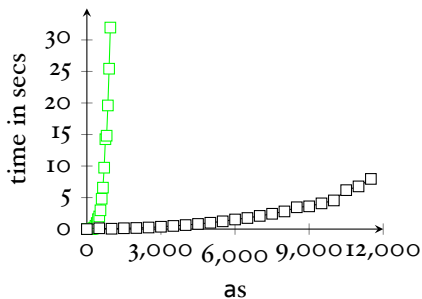
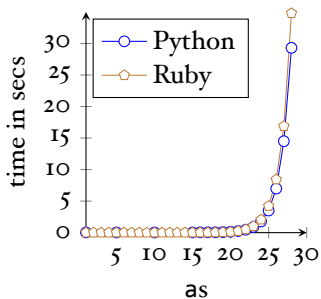


# Automata and Formal Languages (2)

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

# An Efficient Regular Expression Matcher



# Languages, Strings

- **Strings** are lists of characters, for example

$[], abc$  (Pattern match:  $c::s$ )

- A **language** is a set of strings, for example

$\{[], bello, foobar, a, abc\}$

- **Concatenation** of strings and sets

$foo @ bar = foobar$

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

# 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)    |

Th

```
abstract class Rexp
case object NULL extends Rexp
case object EMPTY extends Rexp
case class CHAR(c: Char) extends Rexp
case class ALT(r1: Rexp, r2: Rexp) extends Rexp
case class SEQ(r1: Rexp, r2: Rexp) extends Rexp
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

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

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

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

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

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

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

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

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

# The Meaning of a Regular Expression

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

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

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

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

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

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

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

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

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

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

What is  $L(a^*)$ ?



# Concrete Equivalences

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

$$a + a \equiv a$$

$$a + b \equiv b + a$$

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

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

# Concrete Equivalences

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

$$a + a \equiv a$$

$$a + b \equiv b + a$$

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

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

$$a \cdot a \not\equiv a$$

$$a + (b \cdot c) \not\equiv (a + b) \cdot (a + c)$$

# Corner Cases

$$\begin{array}{lcl} a \cdot \emptyset & \neq & a \\ a + \epsilon & \neq & a \\ \epsilon & \equiv & \emptyset^* \\ \epsilon^* & \equiv & \epsilon \\ \emptyset^* & \neq & \emptyset \end{array}$$

# Simplification Rules

$$r + \emptyset \equiv r$$

$$\emptyset + r \equiv r$$

$$r \cdot \epsilon \equiv r$$

$$\epsilon \cdot r \equiv r$$

$$r \cdot \emptyset \equiv \emptyset$$

$$\emptyset \cdot r \equiv \emptyset$$

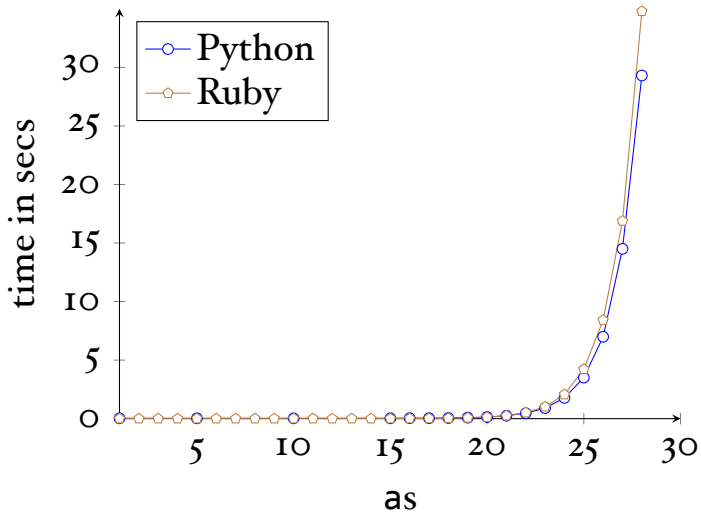
$$r + r \equiv r$$

# The Specification for Matching

A regular expression  $r$  matches a string  $s$   
if and only if

$$s \in L(r)$$

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



# Evil Regular Expressions

- Regular expression Denial of Service (ReDoS)
- Evil regular expressions
  - $(a?\{n\}) \cdot a\{n\}$
  - $(a^+)^+$
  - $([a-z]^+)^*$
  - $(a + a \cdot a)^+$
  - $(a + a?)^+$

# A Matching Algorithm

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

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

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

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

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

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

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



# 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, Brzozowski 1964

# 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)

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

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

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

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

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

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

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

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

# Examples

Given  $r \stackrel{\text{def}}{=} ((a \cdot b) + b)^*$  what is

$$\text{der } a r = ?$$

$$\text{der } b r = ?$$

$$\text{der } c r = ?$$

# The Algorithm

Input:  $r_1, abc$

Step 1: build derivative of  $a$  and  $r_1$  ( $r_2 = \text{der } a r_1$ )

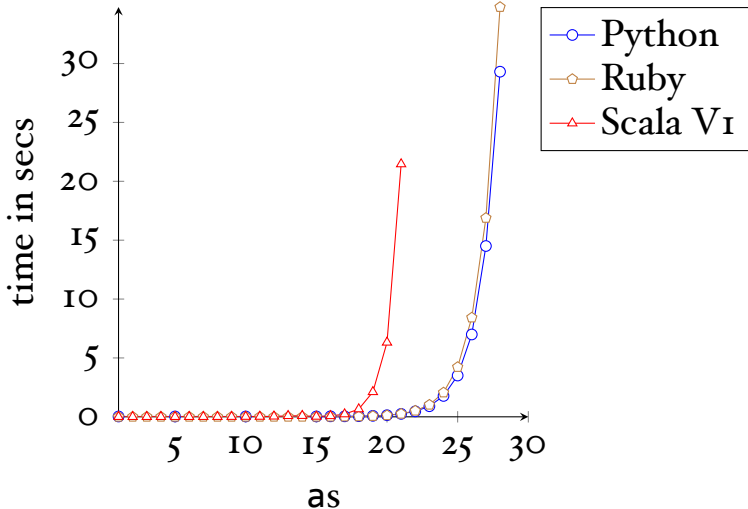
Step 2: build derivative of  $b$  and  $r_2$  ( $r_3 = \text{der } b r_2$ )

Step 3: build derivative of  $c$  and  $r_3$  ( $r_4 = \text{der } c r_3$ )

Step 4: the string is exhausted; test ( $\text{nullable}(r_4)$ )  
whether  $r_4$  can recognise  
the empty string

Output: result of the test  
 $\Rightarrow$  *true* or *false*

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



# A Problem

We represented the “n-times”  $a\{n\}$  as a sequence regular expression:

1:  $a$

2:  $a \cdot a$

3:  $a \cdot a \cdot a$

...

13:  $a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a$

...

20:

This problem is aggravated with  $a?$  being represented as  $\epsilon + a$ .

# Solving the Problem

What happens if we extend our regular expressions

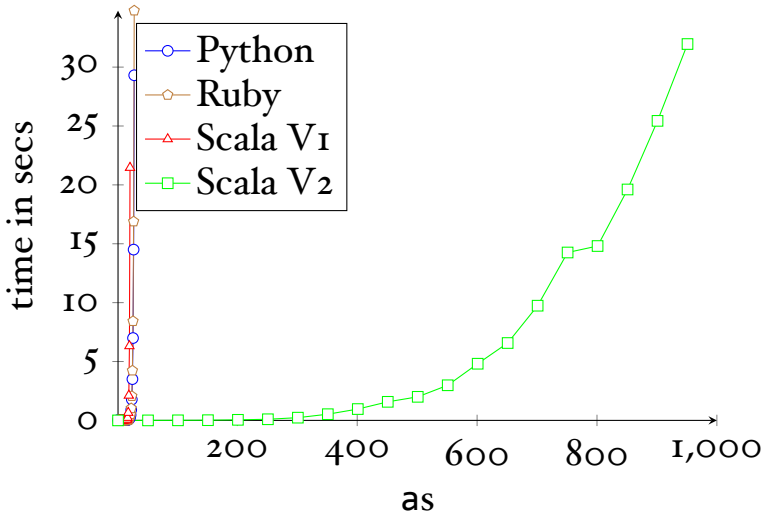
$$r ::= \dots$$

|  |  |          |
|--|--|----------|
|  |  | $r\{n\}$ |
|  |  | $r?$     |

What is their meaning? What are the cases for *nullable* and *der*?



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



# Examples

Recall the example of  $r \stackrel{\text{def}}{=} ((a \cdot b) + b)^*$  with

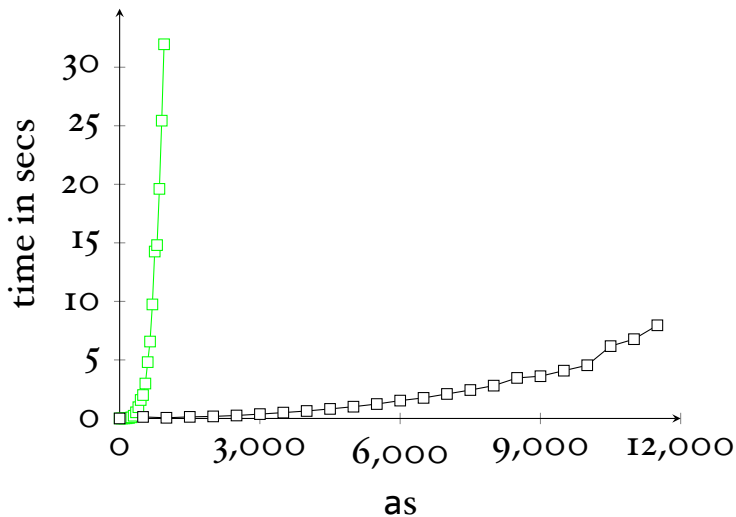
$$\text{der } a r = ((\epsilon \cdot b) + \emptyset) \cdot r$$

$$\text{der } b r = ((\emptyset \cdot b) + \epsilon) \cdot r$$

$$\text{der } c r = ((\emptyset \cdot b) + \emptyset) \cdot r$$

What are these regular expressions equivalent to?

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



# Proofs about Rexps

Remember their inductive definition:

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

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:

*nullable*( $r$ ) if and only if  $\square \in L(r)$

# Proofs about Rexp (4)

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

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

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

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

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

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

We can prove

$$L(\text{rev}(r)) = \{s^{-1} \mid s \in L(r)\}$$

by induction on  $r$ .

# Proofs about Rexp (5)

Let  $Der\ c\ A$  be the set defined as

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

We can prove

$$L(\text{der}\ c\ r) = Der\ c\ (L(r))$$

by induction on  $r$ .



# 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)

We can finally prove

*matches*( $r, s$ ) if and only if  $s \in L(r)$