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.

Strings

Different ways of writing strings:

$$"hello" \qquad [h, e, l, l, o] \qquad h :: e :: l :: l :: o :: Nil$$

$$"" \qquad [] \qquad Nil$$

Strings

Different ways of writing strings:

$$\text{"}hello\text{"} \qquad [h, e, l, l, o] \qquad h :: e :: l :: l :: o :: Nil$$

$$\text{""} \qquad [] \qquad Nil$$

The concatenation operation on strings and sets of strings:

$$\text{"}foo\text{"} \;@\; \text{"}bar\text{"} = \text{"}foobar\text{"}$$

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

Regular Expressions

Their inductive definition:

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

Re

Their indu

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

The Meaning of a Regular Expression

$$
\begin{aligned}
L(\varnothing) &\stackrel{\text{def}}{=} \varnothing\\
L(\epsilon) &\stackrel{\text{def}}{=} \{""\}\\
L(c) &\stackrel{\text{def}}{=} \{"c"\}\\
L(r_1 + r_2) &\stackrel{\text{def}}{=} L(r_1) \cup L(r_2)\\
L(r_1 \cdot r_2) &\stackrel{\text{def}}{=} L(r_1) @ L(r_2)\\
L(r^*) &\stackrel{\text{def}}{=} \bigcup_{n\geq 0} L(r)^n
\end{aligned}
$$

L is a function from regular expressions to sets of strings

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

The Meaning of a Regular Expression

$$
\begin{array}{rcl}
L(\varnothing) & \stackrel{\text{def}}{=} & \varnothing \\
L(\epsilon) & \stackrel{\text{def}}{=} & \{""\} \\
L(c) & \stackrel{\text{def}}{=} & \{"c"\} \\
L(r_1 + r_2) & \stackrel{\text{def}}{=} & L(r_1) \cup L(r_2) \\
L(r_1 \cdot r_2) & \stackrel{\text{def}}{=} & L(r_1) @ L(r_2) \\
L(r^*) & \stackrel{\text{def}}{=} & \bigcup_{n \geq 0} L(r)^n
\end{array}
$$

$$
\begin{array}{rcl}
L(r)^0 & \stackrel{\text{def}}{=} & \{""\} \\
L(r)^{n+1} & \stackrel{\text{def}}{=} & L(r) @ L(r)^n
\end{array}
$$

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

$$
\begin{array}{lrcl}
 & (a + b) + c & \equiv^? & a + (b + c) \\
 & a + a & \equiv^? & a \\
 & (a \cdot b) \cdot c & \equiv^? & a \cdot (b \cdot c) \\
 & a \cdot a & \equiv^? & a \\
 & \epsilon^* & \equiv^? & \epsilon \\
 & \varnothing^* & \equiv^? & \varnothing \\
\forall\ \text{r.} & r \cdot \epsilon & \equiv^? & r \\
\forall\ \text{r.} & r + \epsilon & \equiv^? & r \\
\forall\ \text{r.} & r + \varnothing & \equiv^? & r \\
\forall\ \text{r.} & r \cdot \varnothing & \equiv^? & r \\
 & c \cdot (a + b) & \equiv^? & (c \cdot a) + (c \cdot b) \\
 & a^* & \equiv^? & \epsilon + (a \cdot a^*)
\end{array}
$$

Reg Exp Equivalences

$$
\begin{array}{llcll}
 & (a+b)+c & \equiv^? & a+(b+c) & \text{yes} \\
 & a+a & \equiv^? & a & \\
 & (a\cdot b)\cdot c & \equiv^? & a\cdot(b\cdot c) & \\
 & a\cdot a & \equiv^? & a & \\
 & \epsilon^* & \equiv^? & \epsilon & \\
 & \varnothing^* & \equiv^? & \varnothing & \\
\forall\ \text{r.} & r\cdot\epsilon & \equiv^? & r & \\
\forall\ \text{r.} & r+\epsilon & \equiv^? & r & \\
\forall\ \text{r.} & r+\varnothing & \equiv^? & r & \\
\forall\ \text{r.} & r\cdot\varnothing & \equiv^? & r & \\
 & c\cdot(a+b) & \equiv^? & (c\cdot a)+(c\cdot b) & \\
 & a^* & \equiv^? & \epsilon+(a\cdot a^*) &
\end{array}
$$

Reg Exp Equivalences

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

The Specification for Matching

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

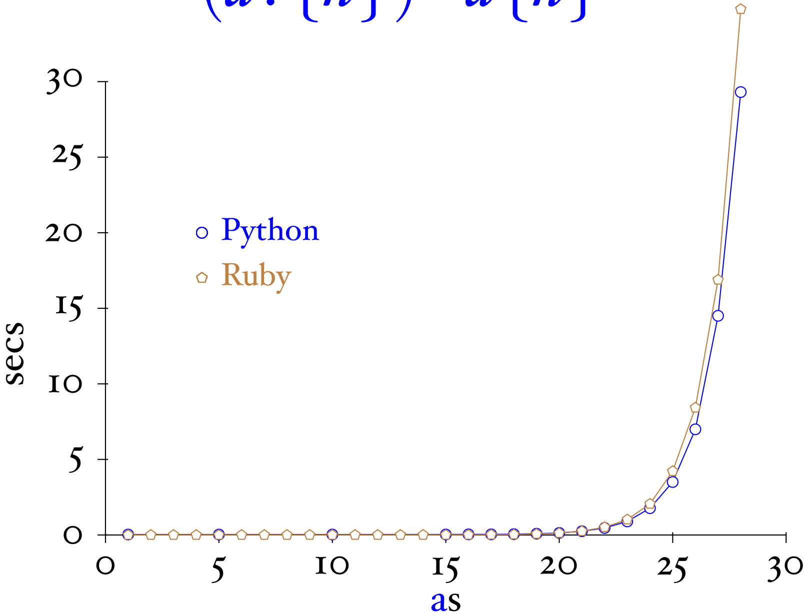
$$s \in L(r)$$

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
  - $(a?\{n\}) \cdot a\{n\}$
  - $(a^+)^+$
  - $([a\text{-}z]^+)^*$
  - $(a + a \cdot a)^+$
  - $(a + a?)^+$

A Matching Algorithm

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

$$
\begin{array}{lcl}
\mathit{nullable}(\varnothing) & \stackrel{\text{def}}{=} & \mathit{false} \\
\mathit{nullable}(\epsilon) & \stackrel{\text{def}}{=} & \mathit{true} \\
\mathit{nullable}(c) & \stackrel{\text{def}}{=} & \mathit{false} \\
\mathit{nullable}(r_1 + r_2) & \stackrel{\text{def}}{=} & \mathit{nullable}(r_1) \vee \mathit{nullable}(r_2) \\
\mathit{nullable}(r_1 \cdot r_2) & \stackrel{\text{def}}{=} & \mathit{nullable}(r_1) \wedge \mathit{nullable}(r_2) \\
\mathit{nullable}(r^*) & \stackrel{\text{def}}{=} & \mathit{true}
\end{array}
$$

A Matching Algorithm

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

$$
\begin{aligned}
nullable(\varnothing) &\stackrel{\text{def}}{=} false\\
nullable(\epsilon) &\stackrel{\text{def}}{=} true\\
nullable(c) &\stackrel{\text{def}}{=} false\\
nullable(r_1 + r_2) &\stackrel{\text{def}}{=} nullable(r_1) \vee nullable(r_2)\\
nullable(r_1 \cdot r_2) &\stackrel{\text{def}}{=} nullable(r_1) \wedge nullable(r_2)
\end{aligned}
$$

nu

```
def nullable (r: Rexp) : Boolean = r match {
  case NULL => false
  case EMPTY => true
  case CHAR(_) => false
  case ALT(r1, r2) => nullable(r1) || nullable(r2)
  case SEQ(r1, r2) => nullable(r1) && nullable(r2)
  case STAR(_) => 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

The Derivative of a Rexp (2)

$$
\begin{array}{lcl}
der\,c\,(\varnothing) & \stackrel{\text{def}}{=} & \varnothing \\
der\,c\,(\epsilon) & \stackrel{\text{def}}{=} & \varnothing \\
der\,c\,(d) & \stackrel{\text{def}}{=} & \text{if}\ c = d\ \text{then}\ \epsilon\ \text{else}\ \varnothing \\
der\,c\,(r_1 + r_2) & \stackrel{\text{def}}{=} & der\,c\,r_1 + der\,c\,r_2 \\
der\,c\,(r_1 \cdot r_2) & \stackrel{\text{def}}{=} & \text{if}\ nullable(r_1) \\
 & & \text{then}\ (der\,c\,r_1) \cdot r_2 + der\,c\,r_2 \\
 & & \text{else}\ (der\,c\,r_1) \cdot r_2 \\
der\,c\,(r^*) & \stackrel{\text{def}}{=} & (der\,c\,r) \cdot (r^*)
\end{array}
$$

The Derivative of a Rexp (2)

$$
\begin{array}{lcl}
der\,c\,(\varnothing) & \stackrel{\text{def}}{=} & \varnothing \\
der\,c\,(\epsilon) & \stackrel{\text{def}}{=} & \varnothing \\
der\,c\,(d) & \stackrel{\text{def}}{=} & \text{if } c = d \text{ then } \epsilon \text{ else } \varnothing \\
der\,c\,(r_1 + r_2) & \stackrel{\text{def}}{=} & der\,c\,r_1 + der\,c\,r_2 \\
der\,c\,(r_1 \cdot r_2) & \stackrel{\text{def}}{=} & \text{if } nullable(r_1) \\
 & & \text{then } (der\,c\,r_1) \cdot r_2 + der\,c\,r_2 \\
 & & \text{else } (der\,c\,r_1) \cdot r_2 \\
der\,c\,(r^*) & \stackrel{\text{def}}{=} & (der\,c\,r) \cdot (r^*) \\
ders\,[]\,r & \stackrel{\text{def}}{=} & r \\
ders\,(c::s)\,r & \stackrel{\text{def}}{=} & ders\,s\,(der\,c\,r)
\end{array}
$$

The Derivative of a Rexp (2)

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

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

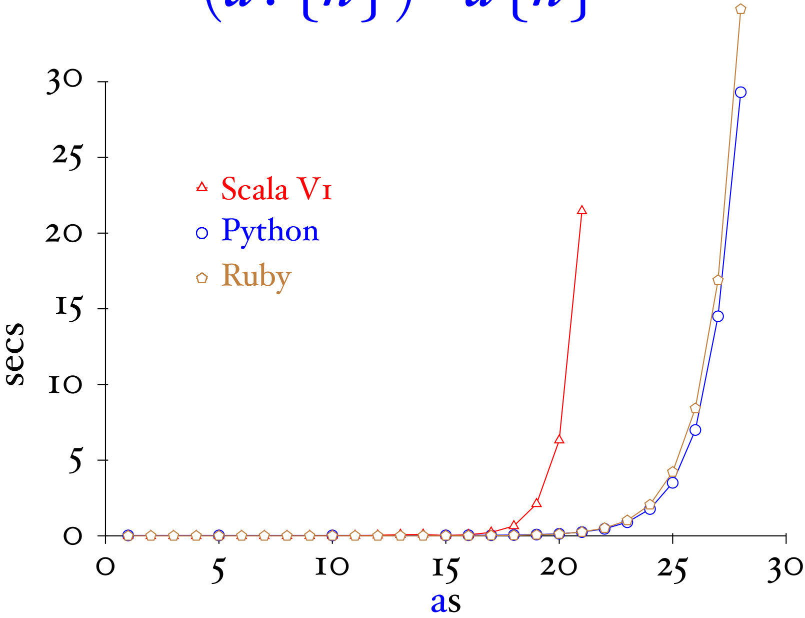
```
def der (r: Rexp, c: Char) : Rexp = r match {
  case NULL => NULL
  case EMPTY => NULL
  case CHAR(d) => if (c == d) EMPTY else NULL
  case ALT(r1, r2) => ALT(der(r1, c), der(r2, c))
  case SEQ(r1, r2) =>
    if (nullable(r1)) ALT(SEQ(der(r1, c), r2), der(r2, c))
    else SEQ(der(r1, c), r2)
  case STAR(r) => SEQ(der(r, c), STAR(r))
}

def ders (s: List[Char], r: Rexp) : Rexp = s match {
  case Nil => r
  case c::s => ders(s, der(c, r))
}
```

Examples

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

$$der\; a\; r$$
$$der\; b\; r$$

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



Proofs about Rexps

Remember their inductive definition:

$$\begin{array}{rcl} r & ::= & \varnothing \\ & \mid & \epsilon \\ & \mid & c \\ & \mid & r_1 \cdot r_2 \\ & \mid & r_1 + r_2 \\ & \mid & r^* \end{array}$$

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

Proofs about Rexp (2)

- P holds for $\varnothing$, ϵ and $\mathbf{c}$
- 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:

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

Proofs about Rexp (4)

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

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

We can prove

$$L(der\,c\,r) = Der\,c\,(L(r))$$

by induction on r.

Proofs about Strings

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

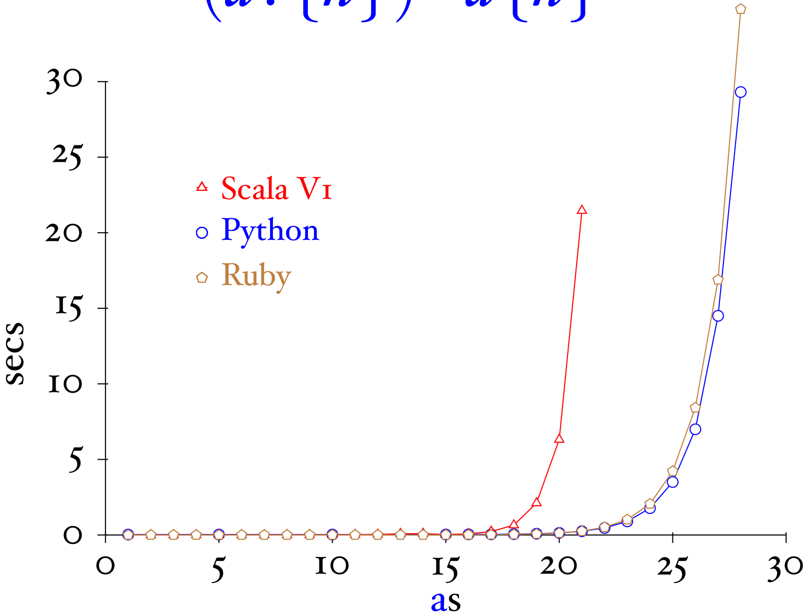
- P holds for the empty string, and
- P holds for the string $c::s$ under the assumption that P already holds for s

Proofs about Strings (2)

We can finally prove

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

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



A Problem

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

$$
\begin{array}{rl}
1: & a \\
2: & a \cdot a \\
3: & a \cdot a \cdot a \\
 & \ldots \\
13: & a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \\
 & \ldots \\
20: &
\end{array}
$$

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

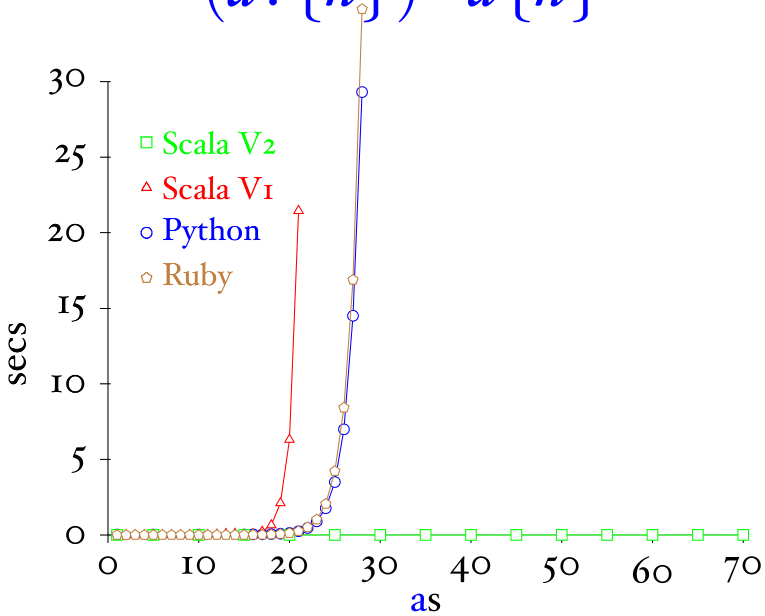
Solving the Problem

What happens if we extend our regular expressions

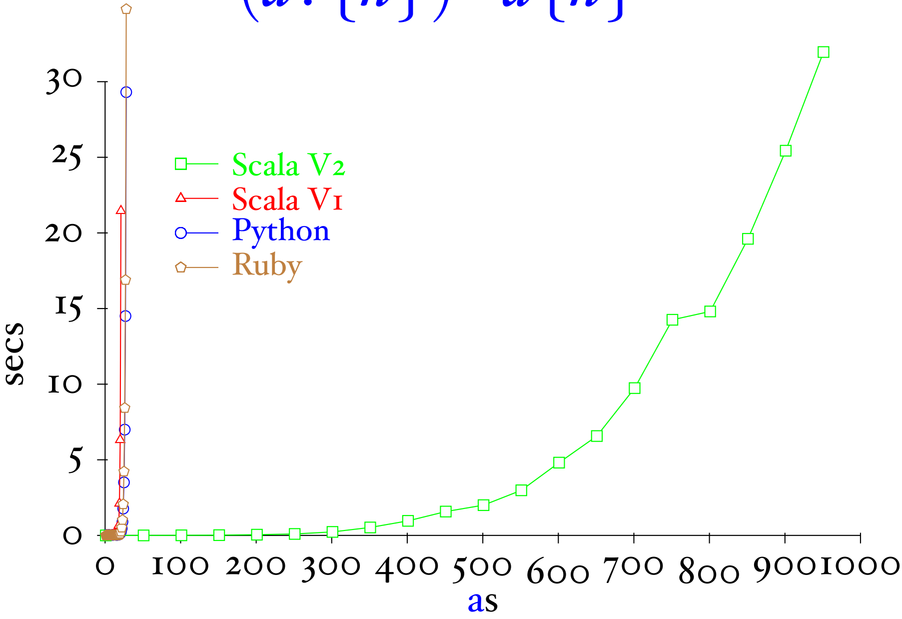
$$
\begin{array}{rcl}
r & ::= & \ldots \\
 & | & r\{n\} \\
 & | & r?
\end{array}
$$

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

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


Scala V2
Scala V1
Python
Ruby
secs
as
0
5
10
15
20
25
30
0
10
20
30
40
50
60
70

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



secs
30
25
20
15
10
5
0
0
100
200
300
400
500
600
700
800
900
1000
as
Scala V2
Scala V1
Python
Ruby

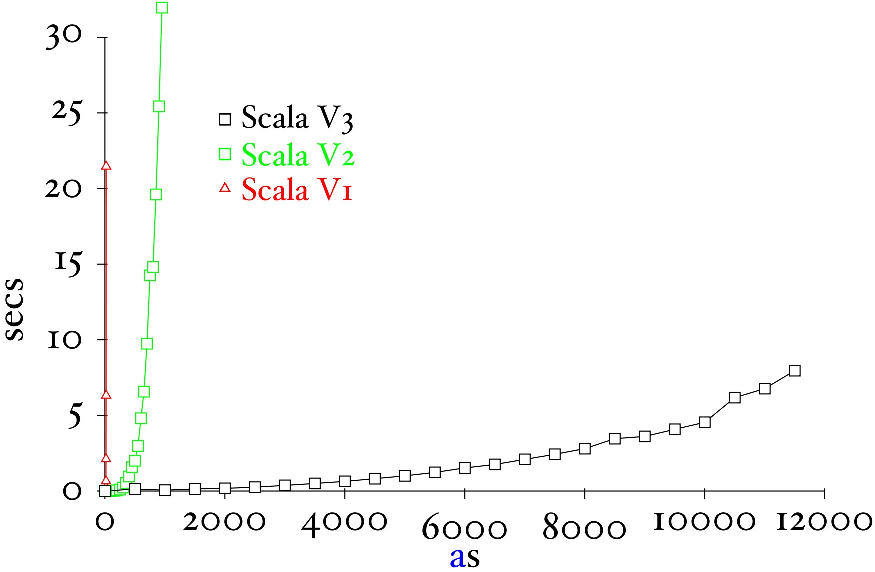
Examples

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

$$
\begin{aligned}
der\, a\, r &= ((\epsilon \cdot b) + \varnothing) \cdot r \\
der\, b\, r &= ((\varnothing \cdot b) + \epsilon) \cdot r
\end{aligned}
$$

What are these regular expressions equal to?

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



Scala V3
Scala V2
Scala V1
secs
30
25
20
15
10
5
0
0
2000
4000
6000
8000
10000
12000
as