуть, виокремити такі: лінійне програмування, принцип максимуму Понтрягіна і теорію локальних екстремумів.

Поряд з класичними ідеями оптимізації в останні десятиріччя почали розвиватись ідеї іншої природи – послідовного аналізу варіантів, їх відбракування, послідовного звуження множини можливих розв'язків. Розглянемо ряд методів вирішення оптимізаційної задачі.

11.2 Повний перебір

Існує очевидний і досить універсальний метод вирішення оптимізаційних задач, який може бути застосований, якщо множина припустимих рішень M обмежена. Це – метод *повного перебору,* який полягає у переборі всіх можливих варіантів. Метод дає гарантований розв'язок якщо множина M скінчена (ситуація, характерна для дискретного програмування), та існує ефективний алгоритм породження будь-якого елемента з M та обчислення на цьому елементі цільової функції.

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 ця 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 вбачається можливим. Тому розвиваються методи і алгоритми, які дозволяють скоротити такий перебір.

У той самий час далеко не завжди слід шукати саме оптимальне рішення. Часто достатньо обмежитися рішенням субоптимальним (наближеним до оптимального) або просто припустимим. У таких випадках достатньо виробити якесь правило, яке б виключало застосування повного перебору і для більшості вхідних даних давало б прийнятний результат. У цьому полягає суть *евристичних алгоритмів*.

Втім далеко не будь-яка інтелектуальна задача припускає очевидне зведення до оптимізаційної, оскільки часто не вдається записати в явному

вигляді цільову функцію або обмеження.

11.3 Методи "сліпого" пошуку

11.3.1 Випадковий пошук

Випадковий пошук реалізується простим алгоритмом. Інформованість при випадковому пошуку обмежується топологією графа простору станів. Стратегія пошуку полягає в тому, що, починаючи з початкового стану, випадково вибирають оператор, що здійснює перехід до наступного стану задачі. Пошук продовжується до тих пір, поки не буде виконаний перехід в цільовий стан.

Нижче приведена процедура випадкового пошуку на псевдомові:

```
Procedure Random_Search;
Begin
n:='початкова вершина';
While n<> 'цільова вершина' Do
Begin
Вибрати випадково оператор, застосовний до 'n'; Застосувати даний оператор до 'n' і одержати нову вершину 'n_New';
N:=n_New;
End.
```

Випадковий вибір оператора не гарантує збіжність алгоритму. Проте, такий метод пошуку забезпечує рішення простих задач, з обмеженим простором пошуку, за наявності шляху між початковою і цільовою вершиною.

Оскільки граф станів обстежується випадковим чином, без запам'ятовування пройденого маршруту, можливий багатократний вибір одних і тих же станів. Поліпшити процедуру пошуку можна введенням певної систематичності вибору переходів стану.

11.3.2. Пошук "в глибину і ширину"

Дані методи виключають даремний повторний вибір одних і тих же вершин. Для цього за допомогою списків OPEN і CLOSED ведеться відповідно облік вже розкритих вершин і вершин, що підлягають розкриттю. Окрім цього, задається певний порядок (систематичність) обстеження графа стану, що дозволяє послідовно пройти всі маршрути по одному разу, без пропусків і повторень. Розглянемо процедуру пошуку в глибину:

```
Procedure Depth_First;
Begin
Помістити початкову вершину в список Open;
Closed:='порожній список';
While OPEN <>'порожній список' Do;
Begin
  n:=first(Open);
  If n='цільова вершина' then Вихід(УСПІХ);
  Перемістити n із списку Open в Closed;
  Розкрити вершину n і помістити всі її дочірні вершини, відсутні в списках Open і Closed , в початок списку Open,
```

```
зв'язавши з кожною дочірньою вершиною покажчик на n;
    End;
    Вихід(НЕВДАЧА);
    End.
```

На початку пошуку список CLOSED порожній, а OPEN містить тільки початкову вершину. Кожного разу із списку Open вибирається для розкриття перша вершина (виклик функції First(Open)). Розкрита вершина переміщається в список Closed, а її дочірні вершини поміщаються в початок списку Open. Таким чином, на наступному кроці розкриватимуться дочірні вершини поточної вершини n. Принцип формування списку Open в цьому випадку відповідає стеку, тобто вершина, додана в список останньої, обробляється першою (LIFO – Last In First Out). Якщо прослідити напрям пошуку по дереву станів, то фронт пошуку росте в глибину. Для побудови зворотного шляху (з цільової вершини в початкову вершину) всі дочірні вершини забезпечуються покажчиками на відповідні батьківські вершини.

Процедура пошуку завширшки відрізняється від розглянутої процедури пошуку в глибину тим, що дочірні вершини, одержувані при розкритті вершини n, поміщаються в кінець списку Open Отже, при пошуку в ширину принцип формування списку Open відповідає черзі, тобто вершина, внесена в список першої, обробляється першою (FIFO – First In First Out). Завдяки цьому фронт пошуку росте вширину, що гарантує знаходження шляху з мінімальною кількістю дуг (ребер) між початковою і цільовою вершинами (свого роду оптимальності).

11.3.3 Алгоритм рівних цін

Інформованість про вирішувану задачу тут вище, ніж у разі пошуку в глибину і ширину. Кожному оператору, що перетворює стан n_i, в стан n_j, ставиться у відповідність деяка функція вартості $c(n_i, n_j)$, що характеризується позитивними значеннями. В ході пошуку шляху на графі враховуються сумарні витрати для досягнення тієї або іншої вершини, тобто вартість шляху до цієї вершини. Вартість шляху до вершини n_j, може бути визначена по наступній формулі

$g(n_j) = g(n_i) + c(n_j, n_i)$

де $g(n_j)$ – вартість шляху з початкової вершини у вершину n_j.

В ході пошуку вершини в списку OPEN сортуються в порядку збільшення вартості. Кожного разу розкривається вершина з якнайменшою вартістю. Це дозволяє знайти шлях мінімальної вартості з початкової вершини в цільову вершину. Слід зазначити, що в процесі пошуку вартість шляху до вершини може мінятися, якщо виявляється більш дешевий шлях. Тому позначатимемо вартість оптимального шляху з початкової вершини у вершину n через $\hat{g}(n)$, а вартість поточного шляху, знайденого алгоритмом, через $g(n)$, тобто $\hat{g}(n)$ – це оцінка вартості шляху $g(n)$. В кінці пошуку $\hat{g}(n) = g(n)$.

Нижче представлена процедура пошуку оптимального шляху (мінімальної вартості), записана на псевдомові:

```
Procedure Optimal Search;
Begin
Помістити початкову вершину в список Open;
Closed:='порожній список';
While OPEN < >'порожній список' Do
   n:=First(Open);
   If n='цільова вершина' Then Вихід(УСПІХ);
   Перемістити вершину n із списку Open в Closed;
   Розкрити вершину n, для кожної дочірньої вершини n_i
обчислити вартість g^(n, n_i);
   Помістити дочірні вершини, яких немає в списках Open і
Closed, в список Open, пов'язавши з кожною вершиною покажчик на
вершину n і покласти g^(n_i)= g^(n, n_i);
   Для кожної з дочірніх вершин, які вже містяться в списку
Open, порівняти поточну вартість g^(n, n_i) з раніше обчисленим
значенням вартості g^(n_i), що зберігається в списку Open, якщо
g^(n,n_i)<g^(n_i), то встановити g^(n_i) = g^(n, n_i). Забезпечити
вказані дочірні вершини покажчиками на вершину n;
   Упорядкувати список Open за збільшенням вартості;
End;
Вихід(НЕВДАЧА);
End.
```

В той момент, коли розкривається деяка вершина *n*, оптимальний шлях до цієї вершини вже знайдений, тобто $\hat{g}(n) = g(n)$. Іншими словами, якщо вершина вже знаходиться на початку списку Open, то для неї неможливо знайти більш дешевший шлях. Отже, переклад вершини в список Closed є остаточним.

Розглянута процедура пошуку гарантує знаходження шляху мінімальної вартості, якщо граф кінцевий і існує шлях з початкової вершини в цільову вершину. Проте сам процес пошуку не є оптимальним, оскільки пошук виконується без урахування положення цільових вершин.

11.4 Евристичний пошук

Основна ідея евристичного пошуку полягає у використовуванні додаткової інформації для управління процесом пошуку. Ця додаткова інформація формується на основі емпіричного досвіду, припущень і інтуїції дослідника, тобто евристик. Використовування евристик дозволяє скоротити кількість варіантів, що проглядаються, при пошуку рішення задачі, що веде до більш швидкого досягнення мети.

В алгоритмах евристичного пошуку список відкритих вершин упорядковується за збільшенням деякій оцінній функції, формованій на основі евристичних правил. Оцінна функція може включати дві складові, одна з яких називається евристичною і характеризує близькість поточної і цільової вершин. Чим менше значення евристичної оцінної функції, тим ближче дана вершина до цільової вершини. Залежно від способу формування оцінної функції виділяють наступні алгоритми евристичного пошуку: алгоритм «підйому на гору», алгоритм глобального обліку відповідності мети, А-алгоритм.

11.4.1 Алгоритм "підйому на гору"

Алгоритм здійснюється цілеспрямований пошук у напрямі найбільшого убування оцінної функції $\hat{h}(n)$. Дана функція забезпечує оцінку (прогноз) вартості найкоротшого шляху від поточної вершини п до найближчої цільової вершини, тобто є мірою вартості шляху, що залишився.

Чим менше значення $\hat{h}(n)$, тим більше «перспективний» шлях, на якому знаходиться вершина *n*.

Подібний алгоритм використовується при пошуку екстремумів функції. Відомо, що положення екстремуму функції багатьох змінних визначається напрямом вектору градієнта функції. Тому пошук екстремуму здійснюють у напрямі найбільшого зростання (убування) функції, тобто в напрямі, співпадаючому (або протилежному) з напрямом градієнта. Пошук максимуму функції в цьому випадку нагадує сходження на пік по найкрутішому маршруту. Тому даний алгоритм і називають алгоритмом «підйому на гору».

На черговому кроці алгоритму для кожної з дочірніх вершин n_1, n_2,..., n_m вершини *n* обчислюється значення оцінної функції $\hat{h}(n_i)$, і для продовження шляху вибирається вершина з якнайменшим значенням $\hat{h}(n_i)$. Процедура пошуку терпить невдачу, якщо для всіх дочірніх вершин $\hat{h}(n_i) > \hat{h}(n)$.

```
Procedure Hill Climbing;
Begin
n:='початкова вершина'; While n<> 'початкова вершина' Do
Begin
   Розкрити вершину n_i для всіх дочірніх вершин n_i обчислити
оцінки h(n_i)
   Вибрати дочірню вершину n_i, з мінімальним значенням h(n_i);
   If h^(n_i) >= h^(n) Then Вихід(НЕВДАЧА);
   n:=n_i;
End;
Вихід(УСПІХ);
End.
```

Існують різні способи побудови евристичної оцінної функції, проте готових рецептів немає. При рішенні кожної конкретної задачі використовують раніше накопичений досвід рішення подібних задач.

11.4.2 Глобальний облік відповідності мети

Цей алгоритм пошуку рішення задачі схожий на попередній. Тут також оцінюється «перспективність» того або іншого шляху за допомогою евристичної оцінної функції $\hat{h}(n)$. Відмінність полягає в тому, що на кожному етапі виконується не локальне порівняння вершин, одержаних при розкритті поточної вершини, а глобальне порівняння всіх вершин-кандидатів, що знаходяться в списку Open. Для цього список відкритих вершин сортується в

порядку зростання значень $\hat{h}(n)$. Якнайкраща вершина для продовження шляху знаходитиметься на першому місці в списку Open. Тому даний алгоритм називають також алгоритмом вибору першої якнайкращої вершини (Best-First)

Процедура пошуку, заснована на даному алгоритмі, схожа на процедуру Optimal Seargh. Відмінність полягає в тому, що замість функції вартості $\hat{g}(n)$ застосовується евристична функція $\hat{h}(n)$, тобто враховуються тільки майбутні витрати:

```
Procedure Best First Searght;
Begin
Помістити початкову вершину в Open; Closed:='порожній список';
While OPEN 'порожній список' Do
Begin
  п:=«First (Open);
  I п='цільова вершина' Then Вихід(УСПІХ);
  Перемістити вершину n із списку Open в список Closed;
  Розкрити вершину n, для кожної дочірньої вершини обчислити
h(ni)
  перемістіть дочірньої вершини яких нема в списку Open  і
Closed.в список Open  взявши з кожної вершини показник на вершину
н.
  Упорядкувати список Open по зросту значень h(n)
End
Вихід (неудача)
End
```

Даний алгоритм дозволяє успішно справлятися з проблемою локальних мінімумів. Проте при цьому знижується ефективність самого процесу пошуку. Ефективність процесу пошуку залишатиметься високої, якщо оцінка вартості найкоротшого шляху $\hat{h}(n)$) з вершини n в цільову вершину буде якомога ближче до реальної вартості $h(n)$.

11.4.3 А-алгоритм

Перевага алгоритму рівних цін полягає в тому, що він гарантує знаходження оптимального шляху. Перевагою алгоритмів, що використовують оцінку майбутніх витрат $\hat{h}(n)$, є можливість усікання дерева пошуку, що підвищує ефективність пошуку. Природним є прагнення комбінувати ці алгоритми і враховувати як зроблені, так і майбутні витрати. Цього можна добитися, якщо скористатися оцінною функцією

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$$

де $\hat{f}(n)$ – оцінка вартості шляху з початкової вершини у вершину n; $\hat{h}(n)$ – оцінка вартості найкоротшого шляху з вершини n в цільову вершину.

Алгоритм, що базується на використовуванні оцінної функції $\hat{f}(n)$

називається А-алгоритмом. Оцінки $\hat{h}(n)$ в ході пошуку не міняють своїх значень. Оцінки $\hat{g}(n)$ також не міняють своїх значень і завжди рівні реальним витратам $g(n)$, якщо пошук шляху виконується на дереві станів. Проте при пошуку шляху на графі станів оцінки $\hat{g}(n)$ можуть мінятися. Отже, в загальному випадку оцінки $\hat{f}(n)$ міняють свої значення в процесі пошуку. Це призводить до того, що вершини із списку Closed можуть переміщатися назад в список Open:

```
Procedure A_Searsh;
Begin
Помістити початкову вершину S в список Open: f^(s)=h^(s)
Closed:='порожній список';
While OPEN 'порожній список' Do
Begin
  n:=First (Open);
  If n='цільова вершина' Then Вихід(УСПІХ);
  Перемістити n з Open в Closed;
  Розкрити n і для всіх її дочірніх вершин і, обчислити оцінку
  f^(n, n_i) = g^(n, n_i) + h^(n_i);
  Помістити дочірні вершини, яких немає в списках Open і
Closed, в список Open, пов'язавши з кожною вершиною покажчик на
вершину n;
  Для дочірніх вершин n, які вже містяться в Open, порівняти
оцінки f^(n, n_i) і f(n_i), якщо f^(n, n_i)<f^(n_i), то пов'язати з
вершиною n_i нову оцінку f^(n, n_i) і покажчик на вершину n; Якщо
вершина n_i, міститься в списку Closed і f^(n, n_i)<f^(n_i), то
пов'язати з вершиною n,  нову оцінку f^(n, n_i), перемістити її в
список Open і встановити покажчик на n;
  Упорядкувати список Open за збільшенням f^(n_i);
End;
Вихід(НЕВДАЧА);
End.
```

Вкажемо властивості А-алгоритма. Якщо для всіх вершин $\hat{h}(n) = 0$, то А-алгоритм відповідатиме алгоритму рівних цін. В цьому випадку гарантується знаходження оптимального шляху, але ефективність пошуку буде низькою, і якщо простір пошуку буде нескінченний, то пошук просто не здійснимо.

А-алгоритм не дає гарантії, що знайдений шлях буде оптимальним. Шлях може бути не оптимальним у випадку, якщо оцінка витрат на шляху з вершини і в цільову вершину буде перебільшена (переоцінена), тобто якщо $\hat{h}(n) > h(n)$.

А-алгорітм забезпечить пошук оптимального шляху, якщо виконується умова

$$\hat{h}(n) < h(n) \tag{11.1}$$

Алгоритм, що враховує умову (2.1), називають А*-алгоритмом або гарантуючим алгоритмом. А*-алгоритм недооцінює витрати на шляху з проміжної вершини в цільову вершину або оцінює їх правильно, але ніколи не

переоцінює.

Нехай A1* і A2* – довільні гарантуючі алгоритми і на всіх вершинах $\hat{h}_1(n) > \hat{h}_2(n)$. Тоді інформованість A1* вище, ніж A2*. Говорять, що в цьому випадку A1* має більшу евристичну силу, ніж A2*. Евристично більш сильний алгоритм A1* більшою мірою скорочує простір пошуку.

Ефективність пошуку за допомогою A*-алгоритма може знижуватися через те, що вершина, що знаходиться в списку Closed, може потрапляти назад в список Open і потім повторно розкриватися. Наступний приклад демонструє випадок багатократного розкриття вершин (рис.11.1).

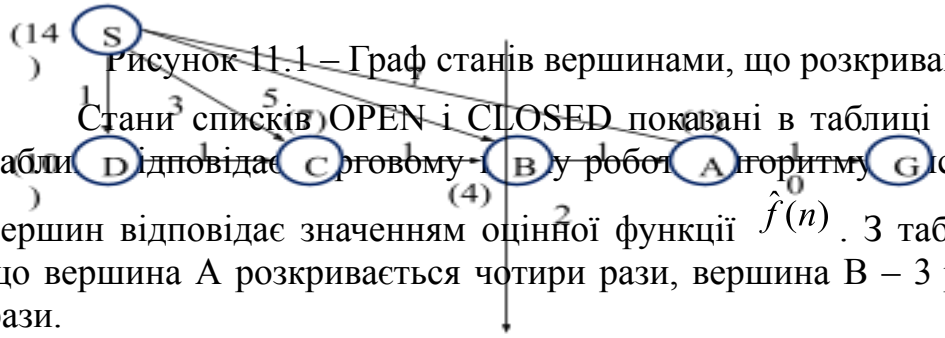


Рисунок 11.1 – Граф станів вершинами, що розкриваються декілька раз

Стани списків OPEN і CLOSED показані в таблиці 11.1. Кожний рядок таблиці відповідає черговому циклу роботи алгоритму. Числа поряд з іменами вершин відповідає значенням оцінної функції $\hat{f}(n)$. З таблиці 11.1 виходить, що вершина A розкривається чотири рази, вершина B – 3 рази, вершина C – 2 рази.

Таблиця 11.1 - Виконання A*-алгоритма

№	OPEN	CLOSED	Коментар
1	S(14)	-	
2	A(8) B(9) C(10) D(11)	s(14)	
3	B(9) C(10) D(11) G(17)	B(9) S(14)	
4	A(7) C(10) D(11) G(17)	B(9) S(14)	Повернення: A
5	C(10) D(11) G(16)	A(7) B(9) S(14)	
6	A(6) B(8) D(11) G(15)	C(10) S(14)	Повернення: A, B
7	B(8) D(11) G(15)	A(6) 3(10) 5(14)	
8	D(11) G(15)	B(8) A(6) C(1 0) S(14)	
9	C(9) G(15)	D(11) B(8) A(6) S(14)	Повернення: C
10	A(5) B(7) G(15)	C(9) D(11) S(14)	Повернення: A, B
11	B(7) G(14)	A(5) C(9) D(11) S(14)	
12	G(14)	B(7) A(5) C(9) D(1 1) S(14)	

Щоб A*-алгоритм не розкривав кілька разів одну і ту ж вершину, необхідно враховувати умову монотонності

$$\hat{h}(n_i) \le \hat{h}(n_j) + c(n_i, n_j) \qquad (11.2)$$

де n_i, – батьківська вершина; n_j – дочірня вершина; $c(n_i n_j)$ – вартість шляху між вершинами n_i і n_j.

Якщо умова монотонності дотримується для всіх дочірніх вершин, то можна довести, що в той момент, коли розкривається деяка вершина n, оптимальний шлях до неї вже знайдений. Отже, оцінна функція $\hat{f}(n)$ для даної вершини надалі не міняє своїх значень, і ні які вершини із списку Closed в список Open не повертаються.

Раніше наголошувалося, що А*-алгоритм недооцінює витрати $h(n)$. Умова монотонності означає, що недооцінка на шляху до мети стає все меншою або, в крайньому випадку, не змінюється. Дійсно, якщо позначити недооцінку через $\hat{u}(n)$, то

$$\hat{u}(n) = h(n) - \hat{h}(n), \quad \hat{u}(n) > 0$$

З умови монотонності виходить, що $\hat{u}(n_j) < \hat{u}(n_i)$, тобто недооцінка $h(n)$ в батьківських вершинах більше або рівно недооцінкам $h(n)$ в дочірніх вершинах. На графі, зображеному на рис 11.1, умова монотонності не виконується для початкової вершини. Наприклад,

$h(S) > h(D) + c(S, D)$, тобто $14 > 10 + 1$.

Якщо $\hat{h}(n) = h(n)$, то інформованість є повною. В цьому випадку ніякого пошуку не відбувається і наближення до мети йде по оптимальному шляху.

Таким чином, властивості А-алгоритму істотно залежать від умов, яким задовольняє або не задовольняє евристична частина оцінної функції $\hat{f}(n)$:

1) А-алгоритм відповідає алгоритму рівних цін, якщо $h(n) = 0$;

2) А-алгоритм гарантує оптимальне рішення, якщо $\hat{h}(n) < h(n)$; в цьому випадку він називається А* - алгоритмом;

3) А-алгоритм забезпечує однократне розкриття вершин, якщо виконується умова монотонності $\hat{h}(n_i) \le \hat{h}(n_j) + c(n_i, n_j)$;

4) алгоритм А1* евристично більш сильний, ніж алгоритм А2* при $\hat{h}_1(n) > \hat{h}_2(n)$;

5) А*-алгоритм повністю інформований, якщо $\hat{h}(n) = h(n)$;

6) при $\hat{h}(n) > h(n)$ А-алгоритм не гарантує отримання оптимального рішення, проте часто рішення виходить швидко.

11.5 Простір задач і простір станів

Методики планування цілеспрямованих дій прийнято розподіляти на два класи: планування в просторі станів і планування в просторі задач.

При *плануванні в просторі станів* заданим вважається деякий набір станів (ситуацій). Відомі дії, які може здійснювати система та які визначають перехід з одного стану до іншого.

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 даного рішення, тобто пошуку найкоротшого шляху. Відповідно, після того як

зведення задачі до формальної моделі проведено можна використовувати відомі алгоритми пошуку шляхів на графах (алгоритми Мура, Дейкстри, гілок і границь і т. п.).

Класичним прикладом планування в просторі станів можна вважати задачу пошуку шляху від одного міста до іншого.

Планування в просторі задач передбачає декомпозицію початкової задачі на підзадачі, поки не буде досягнуто зведення до задач, для яких є готові алгоритми вирішення. Таку декомпозицію зручно уявляти собі у вигляді *І/АБО-графа.*

Існують методики комбінування методів пошуку в просторі станів і в просторі задач.

Пошук рішень в просторі станів може виконуватися в декількох напрямах: з початкового стану в цільовий стан (пошук, керований даними); з цільового стану в початковий стан (пошук, керований метою, або зворотний пошук); одночасно в двох напрямах (комбінований пошук). Оскільки пошукові графи (дерева) для практичних задач – великі, то двонаправлений пошук може бути ефективно однонаправленого.

Пошук рішень в просторі станів – комбінаторна задача. Рішення виходить на основі перебору всіх станів. Проте для багатьох практичних задач повний перебір неможливий. Були запропоновані різні заходи для оцінки складності вирішуваної задачі. Одна з них – це ступінь розгалуження вершин графа стану. Вона рівна середньому числу дочірніх вершин, що доводяться на один стан задачі. Якщо розглядати дерево станів, то можна вивести формулу, що встановлює залежність загального числа станів N від ступеня розгалуження B і глибини дерева L:

$N = B\ (B^L - 1)/(B - 1)$

Наприклад, необхідно підрахувати загальне число станів для гри-головоломки "8". Вкажемо можливі переміщення фішок:

– по два переміщення з кожного кута, тобто для чотирьох кутів вісім переміщень;

– по три переміщення з центральних позицій уздовж кожної із сторін, тобто всього дванадцять переміщень;

– чотири переміщення з центральної позиції.

Загальне число можливих переміщень 24. Розділивши це число на 9 (число різних положень для порожньої ячейки), одержимо ступінь розгалуження 2,67. Тепер можна визначити загальне число станів при прогляданні дерева станів глибини L (табл. 2.2).

Таблиця 11.2 – Число станів при грі "у вісім"

I	2	3	4	5	6	7	8	9	10
N	10	29	80	215	577	1545	4127	11024	29436

Як видно з таблиці 11.2, навіть для такої простої задачі кількість станів, що проглядаються, різко зростає при збільшенні глибини пошуку.

Складність вирішуваної задачі можна також оцінювати розмірами списків

Open і Closed. Для того, щоб ці списки не розростались, можна зберігати в списку Open тільки декілька "найперспективніших" станів. Це, звичайно, скорочує простір пошуку, але виникає небезпека пропуску рішення. Такі стратегії пошуку, що накладають обмеження на розмір списку Open, відповідають променевому пошуку (Beam Search). Більш безпечні стратегії евристичного пошуку, розглянуті вище. Чим більше інформованість про вирішувану задачу, тим більша вага має евристична частина оцінної функції і тим більше скорочується простір пошуку. Слід звернути увагу на те, що обчислення евристик також вимагає певних витрат. В цьому випадку виникає питання мінімізації загального процесорного часу, затрачуваного на пошук рішення.

11.6 Пошук з розповсюдженням обмежень

Існують задачі, де необхідно брати до уваги діючі обмеження на переходи станів, і завдяки цьому цілеспрямовано йти до її рішення. Як приклад можна привести задачу розпізнавання мови або задачу інтерпретації тривимірних контурних малюнків.

Одна з технологій розпізнавання мови заснована на розбитті мовного сигналу на елементарні ділянки і привласненні кожній ділянці мітки у вигляді фонеми. Послідовність фонем, одержувана в результаті такої розмітки, повинна бути злагодженою і не суперечити правилам вимовлення слів. Обмеження, діючі між можливими взаємними поєднаннями фонем, дозволяють виключити з розгляду варіанти, які не формують акустичні аналоги слів. Таким чином, елементарні ділянки мови, представлені фонемами, забезпечують формування правильного слова, а послідовність слів - граматично коректну фразу. В цьому значенні обмеження, що накладаються, дозволяють по елементах відновлювати ціле і їх часто називають обмеженнями цілісності.

Якщо одна і та ж змінна бере участь одночасно в декількох обмеженнях, то утворюються мережі (системи) обмежень. Рішення задачі в цьому випадку полягає у визначенні всіх поєднань значень змінних, які задовольняють обмеженням цілісності.

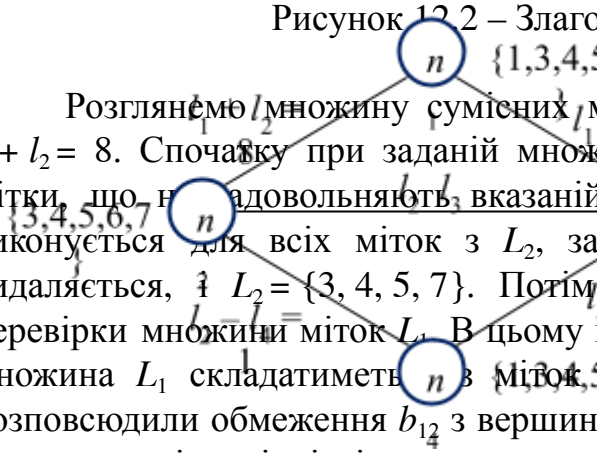
Розглянемо задачу злагодженої розмітки (consistent labeling), при рішенні якої аналізується m елементів, пронумерованих n_1, n_2,..., n_m. Припустимо, що кожний елемент n_i, може бути помічений (інтерпретований) декількома способами. Кількість можливих інтерпретацій кожного елемента задається числами b_1, b_2,…, b_m. Якщо розглядати можливі інтерпретації всієї сукупності елементів і перевіряти їх на дотримання обмежень цілісності (тобто з'ясовувати чи відображають вони допустимі рішення), то необхідно проаналізувати: $b_1 \cdot b_2 \cdot b_3 \cdot \ldots \cdot b_m$ випадків. Хоча дана задача легко представляється у вигляді графа станів, доцільно все ж таки перейти до іншого представлення – до графа, вершини якого є елементи n_i а ребра відповідають бінарним відносинам між відповідними елементами.

Нехай для кожного з елементів n_i, $\in \{n_1, n_2,.., n_m\}$ є кінцева множина інтерпретації (міток) L_i, потужністю G_i. Можливість або неможливість

злагодженої інтерпретації двох елементів n_i і n_j - за допомогою міток $l_i \in L_i$ і $l_j \in L_j$ задається бінарним відношенням b_{ij}. Якщо певна мітка l_i, для елемента n_i, не узгоджується з відповідною міткою l_j для елемента n_j (тобто не задовольняє відношенню b_{ij}, то дана мітка видаляється з множини L_i. Аналогічним чином досліджуються всі мітки, що входять в множину l_j. В ході вказаного процесу з кожною вершиною графа зв'язується мітка (або множина міток). При цьому мітки на кінцях ребер графа повинні задовольняти заданим відносинам.

Розглянемо приклад злагодженої розмітки вершин графа, зображеного на рис.12.2 Біля вершин графа вказана множина можливих міток, а обмеження, яким винні задовольняти відповідні мітки, записані поряд з ребрами. Основна ідея полягає в звуженні множини можливих міток для кожної з вершин, спираючись на задані обмеження.

Рисунок 12.2 – Злагоджена розмітка вузлів графа



Розглянемо множину сумісних міток для вершин n_1 і n_2 при обмеженні $l_1 + l_2 = 8$. Спочатку при заданій множині $L_1 = \{1, 3, 4, 5, 6\}$ видалимо з L_2 всі мітки, що не задовольняють вказаній умові. В даному випадку відношення виконується для всіх міток з L_2, за винятком мітки 6. Отже, дана мітка видаляється, і $L_2 = \{3, 4, 5, 7\}$. Потім відношення B_{12} використовується для перевірки множини міток L_1. В цьому випадку з L_1 також видаляється мітка 6, і множина L_1 складатиметься з міток $\{1, 3, 4, 5\}$. Таким чином, ми спочатку розповсюдили обмеження b_{12} з вершини n_1 у вершину n_2, а потім назад з n_2 в n_1, видаляючи відповідні мітки з множин L_2 і L_1 Мітки, що входять в множини L_2 і L_1, тепер злагоджені.

На наступному кроці для аналізу вибирається вершина, яка пов'язана з найбільшим числом вершин, вже розглянутих до поточного моменту. В даному випадку це вершина n_3. Розповсюдивши обмеження від n_1 до n_3 і від n_2 до n_3, одержимо $L_3=\{1, 3, 5\}$. Потім відповідні обмеження розповсюджуються у зворотний бік. При цьому множини L_2 і L_1 не змінюються. Нарешті, аналогічні операції виконуються з вершиною n_4. У результаті $L_4 = \{3, 4\}$, $L_2 = \{4, 5\}$, $L_3=\{3, 5\}$. Потім обмеження розповсюджуються між n_2 і n_3 (це дає $L_3 = \{3\}$), а також від n_2 і n_3 до n_1. Одержуємо $L_1 = \{3, 4\}$. Поширюючи обмеження назад від n_1 до n_2, n_3 і n_4, одержуємо остаточний результат: $l_1 = 4$ $l_2 = 4$, $l_3 = 3$, $l_4 = 2$. Розповсюдження обмежень продовжується до тих пір, поки на множині міток відбуваються зміни.

В загальному випадку вершина може мати декілька міток. При цьому довільно комбінувати мітки для різних вершин не можна. Необхідно враховувати існуючі обмеження. Очевидно, в цьому випадку задача має декілька рішень.

12 Вступ в експертні системи

У основі стратегії інтелектуальних технологій лежить поняття парадигми – концептуального уявлення на суть проблеми або завдання і принцип її рішення. Центральна парадигма інтелектуальних технологій - це обробка знань. Системи, ядром яких є база знань або модель наочної області, описана на мові надвисокого рівня, наближеній до природної мови називають інтелектуальними. Найчастіше інтелектуальні системи (ІС) застосовуються для вирішення складних завдань, зв'язаних з використанням слабких формалізованих знань фахівців-практиків, а також з логічною обробкою інформації. Наприклад, підтримка ухвалення рішення в складних ситуаціях, аналіз візуальної інформації, управління в електричних ланцюгах електроустаткування і мережах розподілу електроенергії; пошук несправностей в електронних пристроях, діагностика відмов обладнання і так далі. Типовими прикладами ІС є експертні системи (ЕС) і штучні нейронні мережі (ІНС), що беруть на себе вирішення питань витягання і структуризації знань, а також технологічні аспекти розробки систем, заснованих на знаннях.

Експертні системи – це швидко прогресуючий напрям в області штучного інтелекту. Сучасні ЕС є складними програмними комплексами, акумулючими знання фахівців в конкретних наочних областях і поширюючими цей емпіричний досвід для консультування менш кваліфікованих користувачів. Парадигма ЕС припускає наступні об'єкти, а також етапи розробки і функціонування ІС:

– формалізація знань – перетворення експертом проблемного знання у форму, наказану вибраною моделлю представлення знань;

– формування бази знань (БЗ) – вкладення формалізованих знань в програмну систему;

– дедукція – рішення задачі логічного виводу на основі БЗ.

Основні чинники, що впливають на доцільність і ефективність розробки ЕС:

– брак фахівців, що витрачають значний час для надання допомоги іншим;

– виконання невеликого завдання вимагає численного колективу спеціалістів, оскільки жоден з них не володіє достатнім знанням;

– понижена продуктивність, оскільки завдання вимагає повного аналізу складного набору умов, а звичайний фахівець не в змозі проглянути за відведений час всі ці умови;

– велика розбіжність між вирішеннями самих хороших і найгірших виконавців;

– наявність експертів, готових поділитися своїм досвідом.

Порівняльні властивості прикладних завдань для їх вирішення ЕС приведені в табл. 12.1.

Таблиця 12.1 Критерії застосовності ЕС

Застосовані ЕС	Непридатні ЕС
Не можуть бути побудовані строгі алгоритми або процедури, але існують евристичні методи рішення.	Є ефективні алгоритмічні методи.
Є експерти, які здатні вирішити задачу.	Відсутні експерти або число недостатньо.
По своєму характеру завдання відносяться до області діагностики, інтерпретації або прогнозування.	Завдання носять обчислювальний характер.
Доступні дані “зашумлені”	Відомі точні факти і строгі процедури.
Завдання вирішуються методом формальних міркувань.	Завдання вирішуються процедурними методами, за допомогою аналогії або інтуїтивно.
Знання статичні, незмінні.	Знання динамічні міняються з часом.

Недоліки експертних систем перед людиною-експертом:

– експертна система може бути не придатна для застосування користувачем, якщо у нього немає досвіду роботи з такими системами;

– питально-відповідний режим, зазвичай прийнятий в таких системах, уповільнює отримання рішень;

– існує проблема приведення знань, отриманих від експерта, до вигляду, що забезпечує їх ефективну машинну реалізацію;

– людина-експерт при вирішенні завдань зазвичай звертається до своєї інтуїції або здорового глузду, якщо відсутні формальні методи рішення або аналоги таких завдань.

Переваги експертних систем перед людиною-експертом:

– у них немає упереджень, вони не роблять поспішних виводів;

– введені в машину знання зберігаються, людина ж має обмежену базу знань, і якщо дані довгий час не використовуються, то вони забуваються і назавжди втрачаються;

– експерт користується побічними знаннями і легко піддається впливу зовнішніх чинників, які безпосередньо не пов'язані з вирішуваним завданням.

Головна відмінність ЕС від інших програмних засобів – це наявність бази знань, в якій знання зберігаються у формі, зрозумілою спеціалістам наочної області, і можуть бути змінені і доповнені також в зрозумілій формі. Це і є мови представлення знань (МПЗ).

Найбільші труднощі в розробці ЕС викликає не процес машинної реалізації систем, а етап аналізу знань і проектування бази знань. Цим займається спеціальна наука - інженерія знань.

Експертні системи мають дві категорії користувачів і два окремі входи, відповідних різним цілям взаємодії користувачів з ЕС. До першої категорії відносяться звичайні користувачі, яким потрібна консультація ЕС. Другу

категорію представляють експерти в наочній області і інженери знань. У їх функції входить заповнення бази знань за допомогою спеціалізованої діалогової компоненти ЕС – підсистеми придбання знань.

Підсистема придбання знань призначена для додавання в базу знань нових правил і модифікації є. У її завдання входить приведення правила до вигляду, що дозволяє підсистемі виводу застосовувати це правило в процесі роботи. У складніших системах передбачені ще і засоби для перевірки правил, що вводяться або модифікуються, на несуперечність з наявними правилами.

Діалог з ЕС здійснюється через діалоговий процесор – спеціальну компоненту ЕС. Існують дві основні форми діалогу з ЕС – це діалог на обмеженій підмножині природної мови з використанням словника (меню) і діалог на основі з декількох можливих дій.

База знань представляє найбільш важливу компоненту експертної системи. На відміну від всіх останніх компонент ЕС, база знань – є «змінна» частина системи, яка може поповнюватися і модифікуватися інженерами знань і досвіду використання ЕС між консультаціями, а в деяких системах і в процесі консультації. Існує декілька способів представлення знань в ЕС. Загальним для всіх способів є те, що знання представлені в символьній формі (тексти, списки і інші символьні структури). Тим самим, в ЕС реалізується принцип символьної природи міркувань, який полягає в тому, що процес міркування представляється як послідовність символьних перетворень.

Підсистема виводу – програмна компоненту експертних систем, реалізовуюча процес її міркувань на основі бази знань і робочої множини. Вона виконує дві функції: по-перше, проглядання існуючих фактів з робочої множини і правил з бази знань і додавання (в міру можливості) в робочу множину нових фактів і, по-друге, визначення порядку перегляду і застосування правило. Ця підсистема управляє процесом консультації, зберігає для користувача інформацію про отримані висновки, і запрошує у нього інформацію, коли для спрацьовування чергового правила в робочій множині виявляється недостатньо даних.

Мета ЕС – вивести деякий заданий факт, який називається цільовим твердженням. В результаті застосування правив добитися того, щоб цей факт був включений в робочу множину, або спростувати цей факт, тобто переконатися, що його вивести неможливо. Цільове твердження може бути або «закладено» заздалегідь в базу знань системи, або витягується системою з діалогу з користувачем. Робота системи є послідовністю кроків, на кожному з яких з бази вибирається деяке правило, яке застосовується до поточного вмісту робочої множини. Цикл закінчується, коли виведено або спростовано цільове твердження. Цикл роботи експертної системи інакше називається логічним виводом.

Експертні системи як самостійний напрям в штучному інтелекті сформувався в кінці 1970-х рр. Група по експертних системах при Комітеті British Computer Society визначила ЕС як «втілення в ЕОМ компоненти досвіду експерта, заснованою на знаннях, в такій формі, що машина може дати інтелектуальну раду або ухвалити рішення щодо оброблюваної функції».

Однією з важливих властивостей ЕС є здатність пояснити хід своїх міркувань зрозумілим для користувача чином.

Область дослідження ЕС називають «інженерією знань». Цей термін був введений Е. Фейгенбаумом і в його трактуванні означає те, що «привнесло принципів і інструментарію з області штучного інтелекту у вирішення важких прикладних проблем, що вимагають знань експертів».

ЕС охоплюють самі різні наочні області, серед яких лідирують бізнес, виробництво, медицина, проектування і системи управління. У багатьох випадках ЕС є інструментом, що підсилює інтелектуальні здібності експерта. Крім того, ЕС може виступати в ролі:

– консультанта для недосвідчених або непрофесійних користувачів;
– асистента експерта-людини в процесах аналізу варіантів рішень;
– партнера експерта в процесі вирішення завдань, що вимагають залучення знань з різних наочних областей.

Для класифікації ЕС використовуються наступні ознаки:

– спосіб формування рішення;
– спосіб обліку часової ознаки;
– вид використовуваних даних і знань;
– число використовуваних джерел знань.
– партнера експерта в процесі вирішення завдань, що вимагають залучення знань з різних наочних областей.

За способом формування вирішення ЕС можна розділити на ті, що аналізують і синтезують. У системах першого типу здійснюється вибір рішення з множини відомих рішень на основі аналізу знань, в системах другого типу рішення синтезується з окремих фрагментів знань.

Залежно від способу обліку часової ознаки ЕС ділять на статичні і динамічні. Статичні ЕС призначені для вирішення завдань знезмінними в процесі рішення даними і знаннями, а динамічні ЕС допускають такі зміни.

По видах використовуваних даних і знань розрізняють ЕС з детермінованими і невизначеними знаннями. Під невизначеністю знань і даних розуміються їх неповнота, ненадійність, нечіткість.

ЕС можуть створюватися з використанням одного або декількох джерел знань.

Відповідно до перерахованих ознак можна виділити чотири основні класи ЕС (рис. 12.1): класифікуючі, довизначаючі, трансформуючі і мультиагентні.

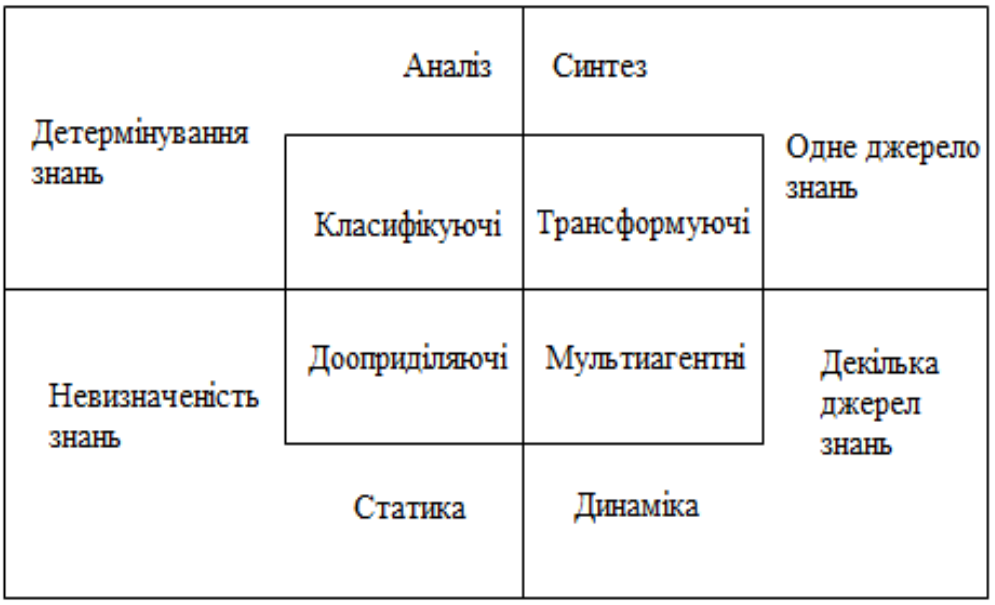


Рисунок. 12.1 – Основні класи експертних систем

Класифікуючі ЕС вирішують завдання розпізнавання ситуацій. Основним методом формування рішень в таких системах є дедуктивний логічний вивід.

Довизначаючі ЕС використовуються для вирішення завдань з не повністю певними даними і знаннями. У таких ЕС виникають завдання інтерпретації нечітких знань і вибору альтернативних напрямів пошуку в просторі можливих рішень. Як методи обробки невизначених знань можуть використовуватися байесовський імовірнісний підхід, коефіцієнти упевненості, нечітка логіка.

Трансформуючі ЕС відносяться до синтезуючих динамічних експертних систем, в яких передбачаєтьсяперетворення знань, що повторюється, в процесі вирішення завдань. У ЕС даного класу використовуються різні способи обробки знань:

– генерація і перевірка гіпотез;

– логіка припущень і умовчань (коли за неповними даними формуються уявлення про об'єкти певного класу, які згодом адаптуються до конкретних умов ситуацій, що змінюються);

– використання метазнань, більш загальних закономірностей для усунення невизначеностей в ситуаціях.

Мультиагентні системи – це динамічні ЕС, засновані на інтеграції декількох різнорідних джерел знань. Ці джерела обмінюються між собою отримуваними результатами в ході вирішення завдань. Системи даного класу мають наступні можливості:

– реалізація альтернативних міркувань на основі використання різних джерел знань і механізму усунення суперечностей;

– розподілене вирішення проблем, декомпозиція на паралельно вирішувані підзадачі з самостійними джерелами знань;

– застосування різних стратегій виведення висновків залежно від типу вирішуваної проблеми;

– обробка великих масивів інформації з баз даних;

– використання математичних моделей і зовнішніх процедур для імітації розвитку ситуацій.

12.1 Технології розробки експертних систем.

12.1.1 Особливості розробки ЕС.

Технологія створення інтелектуального програмного забезпечення відрізняється від розробки традиційних програм з використанням відомих алгоритмічних мов (табл. 12.1). Розглянемо відпрацьовані в даний час елементи технології створення ЕС на прикладі розробки експертних систем. Цей вибір обумовлений тим, що ЕС набули вельми широкого поширення в багатьох сферах людської діяльності, а технології їх створення мають універсальний характер і не вимагають апаратних реалізацій.

Таблиця 12.2 – Відмінності систем штучного інтелекту від звичайних програмних систем.

Характеристика	Програмування в системах штучного інтелекту	Традиційне програмування
Тип обробки	Символьний	Числовий
Метод	Евристичний пошук	Точний алгоритм
Завдання кроків рішення	Неявне	Явне
Шукане рішення	Задовільне	Оптимальне
Управління і дані	Змішані	Розділені
Знання	Неточні	Точні
Модифікації	Часті	Рідкісні

У найперших ЕС не враховувалася зміна знань, використовуваних в процесі рішення конкретної задачі. Їх назвали статичними ЕС. Типова статична ЕС містить наступні основні компоненти (рис. 12.2): базу знань; робочу пам'ять, звану також базою даних; вирішувач (інтерпретатор); систему пояснень; компоненти придбання знань; інтерфейс з користувачем

База знань ЕС призначена для зберігання довгострокових даних, що описують дану область, і правив, що описують доцільні перетворення даних цієї області.

База даних (робоча пам'ять) служить для зберігання поточних даних вирішуваного завдання.

Вирішувач (інтерпретатор) формує послідовність застосування правив і здійснює їх обробку, використовуючи дані з робочої пам'яті і знання з БЗ.

Система пояснень показує, яким чином система отримала рішення задачі, і які знання при цьому використовувалися. Це полегшує тестування системи і підвищує довіру користувача до отриманого результату.

Компоненти придбання знань необхідні для заповнення ЕС знаннями в діалозі з користувачем-експертом, а також для додавання і модифікації

закладених в систему знань.

До розробки ЕС притягуються фахівці з різних наочних областей, а саме:

– експерти тієї проблемної області, до якої відносяться завдання, вирішувані ЕС;

– інженери по знаннях, що є фахівцями з розробки ІС;

– програмісти, що здійснюють реалізацію ЕС.

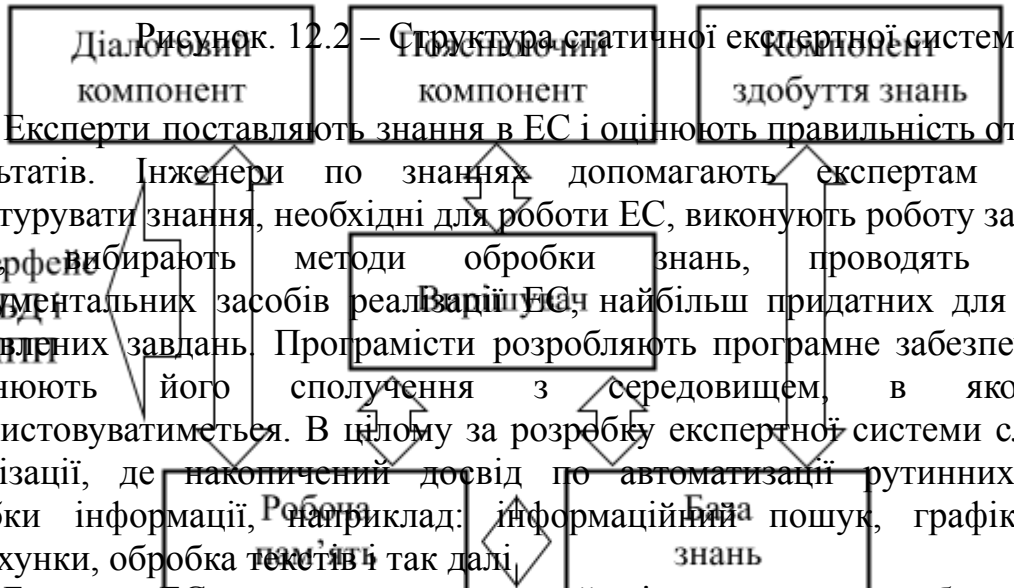


Рисунок. 12.2 – Структура статичної експертної системи.

Експерти поставляють знання в ЕС і оцінюють правильність отримуваних результатів. Інженери по знаннях допомагають експертам виявити і структурувати знання, необхідні для роботи ЕС, виконують роботу за уявленням знань, вибирають методи обробки знань, проводять вибирання інструментальних засобів реалізації ЕС, найбільш придатних для вирішення поставлених завдань. Програмісти розробляють програмне забезпечення ЕС і здійснюють його сполучення з середовищем, в якому воно використовуватиметься. В цілому за розробку експертної системи слід братися організації, де накопичений досвід по автоматизації рутинних процедур обробки інформації, наприклад: інформаційний пошук, графіка, складні розрахунки, обробка текстів і так далі.

Будь-яка ЕС повинна мати, принаймні, два режими роботи. У режимі придбання знань експерт наповнює систему знаннями, які згодом дозволять ЕС самостійно (без допомоги експерта) вирішувати певні завдання з конкретної проблемної області. Експерт описує проблемну область у вигляді сукупності даних і правил. Дані визначають об'єкти, їх характеристики і значення, що існують в області експертизи. Правила визначають взаємні зв'язки, що існують між даними, і способи маніпулювання даними, характерні для даного класу завдань. У режимі консультації користувач ЕС повідомляє системі конкретні дані про вирішуване завдання і прагне отримати з її допомогою результат. Користувачі-неспеціалісти звертаються до ЕС за результатом, не уміючи отримати його самостійно, користувачі-фахівці використовують ЕС для прискорення і полегшення процесу отримання результату. Слід підкреслити, що термін «користувач» є багатозначним, оскільки використовувати ЕС можуть і експерт, і інженер по знаннях, і програміст. Тому, коли хочуть підкреслити, що мова йде про тому, для кого робилася ЕС, використовують термін «кінцевий користувач». У режимі консультації вхідні дані про завдання поступають в робочу пам'ять. Вирішувач на основі вхідних даних з робочої пам'яті і правил з БЗ формує рішення. На відміну від традиційних програм комп'ютерної обробки даних ЕС при рішенні задачі не тільки виконує наказану послідовність операцій, але і сама формує її.

Існує широкий клас додатків, в яких потрібно враховувати зміни, що відбуваються на навколишньому світі за час виконання додатку. Для таких завдань необхідно застосовувати динамічні ЕС. У структуру динамічної ЕС (рис. 12.3) вводяться два компоненти: підсистема моделювання зовнішнього

світу і підсистема зв'язку із зовнішнім оточенням.

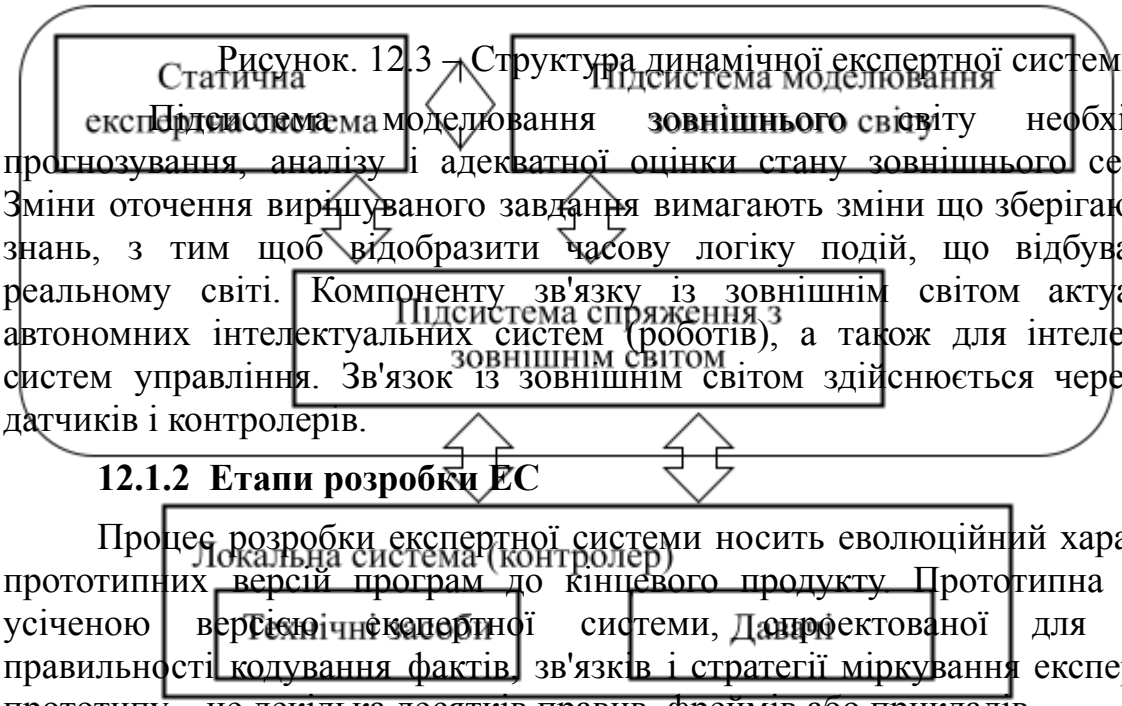


Рисунок. 12.3 – Структура динамічної експертної системи.

Підсистема моделювання зовнішнього світу необхідна для прогнозування, аналізу і адекватної оцінки стану зовнішнього середовища. Зміни оточення вирішуваного завдання вимагають зміни що зберігаються в ЕС знань, з тим щоб відобразити часову логіку подій, що відбуваються на реальному світі. Компоненту зв'язку із зовнішнім світом актуальна для автономних інтелектуальних систем (роботів), а також для інтелектуальних систем управління. Зв'язок із зовнішнім світом здійснюється через систему датчиків і контролерів.

12.1.2 Етапи розробки ЕС

Процес розробки експертної системи носить еволюційний характер – від прототипних версій програм до кінцевого продукту. Прототипна система є усіченою версією експертної системи, спроектованої для перевірки правильності кодування фактів, зв'язків і стратегії міркування експерта. Об'єм прототипу – це декілька десятків правив, фреймів або прикладів.

Промислова технологія створення інтелектуальних систем включає наступні етапи:

- дослідження здійснимості проекту;
- розробку загальної концепції системи;
- розробку і тестування серії прототипів;
- розробку і випробування головного зразка;
- розробку і перевірку розширених версій системи;
- прив'язку системи до реального робочого середовища.

Проектування ЕС засноване на трьох головних принципах:

Потужність експертної системи обумовлена потужністю БЗ і можливостями її поповнення і тільки тоді – використовуваними методами (процедурами) обробки інформації.

Знання, що дозволяють експертові (або експертній системі) отримати якісні і ефективні вирішення завдань, є в основному евристичними, емпіричними, невизначеними, правдоподібними.

Неформальний характер вирішуваних завдань і використовуваних знань робить необхідним забезпечення активного діалогу користувача з ЕС в процесі її роботи.

Перш ніж приступити до розробки ЕС, інженер по знаннях повинен розглянути питання, чи слід розробляти ЕС для даного застосування. Позитивне рішення ухвалюється тоді, коли розробка ЕС можлива, виправдана і методи інженерії знань відповідають вирішуваному завданню. Щоб розробка ЕС була можливою для даного застосування, необхідне виконання, принаймні, наступних вимог:

- існують експерти в даній області, які вирішують задачу значно краще, ніж початкуючі фахівці;

– експерти сходяться в оцінці пропонованого рішення, оскільки інакше буде неможливо оцінити якість розробленої ЕС;

– експерти здатні виразити і пояснити використовувані ними методи, інакше важко розраховувати на те, що знання експертів «витягуватимуть» і закладені в ЕС;

– рішення задачі вимагає тільки міркувань, а не дій;

– завдання не має бути дуже важким (тобто її рішення повинне займати у експерта декілька годин або днів, а не тижнів або років);

– завдання хоча і не має бути виражена у формальному вигляді, але все таки повинна відноситися до достатньо «зрозумілої» і структурованої області, тобто повинна існувати можливість виділення основних понять, стосунків і способів отримання рішення задачі;

– рішення задачі не повинне в значній мірі спиратися на «здоровий глузд» (тобто широкий спектр загальних відомостей про світ і про спосіб його функціонування, які знає і уміє використовувати будь-яка нормальна людина), оскільки подібні знання поки не вдається в достатній кількості закласти в системи штучного інтелекту.

Додаток відповідає методам ЕС, якщо вирішуване завдання володіє сукупністю наступних характеристик:

– завдання може бути природним чином вирішена за допомогою маніпулювання символами (за допомогою символічних міркувань), а не маніпулювання числами, як прийнято в математичних методах і в традиційному програмуванні;

– завдання повинне мати евристичну, а не алгоритмічну природу, тобто її рішення повинне вимагати застосування евристичних правил. Для завдань, які можуть бути гарантовано вирішені (при дотриманні заданих обмежень) за допомогою формальних процедур, існують ефективніші підходи, ніж технології ЕС.

При розробці ЕС, як правило, використовується концепція швидкого прототипу, суть якої полягає в тому, що розробники не намагаються відразу побудувати кінцевий продукт. На початковому етапі вони створюють прототип (можливо, не єдиний) ЕС, що задовольняє двом суперечливим вимогам: уміння вирішувати типові завдання конкретного застосування і незначний час і трудомісткість його розробки. При виконанні цих умов стає можливим паралельно вести процес накопичення і відладки знань, здійснюваний експертом, і процес вибирання (розробки) програмних засобів, що виконується інженером по знаннях і програмістами. Для задоволення вказаним вимогам при створенні прототипу використовуються різноманітні інструментальні засоби, що прискорюють процес проектування.

Традиційна технологія реалізації ЕС включає шість основних етапів (рис. 12.4): ідентифікацію, концептуалізацію, формалізацію, виконання, дослідну експлуатацію.



Рисунок. 12.4 – Етапів розробки експертних систем.

На етапі ідентифікації визначаються завдання, що підлягають рішенню, цілі розробки, експерти і типи користувачів.

На етапі концептуалізації проводиться змістовний аналіз проблемної області, виявляються використовувані поняття і їх взаємозв'язки, визначаються методи вирішення завдань.

На етапі формалізації вибираються інструментальні засоби і способи представлення всіх видів знань, формалізуються основні поняття, визначаються способи інтерпретації знань, моделюється робота системи, оцінюється адекватність системи зафіксованих понять, методів рішення, засобів уявлення і маніпулювання знаннями даної наочної області.

На етапі виконання здійснюється заповнення бази знань. У зв'язку з тим, що основою ЕС є знання, даний етап є одним з найважливіших і самих трудомістких. Процес придбання знань розділяють на витягання знань в Діалозі з експертами; організацію знань, що забезпечує ефективну роботу системи, і представлення знань у вигляді, «зрозумілому» ЕС. Процес придбання знань здійснюється інженером по знаннях на основі аналізу діяльності експерта за рішенням реальних завдань.

На етапі тестування експерт і інженер по знаннях в інтерактивному режимі з використанням діалогових і пояснювальних засобів перевіряють компетентність ЕС. Процес тестування продовжується до тих пір, поки експерт не вирішить, що система досягла необхідного рівня компетентності.

На етапі дослідної експлуатації перевіряється придатність ЕС для кінцевих користувачів. Отримані результати можуть показати необхідність істотної модифікації ЕС.

Процес створення ЕС не зводиться до строгої послідовності перерахованих вище етапів. В ході розробки доводиться неодноразово повертатися на попередні етапи і переглядати ухвалені там рішення.

Інструментальні засоби розрізняються залежно від того, яку технологію розробки ЕС вони допускають. Можна виділити, принаймні, чотири підходи до розробки ЕС:

– підхід, що базується на поверхневих знаннях;
– структурний підхід;
– підхід, заснований на глибинних знаннях;
– змішаний підхід, що спирається на використання поверхневих і глибинних знань.

Поверхневий підхід застосовується для складних завдань, які не можуть бути точно описані. Його суть полягає в отриманні від експертів фрагментів знань, релевантних вирішуваному завданню. При цьому не робиться спроб систематичного або глибинного вивчення області, що зумовлює використання пошуку в просторі станів як універсальний механізм виводу. Зазвичай в ЕС, що використовують даний підхід, як спосіб представлення знань вибираються

правила. Умову кожного правила визначає зразок деякої ситуації, в якій правило може бути виконане. Пошук рішення полягає у виконанні тих правил, зразки яких зіставляються з поточними даними. При цьому передбачається, що в процесі пошуку рішення послідовність формованих таким чином ситуацій не обірветься до отримання рішення, тобто не виникне невідомої ситуації, яка не відповідає жодному правилу. Даний підхід з успіхом застосовується до широкого класу додатків, але виявляється неефективним в тих випадках, коли завдання може структуруватися або для її вирішення може використовуватися деяка модель.

Структурний підхід до побудови ЕС передбачає структуризацію знань проблемної області. Його поява обумовлена тим, що для ряду додатків застосування техніки поверхневих знань не забезпечує рішення задачі. Структурний підхід до побудови ЕС багато в чому схожий на структурне програмування. Проте стосовно ЕС мова йде не про те, щоб структуризація завдання була доведена до точного алгоритму (як в традиційному програмуванні), а передбачається, що частина завдання вирішується за допомогою евристичного пошуку. Структурний підхід в різних застосуваннях доцільно поєднувати з поверхневим або глибинним знанням.

При глибинному підході компетентність ЕС базується на моделі того проблемного середовища, в якому вона працює. Модель може бути визначена різними способами, тобто декларативно або процедурно. Експертні системи, розроблені із застосуванням глибинних знань, при виникненні невідомої ситуації здатні самостійно визначити, які дії слід виконати, за допомогою деяких загальних принципів, справедливих для даної області експертизи. Глибинний підхід вимагає явного опису структури і взаємин між різною суттю проблемної області. У цьому підході необхідно використовувати інструментальні засоби, моделювання, що володіють можливостями: об'єкти з приєднаними процедурами, ієрархічне спадкоємство властивостей, активні знання (програмування, кероване даними), механізм передачі повідомлень об'єктам (об'єктно-орієнтоване програмування) і тому подібне

Змішаний підхід в загальному випадку може поєднувати поверхневий, структурний і глибинний підходи. Наприклад, поверхневий підхід може застосовуватися для пошуку адекватних знань, які потім використовуються деякою глибинною моделлю.

12.2 Класифікаційні ознаки ЕС

У основі класифікації експертних систем лежать наступні параметри: тип додатку, стадія існування, масштаб, тип проблемного середовища, тип вирішуваного завдання.

Тип додатку характеризується наступними ознаками.

1. Можливість взаємодії додатку з іншими програмними засобами

– ізольоване застосування – експертна система, не здатна взаємодіяти з іншими програмними системами (наприклад, з базами даних, електронними таблицями, пакетами прикладних програм, контролерами, датчиками і тому

подібне);

– інтегроване застосування – експертна система і інші програмні системи, з якими вона взаємодіє в ході роботи. Більшість сучасних ЕС, використовуваних для вирішення практично значущих завдань, є інтегрованими.

2. Можливість виконувати додаток на різнорідній апаратурі і переносити його на різні платформи:

– закриті застосування – *виконуються тільки в програмному середовищі даної фірми і можуть бути перенесені на інші платформи тільки шляхом перепрограмування додатку;*

– відкриті застосування – *орієнтовані на виконання в різнорідному програмно-апаратному оточенні і можуть бути перенесені на інші платформи без перепрограмування.*

3. Архітектура додатку:

– централізоване застосування – реалізується на базі центральної ЕОМ, з якою пов'язані термінали;

– розподілене застосування – зазвичай використовується архітектура клієнт-сервер.

Стадія існування характеризує ступінь завершеності розробки ЕС. У неї входять:

– дослідницький прототип – вирішує представницький клас завдань проблемної області, але може бути нестійкий в роботі і не повністю перевірений. За наявності розвинених інструментальних засобів при розробці дослідницького прототипу потрібний приблизно 2-4 місяці. База знань дослідницького прототипу зазвичай містить невелике число тверджень, які виконуються;

– прототип, що діє, – надійно вирішує будь-які завдання проблемної області, але при вирішенні складних завдань може зажадати надмірно багато часу і (або) пам'ять. Доведення системи від початку розробки до стадії прототипу, що діє, вимагає приблизно 6 - 9 місяців, при цьому кількість виконуваних тверджень в базі знань збільшується в порівнянні з дослідницьким прототипом;

– промислова система – забезпечує високу якість вирішення всіх завдань при мінімумі часу і пам'яті. Зазвичай процес перетворення прототипу, що діє, в промислову систему полягає в розширенні бази знань і її ретельній відладці. Доведення ЕС від початку розробки до стадії промислової системи із застосуванням розвинених інструментальних засобів вимагає не менше 12-18 місяців;

– комерційна система – придатна не тільки для використання розробником, але і для продажу різним споживачам. Доведення системи до комерційної стадії вимагає приблизно 1,5-2 року. Приведені тут терміни справедливі для ЕС середньої складності.

Масштаб ЕС характеризує складність вирішуваних завдань і пов'язаний з типом використовуваної ЕОМ.

За цією ознакою розрізняють:

– малі ЕС – призначені для первинного навчання і дослідження можливості застосування технології ЕС для даного класу завдань. Системи такого типу можуть бути реалізовані на персональних комп'ютерах;

– середні ЕС – охоплюють весь спектр необхідних застосувань і зазвичай інтегровані з базами даних, електронними таблицями і так далі. Системи такого масштабу найчастіше реалізуються на робочих станціях;

– великі ЕС – мають доступ до могутніх баз даних і реалізуються на робочих станціях або на спеціалізованих комп'ютерах;

– символьні ЕС – створюються з дослідницькими цілями і реалізуються на спеціалізованих комп'ютерах, орієнтованих на обробку символьних даних.

Поняття проблемного середовища включає опис наочної області (множину сутностей, що описують множини об'єктів, їх характеристик і стосунків між об'єктами) і вирішуваних в ній завдань. Іншими словами, проблемне середовище включає структури даних і вирішувані з ними завдання, представлені у вигляді виконуваних тверджень (правив, процедур, формул і ін.). У зв'язку з цим проблемне середовище визначається характеристиками відповідної наочної області і характеристиками типів вирішуваних в ній завдань.

12.2.1 Характеристики наочної області

1. Тип наочної області:

– статичний – вхідні дані не змінюються за час сеансу роботи додатку, значення інших (не вхідних) даних змінюються тільки самою експертною системою;

– динамічний – вхідні дані, що поступають із зовнішніх джерел, змінюються в часі, значення інших даних змінюються ЕС або підсистемою моделювання зовнішнього оточення.

2. Спосіб опису суті наочної області:

– сукупність атрибутів і їх значень (фіксований склад сутності);

– сукупність класів (об'єктів) і їх екземплярів (змінний склад сутності).

3. Спосіб організації суті в БЗ:

– неструктурована БЗ;

– структуризація суті в БЗ по різних ієрархіях, («приватне – загальне», «частина – ціле», «рід – вид»), що забезпечує спадкоємство властивостей сутностей.

Структуризація БЗ сприяє:

– обмеженню множини сутностей, які повинна розглядатися механізмом виводу, і скороченню кількості варіантів, які переглядаються в процесі вибору рішення;

– забезпеченню спадкоємства властивостей сутностей, тобто передачі властивостей вище розміщених в ієрархії сутностей, розташованим нижче сутностям, що значно спрощує процес придбання і використання знань.

12.2.2 Характеристики типів вирішуваних в проблемній області завдань

Тип вирішуваних завдань:

– завдання аналізу або синтезу. У завданні аналізу задана модель сутності і потрібно визначити невідомі характеристики моделі. У завданні синтезу задаються умови, яким повинні задовольняти характеристики «невідомої» моделі сутності, і потрібно побудувати модель цієї сутності. Рішення задачі синтезу зазвичай включає завдання аналізу як складову частину;

– статичні або динамічні завдання. Якщо завдання, вирішувані ЕС, явно не враховують чинник часу і/або не змінюють в процесі свого рішення знання про навколишній світ, то ЕС вирішує статичні завдання, *інакше* мова йде про вирішенні динамічних завдань. *Враховуючи* значущість часу в динамічних проблемних середовищах, багато фахівців називають їх додатками, що працюють в реальному часі. Зазвичай виділяють наступні системи реального часу: псевдореального *часу*, «м'якого» *реального* часу і «жорсткого» *реального* часу. Системи псевдореального часу, як випливає з назви, не є системами реального часу, проте вони, на відміну від статичних систем, отримують і обробляють дані, що поступають із зовнішніх джерел. Системи псевдореального часу вирішують задачу швидше, ніж відбуваються значущі зміни інформації про навколишній світ.

Спільність виконуваних тверджень:

– часткові виконувані твердження, що містять посилання на конкретну сутність (об'єкти);

– загальні виконувані твердження, що відносяться до будь-якої сутності заданого типу (незалежно від їх числа та імені). Використання загальних тверджень дозволяє значно лаконічніше представляти знання. Проте оскільки загальні твердження не містять явних посилань на конкретну сутність, для їх використання кожного разу потрібно визначати ту сутність, до якої вони повинні застосовуватися.

Не всі поєднання перерахованих вище параметрів, що характеризують проблемне середовище, зустрічаються на практиці. Найбільш поширені наступні типи проблемних середовищ:

– статична наочна область:

– представлення сутностей у вигляді сукупності атрибутів і їх значень, незмінний склад сутності, БЗ не структурована, вирішуються статичні завдання аналізу, використовуються тільки часткові виконувані твердження;

– представлення сутності об'єктами, змінний склад сутності, БЗ структурована, вирішуються статичні завдання аналізу і синтезу, використовуються загальні і часткові виконувані твердження;

– динамічна наочна область:

У ЕС розрізняють наступні типи вирішення завдань:

– інтерпретація даних – процес визначення сенсу даних, результати якого мають бути узгодженими і коректними. Експертні системи, як правило, проводять багатоваріантний аналіз даних;

– діагностика – процес співвідношення об'єкту з деяким класом об'єктів і/або виявлення несправностей в системі (відхилень параметрів системи від нормативних значень);

– моніторинг – безперервна інтерпретація даних в реальному масштабі часу і сигналізація про вихід тих або інших параметрів за допустимі межі;

– проектування – створення раніше не існуючого об'єкту і підготовка специфікацій на створення об'єктів із заздалегідь визначеними властивостями. Ступінь новизни може бути різним і визначається видом знань, закладених в ЕС, і методами їх обробки. Для організації ефективного проектування потрібно формувати не тільки самі проектні рішення, але і мотиви їх ухвалення. ЕС, вирішальні завдання проектування, реалізують процедури виведення рішення і пояснення отриманих результатів;

– прогнозування – прогноз наслідків деяких подій або явищ на основі аналізу наявних даних. Прогнозуючі ЕС логічно виводять вірогідні наслідки із заданих ситуацій. У прогнозуючих ЕС в більшості випадків використовуються динамічні моделі, в яких значення параметрів «підганяються» під задану ситуацію. Наслідки, що виводяться з цих моделей, складають основу для прогнозів з імовірнісними оцінками;

– планування – побудова планів дій об'єктів, здатних виконувати деякі функції. Робота ЕС по плануванню заснована на моделях поведінки реальних об'єктів, які дозволяють проводити логічне виведення наслідків планованої діяльності;

– навчання – використання комп'ютера для навчання яким-небудь дисципліні або предмету. Експертні системи навчання виконують такі функції, як діагностика помилок, підказування правильних рішень; акумуляція знань про гіпотетичного «учня» і його характерні помилки; діагностування слабкості в пізнаннях навчання і знаходження відповідних засобів для їх ліквідації. Системи навчання здатні планувати акт спілкування з учнем залежно від успіхів учня для передачі необхідних знань;

– управління – функція організованої системи, що підтримує відповідний режим її діяльності. Експертні системи даного типу, які призначені для управління поведінкою складних систем відповідно до заданих специфікацій;

– підтримка ухвалення рішень – сукупність процедур, що забезпечує особу, що ухвалює рішення, необхідною інформацією і рекомендаціями, що полегшують процес ухвалення рішення. Такого роду ЕС надають допомогу фахівцям у виборі і/або генерації найбільш раціональної альтернативи з множини можливих при ухваленні відповідальних рішень.

Завдання інтерпретації даних, діагности, підтримка ухвалення рішень відноситься до завдань аналізу, завдання проектування, планування і управління – до завдань синтезу. До комбінованого типу завдань відносяться навчання, моніторинг і прогнозування.

12.2.3 Характеристика інструментальних засобів розробки ІС

Трудомісткість розробки ІС в значній мірі залежить від використовуваних інструментальних засобів (ІЗ). Інструментальні засоби для розробки інтелектуальних застосувань можна класифікувати по наступних основних параметрах:

– рівень використовуваної мови;

– парадигми програмування і механізми реалізації;

– спосіб представлення знань;

– механізми виводу і моделювання;

– засоби придбання знань;

– технології розробки додатків.

Потужність і універсальність мови програмування визначає трудомісткість розробки ЕС:

– традиційні (зокрема об'єктно-орієнтовані) мови програмування типу С, С++ (як правило, вони використовуються не для створення ЕС, а для створення інструментальних засобів);

– спеціальні мови програмування (наприклад, мова LISP**,** орієнтована на обробку списків; мова логічного програмування PROLOG**;** мова рекурсивних функцій РЕФАЛ і ін.). Їх недоліком є слабка пристосованість до об'єднання з програмами, написаними на мовах традиційного програмування;

– інструментальні засоби, що містять багато, але не всі компоненти ЕС. Таке програмне забезпечення призначене для розробників, що володіють технологіями програмування що уміють інтегрувати різнорідні компоненти в програмний комплекс.

– оболонки ЕС загального призначення, що містять всі програмні компоненти, але що не мають знань про конкретні наочні середовища. Засоби цього типу і подальшого не вимагають від розробника додатки знання програмування. Прикладами є АЧ, Leonardo, Nexpert Object, Kappa, EXSYS, GURU, ART, KEE і ін. Останнім часом все рідше вживається термін «оболонка», його замінюють ширшим терміном «середовище розробки». Якщо хочуть підкреслити, що засіб використовується не тільки на стадії розробки додатку, але і на стадіях використання і супроводу, то використовують термін «повне середовище» (complete environment). Для підтримки всього циклу створення і супроводу програм використовуються інтегровані інструментальні системи типу Work Bench, наприклад KEATS, VITAL.

Основними компонентами системи KEATS є:

– ACQUIST – засоби фрагментації текстових джерел знань, що дозволяють розбивати текст або протокол бесіди з експертом на безліч взаємозв'язаних, анотованих фрагментів і створювати поняття (концепти);

– FLIK – мова представлення знань засобами фреймової моделі;

– GIS – графічний інтерфейс, використовуваний для створення гіпертекстів і концептуальних моделей, а також для проектування фреймових систем;

– ERI – інтерпретатор правив, що реалізовує процедури прямого і

зворотного виводу;

– TRI – інструмент візуалізації логічного виводу, що демонструє послідовність виконання правив;

– Tables – інтерфейс маніпулювання таблицями, використовуваними для зберігання знань в БЗ;

– CS – мова опису і розповсюдження обмежень;

– TMS – немонотонна система підтримки істинності.

При використанні інструментарію даного типу можуть виникнути наступні складнощі:

– стратегії, що управляють, які закладені в механізм виводу, можуть не відповідати методам рішення, які використовує експерт, що взаємно-діє з даною системою, що може привести до неефективних, а можливо, і неправильним рішенням;

– спосіб представлення знань, використовуваний в інструментарії, мало підходить для опису знань конкретної наочної області.

Проблемно/предметно-орієнтовані оболонки і середовища (не вимагають знання програмування):

– проблемно-орієнтовані засоби, призначені для вирішення завдань певного класу (завдання пошуку, управління, планування, прогнозування і ін.) і містять відповідні цьому класу функціональні модулі;

– наочно-орієнтовані засоби, що використовують знання про типи наочних областей, що скорочує час розробки БЗ.

При використанні оболонок і середовищ розробник додатку повністю звільняється від програмування, його основні трудовитрати пов'язані з формуванням бази знань.

Способи реалізації механізму виконуваних тверджень часто називають парадигмами програмування. До основних парадигм відносять наступні:

– процедурне програмування;

– програмування, орієнтоване на дані;

– програмування, орієнтоване на правила;

– об'єктно-орієнтоване програмування.

Парадигма процедурного програмування є самій поширеній серед існуючих мов програмування (наприклад, С і Паскаль). У процедурній парадигмі активна роль відводиться процедурам, а не даним; причому будь-яка процедура активізується викликом. Подібні способи завдання поведінки зручні для описів детермінованої послідовності дій одного процесу або декількох взаємозв'язаних процесів.

При використанні програмування, орієнтованого на дані, активна роль належить даним, а не процедурам. Тут із структурами активних даних зв'язують деякі дії (процедури), які активізуються тоді, коли здійснюється звернення до цих даних.

У парадигмі, орієнтованій на правила, поведінка визначається безліччю правил вигляду «умова-дія». Умову задає образ даних, при виникненні якого дія правила може бути виконане. Правила в даній парадигмі грають таку ж роль, як і оператори в процедурній парадигмі. Проте якщо в процедурній парадигмі

поведінка задається детермінованою послідовністю операторів, не залежною від значень оброблюваних даних, то в парадигмі, орієнтованій на правила, поведінка не задається заздалегідь наказаною послідовністю правив, а формується на основі значень Даних, які у нинішній момент обробляються програмою. Підхід, орієнтований на правила, зручний для опису поведінки, гнучко і що різноманітно реагує на велике різноманіття станів даних.

Парадигма об'єктного програмування на відміну від процедурної парадигми не розділяє програму на процедури і дані. Тут програма організовується навколо суті, званої об'єктами, які включають локальні процедури (методи) і локальні дані (змінні). Поведінка (функціонування) в цій парадигмі організовується шляхом пересилки повідомлень між об'єктами. Об'єкт, отримавши повідомлення, здійснює його локальну інтерпретацію, грунтуючись на локальних процедурах і даних. Такий підхід дозволяє описувати складні системи найбільш природним чином, що особливо зручно для інтегрованих ЕС.

Наявність багатьох способів представлення знань викликана прагненням представити різні типи проблемних середовищ з найбільшою ефективністю. Зазвичай спосіб представлення знань в ЕС характеризують моделлю представлення знань. Типовими моделями представлення знань є правила (продукції), фрейми (або об'єкти), семантичні мережі, логічні формули. Інструментальні засоби, що мають в своєму складі більш за одну модель представлення знань, називають гібридними. Більшість сучасних засобів, як правило, використовують об'єктно-орієнтовану парадигму, об'єднану з парадигмою, орієнтованою на правила.

У статичних ЕС єдиним агентом, що змінює інформацію, є механізм виведення експертної системи. У динамічних ЕС зміна даних відбувається не тільки унаслідок функціонування механізму виконуваних тверджень, але також у зв'язку із змінами оточення завдання, які моделюються спеціальною підсистемою або поступають ззовні.

Механізми виводу в різних середовищах можуть відрізнятися способами реалізації наступних процедур.

Структура процесу отримання рішення:

– побудова дерева виводу на основі повчальної вибірки (індуктивні методи придбання знань) і вибір маршруту на дереві виводу в режимі рішення задачі;

– компіляція мережі виводу із специфічних правил в режимі придбання знань і пошук вирішення на мережі виводу в режимі рішення задачі;

– генерація мережі виводу і пошук рішення в режимі рішення задачі, при цьому генерація мережі виводу здійснюється в ході виконання операції зіставлення, що визначає пари «правило – сукупність даних», на яких умови цього правила задовольняються;

– у режимі вирішення завдань ЕС здійснює вироблення правдоподібних припущень (за відсутності достатньої інформації для вирішення); виконання міркувань по обґрунтуванню (спростуванню) пропозицій; генерацію альтернативних мереж виводу; пошук вирішення в

мережах виводу.

Пошук (вибір) рішення:

– напрям пошуку – від даних до мети, від цілей до даних, двонаправлений пошук;

– порядок перебору вершин в мережі виводу

а) «пошук в ширину», при якому спочатку обробляються всі вершини, безпосередньо пов'язані з поточною оброблюваною вершиною G;

б) «пошук в глибину», коли спочатку розкривається одна найбільш значуща вершина – G1, пов'язана з поточною G, потім вершина G1 робиться поточною, і для неї розкривається одна найбільш значуща вершина G2 і так далі.

Процес генерації припущень і мережі виводу:

– режим – генерація в режимі придбання знань, генерація в режимі рішення задачі;

– повнота мережі виводу, що генерується, - операція зіставлення, вживана до всіх правил і до всіх типів вказаної в правилах суті в кожному циклі роботи механізму виводу; використовуються різні засоби для скорочення кількості правив і (або) суті, що бере участь в операції зіставлення; наприклад, застосовується алгоритм зіставлення або використовуються знання більш загального характеру (метазнання).

Механізм виводу для динамічних проблемних середовищ додатково містить: планувальник, керівник діяльністю ЕС відповідно до пріоритетів; засоби, що гарантують отримання кращого рішення в умовах обмеженості ресурсів; систему підтримки істинності значень змінних, що змінюються в часі.

У динамічних інструментальних засобах можуть бути реалізовані наступні варіанти підсистеми моделювання:

– система моделювання відсутня;

– існує система моделювання загального призначення, що є частиною інструментального середовищ;

– існує спеціалізована система моделювання, що є зовнішньою по відношенню до програмного забезпечення, на якому реалізується ЕС.

У інструментальних системах засобу придбання знань характеризуються наступними ознаками:

Рівень мови придбання знань: формальна мова; обмежена природна мова; мова піктограм і зображень; ЕЯ і мова зображень.

Тип знань, що набувають: дані у вигляді таблиць, що містять значення вхідних і вихідних атрибутів, по яких індуктивними методами будується дерево виводу; спеціалізовані правила; загальні спеціалізовані правила.

Тип даних, що набувають: атрибути і значення; об'єкти; класи структурованих об'єктів і їх екземпляри, що одержують значення атрибутів шляхом спадкоємства.

13 Склад і організація даних і знань в ЕС

13.1 Відмінності знань від даних.

Характерною ознакою інтелектуальних систем є наявність знань, необхідних для вирішення завдань конкретної наочної області. При цьому виникає природне питання, що таке знання і чим вони відрізняються від звичайних даних, оброблюваних ЕОМ. Даними називають інформацію фактичного характеру, що описує об'єкти, процеси і явища наочній області, а також їх властивості. У процесах комп'ютерної обробки дані проходять наступні етапи перетворень:

– початкова форма існування даних (результати спостережень і вимірювань, таблиці, довідники, діаграми, графіки і так далі);

– уявлення на спеціальних мовах опису даних, призначених для введення і обробки початкових даних в ЕОМ;

– бази даних на машинних носіях інформації.

Знання є складнішою категорією інформації в порівнянні з даними. Знання описують не тільки окремі факти, але і взаємозв'язки між ними, тому знання іноді називають структурованими даними. Знання можуть бути отримані на основі обробки емпіричних даних. Вони представляють собою результат розумової діяльності людини, направленої на узагальнення його досвіду, отриманого в результаті практичної діяльності.

Для того, щоб наділити ЕС знаннями, їх необхідно представити в певній формі. Існують два способи наділення знаннями програмних систем:

– перший спосіб передбачає помістити знання в програму, написану на звичайній мові програмування. Така система буде єдиним програмним кодом, в якому знання не винесені в окрему категорію. В цьому випадку важко оцінити роль знань і зрозуміти, яким чином вони використовуються в процесі вирішення завдань. Важкою справою є модифікація і супровід подібних програм, а проблема поповнення знань може стати нерозв'язною;

– другий спосіб базується на концепції баз даних і полягає у винесенні знань в окрему категорію, тобто знання представляються в певному форматі і поміщаються в БЗ. База знань легко поповнюється і модифікується. Вона є автономною частиною інтелектуальної системи, хоча механізм логічного виводу, реалізований в логічному блоці, а також засоби ведення діалогу накладають певні обмеження на структуру БЗ і операції з нею. У сучасних ІЇС прийнятий цей спосіб.

При розробці ЕС спочатку здійснюються накопичення і представлення знань, причому на цьому етапі обов'язкова участь людини, а потім знання представляються певними структурами даних, зручними для зберігання і обробки в ЕОМ. Знання в ЕС існують в наступних формах:

– початкові знання (правила, виведені на основі практичного досвіду, математичні і емпіричні залежності, що відображають взаємні зв'язки між фактами; закономірності і тенденції, що описують зміну фактів з часом; функції, діаграми, графи);

– опис початкових знань засобами вибраної моделі представлення знань (множина логічних формул або продукційних правил, семантична мережа, ієрархії фреймів);

– представлення знань структурами даних, які призначені для зберігання і обробки в ЕОМ;

– бази знань на машинних носіях інформації.

В області ЕС даються конкретні визначення знань. «Знання – це закономірності наочної області (принципи, зв'язки, закони), отримані в результаті практичної діяльності і професійного досвіду, дозволяючі фахівцям ставити і вирішувати завдання в цій області». «Знання - це добре структуровані дані або дані про дані, або метадані». «Знання − формалізована інформація, на яку посилаються або використовують в процесі логічного виводу».

Існує множина класифікацій знань. Як правило, за допомогою класифікацій систематизують знання конкретних наочних областей. За своєю природою знання можна розділити на декларативні і процедурні знання. Декларативні знання - це описи фактів і явищ. Процедурні знання − це описи дій, які можливі при маніпулюванні фактами і явищами для досягнення намічених цілей.

Для опису знань на абстрактному рівні розроблені спеціальні мови опису знань. Ці мови також діляться на мови процедурного типу і декларативного. Всі мови описи знань, орієнтовані на використання традиційних комп'ютерів архітектури, є мовами процедурного типу. Розробка мов декларативного типу, зручних для уявлення знань, є актуальною проблемою.

За способом придбання знання можна розділити на факти і евристику (правила, які дозволяють зробити вибір за відсутності точних теоретичних обгрунтувань). Перша категорія знань зазвичай указує на добре відомі в даній наочній області обставини. Друга категорія знань заснована на власному досвіді експерта, що працює в конкретній наочній області, накопиченому в результаті багаторічної практики.

За типом представлення знання діляться на факти і правила. Факти - це знання типу «А - це А, такі знання характерні для баз даних і мережевих моделей. Правила, або продукції, - це знання типу «якщо А, те В».

Окрім фактів і правил існують ще метазнання − знання про знання. Вони необхідні для управління БЗ і для ефективної організації процедур логічного виводу.

Форма представлення знань робить істотний вплив на характеристики ЕС. Бази знань є моделями людських знань. Проте всі знання, які привертає людина в процесі вирішення складних завдань, змоделювати неможливо. Тому в інтелектуальних системах потрібно чітко розділити знання на тих, які призначені для обробки комп'ютером, і знання, використовувані людиною. Очевидно, що для вирішення складних завдань БЗ повинна мати достатньо великий об'єм, інакше неминуче виникають проблеми управління такою базою. Тому при виборі моделі представлення знань слід враховувати такі чинники, як однорідність уявлення і простота розуміння. Однорідність уявлення приводить

до спрощення механізму управління знаннями. Простота розуміння важлива для користувачів інтелектуальних систем і експертів, чиї знання закладаються в ЕС. Якщо форма представлення знань буде важка для розуміння, то ускладнюються процеси придбання і інтерпретації знань.

13.2 Організація даних в робочій пам'яті ЕС

Робоча пам'ять (РП) експертних систем призначена для зберігання даних. Дані в робочій пам'яті діляться на рівні по типах даних. Виділення рівнів ускладнює структуру експертної системи, проте робить її ефективнішою. Наприклад, можна виділити рівень планів, рівень агенди (впорядкованого списку правил, готових до виконання), рівень даних наочної області, рівень рішень. У робочій пам'яті сучасних експертних систем розглядаються як ізольовані, так і зв'язані дані. У першому випадку робоча пам'ять складається з множини простих елементів, а другому − з одного або декількох (при декількох рівнях в РП) складних елементів (наприклад, об'єктів). При цьому складний елемент відповідає множині простих елементів, об'єднаних в єдину суть. Теоретично обидва підходи забезпечують повноту, але використання ізольованих елементів в складних наочних областях приводить до втрати ефективності. Дані в РП в простому випадку є константами і (або) змінними. При цьому змінні можуть розглядатися як характеристики деякого об'єкту, а константи – як значення відповідних характеристик. Якщо в РП потрібно аналізувати одночасно декілька різних об'єктів, що описують поточну проблемну ситуацію, то необхідно указувати, до яких об'єктів відносяться дані характеристики. Одним із способів рішення цієї задачі є явна вказівка того, до якого об'єкту відноситься характеристика. Якщо РП складається з складних елементів, то зв'язок між окремими об'єктами указується явно, наприклад завданням семантичних стосунків. При цьому кожен об'єкт може мати свою внутрішню структуру. Для прискорення пошуку і зіставлення даних в РП використовуються не тільки логічні, але і асоціативні зв'язки.

За показник інтелектуальності системи з погляду представлення знань вважається здатність системи використовувати в потрібний момент необхідні (релевантні) знання. Системи, що не мають засобів для визначення релевантних знань, неминуче стикаються з проблемою «комбінаторного вибуху». Можна стверджувати, що ця проблема є одній з основних причин, що обмежують сферу застосування експертних систем. У проблемі доступу до знань можна виділити три аспекти: зв'язність знань і даних, механізм доступу до знань і спосіб зіставлення.

13.3 Організація знань в БЗ

Зв'язність знань є основним способом, що забезпечує прискорення пошуку релевантних знань. Для цього знання слід організувати навколо найбільш важливих об'єктів (суті) наочної області. Всі знання, що характеризують деяку суть, зв’язуються і представляються у вигляді окремого об'єкту. При подібній організації знань, якщо системі було потрібно інформацію

про деяку суть, то вона шукає об'єкт, що описує цю суть, а потім вже усередині об'єкту відшукує інформацію про дану суть.

У об'єктах доцільно виділяти два типи зв'язок між елементами: зовнішні і внутрішні. Внутрішні зв'язки об'єднують елементи в єдиний об'єкт і призначені для виразу структури об'єкту. Зовнішні зв'язки відображають взаємозалежності, що існують між об'єктами в області експертизи.

Зовнішні зв'язки бувають логічні і асоціативні. Логічні зв'язки виражають семантичні стосунки між елементами знань. Асоціативні зв'язки призначені для забезпечення взаємозв’язків, сприяючих прискоренню процесу пошуку релевантних знань. Основною проблемою при роботі з великою базою знань є проблема пошуку знань, релевантних вирішуваному завданню. Очевидно, що впорядкування і структуризація знань можуть значно прискорити процес пошуку. Знаходження бажаних об'єктів в загальному випадку доречно розглядати як двохетапний процес. На першому етапі, що відповідає процесу вибору по асоціативних зв'язках, здійснюється попередній вибір в базі знань потенційних кандидатів як бажані об'єкти. На другому етапі шляхом виконання операції зіставлення потенційних кандидатів з описами кандидатів здійснюється остаточний вибір шуканих об'єктів. При організації подібного механізму доступу виникають певні труднощі з вибором критерію придатності кандидата, організації роботи в конфліктних ситуаціях і ін. Операція зіставлення може використовуватися не тільки як засіб вибору потрібного об'єкту з безлічі кандидатів, але і для класифікації, підтвердження, декомпозиції і корекції. Для ідентифікації невідомого об'єкту він може бути зіставлений з деякими відомими зразками. Це дозволить класифікувати невідомий об'єкт як такий відомий зразок, при зіставленні з яким були отримані кращі результати. Якщо здійснювати зіставлення деякого відомого об'єкту з невідомим описом, то у разі успішного зіставлення буде здійснена часткова декомпозиція опису.

Операції зіставлення вельми різноманітні. Зазвичай виділяють наступні їх форми: синтаксичне, параметричне, семантичне і вимушене зіставлення.

У разі синтаксичного зіставлення співвідносять форми (зразки), а не зміст об'єктів, Успішним є зіставлення, в результаті якого зразки виявляються ідентичними. Зазвичай вважається, що змінна одного зразка може бути ідентична будь-якій константі (або виразу) іншого зразка. Іноді на змінні, що входять в зразок, накладають вимоги, що визначають тип констант, з якими вони можуть зіставлятися Результат синтаксичного зіставлення є бінарним: зразки зіставляються або не зіставляються.

У параметричному зіставленні вводиться параметр, що визначає ступінь зіставлення.

У разі семантичного зіставлення співвідносяться не зразки об'єктів, а їх функції.

У разі примушуваного зіставлення один зразок, що зіставляється, розглядається з погляду іншого. На відміну від інших типів зіставлення тут завжди може бути отриманий позитивний результат. Питання стоїть в силі примушення і використанні спеціальних процедур, що пов'язуються з об'єктами.

13.4 Моделі представлення знань в ЕС

Вибір моделі представлення знання часто зводять до обговорення балансу між декларативним (ДП) і процедурним представленням (ПП). Відмінність між ДП і ПП можна виразити відмінністю між питаннями «знати, що?» і «знати, як?». Процедурне уявлення засноване на передумові, що інтелектуальна діяльність є знання проблемного середовища, вкладене в програми, тобто знання про те, як можна використовувати ту або іншу суть. Декларативне уявлення засноване на передумові, що знання якоїсь суті «знати, що?» не має глибоких зв'язків з процедурами, що використовуються для обробки цієї суті. При використанні ДП вважається, що інтелектуальність базується на деякій універсальній множині процедур, оброблювальних факти будь-якого типу, і на множині специфічних фактів, що описують приватну галузь знань. Основна гідність ДП в порівнянні з ПП полягає в тому, що в ДП немає необхідності указувати спосіб використання конкретних фрагментів знання. Прості твердження можуть використовуватися декількома способами, і може виявитися незручним фіксувати ці способи заздалегідь. Вказану властивість забезпечує гнучкість і економічність ДП, оскільки дозволяє по-різному використовувати одні і ті ж факти. У ДП знання розглядається як множина незалежних або слабо залежних фактів, що дозволяє здійснювати модифікацію знань і навчання простим додаванням або усуненням тверджень. Для ПП проблема модифікації значно складніша, оскільки тут необхідно враховувати, яким чином використовується дане твердження. Проте відомо, що існує значна кількість суті, яку зручно представити у вигляді процедур і вельми важко, - в чисто декларативному уявленні. Бажання використовувати достоїнства ДП і ПП привело до розробки формалізмів, що використовують змішане уявлення, тобто декларативне уявлення з приєднаними процедурами (наприклад, фрейм-уявлення або мережі з приєднаними процедурами) або процедурне уявлення у вигляді модулів з декларативними зразками. У найбільш доконаному виді ця проблема реалізована в об’єктно-орієнтованому підході.

Моделі представлення знань зазвичай ділять на логічні (формальні) і евристичні (формалізовані) моделі. У логічних моделях, як правило, використовується числення предикатів першого порядку (те, що в думці висловлюється про предмет думки), доповнене рядом евристичних стратегій. Ці методи є системами дедуктивного типу, тобто в них використовується модель отримання виводу із заданої системи посилок за допомогою фіксованої системи правил виводу. Подальшим розвитком предикативних систем є системи індуктивного типу, в яких правила виводу породжуються ЕС на основі обробки кінцевого числа повчальних прикладів. У логічних моделях представлення знань стосунки, що існують між окремими одиницями знань, виражаються тільки за допомогою синтаксичних правил використовуваної формальної системи. На відміну від формальних моделей евристичні моделі мають різноманітний набір засобів, передавальних специфічні особливості тієї або іншої проблемної області. Саме тому евристичні моделі перевершують логічні як по можливості адекватно представити проблемне середовище, так і по

ефективності використовуваних правил виводу. До евристичних моделей, використовуваних в експертних системах, можна віднести мережеві, фреймові, продукційні і об'єктно-орієнтовані моделі. Слід зазначити, що продукційні моделі, використовувані для представлення знань в експертних системах, відрізняються від формальних продукційних систем тим, що вони використовують складніші конструкції правил, а також містять евристичну інформацію про специфіку проблемного середовища, що виражається часто у вигляді семантичних структур.

До типових моделей представлення знань відносяться:

– продукційні моделі;
– семантичні мережі;
– фрейми;
– формальні логічні моделі.

У свою чергу цю множину класів можна розбити на дві великі групи:

– модульні (продукційні моделі і формальні логічні моделі) використовуються для представлення поверхневих знань. Поверхневі знання − знання про видимі взаємозв'язки між окремими подіями і фактами в наочній області;

– мережеві (семантичні мережі і фрейми) використовуються для представлення глибинних знань. Глибинні знання − абстракції, аналогії, схеми, структури, що відображають природу процесів, що протікають в наочній області. Ці знання пояснюють явища і можуть використовуватися для прогнозування поведінки об'єктів.

Таким чином, модульні мови оперують окремими не зв'язаними елементами знань, до них відносяться правила і аксіоми наочної області. Мережеві мови дають можливість зв'язувати ці елементи або фрагменти знань через стосунки в семантичні мережі або мережі фреймів. Найбільш поширеною мовою представлення знань (МПЗ) в ЕС є продукційні моделі. Продукційна модель часто застосовується в експертних системах завдяки своїй наочності, високій модульності, легкістю внесення доповнень і змін, простотою механізму логічного виводу. Є велике число програмних засобів, що реалізовують продукційний підхід, наприклад мова високого рівня CLIPS.

14 Методи придбання знань

14.1 Основні аспекти процесу витягання знань

Витяганням знань називають процес отримання знань від експертів. Витягання знань – це складна і трудомістка процедура, в результаті якої інженерові по знаннях необхідно створити власну модель наочної області на основі інформації, отриманої від експертів. Процес витягання знань розглядають в трьох основних аспектах: психологічному, лінгвістичному і гносеологічному (рис. 14.1).

Психологічний аспект найважливіший зі всіх аспектів, оскільки витягання знань відбувається в процесі спілкування інженера по знаннях з експертами, де психологія грає домінуючу роль. Процес витягання знань для інтелектуальних систем необхідно організувати не як однонаправлений процес передачі повідомлень від експерта аналітику, а як сумісний пошук істини. В процесі розмовного спілкування багато інформації втрачається, тому важливою проблемою є збільшення інформативності спілкування аналітика і експерта за рахунок використання методик, вироблених в психології (рис. 14.2).

Модель спілкування включає учасників спілкування, засоби спілкування і предмет спілкування (знання). Відповідно до цих компонентів виділяються три шару психологічних проблем: контактний, процедурний, когнітивний.

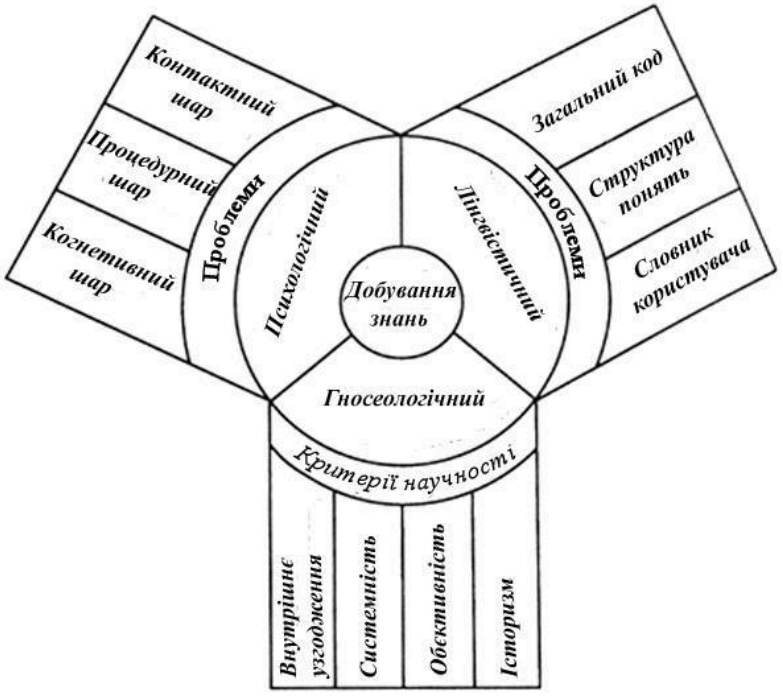


Рисунок 14.1 – Аспекти витягання знань

Ступінь інформативності спілкування аналітика і експерта на рівні контактного шару залежить в основному від підлоги, віку, темпераменту особи і мотивації учасників спілкування. Встановлено, що добрі результати дають гетерогенні пари (чоловік/жінка) і співвідношення віків:

$5 \le (B_e - B_a) \le 20$,

де B_e – вік експерта; B_a – вік аналітика.

Бажано, щоб учасники процесу спілкування володіли наступними якостями: доброзичливістю, хорошою пам'яттю, увагою, спостережливістю, уявою, вразливістю, зібраністю, наполегливістю, товариськістю і винахідливістю.



Рис. 4.2 – [illegible] ти інформації в [illegible] озмовному спілкуванні.

[illegible] амки [illegible] ктно[illegible] ару н[illegible] ьш п[illegible] ум[illegible] ажни[illegible] отирьох класичних типів темпер[illegible] сангвініки і хо[illegible] ки.

[illegible] ієрархічного процедурного шару забезпечують [illegible] ивність витягання знань [illegible] ц[illegible] ам[illegible] ід[illegible] [illegible] я с[illegible] вання (місце, час, три[illegible] у[illegible] ан[illegible] ом[illegible] со[illegible] тле[illegible] меблі); професійні при[illegible] ем[illegible] ь, [illegible] і[illegible] ля[illegible] ення ефективності процесу вит[illegible] на[illegible] ен[illegible] на[illegible] ви[illegible] бр[illegible] чущі для експерта стимули, оскільки експерт передає аналітику один з найцінніших ресурсів - знання.

Когнітивний [illegible] м сем[illegible] простору пам'яті експ[illegible] відт[illegible] ним [illegible] [illegible] стру[illegible] оделі міркувань. Бажано, щоб і аналітики, і експерти володіли наступними когнітивними характеристиками:

– високою поленезалежністю, яка має на увазі здатність виділяти головні аспекти даної проблеми і відкидати все зайве, що не відноситься до поставленого завдання;

– рефлексивністю, що характеризує схильність до розсудливості і самоаналізу (тоді як імпульсна характеризується швидким, часто недостатньо обґрунтованим ухваленням рішень);

– когнітивною еквівалентністю, що визначає здібність людини до розрізнення понять і розбиття їх на класи і підкласи;

– експерти – стійкістю уявлень, тобто жорсткістю структури сприйняття, а аналітики – гнучкістю, тобто умінням легко пристосовуватися до нової обстановки.

Для ефективної побудови ІС інженер по знаннях повинен володіти спеціальними неформальними методами і математичним апаратом, що дозволяють йому відтворювати отримані від експерта знання за допомогою різних моделей, наприклад, таких, як продукційна або фреймова. Не нав'язуючи експертові якої-небудь моделі, аналітик повинен підібрати засоби представлення знань, що мають максимально високу семантичну репрезентативність (показність).

Лінгвістичний аспект визначається тим, що мова є основним засобом спілкування в процесі витягання знань. В області лінгвістичних проблем найбільш важливими є поняття: загальний код, понятійна структура, словник користувача.

Загальним кодом називають спеціальну проміжну мову спілкування між

експертом і інженером по знаннях. Цю мову включає сукупність загальнонаукових і спеціальних понять з професійної літератури, елементів базової мови, неологізмів і ін. Загальний код дозволяє подолати мовний бар'єр в процесі спілкування інженерів по знаннях з експертами. Вироблення загальної коди для партнерів здійснюється відповідно до інформаційних потоків, представлених на рис. 14.3. Надалі загальний код перетвориться в понятійну структуру, або семантичну мережу, яка зв'язує поняття, що зберігаються в пам'яті людини. Виявлення стосунків між елементами знань, представлених поняттями, є одній з найскладніших проблем в процесах витягання знань.

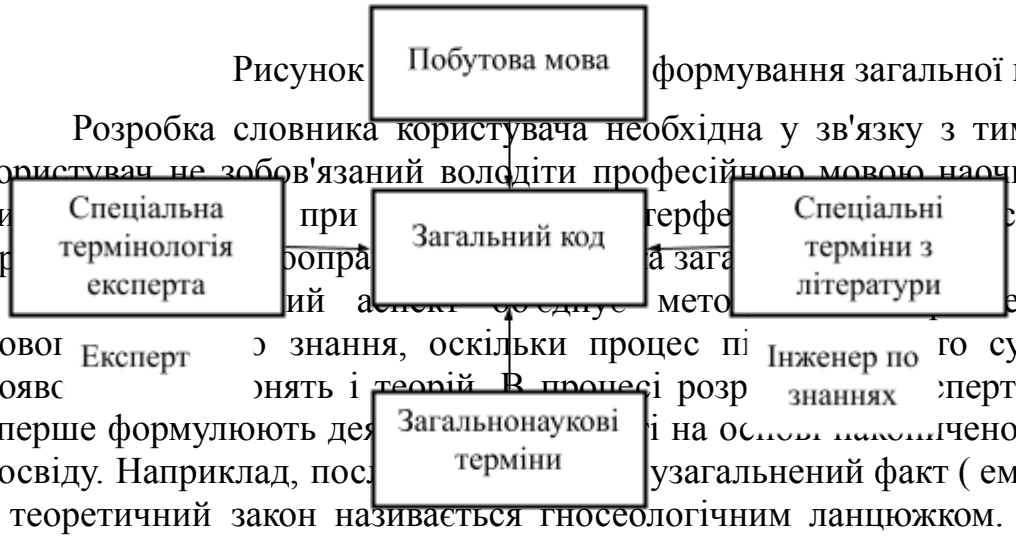


Рисунок формування загальної коди

Розробка словника користувача необхідна у зв'язку з тим, що кінцевий користувач не зобов'язаний володіти професійною мовою наочної області, яка ви при терфе створиється, як пр ропра а заг ий аспект об'єднує мето еми отримання нового знання, оскільки процес п го супроводжується появ онять і теорій. В процесі розр сперти досить часто вперше формулюють дея і на основі накопиченого емпіричного досвіду. Наприклад, пос узагальнений факт (емпіричний закон (теоретичний закон називається гносеологічним ланцюжком. Теорія - це не тільки система узагальнення накопичених знань, але і спосіб отримання нового знання.

В процесі витягання знань інженерів по знаннях перш за все цікавлять емпіричні знання експертів, що є результатами спостережень, які можуть виявитися неузгодженими. Внутрішня узгодженість емпіричних знань характеризується поняттями модальності, суперечності, неповноти. Під модальністю знання розуміється можливість його існування в різних категоріях. Суперечність є природною властивістю емпіричних знань і не завжди може і має бути усунена. Вона може служити відправною крапкою в міркуваннях експертів. Неповнота знань пов'язана з неможливістю вичерпного опису будь-якої наочної області.

На початкових етапах інженер по знаннях, досліджуючи структуру висновків експерта, може використовувати різні теорії і підходи для побудови формальної моделі знань наочної області. Найбільш відомими і часто вживаними прийомами є математична логіка, теорія асоціацій. Математична логіка формує критерії, які гарантують точність, значущість і несуперечність загальних понять, міркувань і виводів У теорії асоціацій мислення представляється у вигляді ланцюжка ідей, зв'язаних загальними поняттями. Тут застосовуються наступні прийоми: асоціації, придбані на основі зв'язків різної природи; залучення минулого досвіду; метод проб і помилок з випадковим успіхом; звичні («автоматичні») реакції і ін.

14.2 Методи витягання знань

Різноманіття завдань, ситуацій і джерел знань зумовило появу великої кількості методів витягання, придбання і формування знань. На першому рівні можливої класифікації методів витягання знань (рис. 14.4) виділено два великі класи. Перший клас утворюють комунікативні методи, які орієнтовані на безпосередній контакт інженера по знаннях з експертом (джерелом знань), другий клас – методи текстологій, засновані на придбанні знань з документів і спеціальної літератури.

Комунікативні методи розділяються на пасивних і активних. У пасивних методах провідну роль грає експерт, а в активних - інженер по знаннях. При вирішенні конкретних завдань, як правило, використовуються як пасивні, так і активні методи. Активні методи діляться на індивідуальних і групових.



Рисунок 14.4 – [illegible] знань

У групових методах знання отримують від множини експертів, а в індивідуальних [illegible] серта. Індивідуальні [illegible] ширше застосування на [illegible] порівнянні з груповими.

Пасивні комунікативні методи включають спостереження, аналіз прото[illegible]олос», про[illegible]я знань з лекцій.

Метод спостереження є одним з найбільш вживаних на початк[illegible]х розробі[illegible] си[illegible] по[illegible]ції всіх [illegible], його рез[illegible]ик і пояснень. При цьому інженер по знаннях не втручає[illegible]у експерт[illegible]остер[illegible]м вирі[illegible]ьних за[illegible]а вирішенням проблем, що [illegible]еальні [illegible]. Спост[illegible]а процес[illegible] реальних завдань дозво[illegible]енерові по знаннях глибше зрозуміти наочну [illegible] експе[illegible]випадку випробовує велику психологічну нап[illegible]очи, [illegible]є не тільки свою професійну діяльність, [illegible]струє [illegible]ерові по знаннях. Спостереження за імітацією процесу знімає ця напруга, але призводить до зниження повноти і якості витягуваних даних. Спостереження за імітацією незамінні в тих випадках, коли спостереження за реальним процесом неможливі із-за специфіки наочної області, що вивчається.

Метод аналізу протоколів «думок вголос» відрізняється від методу спостереження тим, що експерт не тільки коментує свої дії, але і пояснює ланцюжок своїх міркувань, що приводять до рішення. Основною проблемою, що виникає при використанні цього методу, є принципова складність для будь-якої людини словесного опису власних думок і дій. Підвищити повноту і якість витягуваних знань можна за рахунок багатократного уточнюючого протоколювання міркувань експерта.

Метод витягання знань з лекцій припускає, що експерт передає свій

досвід інженерові по знаннях у формі лекцій. При цьому інженер по знаннях може заздалегідь сформулювати теми лекцій. Якщо цього не вдається зробити, то інженер по знаннях конспектує лекції і ставить питання. Якість інформації, наданої експертом в ході лекції, визначається чіткістю сформульованої теми, а також здібностями лектора в структуризації і викладі своїх знань і міркувань.

Наочні області відрізняються рівнем документування і структуризації. Для характеристики наочної області по рівню документування виділяють три класи: добре документовані, средньо документовані і слабо документовані області (рис. 14.5). По ступеню структуризації знань наочні області можуть бути:

– добре структурованими (з чіткою аксіоматизацією, широким застосуванням математичного апарату, сталою термінологією);

– средньо структурованими (з термінологією, що визначилася, теорією, яка розвивається, явними взаємозв'язками між явищами);

– слабо структурованими (з розмитими визначеннями, багатим емпіричним матеріалом, прихованими взаємозв'язками).

Активні індивідуальні методи включають методи анкетування, інтерв'ювання, вільного діалогу і гри з експертом. Перевагою методів анкетування є те, що анкета або запитальник складаються інженером по знаннях заздалегідь і використовуються для опиту експертів. Складання анкети слід проводити з урахуванням рекомендацій, вироблених в соціології і психології. Основними вимогами до анкет є наступні:

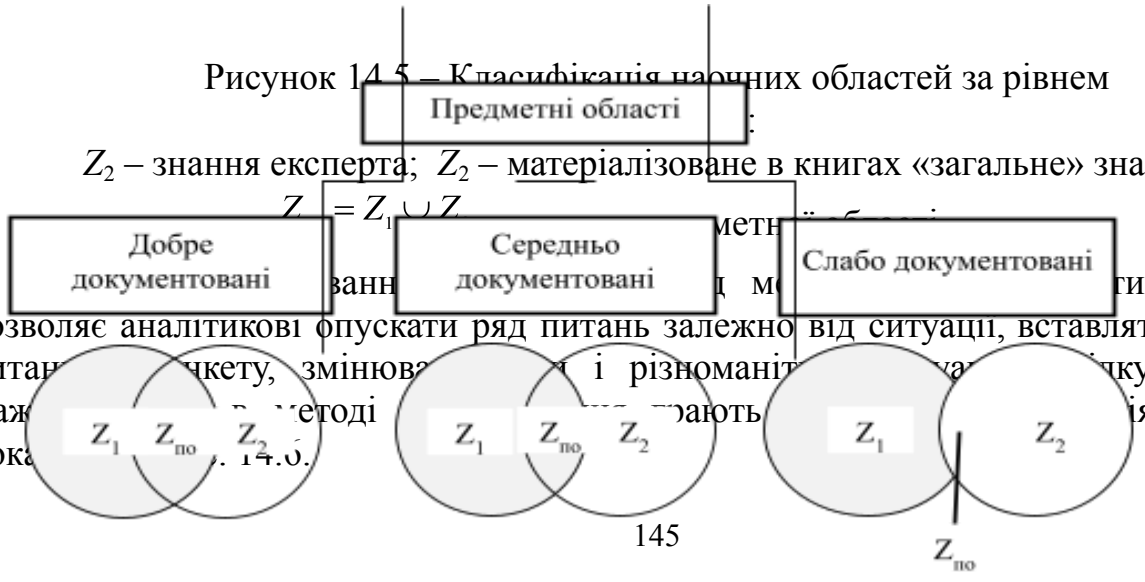
– Анкета не має бути монотонною і одноманітною, щоб не викликати нудьгу або втому. Для цього необхідно різноманітити тематику і форму ставлення питань, застосувати стиль гри;

– Анкета має бути пристосована до мови експерта;

– Слід враховувати, що питання впливають один на одного, тому важливо розташувати їх в правильній послідовності.

– У анкеті повинне міститися оптимальне число надмірних питань, частина яких призначена для контролю правильності відповідей, а інша частина - для зняття напруги.

Рисунок 14.5 – Класифікація наочних областей за рівнем

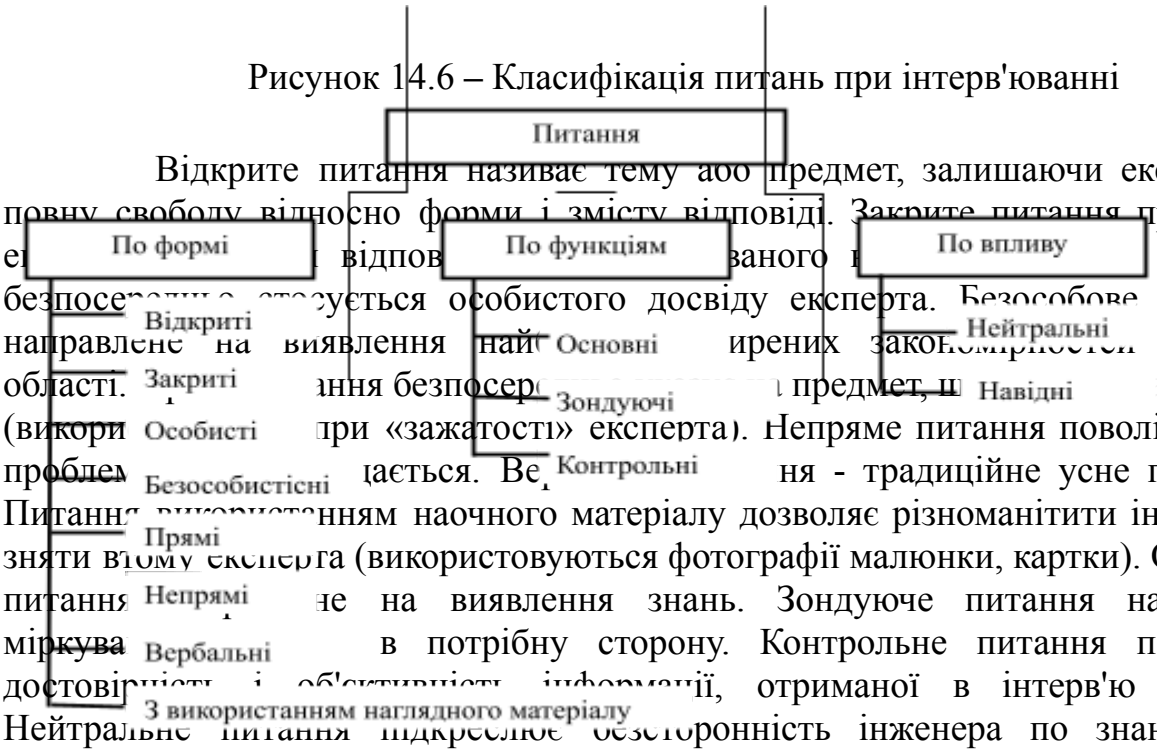


Z_2 – знання експерта; Z_2 – матеріалізоване в книгах «загальне» знання;

$Z = Z_1 \cup Z$

тим, що дозволяє аналітикові опускати ряд питань залежно від ситуації, вставляти нові питан кету, змінюва і різноманіт кування. Важ етоді рають ія яких пока . 14.6.

Рисунок 14.6 – Класифікація питань при інтерв'юванні



Відкрите питання називає тему або предмет, залишаючи експертові повну свободу відносно форми і змісту відповіді. Закрите питання пропонує е відпов ваного питання безпосе ється особистого досвіду експерта. Безособове питання направлене на виявлення най ирених закономірностей наочної області. ання безпосер предмет, ш або тему (викори при «зажатості» експерта). Непряме питання поволі зачіпає пробле ається. Ве ня - традиційне усне питання. Питання використанням наочного матеріалу дозволяє різноманітити інтерв'ю і зняти втому експерта (використовуються фотографії малюнки, картки). Основне питання не на виявлення знань. Зондуюче питання направляє міркува в потрібну сторону. Контрольне питання перевіряє достовірність і об'єктивність інформації, отриманої в інтерв'ю раніше. Нейтральне питання підкреслює безсторонність інженера по знаннях до предмету дослідження. Навідне питання орієнтує експерта взяти до уваги позицію інженера по знаннях. Додатково в інтерв'ю рекомендується включати наступні питання: контактні (що знімають психологічний бар'єр між аналітиком і експертом), буферні (окремі теми інтерв'ю, що розмежовують), оживляючі пам'ять експертів (реконструюючі окремі випадки з практики), «провокуючі» (сприяючі отриманню непідготовлених відповідей). При використанні методу інтерв'ювання слід мати на увазі, що його ефективність багато в чому визначається мовою питань (зрозумілістю, лаконічністю, термінологією); порядком питань (логічна послідовність); доцільністю питань (етичністю і ввічливістю). Перш ніж готувати питання, аналітик повинен оволодіти ключовим набором знань досліджуваної наочної області, оскільки будь-яке питання має сенс тільки в контексті.

Метод вільного діалогу дозволяє витягувати знання у формі бесіди з експертом, тому тут не передбачається використання жорсткого запитальника або плану. В той же час підготовка до вільного діалогу повинна проводитися по спеціальній методиці, в яку входить загальна, спеціальна, конкретна і психологічна підготовка. Загальна підготовка направлена на підвищення наукової ерудиції, оволодіння загальною культурою, знайомство з системною методологією. Спеціальна підготовка зводиться до оволодіння теорією і навиками інтерв'ювання. Конкретна підготовка припускає вивчення наочної області, підготовку ситуації спілкування, знайомство з експертом, тестування експерта. Психологічна підготовка включає знайомство з теорією спілкування і з когнітивною психологією.

Ігри з експертом істотно відрізняються від приведень вище індивідуальних активних методів витягання знанні розглядаються в класі групових активних методів, де особливе місце належить ролевим і експертним методам. Активні групові методи включають «мозковий штурм», дискусії за

круглим столом і ролеві ігри. Групові методи дозволяють творчо інтегрувати знання безлічі експертів. Метод «мозкового штурму» - один з найбільш відомих і широко вживаних методів генерування нових ідей шляхом творчої співпраці групи фахівців. Будучи в деякому розумінні єдиним мозком, група намагається штурмом подолати труднощі, що заважають вирішити дану проблему. В процесі такого штурму учасники висувають і розвивають власні ідеї, стимулюючи появу нових і комбінуючи їх. Для забезпечення максимального ефекту «мозковий штурм» повинен підкорятися певним правилам і грунтуватися на строгому розділенні в часі процесу висунення ідей і процесу їх обговорення і оцінки. На першій стадії штурму забороняється засуджувати висунуті ідеї і пропозиції (вважається, що критичні зауваження відводять до частковостей, переривають творчий процес, заважають висуненню ідей). Роль аналітика полягає в тому, щоб активізувати творче мислення учасників засідання і забезпечити висунення можливо більшого числа ідей. Після висунення ідей виконуються ретельне їх обговорення, оцінка і відбір кращих. На стадії обговорення учасники «мозкового штурму» повинні концентруватися на позитивних сторонах ідей, знайти в них раціональні зерна і запропонувати напрями їх розвитку. Додаткові ідеї, що висуваються в процесі обговорення, можуть базуватися на ідеях інших учасників або, навпаки, служити для них фундаментом, каталізатором. Значний ефект дає комбінування ідей при одночасному виявленні переваг і недоліків варіантів, що синтезуються при цьому. Метод «мозкового штурму» ефективний при вирішенні не дуже складних завдань загального організаційного характеру, коли проблема добре знайома всім учасникам засідання і з даного питання є достатня інформація. Індивідуальний «мозковий штурм» проводиться по тих же правилах, що і колективний, але виконується одним експертом, який одночасно генерує ідеї, дає їм об'єктивну оцінку і критикує їх. Масовий «мозковий штурм» проводиться в масовій аудиторії (до декількох десятків чоловік). Відбір ідей проводиться на проміжних етапах. Експерти групуються по 6 - 8 чоловік, при цьому важливо, щоб безпосереднє відношення до завдання мав лише керівник групи, а останні були лише знайомі з нею (інакше амбіції можуть зіграти негативну роль). Штурм проводиться в два етапи. На першому етапі оперативні групи здійснюють прямий колективний «мозковий штурм» При цьому бажано, щоб кожна група працювала над завданням, найбільш близьким з тематики до профілю фахівців, що увійшли до неї. На другому етапі керівники кожної групи протягом декількох хвилин оцінюють висунуті ідеї, відбирають з них найбільш цікаві і повідомляють їх на «пленарному засіданні». Подвійний «мозковий штурм» органічно сполучає в собі процеси генерування ідей і їх доброзичливої позитивної критики. Зворотний «мозковий штурм» відрізняється від прямого тим, що в нім більше уваги приділяється критиці висловлених ідей.

Метод дискусії за круглим столом припускає рівноправне обговорення експертами поставленої проблеми. Відмітною особливістю методу дискусії є колективний розгляд наочної області з різних точок зору і дослідження спірних гіпотез.

Експертні ігри призначені для витягання знань і базуються на ділових, діагностичних і комп'ютерних іграх. По числу учасників гри підрозділяють на

індивідуальних (ігри з експертом) і групових (ролеві ігри в групі). По застосуванню спеціального устаткування - ігри з тренажерами і ігри без реквізиту. Особливим класом є комп'ютерні ігри. У іграх з експертом інженер по знаннях бере на себе чию-небудь роль в модельованій ситуації. Ролеві ігри в групі передбачають участь в грі декількох фахівців. Учасники гри наділяються певними ролями, а власне гра проводиться по складеному інженером по знаннях сценарію. Комп’ютерні експертні ігри в даний час використовуються в основному в цілях навчання. Вони корисні для «розминки» експертів перед сеансом витягання знань.

Методи текстологій включають методи витягання знань засновані на вивченні текстів підручників, спеціальної літератури і документів. Простий алгоритм витягання знань з текстів включає наступні кроки: складання основного списку літератури для ознайомлення з наочною областю; вибір тексту для витягання знань; збігле знайомство з текстом, проведення консультацій з фахівцями для визначення значень незнайомих слів; формування першої гіпотези про макроструктуру тексту; визначення сенсу прочитаного тексту з випискою ключових слів; визначення зв'язку між ключовими словами, складання реферату; формування нового представлення знань на підставі макроструктури тексту.

ПЕРЕЛІК РЕКОМЕНДОВАНИХ ДЖЕРЕЛ

1. Глибовець М.М., Олецький О.В. Штучний інтелект. Підручник для студ. вищ. навч. закладів, що навчаються за спец. „Комп'ютер. науки" та „Приклад. математика".– К.: Вид.дім „КМ Академія", 2002.– 336с.

2. Адаменко А.Н., Кучуков А.М. Логическое программирование и Visual Prolog.– СПб.: БХВ-Петербург, 2003.– 9921 с.

3. Маллас Дж. Реляционный язык Пролог и его применение.– М.: Наука, 1990.– 464 с.

4. Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с анг.– М.: Мир, 1990.– 560 с.

5. Лорьер Ж.-Л. Системы искусственного интеллекта: Пер. с франц.– М.: Мир, 1991.– 568 с.

6. Девятков В.В. Системы искусственного интеллекта: Учеб. пособие для вузов. − М.: Изд-во МГТУ им. Н.Э.Баумана, 2001. − 352 с., ил. (Сер. Информатика в техническом университете.

7. Базы знаний интеллектуальных систем/ Т.А.Гаврилова, В.Ф.Хорошевский − Спб: Питер, 2000. − 384 с.

8. Люгер, Джордж, Ф. Искусственный интелект: стратегии и методы решения сложных проблем, 4-е издание.: Пер. с англ.− М.: Издательский дом «Вильямс», 2003.− 864 с.