Polymorphism versus overloading

Polymorphic function:
same function usable for many different argument types

```
- fun swap(a,b) = (b,a);
val swap = fn : 'a * 'b -> 'b * 'a
```

Overloaded function:
several different functions, all given name

Resolve overloading to particular function based on:

- static argument types (in ML)
- dynamic argument classes (in object-oriented languages)

```
- 3 + 4;
val it = 7 : int

- 3.0 + 4.5;
val it = 7.5 : real

- (op +);  (* which +? default to int version *)
val it = fn : int*int -> int

- (op +):real*real->real;
val it = fn : real*real -> real
```


Equality types

The = built-in function is polymorphic over all types that "admit equality"

- any type except those containing reals or functions

Use ''a, ''b, etc. to stand for these equality types

```
- fun is_same(x, y) =
    if x = y then "yes" else "no";
val is_same = fn : ''a * ''a -> string
- is_same(3, 4);
val it = "no" : string
- is_same({l=[3,4,5],h=("a","b"),w=nil},
          {l=[3,4,5],h=("a","b"),w=nil});
val it = "yes" : string
- is_same(3.4, 3.4);
Error: operator and operand don't agree
[equality type required]
  operator domain: ''Z * ''Z
  operand:         real * real
  in expression:
    is_same (3.4,3.4)
```