Set-once memory

Set-once memory is a special kind of synchronous memory with the following constraints.

- Multiple fibers can access the memory.
- We initialize the memory at most once, during the first `set` operation.
- We spin until the thunk has evaluated.

Our implementation has the following signature. The `new` operation creates a set-once cell. The `get` operation obtains the value of the cell. This operation spins if the thunk is already evaluating.

```haskell
type 'a set_once_mem
type 'a thunk = unit -> 'a
val new : 'a thunk -> 'a set_once_mem
val get : 'a set_once_mem -> 'a
```

We can completely specify the behavior of set-once memory in terms of futures.

```haskell
type 'a set_once_mem = 'a future
fun set sm =
  case poll sm
  of NONE => touch sm
      SOME v => v
(* end case *)
```