Operacje na liczbach naturalnych

1. nat(k) wiw gdy k jest liczbą naturalną

   nat(0) $\leftarrow$
   nat(s(x)) $\leftarrow$ nat(x)

2. plus(x, y, z) wiw gdy x + y = z

   plus(0, x, x) $\leftarrow$
   plus(s(x), y, s(z)) $\leftarrow$ plus(x, y, z)

   zapytania:

   $\leftarrow$ plus(s(0), s(s(0)), x)
   $\leftarrow$ plus(x, y, s(0))
   $\leftarrow$ plus(x, s(0), s(s(0)))

   odejmowanie

   $\leftarrow$ plus(x, s(0), z)

   (∞ wiele odpowiedzi)

3. minus(x, y, z) wiw gdy x − y = z

   minus(x, y, z) $\leftarrow$ plus(y, z, x)

4. fib(k, f) wiw gdy f jest k-tą liczbą Fibonacciego

   (Def: $f(0) = 1, f(1) = 1,$
   $f(k+2) = f(k) + f(k+1), k \geq 0$)

   fib(0, s(0)) ←
   fib(s(0), s(0)) ←
   fib(s(s(k)), f3) ← fib(k, f1),
   fib(s(k), f2),
   plus(f1, f2, f3)

   zapytania:

   ← fib(s(s(0)), s(0))
   ← fib(s(s(s(0))), x)
   ← fib(x, s(s(s(s((0))))))
   (nieskończona pętla)

Operacje na listach

1. lista(l) wiw gdy l jest listą

   lista(nil) $\leftarrow$
   lista(x.y) $\leftarrow$ lista(y)

2. first(l, e) wiw gdy e jest pierwszym elementem listy l

   first(x.y, x) $\leftarrow$

3. last(l, e) wiw gdy e jest ostatnim elementem listy l

   last(x.nil, x) $\leftarrow$
   last(x.y, z) $\leftarrow$ last(y, z)

4. length(l, k) wiw gdy lista jest k-elementowa

   length(nil, 0) $\leftarrow$
   length(x.y, s(k)) $\leftarrow$ length(y, k)

5. member(e, l) wiw gdy e jest elementem listy l

   member(x, x.y) $\leftarrow$
   member(x, y.z) $\leftarrow$ member(x, z)

6. append(l1, l2, l3) wiw gdy l1 $\circ$ l2 $=$ l3 (konkatenacja list)

   append(nil, x, x) $\leftarrow$
   append(x.y, l, x.z) $\leftarrow$ append(y, l, z)

7. intersect(l1, l2) wiw gdy l1 i l2 mają niepuste przecięcie

   intersect(l1, l2) $\leftarrow$ member(e, l1),
   member(e, l2)

$$\forall\, l1, l2 \quad (\exists e\; member(e, l1) \wedge member(e, l2)) \\ \rightarrow \\ intersect(l1, l2)$$

Przykład.
Połaczenia kolejowe między miastami.

Zapytania typu:
- czy istnieje bezpośrednie połączenie z A do B?
- czy można dojechać z A do B?
- dokąd można dojechać z A?

Połączenia bezpośrednie:

$$direct(warszawa, kraków) \leftarrow$$
$$direct(szczecin, wroclaw) \leftarrow$$
$$direct(kraków, zakopane) \leftarrow$$

Możliwe połączenia:

$$connect(x, y) \leftarrow direct(x, y)$$
$$connect(x, y) \leftarrow direct(x, z), connect(z, y)$$

$$\forall x\, \forall y\, \forall z\, (\, direct(x, z) \wedge connect(z, y) \rightarrow connect(x, y))$$

$$\forall x \forall y\, ((\exists z\, direct(x, z) \wedge connect(z, y)) \rightarrow connect(x, y))$$

Przykład cd.

Przykładowe zapytania:

$\leftarrow connect(warszawa, zakopane)$
odpowiedź: `yes`

$\leftarrow connect(warszawa, x)$
$x = kraków$;
$x = zakopane$;
`no`

$\leftarrow connect(zakopane, x)$
`no`

$\leftarrow connect(x, zakopane)$
$x = kraków$;
$x = warszawa$;
`no`

Programy w logice a programy w Prologu (składnia)

- zamiast symbolu $\leftarrow$ piszemy `:-`
- klauzule unarne bez $\leftarrow$ (`:-`)
- każdą klauzulę kończymy kropką
- nazwy zmiennych od wielkich liter
  (nazwy stałych od małych liter)

Przykład.

```
direct( warszawa, krakow ).
direct( szczecin, wroclaw ).
direct( krakow, zakopane ).

connect(X, Y):- direct(X, Y).
connect(X, Y):- direct(X, Z), connect(Z, Y).
```

Zapytania:

```
:- connect( warszawa, X ).
:- connect( Miasto, zakopane ).
```

Korzystanie z Prologu

(SICStus Prolog, SWI Prolog)

dwa tryby pracy:

- – tryb definiowania procedur
- – tryb wykonywania dyrektyw i zapytań

— początkowo tryb zapytań (zachęta: `| ?-`)

```
| ?- connect(warszawa, X).
X = krakow ?  ; % daj następną odpowiedź
X = zakopane ?
```

— przejście do trybu definiowania:

```
| ?- [user].
| middle(E, L) :- ...       % definicja
| :- middle(3, 1.3.5.nil).  % zapytanie
```

— powrót do trybu zapytań:

```
| Control+D
| {user consulted, ...msec ...bytes}
| ?-
```

zakończenie pracy: **halt/0** lub Control+D

załadowanie programu: `| ?- [plik].`