4. ПРЕДСТАВЛЕННЯ ЗНАНЬ З ВИКОРИСТАННЯМ ЛОГІКИ ПРЕДИКАТІВ

4.1. Логічні моделі й логічне програмування

Одним зі способів представлення знань є мова математичної логіки, що дозволяє формально описувати поняття предметної області й зв'язки між ними.

На відміну від природної мови, яка дуже складна, мова логіки предикатів використовує тільки такі конструкції природної мови, які легко формалізуються.

Тобто логіка предикатів - це мовна система, яка оперує із реченнями природною мовою в межах синтаксичних правил цієї мови.

Мова логіки предикатів використовує *слова*, які описують:

- поняття й об'єкти досліджуваної предметної області;
- властивості цих об'єктів і понять, а також їхнє поводження й стосунки між ними.

У термінах логіки предикатів перший тип слів називається *термами* (поняття), а другий – *предикатами*(властивості та стосунки).

Терми являють собою засоби для позначення індивідуумів, які нас цікавлять, а ***предикати*** виражають відносини між індивідуумами (які позначаються за допомогою термів).

Логічна модель - це множина пропозицій , що виражають різні логічні властивості іменованих відносин.

При логічному програмуванні користувач описує предметну область сукупністю пропозицій у вигляді логічних формул, а комп'ютер, маніпулюючи цими пропозиціями, будує необхідний для розв'язання задачі висновок.

4.2. Найпростіші конструкції мови предикатів

Терм - це знак (символ) або комбінація знаків (символів), що є найменшим значимим елементом мови.

До термів відносяться константи, змінні й функції.

Константа застосовується для позначення конкретних об'єктів реального світу. Приклад: ластівка, птах, один, 2 і т.д.

Змінні використаються для позначення якогось із можливих об'єктів реального миру або їхньої сукупності (у Пролозі починаються із заголовної букви). Приклад: Хтось, X, Who, і т.д.

Функції (структури) - послідовність із декількох констант або змінних, взятих у круглі дужки, що слідують за функціональним символом (функтором). Приклад: сума (1,2); +(1,2); подвоїти (X).

Функтори позначають оператори, які після впливу на об'єкт повертають деяке значення.

Предикат - це логічна функція, що виражає відношення між своїми аргументами й приймає значення «істина», якщо це відношення існує, або «неправда», якщо воно відсутнє.

Укладена в дужки послідовність із n термів, перед якою стоїть предикатний символ, називається *n-місцевим* (або *n-арним*) *предикатом*, що приймає значення «істина» або «неправда» у відповідності зі значенням термів, які є його аргументами.

Приклад:
є (ластівка, птах)
батько (Х, Джон)

Такого типу предикати отримали назву *атомарних предикатів* і відповідають найбільш простим реченням нашої розмовної мови - нерозповсюдженим реченням.

У звичайній мові з *нерозповсюджених* речень за допомогою сполучних займенників, союзів, і інших частин мови будують більш складні конструкції - *складні* речення.

4.3. Предикатні формули

У логіку предикатів складними речення природної мови відповідають предикатні формули. ***Предикатні формули*** утворюються з атомарних предикатів і логічних зв'язувань, які читаються як (таблиця 4.1):

Таблиця 4.1

Логічні зв'язування

,	∨	¬	←	↔
«і»	«або»	«не»	«якщо»	«тоді й тільки»

Логічні зв'язування мають наступний пріоритет використання:

1) ¬

2) , ∨ (

3) ←, ↔

Найбільш часто в логічному програмуванні використаються зв'язування «І», «НЕ» «ЯКЩО».

Приклад предикатної формули, що відповідає складному реченню:

є (ластівка, птах) ← має (ластівка, крила),
володіє (ластівка, гніздо).

де є (_,_); має (_,_); володіють (_,_) - атомарні предикати; «,» й «←» - логічні зв'язування.

Однак наведена конструкція предикатної формули дозволяє робити твердження не тільки про конкретного індивідуума яким є ластівка, але й про всіх індивідуумів із класу птахів, використовуючи замість констант змінні:

є (X, птах) ⟵ має (X, крила), володіє (X, гніздо)

Таким чином, ставлячи змінні замість конкретних імен, ми приходимо до більш загальних понять кортежу довжини n, предиката й логічної формули.

Однак предикат, що містить змінні; наприклад,

має (X, крила)

не може бути оцінений, тобто не можна визначити він неправда або істина, тому що його значення визначається після підстановки в змінну деякої константи.

Однак іноді можна визначити значення предиката не роблячи підстановок використовуючи *квантори спільності* (∀) і *існування* (∃), які позначають «для всіх» й «існує принаймні одне».

Тоді наведена вище логічна формула буде записана у вигляді:

(∀X) [бути (X, птах) ⟵ має (X, крила), володіє (X, гніздо)]

і відповідають пропозиції, що може читатися як: «будь-яке X є птахом якщо це X має крила й володіє гніздом».

Квантори ∀ і ∃ можуть використовуватися й для будь-якого числа змінних. Розглянемо їх використання на прикладі двомісного предиката

любить (X, Y),

який описує відношення «X любить Y»:

- (∀**X**) (∀**Y**) любить (**X, Y**) - всі люди люблять всіх людей;
- (∃**X**) (∀**Y**) любить (**X, Y**) - існує людина, що любить усіх;
- (∀**X**) (∃**Y**) любить (**X, Y**) - для кожної людини існує той, хто її любить;
- (∃**X**) (∃**Y**) любить (**X, Y**) - існує людина, яка кого-небудь любить.

4.4. Визначення правильно побудованої формули

Комбінуючи логічні зв'язування й квантори можна рекурсивно визначити складену формулу логіки предикатів, яка називається *правильно побудованою формулою* (далі просто ППФ або логічна формула).

Якщо говорити стосовно до природної мови, то ППФ описує звичайне речення загального виду.

1. **Термом** є або константа, або змінна, або кортеж з **n** термів, перед яким стоїть функтор.
2. **Предикат** - це кортеж з **n** термів, перед яким стоїть предикатний символ.
3. **Атомарний предикат** є логічною формулою.
4. Якщо **F** й **G** - логічні формули, то (**F**); **F,G**; F∨G; ⌉**F**; **F⟵G**; **F↔G** - також є логічними формулами.

5. Якщо **F(X)** - логічна формула, то обидва вирази (∀**X**) **F**(**X**), (∃**X**) **F**(**X**) є логічними формулами.
6. Всі результати, отримані повторенням кінцевого числа п.1 – п.6, є логічними формулами.

Множина всіх речень, побудованих згідно з даними правилами, утворює мову логіки предикатів першого порядку.

Скориставшись цими визначеннями, можна, наприклад, речення «всі люди смертні» записати у вигляді:

(∀X) [людина (X) ← смертна]

4.5. Логічний висновок

Логічний висновок - це процес одержання з множини правильно побудованих формул (**S**) деякої ППФ (s) шляхом застосування одного або декількох правил висновку.

4.5.1. Правило резолюції для простих речень

Найбільш простий метод логічного висновку використовує тільки одне правило висновку, яке називається *резолюцією*, що застосовують до логічних формул у вигляді:

факт: **A**
заперечення: ⌉(**A_1, ... , A_n**)
імплікація: **A ← B_1, ... , B_m,**

де $\mathbf{A_i}$ (i = 1, n) і $\mathbf{B_j}$ (j = 1, m) - довільні предикати.

Розглянемо найбільш просту з форм резолюції для випадку лише двох $S = \{S_1, S_2\}$ ППФ виглядатимуть:

S_1 (заперечення): ⌉**A**
S_2 (імплікація): **A ← B ,**

у яких предикат **A** з S_1 співпадає з предикатом **A** лівої частини S_2.

У результаті одного кроку висновку з $\mathbf{S_1}$ й $\mathbf{S_2}$ буде отримана нова ППФ виду:

S: ⌉**B**

На цьому кроці висновку ППФ S_1 й S_2 називаються *батьківськими пропозиціями*, а **S** - ***резольвентою***, яка утворюється у результаті застосування резолюції до $\mathbf{S_1}$ й $\mathbf{S_2}$.

Резолюція в цьому найпростішому випадку відповідає правилу висновку modus tollens, що записується у вигляді:

$$\frac{\neg A \quad A \leftarrow B}{\neg B} \qquad (*)$$

і відповідає наступному висновку:

допускаючи, що**:** **НЕ А й А ЯКЩО В,**
робимо висновок**:** **НЕ В**

У ще більш простому випадку, коли S_1 - заперечення, а S_2 - факт, тобто:

S_1: $\neg$**A**
S_2: **A**

застосування правила резолюції дасть резольвенту - *порожнє заперечення*

S: $\square$

і означає протиріччя. Даний крок висновку може бути записаний у вигляді:

$$\frac{\neg A,\ A}{\square} \qquad (**)$$

і резолюцією є міркування типу

допускаючи, що**:** **НЕ А и А**
виводимо протиріччя**.**

4.5.2. Правило резолюції для складних пропозицій

Реальні логічні моделі містять значно більш складні речення. Так заперечення можуть містити кілька предикатів, так само як і праві частини імплікацій. Тому більш загальним є випадок, коли батьківські пропозиції мають вигляд:

S_1: $\neg(A_1,\dots,A_k,\dots,A_n)$
S_2: $A_k \leftarrow (B_1,\dots,B_m)$ **(де $1<k<n$)**

Тут деякий предикат $\mathbf{A_k}$ із заперечення S_1 збігається із предикатом з лівої частини S_2. У цьому випадку крок висновку заміняє A_k в S_1 на праву частину S_2 й як резольвенту одержують заперечення

S: $\neg(A_1, \dots, A_{k-1}, B_1,\dots, B_m, A_{k+1}, \dots, A_n)$

Дане правило проілюструємо змістовним прикладом.

допускаючи, що **Не (темно й зима й холодно)**
і що **Зима якщо Січень**
виводимо, що **НЕ (темно й січень і холодно)**

У тому випадку, якщо S_1 має той же вид, а S_2 - факт

S_2: A_k

причому A_k є одним із предикатів з S_1, крок висновку тільки викреслює A_k з S_1 і утворюється резольвента

S: $\neg(A_1, \dots, A_{k-1}, A_{k+1}, \dots, A_n)$

Даний крок висновку можна ілюструвати наступним прикладом

допускаючи, що **НЕ (темно й зима й холодно)**
і що **Зима**
виводимо, що: **НЕ (темно й холодно)**

4.5.3. Проста резолюція зверху вниз

Розглянуті вище правила застосовуються на кожному кроці висновку тільки до двох батьківських пропозицій. Разом з тим опис будь-якої галузі знання містить множину ППФ.

Розглянемо процес логічного висновку для прикладу, коли знання виражаються двома пропозиціями.

S_2: отримує (студент, стипендію) ← здає (успішно, сесію, студент)
S_3: здає (успішно, сесію, студент)

Задача, яку потрібно вирішити, полягає в тому, щоб відповісти на запит
чи одержує студент стипендію?

Коли використовується звичайна система логічного висновку, то таке питання представляється у вигляді заперечення

S: ⌉отримує (студент, стипендію)

і система повинна відкинути це заперечення за допомогою інших пропозицій, демонструючи, що дане допущення веде до протиріччя.

Цей підхід часто застосовується в математиці й називається *доказом від противного*.

Тепер уявимо собі, що вхідна логічна модель, складається з трьох пропозицій S_1, S_2 ,S_3 надходить на вхід системи логічного висновку комп'ютера.

КРОК 1

Система на першому кроці застосує правило (*) до батьківських пропозицій S_1 й S_2 й отримає резольвенту:

S: ⌉здає (успішно, сесію, студент)

КРОК 2

Використовуючи правило (**) до S й S_3 система виводить протиріччя:

S': ⌉

Таким чином, для доказу суперечливості S_1 S_2 S_3 виявилося досить двох кроків висновку.

Якщо вважати, що S_2 й S_3 не суперечать одне одному, то вони спільно суперечать S_1, тобто

підтверджують заперечення S: ⌉S_1: ⌉(⌉отримує (студент, стипендію))

або іншими словами підтверджують пропозицію:

отримує (студент, стипендію)

і відповіддю на вхідне завдання є «**ТАК**».

Логічний висновок, що породжує послідовність заперечень, таку як (S_1 , S , S`) у розглянутому прикладі, називається *резолюцією зверху вниз*.

4.5.4. Загальна резолюція зверху вниз

У загальному випадку предикати й ППФ у якості термів містять не тільки константи, але й змінні й функції. Розглянемо дві батьківських пропозиції:

S_1: ⌉отримує (студент, Y)
S_2: отримує (X, стипендію) ← здає (успішно, Z, X)

До них безпосередньо не можна застосувати правило резолюції, тому що вони не містять однакових предикатів у лівій частині імплікації й у запереченні. Дані пропозиції містять три змінних **X**, **Y**, **Z**, які неявно універсально квантировані.

Розглянемо першу пропозицію, S_1 , яка стверджує, що:

для всіх Y студент не отримує Y

Причому вираз «для всіх» розуміється як «для всіх індивідуумів з якої або області, обраної для інтерпретації пропозицій». При інтерпретації S_1 й S_2 принаймні один індивідуум **Y** буде зв'язаний з ім'ям «**стипендія**» і тому безпосереднім наслідком S_1 є більш конкретна пропозиція:

S_1`: ⌉отримує (студент, стипендію)

Аналогічно розглядається S_2 на області інтерпретації S_1 й S_2 й, вибираючи для **X**, індивідуум з ім'ям «**студент**», одержуємо більш конкретну пропозицію:

S_2`: отримує (студент, стипендію) ← здає (успішно, Z, студент)

Тепер маємо дві пропозиції S_1` й $S_2$`, які задовольняють умові застосовності правила резолюції, а саме наявності однакових предикатів у лівій частині імплікації й у запереченні. Тому резольвентою S_1` й $S_2$` буде:

S: ⌉здає (успішно, Z, студент)

Предикат

отримує (студент, стипендію)

називається *загальним прикладом батьківських предикатів*

отримує (X, стипендію)
отримує (студент, Y)

і отриманий за допомогою *уніфікатора* виду

Θ = { X:= студент, Y:= стипендія}

4.5.5. Уніфікатори й приклади уніфікації

Уніфікатором називається множина присвоювань вигляду:

$$\Theta = \{X_1 := t_1, ..., X_n := t_n\}$$

де X_i - змінна, а t_i – терм, застосування яких до двох виразів дає однакові загальні приклади.

На практиці уніфікатори визначають порівнюючи по черзі відповідні аргументи предикатів і виписуючи ті присвоювання термів змінним, які зробили б ці аргументи однаковими.

Розглянемо ряд прикладів уніфікації (таблиця 4.1):

Таблиця 4.1

Приклади уніфікації

Батьківські пропозиції	**Уніфікатор**	**Загальний приклад**
p(5), p(5)	Θ- порожня нескінченності (не заміняється жодна змінна)	p(5)
p (x), p(5)	Θ = {x:=5}	p(x)Θ = p(5)Θ = p(5)
p(x), p(y)	Θ = {x:=y}	p(y)
p(x, y), p(5, x)	Θ = {x:=5, y=x} = = {x:=5, y:=5}	p(x,y)Θ = p(5,x)Θ = p(5,5)
⌉p(5, x) p(x, y) ← q(x)	Θ = {x:=5, y:=5}	p(5,5) резольвента: s: ⌉q(x) Θ = ⌉q(s)

4.5.6. Розв'язання задач і побудова відповіді.

Розв'язання задачі з використанням логічного програмування розбивається на 3 етапи:

На першому етапі необхідно сформулювати наші знання й допущення про предметну область у вигляді множини ППФ

На другому етапі потрібно виразити конкретну задачу, поставлену на предметній області, як запит про одну або декілька відносин. Звичайно запит ставиться як вхідне заперечення.

Складанням допущень і вхідного запиту завершується робота розробника, мета якого - сформулювати задачу.

Третій крок виконується комп'ютером, що намагається вирішити задачу, будуючи доказ від противного. Він будує висновок зверху вниз, починаючи з вхідного заперечення, і породжує послідовність заперечень

$$\mathbf{D_1, D_2, \dots , D}$$

Якщо може бути побудований висновок, що закінчується запереченням

$$\mathbf{D_n = ⌉(протиріччя)},$$

то цей висновок, називаний *успішним висновком*, відразу дає розв'язок поставленої задачі.

Розглянемо всі ці три етапи на прикладі обчислювальної задачі знаходження значення факторіала деякого числа.

На першому етапі необхідно у вигляді ППФ сформулювати наші знання про обчислення факторіала, які можна представити двома пропозиціями:

S_1 : факторіал (0,1)
S_2 : факторіал (x,y) x>0, факторіал (x-1, y), y=y x

На другому етапі конкретну задачу, наприклад, обчислення значення факторіала трьох (тобто 3!), формулюємо у вигляді запиту як вхідного заперечення:

D_1: ⌉ факторіал (3, z)

На третьому етапі система логічного висновку виконає наступну послідовність етапів висновку на основі резолюції (таблиця 4.2)

Таблиця 4.2

Вивід на основі резолюції

Крок висновку	Батьківські пропозиції	Уніфікатор	Заперечення (резольвента)
1	D_1, S_2	Θ = {x:=3, y:=z, y_1:=y/x} = = {x:=3 , y_1:= z/3}	D_2: ⌉ факторіал (2, z/3)
2	D_2, S_2	Θ = {x:=2, y:=z/3, y_1:=y/x} = = {x:=2 , y_1:= z/6}	D_3: ⌉ факторіал (1, z/6)
3	D_3, S_2	Θ = {x:=1, y:=z/6, y_1:=y/x} = = {x:=1 , y_1:= z/6}	D_4: ⌉ факторіал (0, z/6)
4	D_4, S_1	Θ = {z/6:=1} = {z:=6}	⌉

Отримане на 4 кроці протиріччя підтверджує заперечення D_1, а стало бути висновок є успішним і дає рішення:

факторіал (3,z), з уніфікатором Θ = {z:=6}

т. т. **факторіал (3,6)**

Звідки відповідь, видавана системою логічного висновку: **z:=6**