# DDC: peekOn and pokeOn

Ben Lippmeier
University of New South Wales, Australia
Haskell Implementors Workshop
2010/10/1

# Macros + Token Pasting == Polymorphism

```
#define MAKE_UNBOXFUN(Type) \
   static inline \
   Type _unbox##Type (Obj* obj) \
   {  DataRS* data = (DataRS*) _force (obj); \
      Type* x      = (Type*) data->payload; \
      Type  v      = *x; \
      return v; \
   }

// -- Unboxing Functions
MAKE_UNBOXFUN (Word32)
MAKE_UNBOXFUN (Int32);
MAKE_UNBOXFUN (Int64);
MAKE_UNBOXFUN (Float32);
MAKE_UNBOXFUN (Float64);
MAKE_UNBOXFUN (Char32);
```

# Unboxing in Disciple

```
unboxInt32 :: Int32 -> Int32#
unboxInt32 w = do
  w' = force w
  peek (castToPtrInt32 (dataRS_payload w'))
```

```
force              :: b -> b
dataRS_payload     :: b -> Ptr# Word8#
castToPtrInt32     :: Ptr# a -> Ptr# Int32#
peek               :: Ptr# c -> c
```

# Region and Effect inference

```
unboxInt32 :: Int32 %r1 -(!Read %r1)> Int32#
unboxInt32 w = do
   w' = force w
   peek (castToPtrInt32 (dataRS_payload w'))
```

```
force              :: b -> b
dataRS_payload     :: b -> Ptr# Word8#
castToPtrInt32     :: Ptr# a -> Ptr# Int32#
peek               :: Ptr# c -> c
```

# peek in Haskell

```
peek
  :: forall a
.   Ptr a -> IO a
```

# peekOn in Disciple

```
peekOn
  :: forall (t :: % -> *) a
.    t %r1 -> Ptr# a -(!Read %r1)> a
```

# peekOn in Disciple

```
peekOn
  :: forall (t :: % -> *) a
.    t %r1 -> Ptr# a -(!Read %r1)> a
```

Read from this pointer

The read affects this object.

# Reading the argument

```
unboxInt32 :: Int32 %r1 -(!Read %r1)> Int32#
unboxInt32 w = do
   w' = force w
   peekOn w' (castToPtrInt32
                        (dataRS_payload w'))
```

```
peekOn
  :: forall (t :: % -> *) a
  .   t %r1 -> Ptr# a -(!Read %r1)> a
```

# pokeOn

```
pokeOn
  :: forall (t :: % -> *) a
  .  t %r1 -> Ptr# a -> a -(!Write %r1)> ()
  :- Mutable %r1
```

# DDC: Disciplined Disciple Compiler.

- Strictness as default + region and effect typing.

- Many Haskell programs are also Disciple programs.