Type inference

aka: how in the world does Ocaml figure out all the types for me ???

Example 1

```ocaml
let x = 2 + 3;;
let y = string_of_int x;;
```

Example 2

```ocaml
let x = 2 + 3;;
let inc y = x + y;;
```

Example 3

```ocaml
let foobar x =
  let (y,z) = x in
  z-y;;
```

```ocaml
let foobar x =
  let (y,z) = x in
  z-y;;
```

```
Ty
Γx = int ⊑ T
Ty
Γinc = T ⊑ Ty
Ty
Γy = int ⊑ int
Ty
Γ = int ⊑ int
```
Example 4

```ocaml
let rec cat l = match l with
  | [] -> ""
  | h::t -> h^(cat t)
```

Example 4

ML doesn't know what the function does, or even that it terminates. ML only knows its type!

Example 5

```ocaml
let rec map f l = match l with
  | [] -> []
  | h::t -> (f h)::(map f t)
```

Example 5

Inferring types with ‘a

- Introduce unknown type vars
- Figure out equalities that must hold, and solve these equalities
- Remaining types vars get a forall and thus become the ‘a, ‘b, etc.

Example 6

```ocaml
let compose (f, g) x = f (g x)
```
Example 6

```
let compose (f, g) x = f (g x)
```

Example 7

```
let rec fold f cur l =
  match l with
  | [] -> cur
  | h::t -> fold f (f h cur) t
```

Example 7

```
let rec fold f cur l =
  match l with
  | [] -> cur
  | h::t -> fold f (f h cur) t
```