

Automata and Formal Languages (2)

Email: christian.urban at kcl.ac.uk

Office: S1.27 (1st floor Strand Building)

Slides: KEATS

Languages

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

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

Regular Expressions

Their inductive definition:

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

Regular Expressions

Their implementation in Scala:

```
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
```

The Meaning of a Regular Expression

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

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

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

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

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

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

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

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

The Meaning of a Regular Expression

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

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

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

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

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

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

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

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

$A @ B$

...you take out every string from A and concatenate it with every string in B

The Meaning of a Regular Expression

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

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

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

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

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

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

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

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

$A @ B$

...you take out every string from A and concatenate it with every string in B

L is a function from regular expressions to sets of strings

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

What is $L(a^*)$?

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

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

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

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

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

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

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

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

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

	$(a + b) + c$	$\equiv^?$	$a + (b + c)$	yes
	$a + a$	$\equiv^?$	a	yes
	$(a \cdot b) \cdot c$	$\equiv^?$	$a \cdot (b \cdot c)$	yes
	$a \cdot a$	$\equiv^?$	a	no
	ϵ^*	$\equiv^?$	ϵ	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	
	$c \cdot (a + b)$	$\equiv^?$	$(c \cdot a) + (c \cdot b)$	
	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
	ϵ^*	$\equiv?$	ϵ	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
	ϵ^*	$\equiv^?$	ϵ	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
	ϵ^*	$\equiv^?$	ϵ	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 Meaning of Matching

a regular expression r matches a string s
is defined as

$$s \in L(r)$$

The Meaning of Matching

a regular expression r matches a string s
is defined as

$$s \in L(r)$$

if $r_1 \equiv r_2$, then $s \in L(r_1)$ iff $s \in L(r_2)$

A Matching Algorithm

- given a regular expression r and a string s , say yes or no for whether

$$s \in L(r)$$

or not.

A Matching Algorithm

- given a regular expression r and a string s , say yes or no for whether

$$s \in L(r)$$

or not.

A Matching Algorithm

- given a regular expression r and a string s , say yes or no for whether

$$s \in L(r)$$

or not.

- Identifiers (strings of letters or digits, starting with a letter)
- Integers (a non-empty sequence of digits)
- Keywords (else, if, while, ...)
- White space (a non-empty sequence of blanks, newlines and tabs)

A Matching Algorithm

whether a regular expression matches the empty string:

```
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 ?

$\text{der } c \ r$ gives the answer

The Derivative

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

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

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

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

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

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

The Derivative

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

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

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

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

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

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

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

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

The Derivative

```
1 def deriv (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(deriv(r1, c), deriv(r2, c))
6   case SEQ(r1, r2) =>
7     if (nullable(r1)) ALT(SEQ(deriv(r1, c), r2), deriv(r2, c))
8     else SEQ(deriv(r1, c), r2)
9   case STAR(r) => SEQ(deriv(r, c), STAR(r))
10 }
```

The Rexp Matcher

```
1 def matches(r: Rexp, s: String) : Boolean =
2   nullable(derivs(r, s.toList))
3
4
5  /* Examples */
6
7  println(matches(SEQ(SEQ(CHAR('c'), CHAR('a')), CHAR('b')), "cab"))
8  println(matches(STAR(CHAR('a')), "aaa"))
9
10 /* Convenience using implicits */
11 implicit def string2rexp(s : String) : Rexp = {
12   s.foldRight (EMPTY: Rexp) ( (c, r) => SEQ(CHAR(c), r) )
13 }
14
15 println(matches("cab" , "cab"))
16 println(matches(STAR("a"), "aaa"))
17 println(matches(STAR("a"), "aaab"))
```