TYPES, TYPE INFERENCE AND UNIFICATION

WHAT IS A TYPE?

collection of computable values that share some structural property

ex

$$\left.\begin{array}{l} \text{int} \\ \text{string} \\ \text{int} \rightarrow \text{bool} \\ (\text{int} \rightarrow \text{int}) \rightarrow \text{bool} \\ [a] \rightarrow a \\ [a] * a \rightarrow [a] \end{array}\right\} \text{types.}$$

$$\left.\begin{array}{l} \{3,\ \text{true},\ \backslash x \rightarrow x\} \\ \text{even integers} \\ \{f : \text{int} \rightarrow \text{int} \\ \quad \mid x > 3 \Rightarrow f(x) > x * (x+1)\} \end{array}\right\} \text{not types}$$

distinction between

- sets of values that are types, and
- Sets that are not types

is language dependent.

types help identify & prevent errors.

TYPE ERRORS ARE LANGUAGE DEPENDENT

- array out of bound access

C/C++	runtime error
OCaml/Java	dynamic type error.

- null pointer ~~def~~reference

C/C++	run-time error
OCaml	pointers are hidden inside datatypes