Automata and Formal Languages (5)

Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)
Slides: KEATS (also home work is there)

Last Week
Regexes and Values

Regular expressions and their corresponding values:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
r & ::= & \varnothing & v & ::= & \\
 & | & \epsilon & & & \mathit{Empty} \\
 & | & c & & | & \mathit{Char}(c) \\
 & | & r_1 \cdot r_2 & & | & \mathit{Seq}(v_1, v_2) \\
 & | & r_1 + r_2 & & | & \mathit{Left}(v) \\
 & & & & | & \mathit{Right}(v) \\
 & | & r^* & & | & [v_1, \ldots\ v_n]
\end{array}$$

$$\begin{tikzcd}
r_1 \arrow[r, "\mathit{der}\ a"] \arrow[d] & r_2 \arrow[r, "\mathit{der}\ b"] \arrow[d] & r_3 \arrow[r, "\mathit{der}\ c"] \arrow[d] & r_4\ \mathit{nullable} \arrow[d, "\mathit{mkeps}"] \\
v_1 & v_2 \arrow[l, "\mathit{inj}\ a"'] & v_3 \arrow[l, "\mathit{inj}\ b"'] & v_4 \arrow[l, "\mathit{inj}\ c"']
\end{tikzcd}$$

r_1: $a \cdot (b \cdot c)$
r_2: $\epsilon \cdot (b \cdot c)$
r_3: $(\varnothing \cdot (b \cdot c)) + (\epsilon \cdot c)$
r_4: $(\varnothing \cdot (b \cdot c)) + ((\varnothing \cdot c) + \epsilon)$

$$\begin{tikzcd}
r_1 \arrow[r, "\mathit{der}\,a"] \arrow[d] & r_2 \arrow[r, "\mathit{der}\,b"] \arrow[d] & r_3 \arrow[r, "\mathit{der}\,c"] \arrow[d] & r_4\ \mathit{nullable} \arrow[d, "\mathit{mkeps}"] \\
v_1 & v_2 \arrow[l, "\mathit{inj}\,a"'] & v_3 \arrow[l, "\mathit{inj}\,b"'] & v_4 \arrow[l, "\mathit{inj}\,c"']
\end{tikzcd}$$

v_1: $\mathit{Seq}(\mathit{Char}(a), \mathit{Seq}(\mathit{Char}(b), \mathit{Char}(c)))$
v_2: $\mathit{Seq}(\mathit{Empty}, \mathit{Seq}(\mathit{Char}(b), \mathit{Char}(c)))$
v_3: $\mathit{Right}(\mathit{Seq}(\mathit{Empty}, \mathit{Char}(c)))$
v_4: $\mathit{Right}(\mathit{Right}(\mathit{Empty}))$

Mkeps

Finding a (posix) value for recognising the empty string:

$$
\begin{array}{lcl}
\mathit{mkeps}\,\epsilon & \stackrel{\text{def}}{=} & \mathit{Empty}\\
\mathit{mkeps}\,r_1 + r_2 & \stackrel{\text{def}}{=} & \text{if}\ \mathit{nullable}(r_1)\\
 & & \text{then}\ \mathit{Left}(\mathit{mkeps}(r_1))\\
 & & \text{else}\ \mathit{Right}(\mathit{mkeps}(r_2))\\
\mathit{mkeps}\,r_1 \cdot r_2 & \stackrel{\text{def}}{=} & \mathit{Seq}(\mathit{mkeps}(r_1), \mathit{mkeps}(r_2))\\
\mathit{mkeps}\,r^* & \stackrel{\text{def}}{=} & []
\end{array}
$$

Inject

Injecting ("Adding") a character to a value

$$
\begin{array}{ll}
\mathit{inj}\,(c)\,c\,\mathit{Empty} & \overset{\mathrm{def}}{=} \mathit{Char}\,c \\
\mathit{inj}\,(r_1 + r_2)\,c\,\mathit{Left}(v) & \overset{\mathrm{def}}{=} \mathit{Left}(\mathit{inj}\,r_1\,c\,v) \\
\mathit{inj}\,(r_1 + r_2)\,c\,\mathit{Right}(v) & \overset{\mathrm{def}}{=} \mathit{Right}(\mathit{inj}\,r_2\,c\,v) \\
\mathit{inj}\,(r_1 \cdot r_2)\,c\,\mathit{Seq}(v_1, v_2) & \overset{\mathrm{def}}{=} \mathit{Seq}(\mathit{inj}\,r_1\,c\,v_1, v_2) \\
\mathit{inj}\,(r_1 \cdot r_2)\,c\,\mathit{Left}(\mathit{Seq}(v_1, v_2)) & \overset{\mathrm{def}}{=} \mathit{Seq}(\mathit{inj}\,r_1\,c\,v_1, v_2) \\
\mathit{inj}\,(r_1 \cdot r_2)\,c\,\mathit{Right}(v) & \overset{\mathrm{def}}{=} \mathit{Seq}(\mathit{mkeps}(r_1), \mathit{inj}\,r_2\,c\,v) \\
\mathit{inj}\,(r^*)\,c\,\mathit{Seq}(v, vs) & \overset{\mathrm{def}}{=} \mathit{inj}\,r\,c\,v\ ::\ vs
\end{array}
$$

inj: 1st arg ↦ a rexp; 2nd arg ↦ a character; 3rd arg ↦ a value

Flatten

Obtaining the string underlying a value:

$$
\begin{array}{lcl}
|\mathit{Empty}| & \stackrel{\text{def}}{=} & [] \\
|\mathit{Char}(c)| & \stackrel{\text{def}}{=} & [c] \\
|\mathit{Left}(v)| & \stackrel{\text{def}}{=} & |v| \\
|\mathit{Right}(v)| & \stackrel{\text{def}}{=} & |v| \\
|\mathit{Seq}(v_1, v_2)| & \stackrel{\text{def}}{=} & |v_1| \,@\, |v_2| \\
|[v_1, \ldots, v_n]| & \stackrel{\text{def}}{=} & |v_1| \,@ \ldots @\, |v_n|
\end{array}
$$

r_1: $a \cdot (b \cdot c)$
r_2: $\epsilon \cdot (b \cdot c)$
r_3: $(\varnothing \cdot (b \cdot c)) + (\epsilon \cdot c)$
r_4: $(\varnothing \cdot (b \cdot c)) + ((\varnothing \cdot c) + \epsilon)$

$$\begin{tikzcd}
r_1 \arrow[r, "der\ a"] \arrow[d] & r_2 \arrow[r, "der\ b"] \arrow[d] & r_3 \arrow[r, "der\ c"] \arrow[d] & r_4\ nullable \arrow[d, "mkeps"] \\
v_1 & v_2 \arrow[l, "inj\ a"] & v_3 \arrow[l, "inj\ b"] & v_4 \arrow[l, "inj\ c"]
\end{tikzcd}$$

v_1: $Seq(Char(a), Seq(Char(b), Char(c)))$
v_2: $Seq(Empty, Seq(Char(b), Char(c)))$
v_3: $Right(Seq(Empty, Char(c)))$
v_4: $Right(Right(Empty))$

$|v_1|$: abc
$|v_2|$: bc
$|v_3|$: c
$|v_4|$: $[]$

Simplification

- If we simplify after the derivative, then we are builing the value for the simplified regular expression, but *not* for the original regular expression.

$$\begin{tikzcd}
r_1 \arrow[r, "der\,a"] \arrow[d] & r_2 \arrow[r, "der\,b"] \arrow[d] & r_3 \arrow[r, "der\,c"] \arrow[d] & r_4\ \mathit{nullable} \arrow[d, "mkeps"] \\
v_1 & v_2 \arrow[l, "inj\,a"] & v_3 \arrow[l, "inj\,b"] & v_4 \arrow[l, "inj\,c"]
\end{tikzcd}$$

$$(\varnothing \cdot (b \cdot c)) + ((\varnothing \cdot c) + \epsilon) \mapsto \epsilon$$

Rectification

```
$simp(r)$:
    case $r = r_1 + r_2$
        let $(r_{1s}, f_{1s}) = simp(r_1)$
            $(r_{2s}, f_{2s}) = simp(r_2)$
        case $r_{1s} = \varnothing$: return $(r_{2s}, \lambda v. Right(f_{2s}(v)))$
        case $r_{2s} = \varnothing$: return $(r_{1s}, \lambda v. Left(f_{1s}(v)))$
        case $r_{1s} = r_{2s}$: return $(r_{1s}, \lambda v. Left(f_{1s}(v)))$
        otherwise: return $(r_{1s} + r_{2s}, f_{alt}(f_{1s}, f_{2s}))$
```

```
$f_{alt}(f_1, f_2) \stackrel{\text{def}}{=}$
    $\lambda v.$ case $v = Left(v')$: return $Left(f_1(v'))$
           case $v = Right(v')$: return $Right(f_2(v'))$
```

Rectification

```
$simp(r)$:...
    case $r = r_1 \cdot r_2$
        let $(r_{1s}, f_{1s}) = simp(r_1)$
            $(r_{2s}, f_{2s}) = simp(r_2)$
        case $r_{1s} = \varnothing$: return $(\varnothing, f_{error})$
        case $r_{2s} = \varnothing$: return $(\varnothing, f_{error})$
        case $r_{1s} = \epsilon$: return $(r_{2s}, \lambda v.\, Seq(f_{1s}(Empty), f_{2s}(v)))$
        case $r_{2s} = \epsilon$: return $(r_{1s}, \lambda v.\, Seq(f_{1s}(v), f_{2s}(Empty)))$
        otherwise: return $(r_{1s} \cdot r_{2s}, f_{seq}(f_{1s}, f_{2s}))$
```

```
$f_{seq}(f_1, f_2) \overset{\text{def}}{=}$
    $\lambda v.$ case $v = Seq(v_1, v_2)$: return $Seq(f_1(v_1), f_2(v_2))$
```

Lexing with Simplification

$$\begin{array}{ll}
\textit{lex}\,r\,[\,] & \stackrel{\text{def}}{=} \text{if } \textit{nullable}(r) \text{ then } \textit{mkeps}(r) \text{ else } \textit{error} \\
\textit{lex}\,r\,c::s & \stackrel{\text{def}}{=} \text{let } (r', \textit{frect}) = \textit{simp}(\textit{der}(c, r)) \\
& \quad \textit{inj}\,r\,c\,(\textit{frect}(\textit{lex}(r', s)))
\end{array}$$

$$\begin{tikzcd}
r_1 \arrow[r, "der\ a"] \arrow[d] & r_2 \arrow[r, "der\ b"] \arrow[d] & r_3 \arrow[r, "der\ c"] \arrow[d] & r_4\ nullable \arrow[d, "mkeps"] \\
v_1 & v_2 \arrow[l, "inj\ a"'] & v_3 \arrow[l, "inj\ b"'] & v_4 \arrow[l, "inj\ c"']
\end{tikzcd}$$

Records

- new regex: $(x : r)$ new value: $Rec(x, v)$

Records

- new regex: $(x : r)$ new value: $Rec(x, v)$
- $nullable(x : r) \stackrel{\text{def}}{=} nullable(r)$
- $der\,c\,(x : r) \stackrel{\text{def}}{=} (x : der\,c\,r)$
- $mkeps(x : r) \stackrel{\text{def}}{=} Rec(x, mkeps(r))$
- $inj\,(x : r)\,c\,v \stackrel{\text{def}}{=} Rec(x, inj\,r\,c\,v)$

Records

- new regex: $(x : r)$ new value: $Rec(x, v)$
- $nullable(x : r) \stackrel{\text{def}}{=} nullable(r)$
- $der\,c\,(x : r) \stackrel{\text{def}}{=} (x : der\,c\,r)$
- $mkeps(x : r) \stackrel{\text{def}}{=} Rec(x, mkeps(r))$
- $inj\,(x : r)\,c\,v \stackrel{\text{def}}{=} Rec(x, inj\,r\,c\,v)$

for extracting subpatterns $(z : ((x : ab) + (y : ba))$

While Tokens

$$
\begin{array}{rl}
\mathtt{WHILE\_REGS} \overset{\text{def}}{=} & ((\text{”}\mathtt{k}\text{”} : \mathtt{KEYWORD}) + \\
 & \;(\text{”}\mathtt{i}\text{”} : \mathtt{ID}) + \\
 & \;(\text{”}\mathtt{o}\text{”} : \mathtt{OP}) + \\
 & \;(\text{”}\mathtt{n}\text{”} : \mathtt{NUM}) + \\
 & \;(\text{”}\mathtt{s}\text{”} : \mathtt{SEMI}) + \\
 & \;(\text{”}\mathtt{p}\text{”} : (\mathtt{LPAREN} + \mathtt{RPAREN})) + \\
 & \;(\text{”}\mathtt{b}\text{”} : (\mathtt{BEGIN} + \mathtt{END})) + \\
 & \;(\text{”}\mathtt{w}\text{”} : \mathtt{WHITESPACE}))^*
\end{array}
$$

```
”if true then then 42 else +”
```

```
KEYWORD(if),
WHITESPACE,
IDENT(true),
WHITESPACE,
KEYWORD(then),
WHITESPACE,
KEYWORD(then),
WHITESPACE,
NUM(42),
WHITESPACE,
KEYWORD(else),
WHITESPACE,
OP(+)
```

”if true then then 42 else +”

```
KEYWORD(if),
IDENT(true),
KEYWORD(then),
KEYWORD(then),
NUM(42),
KEYWORD(else),
OP(+)
```

There is one small problem with the tokenizer. How should we tokenize:

```
”x - 3”
```

```
OP:
  ”+”, ”-”
NUM:
  (NONZERODIGIT · DIGIT*) + ”0”
NUMBER:
  NUM + (”-” · NUM)
```

Two Rules

- Longest match rule (“maximal munch rule”): The longest initial substring matched by any regular expression is taken as next token.

- Rule priority: For a particular longest initial substring, the first regular expression that can match determines the token.

Environment

Obtaining the “recorded” parts of a regular expression:

$$
\begin{array}{lcl}
\mathit{env}(\mathit{Empty}) & \stackrel{\text{def}}{=} & [] \\
\mathit{env}(\mathit{Char}(c)) & \stackrel{\text{def}}{=} & [] \\
\mathit{env}(\mathit{Left}(v)) & \stackrel{\text{def}}{=} & \mathit{env}(v) \\
\mathit{env}(\mathit{Right}(v)) & \stackrel{\text{def}}{=} & \mathit{env}(v) \\
\mathit{env}(\mathit{Seq}(v_1, v_2)) & \stackrel{\text{def}}{=} & \mathit{env}(v_1) \mathbin{@} \mathit{env}(v_2) \\
\mathit{env}([v_1, \ldots, v_n]) & \stackrel{\text{def}}{=} & \mathit{env}(v_1) \mathbin{@} \ldots \mathbin{@} \mathit{env}(v_n) \\
\mathit{env}(\mathit{Rec}(x : v)) & \stackrel{\text{def}}{=} & (x : |v|) :: \mathit{env}(v)
\end{array}
$$

- Regular expression for email addresses

$$
\begin{aligned}
&(\text{name: } [a\text{-}z0\text{-}9\_\ .-]^{+})\cdot @\cdot\\
&\quad(\text{domain: } [a\text{-}z0\text{-}9\ .-]^{+})\cdot .\cdot\\
&\qquad(\text{top\_level: } [a\text{-}z\ .]^{\{2,6\}})
\end{aligned}
$$

`christian.urban@kcl.ac.uk`

- result environment:

$$
\begin{aligned}
&[(\textit{name} : \texttt{christian.urban}),\\
&\ (\textit{domain} : \texttt{kcl}),\\
&\ (\textit{top\_level} : \texttt{ac.uk})]
\end{aligned}
$$

Coursework

$$
\begin{array}{lcl}
\mathit{nullable}([c_1 c_2 \ldots c_n]) & \stackrel{\text{def}}{=} & ?\\
\mathit{nullable}(r^+) & \stackrel{\text{def}}{=} & ?\\
\mathit{nullable}(r^?) & \stackrel{\text{def}}{=} & ?\\
\mathit{nullable}(r^{\{n,m\}}) & \stackrel{\text{def}}{=} & ?\\
\mathit{nullable}(\sim r) & \stackrel{\text{def}}{=} & ?\\
\\
\mathit{der}\,c\,([c_1 c_2 \ldots c_n]) & \stackrel{\text{def}}{=} & ?\\
\mathit{der}\,c\,(r^+) & \stackrel{\text{def}}{=} & ?\\
\mathit{der}\,c\,(r^?) & \stackrel{\text{def}}{=} & ?\\
\mathit{der}\,c\,(r^{\{n,m\}}) & \stackrel{\text{def}}{=} & ?\\
\mathit{der}\,c\,(\sim r) & \stackrel{\text{def}}{=} & ?
\end{array}
$$