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	$::=$	$\varnothing$	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)

Regular Expressions

Their implementation in Scala:

```
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
```

The Meaning of a Regular Expression

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

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

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

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

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

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

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

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

The Meaning of a Regular Expression

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

A @ B

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

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

The Meaning of a Regular Expression

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

A @ B

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

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

L is a function from regular expressions to sets of strings

L : Rexp $\Rightarrow$ Set[String]

What is $L(a^*)$?

Reg Exp Equivalences

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

Reg Exp Equivalences

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

	$(a + b) + c$	$\stackrel{?}{\equiv}$	$a + (b + c)$	yes
	$a + a$	$\stackrel{?}{\equiv}$	a	yes
	$(a \cdot b) \cdot c$	$\stackrel{?}{\equiv}$	$a \cdot (b \cdot c)$	yes
	$a \cdot a$	$\stackrel{?}{\equiv}$	a	
	ϵ^*	$\stackrel{?}{\equiv}$	ϵ	
	$\varnothing^*$	$\stackrel{?}{\equiv}$	$\varnothing$	
$\forall\, r.$	$r \cdot \epsilon$	$\stackrel{?}{\equiv}$	r	
$\forall\, r.$	$r + \epsilon$	$\stackrel{?}{\equiv}$	r	
$\forall\, r.$	$r + \varnothing$	$\stackrel{?}{\equiv}$	r	
$\forall\, r.$	$r \cdot \varnothing$	$\stackrel{?}{\equiv}$	r	
	$c \cdot (a + b)$	$\stackrel{?}{\equiv}$	$(c \cdot a) + (c \cdot b)$	
	a^*	$\stackrel{?}{\equiv}$	$\epsilon + (a \cdot a^*)$	

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
$\forall$ r.	$r \cdot \epsilon$	$\equiv^?$	r	yes
$\forall$ r.	$r + \epsilon$	$\equiv^?$	r	no
$\forall$ 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	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

	$(a + b) + c$	$\stackrel{?}{\equiv}$	$a + (b + c)$	yes
	$a + a$	$\stackrel{?}{\equiv}$	a	yes
	$(a \cdot b) \cdot c$	$\stackrel{?}{\equiv}$	$a \cdot (b \cdot c)$	yes
	$a \cdot a$	$\stackrel{?}{\equiv}$	a	no
	ϵ^*	$\stackrel{?}{\equiv}$	ϵ	yes
	$\varnothing^*$	$\stackrel{?}{\equiv}$	$\varnothing$	no
$\forall\, r.$	$r \cdot \epsilon$	$\stackrel{?}{\equiv}$	r	yes
$\forall\, r.$	$r + \epsilon$	$\stackrel{?}{\equiv}$	r	no
$\forall\, r.$	$r + \varnothing$	$\stackrel{?}{\equiv}$	r	yes
$\forall\, r.$	$r \cdot \varnothing$	$\stackrel{?}{\equiv}$	r	no
	$c \cdot (a + b)$	$\stackrel{?}{\equiv}$	$(c \cdot a) + (c \cdot b)$	yes
	a^*	$\stackrel{?}{\equiv}$	$\epsilon + (a \cdot a^*)$	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:

```
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

$$
\begin{aligned}
\mathrm{der}\ c\ (\varnothing) &\stackrel{\mathrm{def}}{=} \varnothing \\
\mathrm{der}\ c\ (\epsilon) &\stackrel{\mathrm{def}}{=} \varnothing \\
\mathrm{der}\ c\ (d) &\stackrel{\mathrm{def}}{=} \mathrm{if}\ c = d\ \mathrm{then}\ [\,]\ \mathrm{else}\ \varnothing \\
\mathrm{der}\ c\ (r_1 + r_2) &\stackrel{\mathrm{def}}{=} (\mathrm{der}\ c\ r_1) + (\mathrm{der}\ c\ r_2) \\
\mathrm{der}\ c\ (r_1 \cdot r_2) &\stackrel{\mathrm{def}}{=} ((\mathrm{der}\ c\ r_1) \cdot r_2) + \\
&\qquad (\mathrm{if}\ \mathrm{nullable}\ r_1\ \mathrm{then}\ \mathrm{der}\ c\ r_2\ \mathrm{else}\ \varnothing) \\
\mathrm{der}\ c\ (r^*) &\stackrel{\mathrm{def}}{=} (\mathrm{der}\ c\ r) \cdot (r^*)
\end{aligned}
$$

The Derivative

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

The Derivative

```
def deriv (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(deriv(r1, c), deriv(r2, c))
  case SEQ(r1, r2) =>
    if (nullable(r1)) ALT(SEQ(deriv(r1, c), r2), deriv(r2, c))
    else SEQ(deriv(r1, c), r2)
  case STAR(r) => SEQ(deriv(r, c), STAR(r))
}
```

The Rexp Matcher

```
def matches(r: Rexp, s: String) : Boolean =
  nullable(derivs(r, s.toList))


/* Examples */

println(matches(SEQ(SEQ(CHAR('c'), CHAR('a')), CHAR('b')),"cab"))
println(matches(STAR(CHAR('a')),"aaa"))

/* Convenience using implicits */
implicit def string2rexp(s : String) : Rexp = {
  s.foldRight (EMPTY: Rexp) ( (c, r) => SEQ(CHAR(c), r) )
}

println(matches("cab" ,"cab"))
println(matches(STAR("a"),"aaa"))
println(matches(STAR("a"),"aaab"))
```

Proofs about Rexp

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 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 "" $\in L$(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

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