

# Automata and Formal Languages (2)

Email: christian.urban at kcl.ac.uk  
Office: SI.27 (1st floor Strand Building)  
Slides: KEATS

# Languages

A **language** is a set of strings.

A **regular expression** specifies a set of strings, or language.

# Strings

Different ways of writing strings:

*”hello”*      *[h, e, l, l, o]*      *h::e::l::l::o::Nil*  
*"""*                      *[]*                      *Nil*

# Strings

Different ways of writing strings:

*”hello”*      $[h, e, l, l, o]$       $h :: e :: l :: l :: o :: Nil$   
*””*      $[]$       $Nil$

The concatenation operation on strings and sets of strings:

*”foo” @ ”bar” = ”foobar”*

$A @ B \stackrel{\text{def}}{=} \{s_1 @ s_2 \mid s_1 \in A \wedge s_2 \in B\}$

# Regular Expressions

Their inductive definition:

$r ::=$	$\emptyset$	null
	$\epsilon$	empty string / "" / []
	$c$	character
	$r_1 \cdot r_2$	sequence
	$r_1 + r_2$	alternative / choice
	$r^*$	star (zero or more)

# Re

Their indu

```
1 abstract class Rexp
2
3 case object NULL extends Rexp
4 case object EMPTY extends Rexp
5 case class CHAR(c: Char) extends Rexp
6 case class ALT(r1: Rexp, r2: Rexp) extends Rexp
7 case class SEQ(r1: Rexp, r2: Rexp) extends Rexp
8 case class STAR(r: Rexp) extends Rexp
```

$r ::= \emptyset$

|  $\epsilon$

|  $c$

|  $r_1 \cdot r_2$

|  $r_1 + r_2$

|  $r^*$

null

empty string / "" / []

character

sequence

alternative / choice

star (zero or more)

# The Meaning of a Regular Expression

$$L(\emptyset) \stackrel{\text{def}}{=} \emptyset$$

$$L(\epsilon) \stackrel{\text{def}}{=} \{""\}$$

$$L(c) \stackrel{\text{def}}{=} \{ "c" \}$$

$$L(r_1 + r_2) \stackrel{\text{def}}{=} L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) \stackrel{\text{def}}{=} L(r_1) @ L(r_2)$$

$$L(r^*) \stackrel{\text{def}}{=} \bigcup_{n \geq 0} L(r)^n$$

$L$  is a function from  
regular expressions to sets  
of strings

$$L : \text{Rexp} \Rightarrow \text{Set}[\text{String}]$$

# The Meaning of a Regular Expression

$$L(\emptyset) \stackrel{\text{def}}{=} \emptyset$$

$$L(\epsilon) \stackrel{\text{def}}{=} \{""\}$$

$$L(c) \stackrel{\text{def}}{=} \{ "c" \}$$

$$L(r_1 + r_2) \stackrel{\text{def}}{=} L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) \stackrel{\text{def}}{=} L(r_1) @ L(r_2)$$

$$L(r^*) \stackrel{\text{def}}{=} \bigcup_{n \geq 0} L(r)^n$$

$$L(r)^0 \stackrel{\text{def}}{=} \{""\}$$

$$L(r)^{n+1} \stackrel{\text{def}}{=} L(r) @ L(r)^n$$

$L$  is a function from  
regular expressions to sets  
of strings

$$L : \text{Rexp} \Rightarrow \text{Set}[\text{String}]$$



What is  $L(\mathbf{a}^*)$ ?

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c)$$

$$a + a \equiv? a$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c)$$

$$a \cdot a \equiv? a$$

$$\epsilon^* \equiv? \epsilon$$

$$\emptyset^* \equiv? \emptyset$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c)$$

$$a \cdot a \equiv? a$$

$$\epsilon^* \equiv? \epsilon$$

$$\emptyset^* \equiv? \emptyset$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c)$$

$$a \cdot a \equiv? a$$

$$\epsilon^* \equiv? \epsilon$$

$$\emptyset^* \equiv? \emptyset$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a$$

$$\epsilon^* \equiv? \epsilon$$

$$\emptyset^* \equiv? \emptyset$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a \quad \text{no}$$

$$\epsilon^* \equiv? \epsilon$$

$$\emptyset^* \equiv? \emptyset$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a \quad \text{no}$$

$$\epsilon^* \equiv? \epsilon \quad \text{yes}$$

$$\emptyset^* \equiv? \emptyset$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a \quad \text{no}$$

$$\epsilon^* \equiv? \epsilon \quad \text{yes}$$

$$\emptyset^* \equiv? \emptyset \quad \text{no}$$

$$\forall r. \quad r \cdot \epsilon \equiv? r$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$



# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a \quad \text{no}$$

$$\epsilon^* \equiv? \epsilon \quad \text{yes}$$

$$\emptyset^* \equiv? \emptyset \quad \text{no}$$

$$\forall r. \quad r \cdot \epsilon \equiv? r \quad \text{yes}$$

$$\forall r. \quad r + \epsilon \equiv? r$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a \quad \text{no}$$

$$\epsilon^* \equiv? \epsilon \quad \text{yes}$$

$$\emptyset^* \equiv? \emptyset \quad \text{no}$$

$$\forall r. \quad r \cdot \epsilon \equiv? r \quad \text{yes}$$

$$\forall r. \quad r + \epsilon \equiv? r \quad \text{no}$$

$$\forall r. \quad r + \emptyset \equiv? r$$

$$\forall r. \quad r \cdot \emptyset \equiv? r$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

$$(a + b) + c \equiv? a + (b + c) \quad \text{yes}$$

$$a + a \equiv? a \quad \text{yes}$$

$$(a \cdot b) \cdot c \equiv? a \cdot (b \cdot c) \quad \text{yes}$$

$$a \cdot a \equiv? a \quad \text{no}$$

$$\epsilon^* \equiv? \epsilon \quad \text{yes}$$

$$\emptyset^* \equiv? \emptyset \quad \text{no}$$

$$\forall r. \quad r \cdot \epsilon \equiv? r \quad \text{yes}$$

$$\forall r. \quad r + \epsilon \equiv? r \quad \text{no}$$

$$\forall r. \quad r + \emptyset \equiv? r \quad \text{yes}$$

$$\forall r. \quad r \cdot \emptyset \equiv? r \quad \text{no}$$

$$c \cdot (a + b) \equiv? (c \cdot a) + (c \cdot b)$$

$$a^* \equiv? \epsilon + (a \cdot a^*)$$

# Reg Exp Equivalences

	$(a + b) + c$	$\equiv^?$	$a + (b + c)$	yes
	$a + a$	$\equiv^?$	$a$	yes
	$(a \cdot b) \cdot c$	$\equiv^?$	$a \cdot (b \cdot c)$	yes
	$a \cdot a$	$\equiv^?$	$a$	no
	$\epsilon^*$	$\equiv^?$	$\epsilon$	yes
	$\emptyset^*$	$\equiv^?$	$\emptyset$	no
$\forall r.$	$r \cdot \epsilon$	$\equiv^?$	$r$	yes
$\forall r.$	$r + \epsilon$	$\equiv^?$	$r$	no
$\forall r.$	$r + \emptyset$	$\equiv^?$	$r$	yes
$\forall r.$	$r \cdot \emptyset$	$\equiv^?$	$r$	no
	$c \cdot (a + b)$	$\equiv^?$	$(c \cdot a) + (c \cdot b)$	
	$a^*$	$\equiv^?$	$\epsilon + (a \cdot a^*)$	

# Reg Exp Equivalences

	$(a + b) + c$	$\equiv^?$	$a + (b + c)$	yes
	$a + a$	$\equiv^?$	$a$	yes
	$(a \cdot b) \cdot c$	$\equiv^?$	$a \cdot (b \cdot c)$	yes
	$a \cdot a$	$\equiv^?$	$a$	no
	$\epsilon^*$	$\equiv^?$	$\epsilon$	yes
	$\emptyset^*$	$\equiv^?$	$\emptyset$	no
$\forall r.$	$r \cdot \epsilon$	$\equiv^?$	$r$	yes
$\forall r.$	$r + \epsilon$	$\equiv^?$	$r$	no
$\forall r.$	$r + \emptyset$	$\equiv^?$	$r$	yes
$\forall r.$	$r \cdot \emptyset$	$\equiv^?$	$r$	no
	$c \cdot (a + b)$	$\equiv^?$	$(c \cdot a) + (c \cdot b)$	yes
	$a^*$	$\equiv^?$	$\epsilon + (a \cdot a^*)$	

# Reg Exp Equivalences

	$(a + b) + c$	$\equiv^?$	$a + (b + c)$	yes
	$a + a$	$\equiv^?$	$a$	yes
	$(a \cdot b) \cdot c$	$\equiv^?$	$a \cdot (b \cdot c)$	yes
	$a \cdot a$	$\equiv^?$	$a$	no
	$\epsilon^*$	$\equiv^?$	$\epsilon$	yes
	$\emptyset^*$	$\equiv^?$	$\emptyset$	no
$\forall r.$	$r \cdot \epsilon$	$\equiv^?$	$r$	yes
$\forall r.$	$r + \epsilon$	$\equiv^?$	$r$	no
$\forall r.$	$r + \emptyset$	$\equiv^?$	$r$	yes
$\forall r.$	$r \cdot \emptyset$	$\equiv^?$	$r$	no
	$c \cdot (a + b)$	$\equiv^?$	$(c \cdot a) + (c \cdot b)$	yes
	$a^*$	$\equiv^?$	$\epsilon + (a \cdot a^*)$	yes

# The Specification of Matching

a regular expression  $r$  matches a string  $s$  is defined as

$$s \in L(r)$$

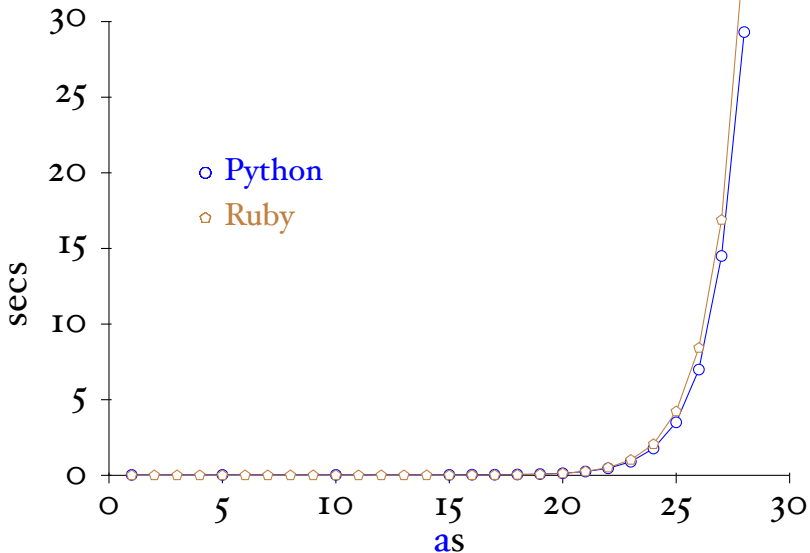
# The Specification of Matching

a regular expression  $r$  matches a string  $s$  is defined as

$$s \in L(r)$$



$$(a^{\{n\}}) \cdot a\{n\}$$



# A Matching Algorithm

...whether a regular expression can match the empty string:

$$\mathit{nullable}(\emptyset) \stackrel{\text{def}}{=} \mathit{false}$$

$$\mathit{nullable}(\epsilon) \stackrel{\text{def}}{=} \mathit{true}$$

$$\mathit{nullable}(c) \stackrel{\text{def}}{=} \mathit{false}$$

$$\mathit{nullable}(r_1 + r_2) \stackrel{\text{def}}{=} \mathit{nullable}(r_1) \vee \mathit{nullable}(r_2)$$

$$\mathit{nullable}(r_1 \cdot r_2) \stackrel{\text{def}}{=} \mathit{nullable}(r_1) \wedge \mathit{nullable}(r_2)$$

$$\mathit{nullable}(r^*) \stackrel{\text{def}}{=} \mathit{true}$$

# A Matching Algorithm

...whether a regular expression can match the empty string:

$nullable(\emptyset) \stackrel{\text{def}}{=} false$

$nullable(\epsilon) \stackrel{\text{def}}{=} true$

$nullable(c) \stackrel{\text{def}}{=} false$

$nullable(r_1 + r_2) \stackrel{\text{def}}{=} nullable(r_1) \vee nullable(r_2)$

$nullable(r_1 \cdot r_2) \stackrel{\text{def}}{=} nullable(r_1) \wedge nullable(r_2)$

$nullable(r^*) \stackrel{\text{def}}{=} true$

```
1 def nullable (r: Rexp) : Boolean = r match {
2   case NULL => false
3   case EMPTY => true
4   case CHAR(_) => false
5   case ALT(r1, r2) => nullable(r1) || nullable(r2)
6   case SEQ(r1, r2) => nullable(r1) && nullable(r2)
7   case STAR(_) => true
8 }
```

# The Derivative of a Rexp

If  $r$  matches the string  $c::s$ , what is a regular expression that matches  $s$ ?

$der\ c\ r$  gives the answer

# The Derivative of a Rexp (2)

$$\mathit{der} \ c (\emptyset) \stackrel{\text{def}}{=} \emptyset$$

$$\mathit{der} \ c (\epsilon) \stackrel{\text{def}}{=} \emptyset$$

$$\mathit{der} \ c (d) \stackrel{\text{def}}{=} \text{if } c = d \text{ then } \epsilon \text{ else } \emptyset$$

$$\mathit{der} \ c (r_1 + r_2) \stackrel{\text{def}}{=} \mathit{der} \ c r_1 + \mathit{der} \ c r_2$$

$$\mathit{der} \ c (r_1 \cdot r_2) \stackrel{\text{def}}{=} \text{if } \mathit{nullable}(r_1) \\ \text{then } (\mathit{der} \ c r_1) \cdot r_2 + \mathit{der} \ c r_2 \\ \text{else } (\mathit{der} \ c r_1) \cdot r_2$$

$$\mathit{der} \ c (r^*) \stackrel{\text{def}}{=} (\mathit{der} \ c r) \cdot (r^*)$$

# The Derivative of a Rexp (2)

$$\mathit{der} \ c (\emptyset) \stackrel{\text{def}}{=} \emptyset$$

$$\mathit{der} \ c (\epsilon) \stackrel{\text{def}}{=} \emptyset$$

$$\mathit{der} \ c (d) \stackrel{\text{def}}{=} \text{if } c = d \text{ then } \epsilon \text{ else } \emptyset$$

$$\mathit{der} \ c (r_1 + r_2) \stackrel{\text{def}}{=} \mathit{der} \ c r_1 + \mathit{der} \ c r_2$$

$$\mathit{der} \ c (r_1 \cdot r_2) \stackrel{\text{def}}{=} \text{if } \mathit{nullable}(r_1) \\ \text{then } (\mathit{der} \ c r_1) \cdot r_2 + \mathit{der} \ c r_2 \\ \text{else } (\mathit{der} \ c r_1) \cdot r_2$$

$$\mathit{der} \ c (r^*) \stackrel{\text{def}}{=} (\mathit{der} \ c r) \cdot (r^*)$$

$$\mathit{ders} \ [] r \stackrel{\text{def}}{=} r$$

$$\mathit{ders} (c :: s) r \stackrel{\text{def}}{=} \mathit{ders} s (\mathit{der} \ c r)$$

# The Derivative of a Rexp (2)

$der\ c(\emptyset) \stackrel{\text{def}}{=} \emptyset$

$der\ c(\epsilon) \stackrel{\text{def}}{=} \emptyset$

```
1  def der (r: Rexp, c: Char) : Rexp = r match {
2    case NULL => NULL
3    case EMPTY => NULL
4    case CHAR(d) => if (c == d) EMPTY else NULL
5    case ALT(r1, r2) => ALT(der(r1, c), der(r2, c))
6    case SEQ(r1, r2) =>
7      if (nullable(r1)) ALT(SEQ(der(r1, c), r2), der(r2, c))
8      else SEQ(der(r1, c), r2)
9    case STAR(r) => SEQ(der(r, c), STAR(r))
10 }
11
12 def ders (s: List[Char], r: Rexp) : Rexp = s match {
13   case Nil => r
14   case c::s => ders(s, der(c, r))
15 }
```

# The Rexp Matcher

```
1  abstract class Parser[I, T] {
2    def parse(ts: I): Set[(T, I)]
3
4    def parse_all(ts: I) : Set[T] =
5      for ((head, tail) <- parse(ts); if (tail.isEmpty))
6        yield head
7
8    def || (right : => Parser[I, T]) : Parser[I, T] =
9      new AltParser(this, right)
10   def ==>[S] (f: => T => S) : Parser [I, S] =
11     new FunParser(this, f)
12   def ~[S] (right : => Parser[I, S]) : Parser[I, (T, S)] =
13     new SeqParser(this, right)
14 }
```



# Proofs about Rexp

Remember their inductive definition:

$$\begin{array}{l} r ::= \emptyset \\ | \epsilon \\ | c \\ | r_1 \cdot r_2 \\ | r_1 + r_2 \\ | r^* \end{array}$$

If we want to prove something, say a property  $P(r)$ , for all regular expressions  $r$  then ...

# Proofs about Rexp (2)

- $P$  holds for  $\emptyset$ ,  $\epsilon$  and  $c$
- $P$  holds for  $r_1 + r_2$  under the assumption that  $P$  already holds for  $r_1$  and  $r_2$ .
- $P$  holds for  $r_1 \cdot r_2$  under the assumption that  $P$  already holds for  $r_1$  and  $r_2$ .
- $P$  holds for  $r^*$  under the assumption that  $P$  already holds for  $r$ .

# Proofs about Rexp (3)

Assume  $P(r)$  is the property:

$\text{nullable}(r)$  if and only if  $\epsilon \in L(r)$

# Proofs about Strings

If we want to prove something, say a property  $P(s)$ , for all strings  $s$  then ...

- $P$  holds for the empty string, and
- $P$  holds for the string  $c::s$  under the assumption that  $P$  already holds for  $s$

# Proofs about Strings (2)

Let  $\text{Der } c \ A$  be the set defined as

$$\text{Der } c \ A \stackrel{\text{def}}{=} \{ s \mid c::s \in A \}$$

Assume that  $L(\text{der } c \ r) = \text{Der } c \ (L(r))$ . Prove that

$\text{matcher}(r, s)$  if and only if  $s \in L(r)$

# Regular Languages

A language (set of strings) is **regular** iff there exists a regular expression that recognises all its strings.

# Automata

A deterministic finite automaton consists of:

- a set of states
- one of these states is the start state
- some states are accepting states, and
- there is transition function

which takes a state as argument and a character and produces a new state

this function might not always be defined