

# CATCH<sup>1</sup>: Case and Termination Checking for Haskell

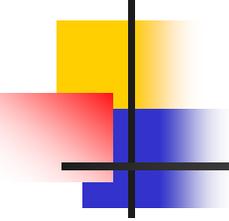
---

Neil Mitchell

(supervised by Colin Runciman)



<sup>1</sup> Name courtesy of Mike Dodds



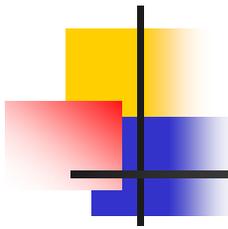
# Termination Checkers

---

**Q)** Does function  $f$  terminate?

**A)** {Yes, Don't know}

- Typically look for decreasing size
  - Primitive recursive
  - Walther recursion
  - Size change termination



# Does this terminate?

---

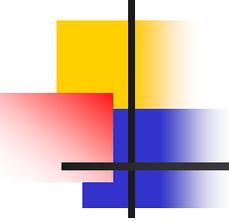
`fib :: Integer -> Integer`

`fib(1) = 1`

`fib(2) = 1`

`fib(n) = fib(n-1) + fib(n-2)`

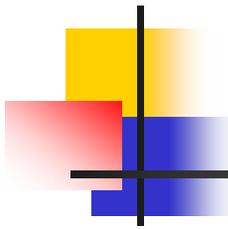
`fib(0) = ⊥NT`



# Remember the value!

---

- A function only stops terminating when its given a *value*
- Perhaps the question is wrong:
  - Q) Given a function  $f$  and a value  $x$ , does  $f(x)$  terminate?
  - Q) Given a function  $f$ , for what values of  $x$  does  $f(x)$  terminate?

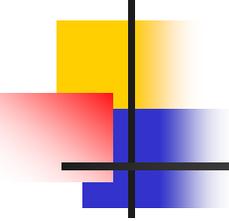


## But that's wrong...

---

```
fib n | n <= 0 =  
    error "bad programmer!"
```

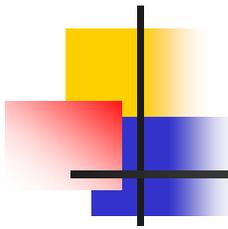
- A function should *never* non-terminate
- It *should* give an helpful error message
- There may be a few exceptions
  - But probably things that can't be proved
  - i.e. A Turing machine simulator



# CATCH: Haskell

---

- Haskell is:
  - A functional programming language
  - Lazy – not strict
- Only evaluates what is required
- Lazy allows:
  - Infinite data structures

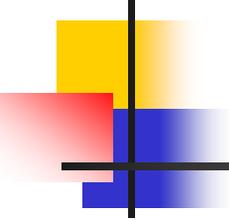


# Productivity

---

$[1..] = [1, 2, 3, 4, 5, 6, \dots]$

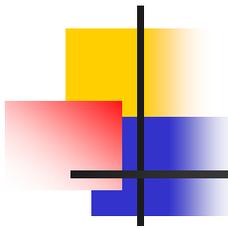
- Not terminating
- But is *productive*
  - Always another element
  - Time to generate “next result” is always finite



# The blame game

---

- `last [1..]` is  $\perp^{NT}$
- `last` is a useful function
- `[1..]` is a useful value
  
- Who is at fault?
  - The *caller* of `last`



# A Lazy Termination Checker

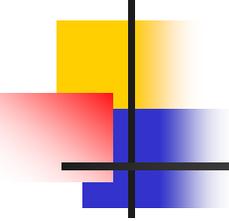
---

- All data/functions must be productive
- Can easily encode termination

`i sTerm :: [a] -> Bool`

`i sTerm [] = True`

`i sTerm (x: xs) = i sTerm xs`



# NF, WHNF

---

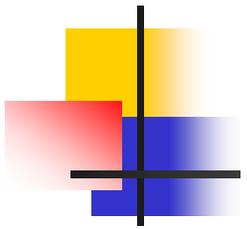
- Normal Form (NF)
  - Fully defined data structure
  - Possibly infinite
  - $\text{value}\{*\}$
- Weak Head Normal Form (WHNF)
  - Outer lump is a constructor
  - $\text{value}\{?\}$
- $\text{value}\{*\} \Rightarrow \text{value}\{?\}$

last x = case x of

(:) -> case x.tl of

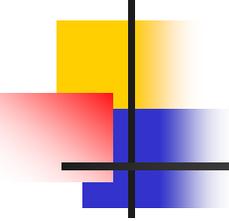
[] -> x.hd

(:) -> last x.tl


$$(\text{last } x)\{?\} = x\{[]\} \vee \left( \begin{array}{l} (x.tl\{:\}) \vee (x.hd\{?\}) \\ \wedge \\ (x.tl\{[]\}) \vee (\text{last } x.tl)\{?\} \end{array} \right)$$

$$\begin{aligned} (\text{last } x)\{?\} &= x\{[]\} \vee x.tl\{[]\} \vee (\text{last } x.tl)\{?\} \\ &= x\{[]\} \vee x.tl\{[]\} \vee x.tl.tl\{[]\} \vee \dots \\ &= \exists i \in L(tl^*), x.i\{[]\} \\ &= x.tl^\exists\{[]\} \end{aligned}$$

$$\begin{aligned} (\text{last } x)\{*\} &= (\text{last } x)\{?\} \wedge (x\{[]\} \vee x.tl\{[]\} \vee (\text{last } x.tl)\{*\}) \\ &= x.tl^\exists\{[]\} \end{aligned}$$



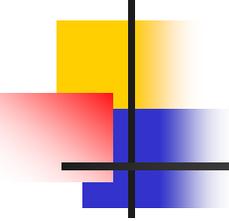
## And the result:

---

$$(\text{last } x)\{*\} = x\{*\} \wedge x.\text{tl}^{\exists}\{[]\}$$

- $x$  is defined
- $x$  has a  $[]$ ,  $x$  is finite

A nice result 😊



# Ackermann's Function

---

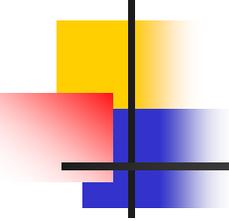
data Nat = S Nat | Z

ack Z n = S n

ack (S m) Z = ack m (S Z)

ack (S m) (S n) = ack m (ack (S m) n)

- $(\text{ack } m \ n)\{?\} = m.p^{\exists}\{Z\} \wedge m\{*\} \wedge n\{*\}$
- $\text{ack } 1 \ \infty = ?$  (answer is  $\infty$ )
- $\text{ack } \infty \ 1 = \perp^{\text{NT}}$



# Conclusion

---

- What lazy termination might mean
  - Productivity
  - Constraints on arguments
  - WHNF vs NF
- Lots to do!
  - Check it
  - Prove it
  - Implement it