Programowanie funkcyjne:

Pierwszym językiem funkcyjnym był LISP (ok. 1960); on też był pierwszym językiem programowania w sztucznej inteligencji.

Przedstawię programowanie w SML (Standard ML, ok. 1985).

- *nie ma* przypisań wartości zmiennym;
- *nie ma* pętli;
- *nie ma* procedur;
- *są* stałe, którym można nadać wartość (ale nie można tej wartości potem zmienić);
- *są* funkcje;
- *jest* rekursja;
- *jest* bardzo bogaty system typów;
- funkcje *są* obywatelami pierwszej klasy.

Programowanie funkcyjne:

```
- val x = 1;                    — def. stałej całkowitej
val x = 1 : int

- val f = (fn(x)=> 1+x);        — def. stałej funkcyjnej
val f = fn : int -> int

- fun f(x) = 1+x;               — j.w. „polukrowane"
val f = fn : int -> int

- f x;                          — zastosowanie funkcji
val it = 2 : int                  do argumentu

- fun sil n =                   — definicja silni
=   if n=0 then 1
=   else n * sil(n-1);
val sil = fn : int -> int

- sil 5;                        — silnia od argumentu
val it = 120 : int

-
```

Programowanie funkcyjne:

```
- sil sil;                                          — BŁĄD!
Error: operator and operand
 don't agree
   operator domain: int
   operand:         int -> int
   in expression:
     sil sil
```

```
- fun divmod(n,k) =                                 — iloraz całkowity
=   if n < k then (0,n)                                i reszta
=   else let val (q,r) = divmod(n-k,k)                 (naraz)
=        in (q+1, r)
=        end;
val divmod = fn : int * int -> int * int
```

```
- divmod (17,3);                                    — zastosowanie
val it = (5,2) : int * int
-
```


Listy:

```
[2, 3, 5, 7, 11, 13] : int list
[true, true, false, true, false] : bool list
[[1, 2], [ ], [5, 3, 6]] : (int list) list
[true, 1, 6.3]                    — BŁĄD!
```

Dołączenie elementu z przodu listy:

$$x :: [x_1, x_2, \ldots, x_n] \stackrel{\text{def}}{=} [x, x_1, x_2, \ldots, x_n]$$

Suma elementów na liście:

```
fun sum [ ] = 0
  | sum (x :: reszta) = x + sum(reszta);
val sum = fn : int list -> int

- sum [2, 3, 5, 7, 11, 13];
val it = 41 : int
```

Obliczenie wg definicji rekursywnej:

```
fun sum [ ] = 0
  | sum (x :: reszta) = x + sum(reszta);

sum [2, 3, 5, 7, 11, 13] =
   = sum (2 :: [3, 5, 7, 11, 13]) =
   = 2 + sum [3, 5, 7, 11, 13] =
   = 2 + sum (3 :: [5, 7, 11, 13]) =
   = 2 + 3 + sum [5, 7, 11, 13] =
   = 2 + 3 + 5 + sum [7, 11, 13] =
   = 2 + 3 + 5 + 7 + sum [11, 13] =
   = 2 + 3 + 5 + 7 + 11 + sum [13] =
   = 2 + 3 + 5 + 7 + 11 + 13 + sum [] =
   = 2 + 3 + 5 + 7 + 11 + 13 + 0 =
   = 41
```


Języki programowania do sztucznej inteligencji — SML

Czy lista zawiera dany element?

```
fun zawiera x [ ] = false
  | zawiera x (y::reszta) =
       if x=y then true
       else zawiera x reszta;
val zawiera = fn : ''a -> ''a list -> bool

- zawiera false [true, true, false, true, false];
val it = true : bool

- zawiera false [true, true, true];
val it = false : bool

- zawiera 4 [1, 2, 3, 4, 5, 6];
val it = true : bool
```

Czy wszystkie elementy listy spełniają warunek?

```
fun wszystkie war [ ] = true
  | wszystkie war ((x:real) :: reszta) =
     (war x) andalso (wszystkie war reszta);
val wszystkie = fn : (real -> bool) -> real list -> bool

- wszystkie (fn(x)=>x<10.0) [1.0,  5.1, 3.14, 0.0];
val it = true : bool

- wszystkie (fn(x)=>x<0.0) [1.0,  5.1, 3.14, 0.0];
val it = false : bool
```