1 Mutable Objects

We now implement mutable objects. We start with inheritance:

```
data BaseCons x = BaseCons x

instance (o ~ (BaseCons bo   C o n s   no), Record o Ref ST s => Object o Ref ST s where
  type Base o = bo
  get_base self_ref =
        Ref(do ((BaseCons base)   C o n s   tl) <- eval self_ref
               return base)
              (\b a s e   -> do (_   C o n s   tl) <- eval self_ref
               self_ref := ((BaseCons b a s e )   C o n s   tl)
               return ())

  ...

instance (o ~ (Unit   C o n s   so), Record o Ref ST s => Object o Ref ST s where
  type Base o = o
  get_base self_ref = self_ref

  ...
```

In our mutable encoding the first field of the object must be either the value of the inherited value or unit when the object does not inherit anything.

Methods enjoy the same implementation in both cases, so we just give one:

```
data MethodCons t a b = MethodCons(t -> a -> (b,t))

instance (o ~ (Unit   C o n s   so), Record o Ref ST s => Object o Ref ST s where
  ...

  type Method ro ref st a b = MethodCons ro a b
  self_ref <=| (Label read write) =
              \x -> do self <- eval self_ref
                       let (MethodCons m) = read self
                           (res, s e l f  ) = m self x
                       self_ref := s e l f
                       return res
```

It can prove very useful to take advantage of our existing infrastructure to create methods from references and statements, so that the user of our system will not be forced to define methods by explicitly tracking mutations to the value of *this*; for this reason we add a function to the *Object* predicate that converts a method from reference to state into our format (the implementation here is the same for both instances of *Object*, so we provide only one:

```
class (Record o ref st s, Recursive o, ro~Rec o) => Object o ref st s where
  ...
  mk_method :: (ref o o -> a -> st o b) -> Method ro ref st a b

instance (o ~ (Unit   C o n s   so), Record o Ref ST s => Object o Ref ST s where
  ...
  mk_method m = Method(\this->\args->
                       let (ST res_st) = m state_ref args
                       in res_st this)
```