MOST GENERAL TYPE

- type inference produces the most general type

```
let rec map f arg = function
    [] -> []
  | hd :: tl  ->  f hd :: (map f tl)
```

val map : ('a → 'b) → 'a list → 'b list = <fun>

- functions may have many less general types

val map : (t_1 → int, [t_1]) → [int]

val map : (bool → t_2, [bool]) → [t_2]

val map : (char → int, [char]) → [int]

- less general types are all instances of most general type, called the principal type.

INFORMATION FROM TYPE INFERENCE

- consider this function

```
let reverse ls = match ls with
    [] -> []
    | x :: xs -> reverse xs
```

and its most general type:

:- reverse :: list 't1 → list 't2 $\textcircled{x}$

- what does this type mean?

reversing a list should not change its type, so there must be an error in the definition of reverse.