

MOST GENERAL TYPE

type inference produces the most general type

let rec map f arg = function $[] \rightarrow []$

| hd: te -> f hd: (map f te)

val map: ('a → 'b) → 'a list → 'b list = (60m)

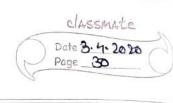
- functions may have many less general types

val map: (t, > int [t,]) -> [int]

val map: (bool > tz, [bool]) > [tz]

val map: (char > int, Echar]) > [int]

general type, anded the principal type.



INFORMATION FROM TYPE INFERENCE

consider this function

Cet reverse ls = match le with

(et reverse is = match is with

1 72: 72s -> reverse 76s

and its most general type:

i manage in Andrew I am light in

:- reverse :: list 't, -> list 'tags

- what does this type mean?

reversing a list should not change its type, so there must be an error

in the definition of reverse.