**Лабораторна робота №1**

**Складання правила на мові Prolog**

**Управління програмою**

**Цілі роботи:**

* ознайомлення з основами мові Prolog;
* вивчення правил уніфікації змінних;
* одержання практичних навичок складання найпростіших правил на

мові Prolog.

* ознайомлення з механизмами управління процесом пошуку рішень

у мові Prolog;

**Завдання**

* Скласти правило для визначення предикату відповідно варіанту на основі фактів: чоловік (ім’я), жінка (ім’я), батьки (ім’я\_батька, ім’я\_дитини).
* Для отримання унікального варіанту рішення;
* Для отримання всіх можливих рішень.



Короткі теоретичні відомості

Prolog – це найбільш відома мова логічного програмування, яка використовується для вирішення задач, в яких діють об’єкти та відношення між цими об’єктами. Програмування на мові Prolog відноситься до декларативного стилю програмування і має спільні властивості з функціональним програмуванням.

Теоретичною основою логічного програмування є логіка. Логіка предикатів займається дослідженням відношень між об’єктами та доказами їхнього існування.

*Склад та структура програми*

Програма на мові Prolog складається з висловлювань трьох типів: факти, правила, питання.

Твердження складається з двох частин: ліва частина – висновок, права частина – умова, які розділяються знаком «:-».

Факт – це твердження, що є безумовною істиною, який складається з висновку за яким не виходить ніяких умов, наприклад: лікар (василь).

Сукупність фактів складають деякий світ, у межах якого дійсні твердження програми.

*Запитання* – це твердження, що містить лише умову без висновку. Запитання складається з одній або декількома цілей. Якщо всі цілі досягненні, Prolog дає позитивну відповідь – Так (Yes), у протилежному випадку – негативну відповідь Ні (No). Якщо у параметрах предикатів існують зміни, тоді, при позитивній відповіді, зміни конкретизуються.

*Правила* - це залежні відношення і дозволяють Prolog робити виведення, порівнюючи одну частину інформації з іншою.

*Відповідь на запитання і пошук рішень*

Відповідь на запитання здійснюється на основі фактів і правил, з яких складається програма. Необхідно звернути увагу на те, що повнота відповіді залежить від повноти складу бази даних програми (фактів), що описують «світ» програми.

Основний механізм, на якому базується пошук рішень, є співставлення термів, і називається уніфікацією.

Уніфікація може бути успішною або ні. Якщо уніфікація успішна, тоді ціль досягнуто. Успіх уніфікації залежить від складу термів, які співставленні. Крім того, якщо терм є зміною або у складі терму входить зміна, в результаті уніфікації зміна може бути конкретизована (отримати значення), якщо уніфікація успішна. Prolog керується наступними правилами:

1) Якщо уніфікуються дві константи, тоді уніфікація є успішною, якщо обидві константи однакові.

2) Якщо уніфікуються зміна з константою або структурою, уніфікація є успішною, а зміна конкретизується значенням константи або структури.

3) Якщо уніфікуються дві вільні зміни, у подальшому вони сприймаються як одна, тобто, якщо одна із змін конкретизується, друга зміна теж конкретизується тим же значенням.

**Механізм повернення**

При відповіді на запитання Prolog-система розглядає всі факти і правила як множина аксіом, запитання – як теорему. Потім Prolog намагається доказати цю теорему, тобто показати, що теорема логічно виводиться з аксіом.

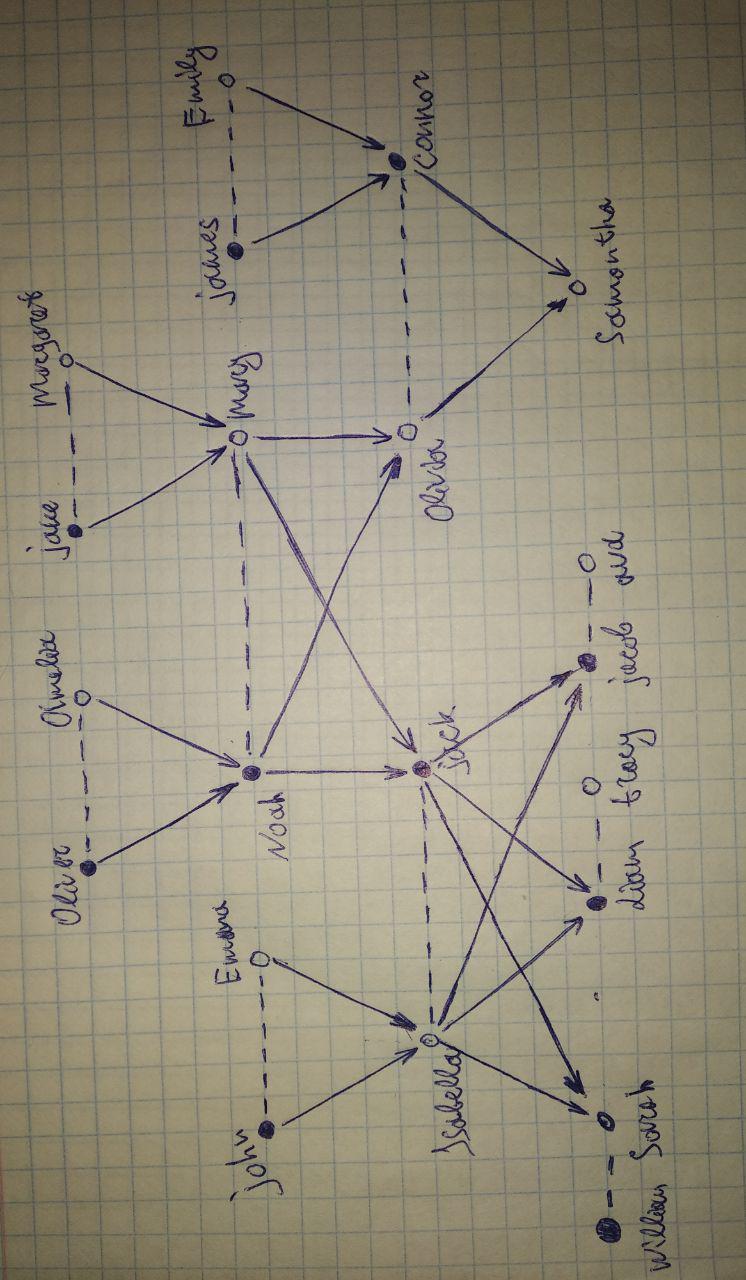
Якщо запитання складається з декількох цілей, які зв’язані кон’юнкціею, Prolog послідовно, зліва направо, виконує пошук рішення (до успішної уніфікації) для кожної з них. При цьому, якщо ціль містить зміну, вона конкретизується. При досягненні поточної цілі Prolog встановлює маркер і переходить до пошуку рішення для цілі, яка знаходиться праворуч. Зміни, що були конкретизовані, рахуються як константи. Такий процес продовжується до останньої цілі твердження, після чого Prolog система виводить значення конкретизованих змін.

Якщо будь-яка ціль не досягнуто, тоді Prolog намагається знайти друге рішення. Для цього виконується повернення до попередньої цілі (ліворуч), відміняється конкретизація змін (якщо вона була), і, починаючи з останнього маркера, продовжується пошук рішення. Якщо пошук закінчується невдачею, знову виконується повернення до цілі, яка знаходиться ліворуч. Такий процес називається зворотним трасуванням (back tracking) або механізмом повернення.

Якщо досягнуто остання ціль твердження, однак потенційно в базі даних можна знайти ще рішення, тобто не всі факти пройдені, система звертається до користувача за вказівкою, продовжити пошук чи ні. У рази необхідності отримання всіх рішень слід запустити механізм повернення після отримання кожного рішення. Для цього у мові Prolog існує предикат fail, який завжди є хибним, тобто закінчується невдачею.

**Рішення**

Схема зв’язків між об’єктами



Формування і запис правила з використанням квантору загальності

Якщо

M(x) – x це чоловік

F(x) – x це жінка

C(x, y) – х та у це пара

P(x, y) – y дитина х

То маємо:

тобто m – свекруха h.

Текст програми

male(oliver).

male(jake).

male(noah).

male(james).

male(jack).

male(connor).

male(liam).

male(john).

male(william).

male(jacob).

female(amelia).

female(margaret).

female(emma).

female(mary).

female(emily).

female(isabella).

female(olivia).

female(samantha).

female(sarah).

female(tracy).

female(ava).

couple(oliver, amelia).

couple(jake, margaret).

couple(john, emma).

couple(noah, mary).

couple(james, emily).

couple(isabella, jack).

couple(olivia, connor).

couple(william, sarah).

couple(liam, tracy).

couple(jacob, ava).

parent(oliver, noah).

parent(amelia, noah).

parent(jake, mary).

parent(margaret, mary).

parent(john, isabella).

parent(emma, isabella).

parent(noah, jack).

parent(noah, olivia).

parent(mary, jack).

parent(mary, olivia).

parent(james, connor).

parent(emily, connor).

parent(isabella, liam).

parent(isabella, sarah).

parent(isabella, jacob).

parent(jack, liam).

parent(jack, sarah).

parent(jack, jacob).

parent(olivia, samantha).

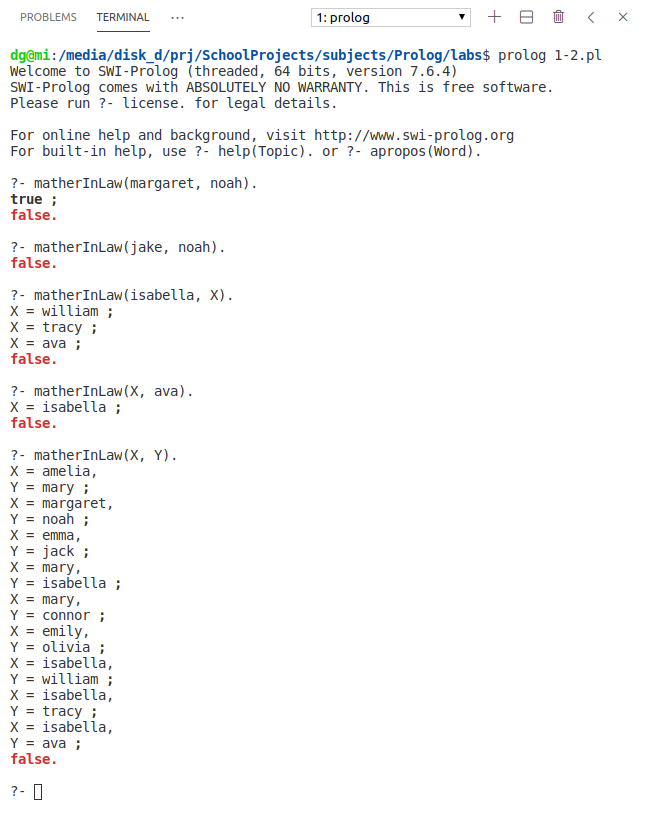
parent(connor, samantha).

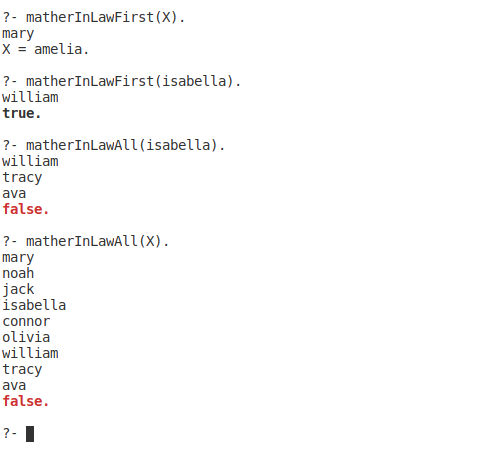
matherInLaw(M,H) :- female(M), (couple(H,K) ; couple(K,H)), parent(M,K).

matherInLawAll(M) :- matherInLaw(M, X), write(X), nl, fail.

matherInLawFirst(M) :- matherInLaw(M, X), write(X), !.

Результат виконання програми





**Висновок:** на цій лабораторній роботі я отримав навички програмування на мові F# використовуючи принципи функціонального програмування; засвоїв методи рішення складних задач на основі декомпозиції.