



List implementation

```
module type Set =
sig
  type t
  val empty : t
  val add : t → int → t
  val mem : t → int → bool
end

module IntSet : Set =
struct
  type t = int list
  let empty = []
  let add s i = i :: s
  let mem s i = List.mem i s
end
```



Bitmask implementation

```
module type Set =
sig
  type t
  val empty : t
  val add : t → int → t
  val mem : t → int → bool
end

module IntSet : Set =
struct
  type t = int
  let empty = 0
  let add s i = s | (1 « i)
  let

```





Syntax for existential

$e ::= t \mid t_1, e \text{ as } t_2$

