

# Compilers and Formal Languages

Email: christian.urban at kcl.ac.uk

Office Hour: Thursdays 15 – 16

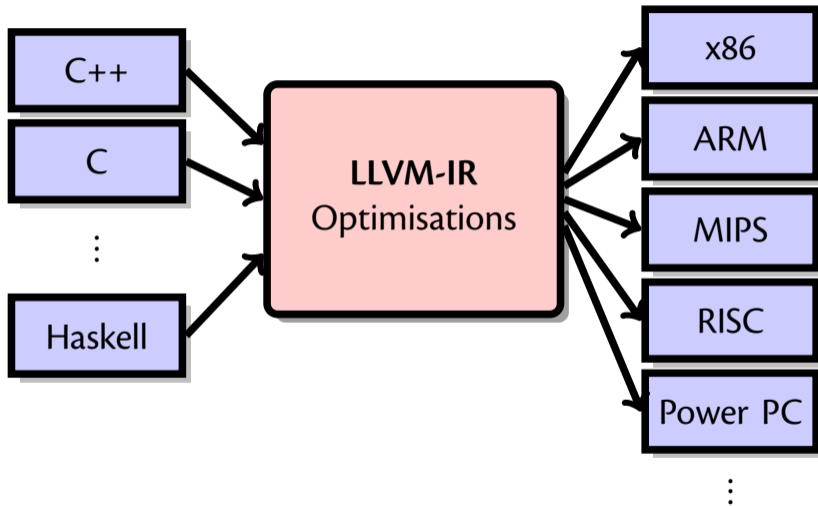
Location: N7.07 (North Wing, Bush House)

Slides & Progs: KEATS

Pollev: <https://pollev.com/cfl1tutoratki576>

1 Introduction, Languages	6 While-Language
2 Regular Expressions, Derivatives	7 Compilation, JVM
3 Automata, Regular Languages	8 Compiling Functional Languages
4 Lexing, Tokenising	9 Optimisations
5 Grammars, Parsing	10 LLVM

# LLVM: Overview



# Static Single-Assignment

$(1 + a) + (3 + (b * 5))$

```
1 let tmp0 = add 1 a in
2 let tmp1 = mul b 5 in
3 let tmp2 = add 3 tmp1 in
4 let tmp3 = add tmp0 tmp2
5   in tmp3
```

```
1 define i32 @fact (i32 %n) {  
2     %tmp_20 = icmp eq i32 %n, 0  
3     br i1 %tmp_20, label %if_branch_24, label %else_branch_25  
4 if_branch_24:  
5     ret i32 1  
6 else_branch_25:  
7     %tmp_22 = sub i32 %n, 1  
8     %tmp_23 = call i32 @fact (i32 %tmp_22)  
9     %tmp_21 = mul i32 %n, %tmp_23  
10    ret i32 %tmp_21  
11 }
```

```
def fact(n) = if n == 0 then 1 else n * fact(n - 1)
```

# LLVM Types

boolean	i1
byte	i8
short	i16
char	i16
integer	i32
long	i64
float	float
double	double
*_	pointer to
**_	pointer to a pointer to
[_]	arrays of

```
br i1 %var, label %if_br, label %else_br
```

```
icmp eq i32 %x, %y ; for equal
```

```
icmp sle i32 %x, %y ; signed less or equal
```

```
icmp slt i32 %x, %y ; signed less than
```

```
icmp ult i32 %x, %y ; unsigned less than
```

```
%var = call i32 @foo(...args...)
```

# Abstract Syntax Trees

```
// Fun language (expressions)
abstract class Exp
abstract class BExp

case class Call(name: String, args: List[Exp]) extends Exp
case class If(a: BExp, e1: Exp, e2: Exp) extends Exp
case class Write(e: Exp) extends Exp
case class Var(s: String) extends Exp
case class Num(i: Int) extends Exp
case class Aop(o: String, a1: Exp, a2: Exp) extends Exp
case class Sequence(e1: Exp, e2: Exp) extends Exp
case class Bop(o: String, a1: Exp, a2: Exp) extends BExp
```

# K-(Intermediate)Language

```
abstract class KExp
```

```
abstract class KVal
```

```
// K-Values
```

```
case class KVar(s: String) extends KVal
```

```
case class KNum(i: Int) extends KVal
```

```
case class Kop(o: String, v1: KVal, v2: KVal) extends KVal
```

```
case class KCall(o: String, vrs: List[KVal]) extends KVal
```

```
case class KWrite(v: KVal) extends KVal
```

```
// K-Expressions
```

```
case class KIf(x1: String, e1: KExp, e2: KExp) extends KExp
```

```
case class KLet(x: String, v: KVal, e: KExp) extends KExp
```

```
case class KReturn(v: KVal) extends KExp
```



# KLet

```
tmp0 = add 1 a  
tmp1 = mul b 5  
tmp2 = add 3 tmp1  
tmp3 = add tmp0 tmp2
```

```
KLet tmp0 , add 1 a in  
  KLet tmp1 , mul b 5 in  
    KLet tmp2 , add 3 tmp1 in  
      KLet tmp3 , add tmp0 tmp2 in  
        ...
```

```
case class KLet(x: String, e1: KVal, e2: KExp)
```

# KLet

```
tmp0 = add 1 a  
tmp1 = mul b 5  
tmp2 = add 3 tmp1  
tmp3 = add tmp0 tmp2
```

```
let tmp0 = add 1 a in  
  let tmp1 = mul b 5 in  
    let tmp2 = add 3 tmp1 in  
      let tmp3 = add tmp0 tmp2 in  
        ...
```

```
case class KLet(x: String, e1: KVal, e2: KExp)
```

# CPS-Translation

```
def CPS(e: Exp)(k: KVal => KExp) : KExp =  
  e match { ... }
```

the continuation k can be thought of:

```
let tmp0 = add 1 a in  
let tmp1 = mul □ 5 in  
let tmp2 = add 3 tmp1 in  
let tmp3 = add tmp0 tmp2 in  
KReturn tmp3
```

```
def fact(n: Int) : Int = {  
  if (n == 0) 1 else n * fact(n - 1)  
}
```

```
def factC(n: Int, ret: Int => Int) : Int = {  
  if (n == 0) ret(1)  
  else factC(n - 1, x => ret(n * x))  
}
```

```
fact(10)
```

```
factC(10, identity)
```

```
def fibC(n: Int, ret: Int => Int) : Int = {  
  if (n == 0 || n == 1) ret(1) else  
  fibC(n - 1,  
    r1 => fibC(n - 2,  
      r2 => ret(r1 + r2)))  
}  
  
fibC(10, identity)
```



































Are there more strings in

$L(a^*)$  or  $L((a + b)^*)$ ?

# Can you remember this HW?

- (1) How many basic regular expressions are there to match the string *abcd*?
- (2) How many if they cannot include **1** and **0**?
- (3) How many if they are also not allowed to contain stars?
- (4) How many if they are also not allowed to contain  $\_ + \_$ ?

**There are more problems, than  
there are programs.**

**There are more problems, than  
there are programs.**

**There must be a problem for which  
there is no program.**



# Subsets

If  $A \subseteq B$  then  $A$  has fewer or equal elements than  $B$

$A \subseteq B$  and  $B \subseteq A$

then  $A = B$

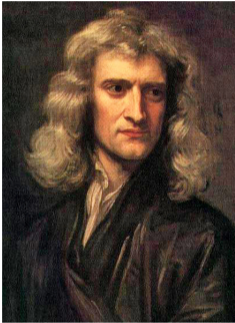


5 elements

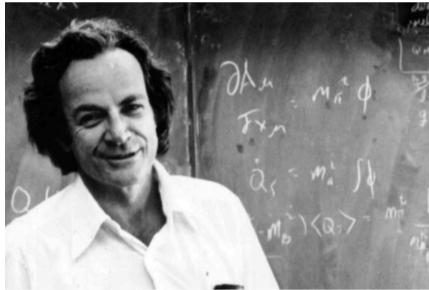


3 elements

# Newton vs Feynman



classical physics



quantum physics

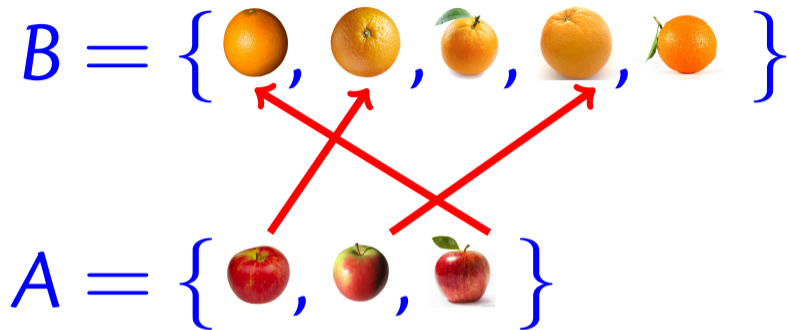
# The Goal of the Talk

- show you that something very unintuitive happens with very large sets
- convince you that there are more **problems** than **programs**

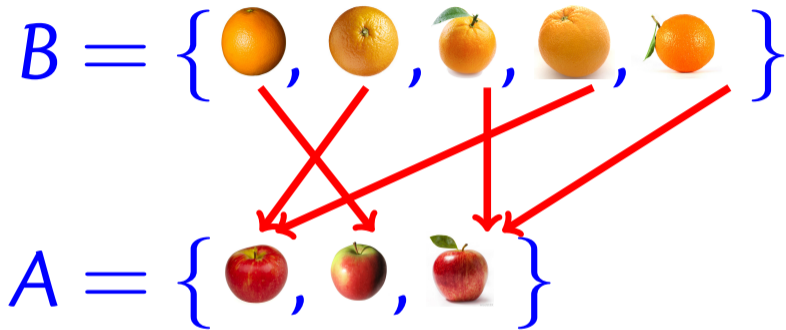
$$B = \{ \text{orange}, \text{orange}, \text{orange}, \text{orange}, \text{orange} \}$$

$$A = \{ \text{apple}, \text{apple}, \text{apple} \}$$

$$|A| = 5, |B| = 3$$



then  $|A| \leq |B|$



for  $=$  has to be a **one-to-one** mapping

# Cardinality

$|A| \stackrel{\text{def}}{=} \text{“how many elements”}$

$$A \subseteq B \Rightarrow |A| \leq |B|$$



# Cardinality

$|A| \stackrel{\text{def}}{=} \text{“how many elements”}$

$$A \subseteq B \Rightarrow |A| \leq |B|$$

if there is an injective function

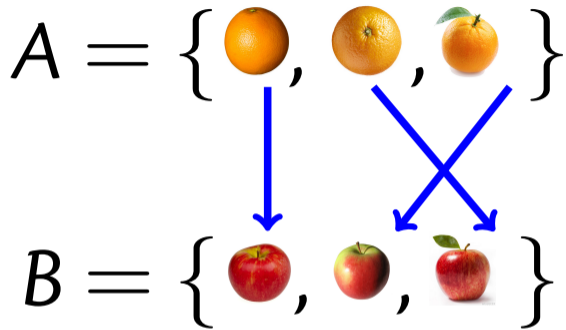
$$f: A \rightarrow B \text{ then } |A| \leq |B|$$

$$\forall xy. f(x) = f(y) \Rightarrow x = y$$

$A = \{ \text{orange}, \text{orange}, \text{orange} \}$

$B = \{ \text{apple}, \text{apple}, \text{apple} \}$





$$A = \{ \text{orange}, \text{orange}, \text{orange} \}$$

$$B = \{ \text{apple}, \text{apple}, \text{apple} \}$$

then  $|A| = |B|$

# Natural Numbers

$$\mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, 3, \dots\}$$

# Natural Numbers

$$\mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, 3, \dots\}$$

$A$  is **countable** iff  $|A| \leq |\mathbb{N}|$

# First Question

$$|\mathbb{N} - \{0\}| \quad ? \quad |\mathbb{N}|$$

$\geq$  or  $\leq$  or  $=$  ?

# First Question

$$|\mathbb{N} - \{0\}| \quad ? \quad |\mathbb{N}|$$

$\geq$  or  $\leq$  or  $=$  ?

$$x \mapsto x + 1,$$

$$|\mathbb{N} - \{0\}| = |\mathbb{N}|$$



$$|\mathbb{N} - \{0, 1\}| \quad ? \quad |\mathbb{N}|$$

$$|\mathbb{N} - \{0, 1\}| \quad ? \quad |\mathbb{N}|$$

$$|\mathbb{N} - \mathbb{O}| \quad ? \quad |\mathbb{N}|$$

$$\mathbb{O} \stackrel{\text{def}}{=} \text{odd numbers} \quad \{1, 3, 5, \dots\}$$

$$|\mathbb{N} - \{0, 1\}| \quad ? \quad |\mathbb{N}|$$

$$|\mathbb{N} - \mathbb{O}| \quad ? \quad |\mathbb{N}|$$

$\mathbb{O} \stackrel{\text{def}}{=} \text{odd numbers} \quad \{1, 3, 5, \dots\}$

$\mathbb{E} \stackrel{\text{def}}{=} \text{even numbers} \quad \{0, 2, 4, \dots\}$

$$|\mathbb{N} \cup -\mathbb{N}| \quad ? \quad |\mathbb{N}|$$

$\mathbb{N} \stackrel{\text{def}}{=} \text{positive numbers} \quad \{0, 1, 2, 3, \dots\}$

$-\mathbb{N} \stackrel{\text{def}}{=} \text{negative numbers} \quad \{0, -1, -2, -3, \dots\}$

$A$  is **countable** if there exists an injective  $f : A \rightarrow \mathbb{N}$

$A$  is **uncountable** if there does not exist an injective  $f : A \rightarrow \mathbb{N}$

countable:  $|A| \leq |\mathbb{N}|$

uncountable:  $|A| > |\mathbb{N}|$

$A$  is **countable** if there exists an injective  $f : A \rightarrow \mathbb{N}$

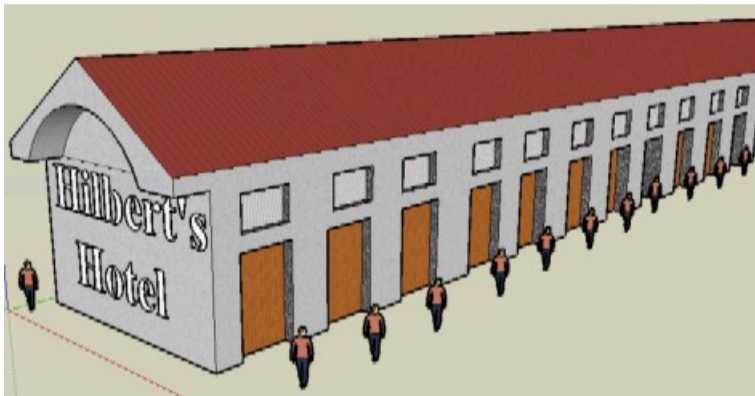
$A$  is **uncountable** if there does not exist an injective  $f : A \rightarrow \mathbb{N}$

countable:  $|A| \leq |\mathbb{N}|$

uncountable:  $|A| > |\mathbb{N}|$

Does there exist such an  $A$  ?

# Hilbert's Hotel



- ...has as many rooms as there are natural numbers

# Real Numbers between 0 and 1

1	3	3	3	3	3	3	...	...
2	1	2	3	4	5	6	7	
3	0	1	0	1	0	...		
4	7	8	5	3	9	...		

...



# Real Numbers between 0 and 1

1	4	3	3	3	3	3	...	...
2	1	2	3	4	5	6	7	
3	0	1	0	1	0	...		
4	7	8	5	3	9	...		

...

# Real Numbers between 0 and 1

1	4	3	3	3	3	3	...	...
2	1	3	3	4	5	6	7	
3	0	1	0	1	0	...		
4	7	8	5	3	9	...		

...

# Real Numbers between 0 and 1

1	4	3	3	3	3	3	...	...
2	1	3	3	4	5	6	7	
3	0	1	1	1	0	...		
4	7	8	5	3	9	...		

...

# Real Numbers between 0 and 1

1	4	3	3	3	3	3	...	...
2	1	3	3	4	5	6	7	
3	0	1	1	1	0	...		
4	7	8	5	4	9	...		

...

# Real Numbers between 0 and 1

1	4	3	3	3	3	3	...	...
2	1	3	3	4	5	6	7	
3	0	1	1	1	0	...		
4	7	8	5	4	9	...		

...

$$|\mathbb{N}| < |\mathbb{R}|$$

# The Set of Problems

 $\mathcal{N}_0$ 

	0	1	2	3	4	5	...
1	0	1	0	1	0	1	...
2	0	0	0	1	1	0	0
3	0	0	0	0	0	...	
4	1	1	0	1	1	...	

...

# The Set of Problems

 $\aleph_0$ 

	0	1	2	3	4	5	...
1	0	1	0	1	0	1	...
2	0	0	0	1	1	0	0
3	0	0	0	0	0	...	
4	1	1	0	1	1	...	

...

$$|\text{Progs}| = |\mathbb{N}| < |\text{Probs}|$$

# Halting Problem

Assume a program  $H$  that decides for all programs  $A$  and all input data  $D$  whether

- $H(A, D) \stackrel{\text{def}}{=} 1$  iff  $A(D)$  terminates
- $H(A, D) \stackrel{\text{def}}{=} 0$  otherwise



# Halting Problem (2)

Given such a program  $H$  define the following program  $C$ : for all programs  $A$

- $C(A) \stackrel{\text{def}}{=} 0$  iff  $H(A, A) = 0$
- $C(A) \stackrel{\text{def}}{=} \text{loops}$  otherwise

# Contradiction

$H(C, C)$  is either 0 or 1.

- $H(C, C) = 1 \xRightarrow{\text{def } H} C(C) \downarrow \xRightarrow{\text{def } C} H(C, C) = 0$

- $H(C, C) = 0 \xRightarrow{\text{def } H} C(C) \text{ loops} \xRightarrow{\text{def } C}$

$$H(C, C) = 1$$

Contradiction in both cases. So  $H$  cannot exist.

# Take Home Points

- there are sets that are more infinite than others
- even with the most powerful computer we can imagine, there are problems that cannot be solved by any program
- in CS we actually hit quite often such problems (halting problem)













































# Big Thank You!

- It is always fun to learn new things in CFL
- I want to add Higher-Order Functions and Algebraic Datatypes to Fun





# Big Thank You!

- Thanks for ALL the EoY feedback:

*“If all modules were as good as this one I would start recommending KCL over basically every single university instead of suggesting people look somewhere else.”*

