# Jednostka 9 - Paradygmat logiczny

Joanna Dagil

Grupa TCH-1

28 listopada 2025

## 1 Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się ze złożonymi zapytaniami, backtrackingiem i unifikacją w paradygmacie logicznym. Wykorzystujemy do tego język Prolog, w którym pokrywamy także zagadnienia takie jak rekurencja, struktury danych i operacje na nich.

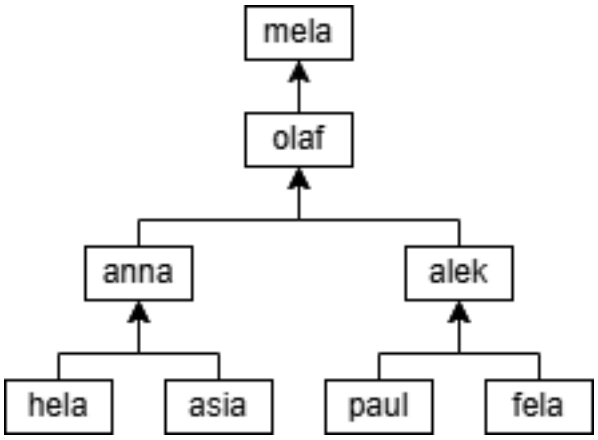
## 2 Zadania

### 2.1 Zadanie 1

Tworzenie faktów `dziecko/2`

```
dziecko(hela, anna).
dziecko(asia, anna).
dziecko(anna, olaf).
dziecko(paul, alek).
dziecko(fela, alek).
dziecko(alek, olaf).
dziecko(olaf, mela).
```

Diagram rodziny




Tworzenie reguły `rodzeństwo/2`

```
rodzenstwo(A, B) :- dziecko(A, Rodzic), dziecko(B, Rodzic).
```

Tworzenie reguły `dziadek/2`

Między dziadkiem a wnukiem musi być jedna instancja pośrednika - rodzica.

```
dziadek(Dziadek, Wnuk) :- dziecko(Rodzic, Dziadek), rodzic(Wnuk, Rodzic).
```

Tworzenie reguł `przodek/2`

Przodek może być albo bezpośrednio rodzicem potomka, albo rekurencyjnie rodzicem przodka potomka.

```
przodek(Przodek, Potomek) :- dziecko(Potomek, Przodek).
przodek(Przodek, Potomek) :- dziecko(Posrednik, Przodek), przodek(Posrednik, Potomek).
```

### 2.1.1 Przykłady zapytań prezentujące funkcjonowanie relacji rodzinnych

Czy mela jest przodkiem feli?

```
?- przodek(mela, fela).
true .
```

Jakich przodków ma fela?

```
?- przodek(Kto, fela).
Kto = alek ;
Kto = olaf ;
Kto = mela .
```

Czyim przodkiem jest olaf?

```
przodek(olaf, Kto).
Kto = anna ;
Kto = alek ;
Kto = hela ;
Kto = asia ;
Kto = paul ;
Kto = fela .
```

Czy alek jest przodkiem olafa?

```
?- przodek(alek, olaf).
false.
```

## 2.2 Zadanie 2

Reguła `nalezy/2`

```
nalezy(A, [A|_]).
nalezy(A, [_|T]) :- nalezy(A, T).
```

Regułą `dlugosc/2`

```
dlugosc([], 0).
dlugosc([_|T], N) :- dlugosc(T, N2), N is N2 + 1.
```

### 2.2.1 Przykłady zapytań prezentujące funkcjonowanie powyższych reguł

```
?- nalezy(3, [1,2,3,4]).
true.

?- nalezy(5, [1,2,3,4]).
false.

?- dlugosc([], N).
N = 0.

?- dlugosc([a,b,c,d], N).
N = 4.
```

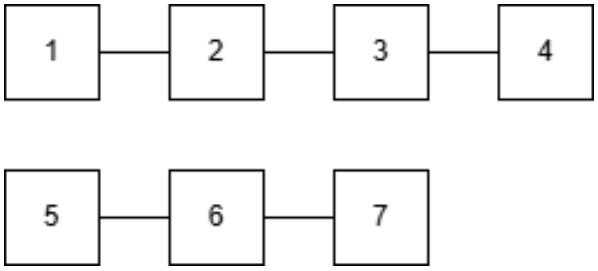
## 2.3 Zadanie 3

Tworzenie grafu. `krawedz/2)`

```
krawedz(1, 2).
krawedz(2, 3).
krawedz(3, 4).

krawedz(5, 6).
krawedz(6, 7).
studiuje(alek, matematyka).
```

Diagram grafu




Tworzenie reguły `sciezka/3`

```
sciezka(Start, End, [Start, End]) :- krawedz(Start, End).
sciezka(Start, End, [Start|Tail]) :- krawedz(Start, X), sciezka(X, End, Tail).
```

### 2.3.1 Przykłady zapytań prezentujące funkcjonowanie powyższych reguł

```
?- sciezka(1, 3, P).
P = [1, 2, 3].

?- sciezka(1, 2, P).
P = [1, 2].

?- sciezka(1, 5, P).
false.
```

# 3 Wnioski

Backtracking miał bezpośredni wpływ na sposób uzyskiwania wyników – po zadaniu jednego zapytania Prolog automatycznie szukał kolejnych rozwiązań i po wciśnięciu `;` zwracał następne możliwe dopasowania. Było to wyraźnie widoczne przy wyszukiwaniu wszystkich przodków danej osoby oraz wszystkich ścieżek w grafie. Z jednej strony ułatwia to eksplorację wielu rozwiązań, z drugiej wymaga ostrożnego definiowania reguł (np. unikania cykli w grafie), aby obliczenia nie zapętlały się.

Unifikacja decydowała o tym, czy dane wywołanie predykatu pasuje do faktu lub reguły. Przy zapytaniach takich jak `przodek(Kto, fela)` czy `sciezka(1, 3, P)` Prolog automatycznie podstawiał wartości za zmienne tak, aby lewa i prawa strona dopasowania były zgodne. Dzięki temu nie trzeba ręcznie „przechodzić" po strukturach danych – wystarczy zdefiniować ogólny wzorzec, a unifikacja zajmowała się doborem konkretnych wartości i wiązaniem zmiennych.