PATTERN MATCHING

- matching with multiple cases

```
let isempty l = match l with
      | [] → true
      | _  → false
```

- infer type of each case

• first case

$[t_1] \rightarrow$ bool

• second case

$t_2 \rightarrow$ bool.

- combine by unification of the types of the cases

val isempty : $[t_1] \rightarrow$ bool = <fun>

BAD PATTERN MATCHING

- matching with multiple cases

```
let isempty l = match l with
    | [] -> true
    | _  -> 0
```

- infer type of each case

  - first case
    $[t_1] \to bool$

  - second case
    $t_2 \to int$

- combine by unification of the types of the cases

type error: cannot unify bool and int.