

# PEP Scala (5)

Email: christian.urban at kcl.ac.uk

Slides & Code: KEATS

Office Hour: Fridays 13:00 – 14:00

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

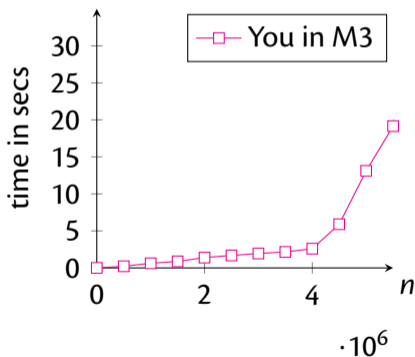
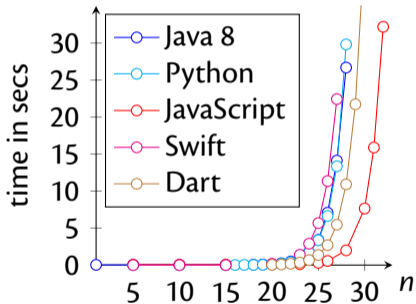
Pollev: <https://pollev.com/cfltutoratki576>

# Housekeeping

- SGTs still ongoing next week

# Main 3: Regexes

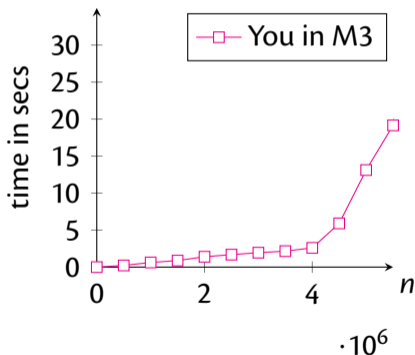
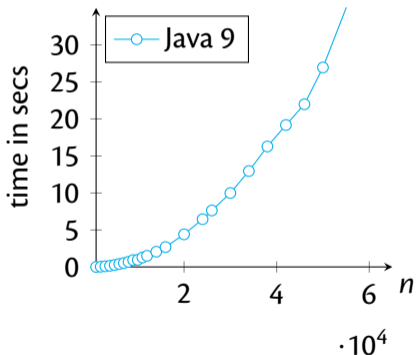
Graphs: regex  $(a^*)^*b$  and strings  $\underbrace{a \dots a}_n$



<https://vimeo.com/112065252>

# Main 3: Regexes

Graphs: regex  $(a^*)^*b$  and strings  $\underbrace{a \dots a}_n$



<https://vimeo.com/112065252>

# Polyorphic Types

```
def length_string_list(lst: List[String]): Int =  
  lst match {  
    case Nil => 0  
    case x::xs => 1 + length_string_list(xs)  
  }
```

```
def length_int_list(lst: List[Int]): Int =  
  lst match {  
    case Nil => 0  
    case x::xs => 1 + length_int_list(xs)  
  }
```

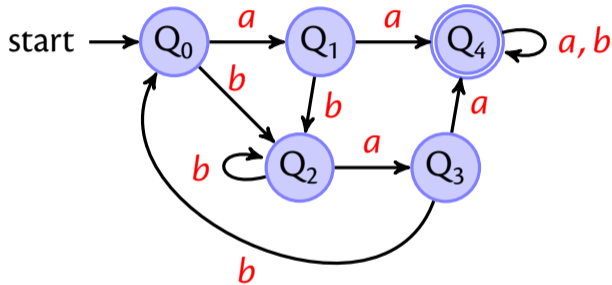
# Polyorphic Types

```
def length[A](lst: List[A]): Int = lst match {  
  case Nil => 0  
  case x::xs => 1 + length(xs)  
}
```

```
length(List("1", "2", "3", "4"))  
length(List(1, 2, 3, 4))
```

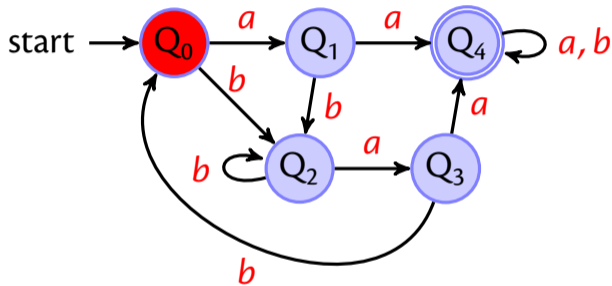
```
def map[A, B](lst: List[A], f: A => B): List[B] =  
  lst match {  
    case Nil => Nil  
    case x::xs => f(x)::map(xs, f)  
  }
```

# DFA's



*abaaa*

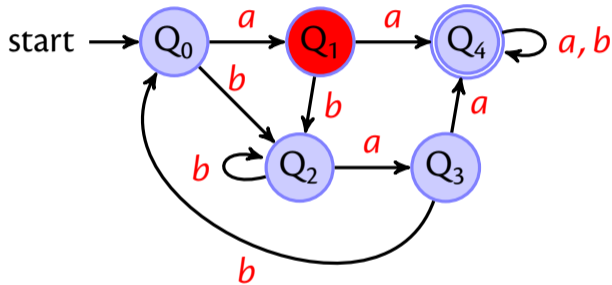
# DFA's



*abaaa*

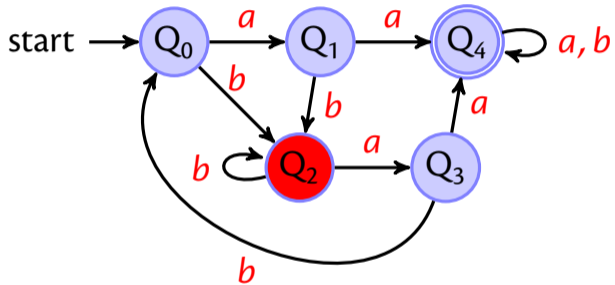


# DFA's



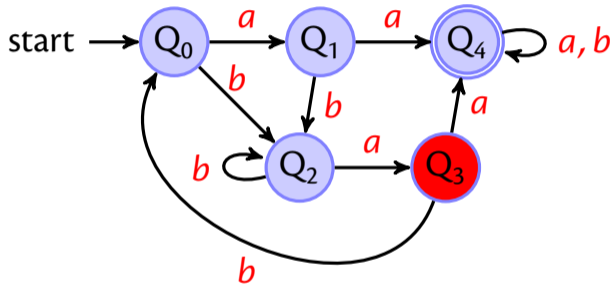
**a**baaa

# DFAs



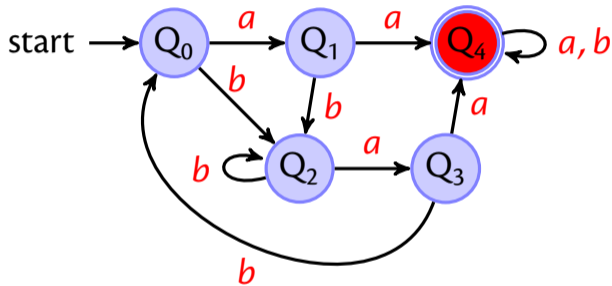
**ab**aaa

# DFA's



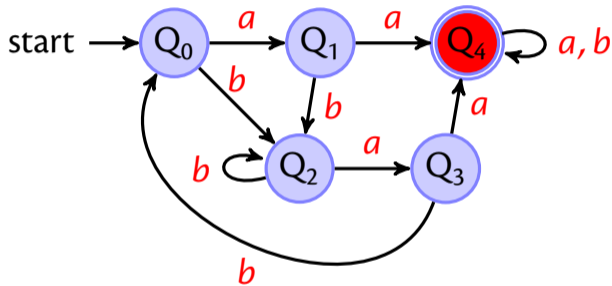
**aba**aa

# DFA's



**abaaa**

# DFA's



***abaaa*  $\Rightarrow$  *yes***

# DFAs

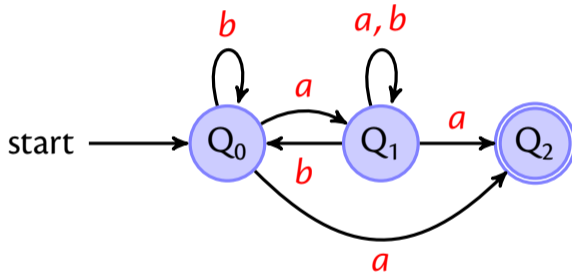
A **deterministic finite automaton** (DFA) consists of 5 things:

- an alphabet  $\Sigma$
- a set of states  $Q_s$
- one of these states is the start state  $Q_0$
- some states are accepting states  $F$ , and
- there is transition function  $\delta$

which takes a state and a character as arguments and produces a new state; this function might not be everywhere defined

$$A(\Sigma, Q_s, Q_0, F, \delta)$$

# NFAs



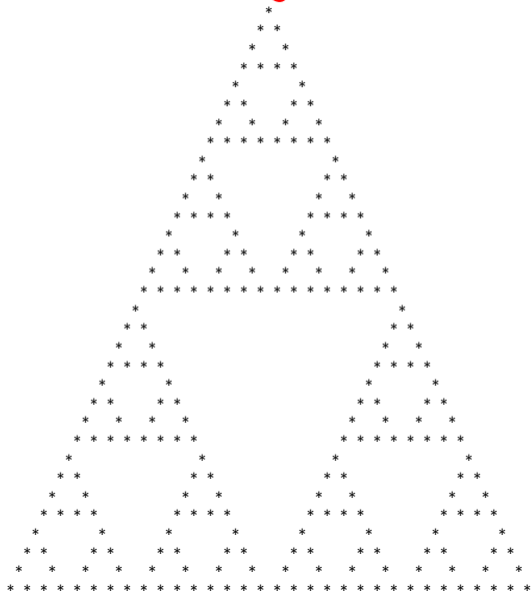
# Where to go on from here?

- Martin Odersky (EPFL) developed now Scala 3
- I use Ammonite by Haoyi Li
- Elm (<http://elm-lang.org>)...web applications with style
- Haskell, Ocaml, Standard ML, Scheme, ...





# Questions?



```
+++++++[ >+>++++< <- ]>+>>  
+<[ - [ >>+< <- ]+>> ]>+ [ - <<< [ -  
> [ + [ - ]+>+>>> - << ] < [ < ]>>+>  
+++++ [ <<++++>> - ]+<<+>. [ - ]  
<< ]>. >+ [ >> ]>+ ]
```





















