# Programowanie w Logice
# Fakty i reguły
# (Lista 1)

Przemysław Kobylański

## Wstęp

### Fakty

Niech $p$ będzie nazwą $n$-argumentowego predykatu, natomiast $t_1, t_2, \ldots, t_n$ niech będą termami opisującymi $n$ obiektów.

Wówczas fakt, że między obiektami opisanymi termami $t_1, t_2, \ldots, t_n$ zachodzi relacja $p$, zapisywać będziemy w Prologu następująco:

$$p(t_1, t_2, \ldots, t_n).$$

#### Przykłady

```
lubi(janek, jabłka).
lubi(monika, jabłecznik).
lubi(franek, jabola).
```

### Reguły

Jeśli *head* jest konkluzją reguły posiadającą postać formuły atomowej $p(t_1, t_2, \ldots, t_n)$ a *body* jest przesłanką reguły, tj. prostym lub złożonym warunkiem z którego logicznie wynika konkluzja *head*, to regułę taką zapisywać będziemy w Prologu następująco:

*head* `:-` *body*.

Przesłanka *body* może być następującej postaci:

- prosta atomowa formuła postaci $p(t_1, t_2, \ldots, t_n)$,
- koniunkcja formuł oddzielonych przecinkiem,
- alternatywa formuł oddzielonych średnikiem,
- negacją formuły (operatorem[1] negacji jest `\+`),

[1] Operator negacji jest jednoargumentowy, zatem jeśli ma negować formułę ujętą w nawiasy, to koniecznie oddziel go od nawiasu otwierającego przynajmniej jedną spacją.

Jeśli treść reguły jest formułą złożoną, to przyjęło się wymieniać kolejne jej człony w kolejnych wierszach robiąc tabulacją wcięcie.

**Przykłady**

```
hamlet(Decyzja) :-
        Decyzja = być;
        Decyzja = nie_być.
hamlet_który_się_rozmyślił(Decyzja_Ostateczna) :-
        hamlet(Decyzja),
        przeciwna(Decyzja, Decyzja_Ostateczna).
dobra(Decyzja) :-
        \+ zła(Decyzja).
```

## Klauzula

Klauzula to fakt albo reguła. Program w Prologu składa się z klauzul.

Przyjęło się zapisywać kolejne predykaty w postaci kolejnych klauzul, przy czym klauzule stanowiące definicję jednego predykatu pisane są jedna za drugą (nie miesza się klauzul z różnych predykatów).

**Przykłady**

```
lubi(adam, ewa).
lubi(ewa, abel).
lubi(ewa, kain).

nie_lubi(kain, abel).
nie_lubi(adam, jabłka).
```

# Zadania

## Zadanie 1 (1 pkt)

Ćwiczenie 1.3 z książki [1].

Inspiracją do tego ćwiczenia było ćwiczenie z książki „Logic for Problem Solving” (*Logika w rozwiązywaniu problemów*) Roberta Kowalskiego wydanej przez North Holland w 1979 roku. Załóżmy, że zapisano w formie klauzul Prologa następujące relacje:

```
ojciec(X, Y)          /* X jest ojcem Y       */
matka(X, Y)           /* X jest matką Y       */
mężczyzna(X)          /* X jest mężczyzną     */
kobieta(X)            /* X jest kobietą       */
rodzic(X, Y)          /* X jest rodzicem Y    */
diff(X, Y)            /* X i Y są różne       */
```

Rysunek 1: Martwa natura (źródło [2])

Zamiast warunku `diff(X, Y)` użyj predykatu `X \= Y`.

Należy zapisać klauzule definiujące relacje:

```
jest_matką(X)          /* X jest matką          */
jest_ojcem(X)          /* X jest ojcem          */
jest_synem(X)          /* X jest synem          */
siostra(X, Y)          /* X jest siostrą Y      */
dziadek(X, Y)          /* X jest dziadkiem Y    */
rodzeństwo(X, Y)       /* X i Y są rodzeństwem */
```

Przykładowo można byłoby zapisać regułę `ciotka` korzystając z danych wcześniej reguł `kobieta`, `rodzeństwo` i `rodzic`:

```
ciotka(X, Y) :- kobieta(X), rodzeństwo(X, Z), rodzic(Z, Y).
```

Można też tę regułę zapisać inaczej:

```
ciotka(X, Y) :- siostra(X, Z), rodzic(Z, Y).
```

## Zadanie 2 (1 pkt)

Stos bloków może być opisany przez zbiór faktów `on(Block1, Block2)`, który jest prawdziwy jeśli `Block1` leży na `Block2`. Zdefiniuj predykat `above(Block1, Block2)`, który jest prawdziwy jeśli `Block1` jest na stosie powyżej `Block2`. (Wskazówka: `above` jest tranzytywnym domknięciem `on`.)

Ćwiczenie 2.3.1 (i) z [2].

## Zadanie 3 (1 pkt)

Opisz rozmieszczenie obiektów z rysunku 1 w postaci faktów wyrażających predykaty: `left_of(Object1, Object2)` i `above(Object1, Object2)`.

Zdefiniuj nowe predykaty `right_of(Object1, Object2)` i `below(Object1, Object2)` w terminach `left_of` i `above`.

Dodaj rekurencyjne reguły dla `left_of` i `above`. Zdefiniuj `higher(Object1, Object2)`, który jest prawdziwy jeśli `Obiekt1` jest w wyżej położonym wierszu niż niż `Object2`. Dla przykładu `bicycle` jest wyżej niż `fish`.

Ćwiczenia 2.1.1 (iii) i 2.3.1 (ii) z [2].

### Zadanie 4 (2 pkt)

Załóżmy, że relacja częściowego porządku `le(X, Y)`, gdy $X \preceq Y$, zapisana jest w postaci faktów (pamiętaj, że częściowy porządek jest zwrotny, przechodni i słabo antysymetryczny).

Napisz definicje następujących relacji:

`maksymalny(X)` gdy X jest elementem maksymalnym,

`największy(X)` gdy X jest elementem największym,

`minimalny(X)` gdy X jest elementem minimalnym,

`najmniejszy(X)` gdy X jest elementem najmniejszym.

### Zadanie 5 (2 pkt)

Załóżmy, że dana jest w postaci faktów relacja `le(X, Y)`.

Napisz bezargumentowy predykat `częściowy_porządek/0`, który jest spełniony gdy relacja `le/2` jest częściowym porządkiem.

### Zadanie 6 (3 pkt)

Napisz predykat `prime(LO, HI, N)`, który jest spełniony przez te liczby całkowite $LO \leq N \leq HI$, że $N$ jest liczbą pierwszą (liczby $LO$ i $HI$ są dane).

Przykład użycia:

```
?- prime(10, 20, X).
X = 11 ;
X = 13 ;
X = 17 ;
X = 19 ;
false.
```

Pomyśl o jak najefektywniejszym znajdowaniu tych liczb.

## Literatura


[1] W.F. Clocksin, C.S. Mellish. Prolog. Programowanie. Helion, 2003.

[2] L. Sterling, E. Shapiro. The Art of Prolog. The MIT Press, 1994.