PEP Scala (5)

Email: christian.urban at kcl.ac.uk

Slides & Code: KEATS

Office Hour: Fridays 11:00 – 12:00

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

Pollev: https://pollev.com/cfltutoratki576

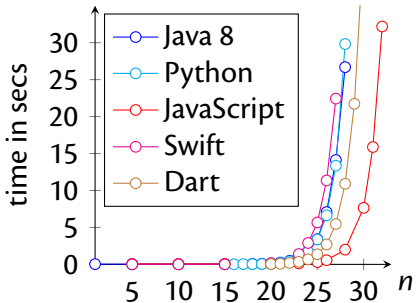
Housekeeping

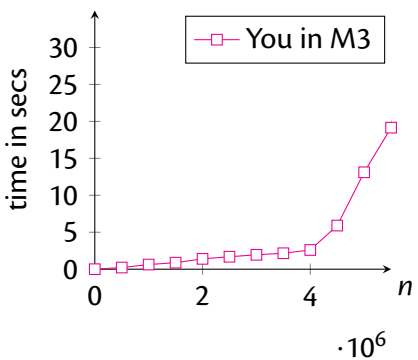
- SGTs still ongoing next week
- LGT next week online Ask-Me-Anything (will be recorded, TEAMS link will be emailed and published on KEATS)
- tests might break over Christmas

Junhuai Hou Chin Wan

Main 3: Regexes

Graphs: regex `(a*)*b` and strings $\underbrace{\texttt{a}\ldots\texttt{a}}_{n}$

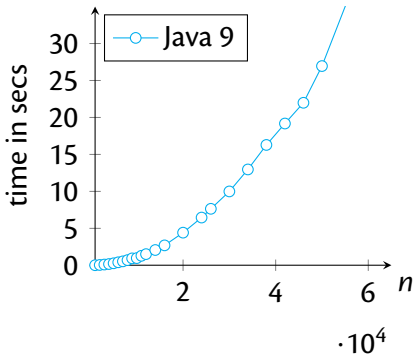


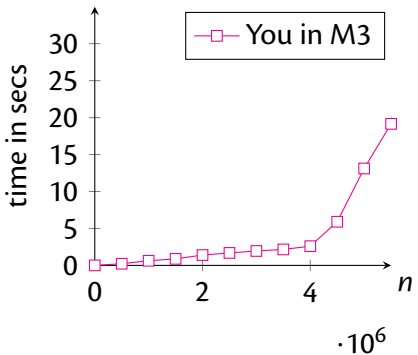


https://vimeo.com/112065252

Main 3: Regexes

Graphs: regex `(a*)*b` and strings $\underbrace{\texttt{a}\ldots\texttt{a}}_{n}$





https://vimeo.com/112065252

Plan for Today

- Implicits
- Polymorphic Types
- Immutable OOP
- Making Fun about Scala

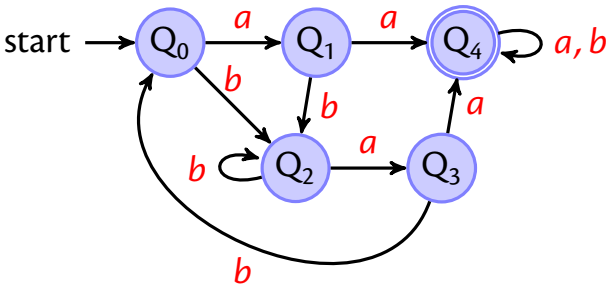
Polyorphic Types

```
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

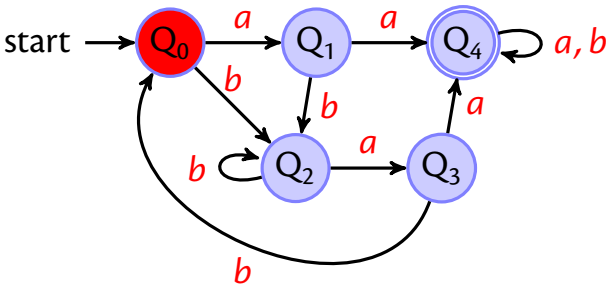
```
def length[A](lst: List[A]): Int = lst match {
  case Nil => 0
  case x::xs => 1 + length(xs)
}

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)
 }
```

DFAs



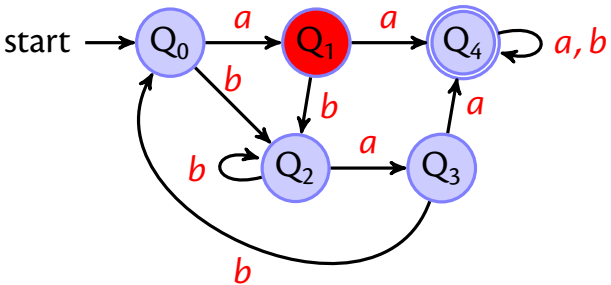
abaaa

DFAs



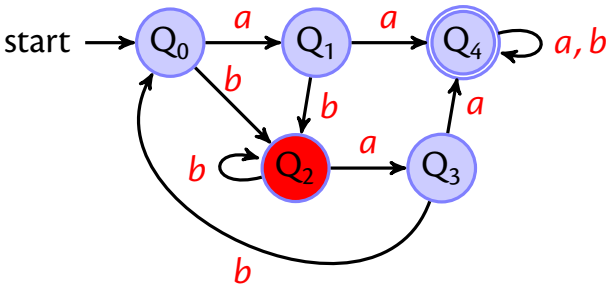
abaaa

DFAs



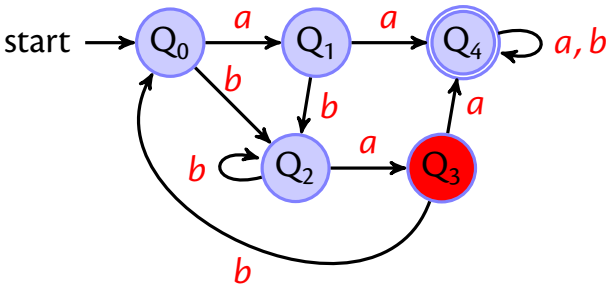
***a**baaa*

DFAs



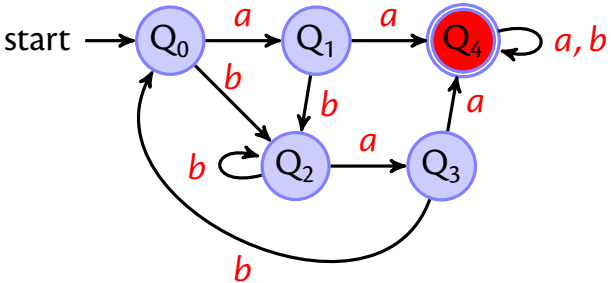
abaaa

DFAs



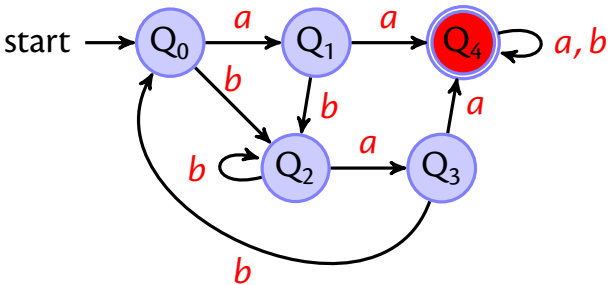
abaaa

DFAs



abaaa

DFAs



abaaa $\Rightarrow$ yes

DFAs

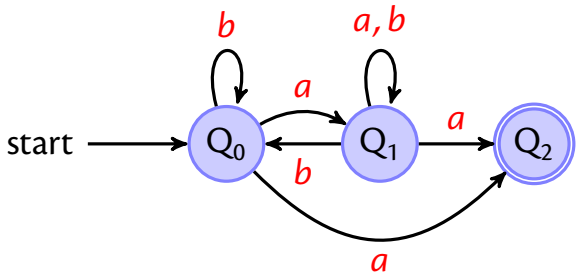
A **deterministic finite automaton** (DFA) consists of 5 things:

- an alphabet Σ
- a set of states Qs
- one of these states is the start state Q_0
- some states are accepting states F, and
- there is transition function δ

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

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

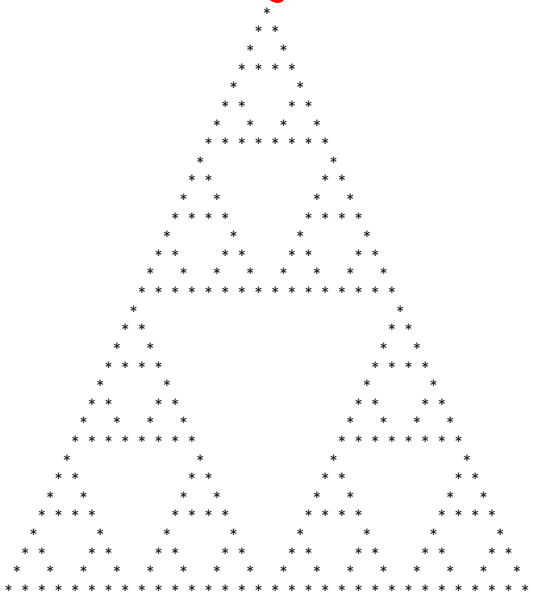
NFAs



Where to go on from here?

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

Questions?



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