

The Poisoning Problem

Ben Lippmeier

Australian National University

FP-SYD 2009/06/18

What is the type of x in this expression?

```
if b then 5  
      else x
```

What is the type of x in this expression?

```
if b then (5 :: Int)
      else x
```

What is the type of x in this expression?

```
if b then (5 :: Int)
      else (x :: Int)
```

Both alternatives of an **if** have the same type.

$$\Gamma \vdash t_1 :: \text{Bool} \quad \Gamma \vdash t_2 :: T \quad \Gamma \vdash t_3 :: T$$

$$\Gamma \vdash \mathbf{if} \ t_1 \ \mathbf{then} \ t_2 \ \mathbf{else} \ t_3 \ :: T$$

π is a useful constant...

$$\pi = 3.1415926535\dots$$

... which is used in a useful program.

```
pi = 3.1415926535...
```

```
fun
```

```
  = do
```

```
    total = 0
```

```
    ...
```

```
    total := total + 5
```

```
    ...
```

```
    if b then total
```

```
        else pi
```

total is updated, so it must be Mutable...

```
pi = 3.1415926535...
```

```
fun
```

```
  = do total :: Mutable r => Float r
```

```
    total = 0
```

```
    ...
```

```
    total := total + 5
```

```
    ...
```

```
    if b then total
```

```
        else pi
```


total is updated, so it must be Mutable...

```
pi = 3.1415926535...
```

```
fun
```

```
  = do total :: Mutable r => Float r
```

```
      total = 0
```

```
      ...
```

```
      total := total + 5
```

```
      ...
```

```
      if b then total
```

```
          else pi
```


← same type

... and `pi` becomes `Mutable` also.

```
pi :: Mutable r => Float r  
pi = 3.1415926535...
```

```
fun  
  = do total :: Mutable r => Float r  
      total = 0  
      ...  
      total := total + 5  
      ...  
      if b then total  
        else pi
```

← same type



... and pi becomes Mutable also.

OH NOES!

```
pi :: Mutable r => Float r  
pi = 3.1415926535...
```

```
fun  
  = do total :: Mutable r => Float r  
      total = 0  
      ...  
      total := total + 5  
      ...  
      if b then total  
        else pi
```

... and `pi` becomes `Mutable` also.

OH NOES!

```
pi :: Mutable r => Float r
```

```
pi = 3.1415926535...
```

the type of `pi` is poisoned

```
fun
```

```
  = do total :: Mutable r => Float r
```

```
      total = 0
```

```
      ...
```

```
      total := total + 5
```

```
      ...
```

```
      if b then total
```

```
          else pi
```

Drinking from the well.

```
pi :: Mutable r => Float r
```

```
pi = 3.1415926535...
```

```
e = 2.71828183...
```

```
thing
```

```
  = if b then pi  
    else e
```

Drinking from the well.


```
pi :: Mutable r => Float r
```

```
pi = 3.1415926535...
```

```
e = 2.71828183...
```

```
thing
```

```
    = if b then pi  
      else e
```



same type

Drinking from the well.

```
pi :: Mutable r => Float r
```

```
pi = 3.1415926535...
```

```
e :: Mutable r => Float r
```

```
e = 2.71828183...
```

POISONED!

```
thing
```

```
  = if b then pi
```

```
    else e
```

← same type



pi is supposed to be constant.

```
pi :: Const s => Float s
```

```
pi = 3.1415926535...
```


total still has to be Mutable...

```
pi :: Const s => Float s
```

```
pi = 3.1415926535...
```

```
fun
```

```
  = do total :: Mutable r => Float r
```

```
      total = 0
```

```
      ...
```

```
      total := total + 5
```

```
      ...
```

```
      if b then total
```

```
          else pi
```

... but what type do we give the result?

```
pi :: Const s => Float s
```

```
pi = 3.1415926535...
```

```
fun
```

```
  = do total :: Mutable r => Float r
```

```
      total = 0
```

```
      ...
```

```
      total := total + 5
```

```
      ...
```

```
      if b then total :: ????????
```

```
      else pi
```

A new type for the result.

```
pi      :: Const s    => Float s
total   :: Mutable r  => Float r
```

```
(if b then total
  else pi)
  :: ???????
```

A new type for the result.

```
pi      :: Const s    => Float s
total   :: Mutable r  => Float r
```

```
(if b then total
 else pi)
:: ???????
```

pi and total are in
different regions

A new type for the result.

```
pi      :: Const s    => Float s
total  :: Mutable r  => Float r
```

```
(if b then total
else pi)
  :: (r <: q, s <: q)
  => Float q
```

the result could be in either region

A new type for the result.

```
pi    :: Const s    => Float s
```

```
total :: Mutable r => Float r
```

```
(if b then total
```

```
  else pi)
```

```
  :: (r <: q, s <: q, Blocked q)
```

```
  => Float q
```

the result could be in either region
and either constant or mutable

A new type for the result.

```
pi    :: Const s    => Float s
```

```
total :: Mutable r  => Float r
```

```
(if b then total
```

```
  else pi)
```

```
  :: (r <: q, s <: q, Blocked q)
```

```
  => Float q
```

the result could be in either region
and either constant or mutable
and you can't update it.

Further Reading

- **Witnessing Purity, Constancy and Mutability**
Ben Lippmeier
Submitted to APLAS 2009.
- **Once Upon a Polymorphic Type**
Keith Wansbrough, Simon Peyton Jones
POPL, 1999.
- **Monads, Effects and Transformations**
Nick Benton and Andrew Kennedy
Electronic Notes in Theoretical Computer Science, 1999.