„Normale“ Listen

datatype 'a list = nil | :: of 'a * 'a list

Generation von Listen:
Ausgehend von einem Anfangswert a wird mithilfe einer Funktion f eine Liste der Form

[a, f(a), f(f(a)), ... , f_k(a)] bis zu einem Index k gebildet.

```
- fun generate f a 0 = [a] | generate f a k = [a]@generate f (f a) (k-1)
val generate = fn : ('a -> 'a) -> 'a -> int -> 'a list

- generate (fn x => x+3) 0 10;
val it = [0,3,6,9,12,15,18,21,24,27,30] : int list
```

Unendliche Listen
(bei call-by-need Substitution)

datatype 'a list = nil | :: of 'a * 'a list

```
- fun from n = n::(from (n+1))      (* unendliche Liste [n, n+1, n+2, ...] *)
val from = fn : int -> int list

- fun take 0  l       = [ ] |
        take k [ ]    = [ ] |
        take k (x::y)= x::(take (k-1) y);
val take = fn : int -> 'a list -> 'a list

- take 4 (from 10);
val it = [10,13,16,19] : int list      (* in ML so nicht möglich !!!! *)
```

Lazylist in ML

```
datatype 'type lazylist = empty | c of 'type * (unit -> 'type lazylist);

- fun from n = c(n,fn () => from(n+1));
val from = fn : int -> int lazylist

- fun take 0 l     = [ ] |
    take k empty = [ ] |
    take k (c(x,y))=x::(take (k-1) (y()));
val take = fn : int -> 'a lazylist -> 'a list

- fun generate f a = c(a,   fn () => generate f (f a)    );
val generate = fn : ('a -> 'a) -> 'a -> 'a lazylist

- val from = generate (fn x => x+3);
val from = fn : int -> int lazylist

- take 4 (from 10);
val it = [10,13,16,19] : int list
```