References - L-values versus R-values

```
val a = ref 10;
val a = ref 10 : int ref
val b = ref true;
val b = ref true : bool ref
```

A reference is similar to a pointer in C. Like a pointer, a reference points to data allocated in the heap. Unlike in C, pointer arithmetic is not allowed:

```
b + 2;
Error: operator and operand do not agree
```

To dereference a reference use !. Using ! is the same as * in C++.

```
!;
val it = fn : 'a ref -> 'a
!a;
val it = 10 : int
!b;
val it = true : bool
(op :=);
val it = fn : 'a ref * 'a -> unit
a := a + 1;
Error
a:= !a +1 ;
val it = () : unit
```

Note that the assignment returns unit because that is the value of the side effect. The value variable `a` points to now to 11.

References in ML follow uniform representation. This means that when space is allocated for a reference, it will be the same size regardless of the type.

Note that `:=` takes a reference to `a` as the L-value and an `a` as the R-value, then returns a unit. This differs from `=` in that `:=` is updating a value and `=` is introducing a new definition.