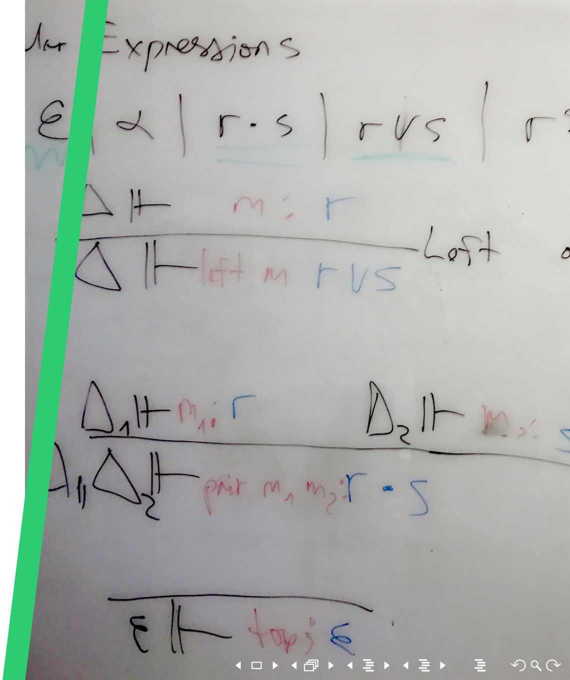
futurice

Regular expressions of types

Oleg Grenrus

2018-09-13 • SMALL FP CONF • HELSINKI



Me

- Oleg Grenrus, `@phadej`

Me

- Oleg Grenrus, @phadej
- I work at futurice

Me

- Oleg Grenrus, `@phadej`
- I work at futurice
- I co-organise the HaskHel-meetup

Me

- Oleg Grenrus, `@phadej`
- I work at futurice
- I co-organise the HaskHel-meetup
- I'm the "release manager" of `servant`

find - search for files

```
find [-H|-L] path... [operand_expression...]
```

find - search for files

```
find [-H|-L] path... [operand_expression...]
```

- is it just stringly typed?

find - search for files

```
find [-H|-L] path... [operand_expression...]
```

- is it just stringly typed?
- is it

$$\begin{aligned}(s/cat\ &{:}link\ (s/?\ (s/alt\ {:}H\ {:}H\ {:}L\ {:}L))\\ &{:}path\ (s/*\ string?)\\ &{:}expr\ \ operand\text{-}expression?)\end{aligned}$$

Using find

- Shell:

```
find -L src -type f
```

Using find

- Shell:

  ```
  find -L src -type f
  ```

- Clojure?

  $(\mathit{find}\ :L\ \texttt{"src"}\ :\mathit{type}\ :\mathit{file})$

Using find

- Shell:

```
find -L src -type f
```

- Clojure?

$$(\mathit{find}\ :L\ \texttt{"src"}\ :\mathit{type}\ :\mathit{file})$$

- Haskell!

$$(\!|\ \mathit{find\_}\ \#L, \texttt{"src"}, \#\mathit{type}, \#\mathit{file}\ |\!)$$

Types

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\Gamma \vdash x : A \qquad \Gamma \vdash y : B}{\vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

Let’s walk through the notation.

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\text{Premises}}{\text{Conclusion}}\ \text{Rulename}$$

We can conclude proposition below the line, if we have proof of propositions on top of the line

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma} \text{Pair}$$

Γ is a context, "what is in the scope", or "what we know".
For example $x : \mathsf{Int}, y : \mathsf{Bool}$ - there are x and y in scope.

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash} \text{Pair}$$

$\vdash$, turnstile, “entails”. $\Gamma \vdash \ldots$ is pronounced “In the context $\Gamma \ldots$”

futurice

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \qquad\qquad : \qquad\qquad} \text{Pair}$$

: colon, “has type”

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \color{red}{\mathsf{pair}\ x\ y}\ :}\ \text{Pair}$$

On the left hand side are expressions, I color them in red.

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \qquad\qquad : {\color{blue} A \times B}} \text{ Pair}$$

On the right hand side are types, I color them blue

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \mathsf{pair}\; x\; y : A \times B}\;\text{Pair}$$

On the left hand side are expressions, I color them in red. On the right hand side are types, I color them blue

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\overline{\qquad\qquad\qquad} \text{ Pair}$$

So…

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash} \text{Pair}$$

In (any) context Γ,

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \mathsf{pair}\ x\ y}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \mathsf{pair}\ x\ y :}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ **has type**

futurice

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ has type $A \times B$;

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ has type $A \times B$; **given that in the same context** Γ,

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ has type $A \times B$; given that in the same context Γ, x **has type** A

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\Gamma \vdash x : A \qquad \Gamma}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ has type $A \times B$; given that in the same context Γ, x has type A and

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\Gamma \vdash x : A \qquad \Gamma \vdash y : B}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ has type $A \times B$; given that in the same context Γ, x has type A and y **has type** B.

Lambda calculus

Example rule: introducing (making) a pair (or a product)

$$\frac{\Gamma \vdash x : A \qquad \Gamma \vdash y : B}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair}$$

In (any) context Γ, $\mathsf{pair}\ x\ y$ has type $A \times B$; given that in the same context Γ, x has type A and y has type B. $\square$

Lambda calculus: more rules

$$\frac{}{\Gamma, x : A \vdash x : A}\ \text{Var}$$

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash x : A}{\Gamma \vdash f\ x : B}\ \text{App} \qquad \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x \colon A \to e : A \to B}\ \text{Lam}$$

$$\frac{\Gamma \vdash x : A \qquad \Gamma \vdash y : B}{\Gamma \vdash \mathsf{pair}\ x\ y : A \times B}\ \text{Pair} \qquad \frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \mathsf{fst}\ p : A}\ \text{Fst} \qquad \frac{\Gamma \vdash p : A \times B}{\Gamma \vdash \mathsf{snd}\ p : B}\ \text{Snd}$$

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \mathsf{inl}\ x : A + B}\ \text{Left} \qquad \frac{\Gamma \vdash y : B}{\Gamma \vdash \mathsf{inr}\ y : A + B}\ \text{Right} \qquad \frac{\Gamma \vdash \cdots}{\Gamma \vdash \cdots}\ \text{Either}$$

futurice

Decidability of typed calculi

For the various type systems, we can ask three typical questions.

Decidability of typed calculi

1. Type-checking

$$\frac{?}{\Gamma \vdash x : \alpha}$$

Decidability of typed calculi

2. Typability, or type-inference

$$\frac{?}{\Gamma \vdash x : ?}$$

Decidability of typed calculi

3. Inhabitation, or proof search

$$\frac{?}{\Gamma \vdash {\color{red}?} : {\color{blue}\alpha}}$$

futurice

Inhabitation, an example

Really nice feature of the system above is that everything is decidable
In particular: inhabitation.

Inhabitation, an example

$$\frac{}{\vdash \; ? : (A \times B \to C) \to (A \to B \to C)}$$

Inhabitation, an example

$$\dfrac{\dfrac{}{f : A \times B \to C \vdash ? : A \to B \to C}}{\vdash \lambda f \to ? : (A \times B \to C) \to (A \to B \to C)}\ \text{Lam}$$

futurice

Inhabitation, an example

$$
\dfrac{\dfrac{\dfrac{}{\Gamma \vdash ? : C}}{f : A \times B \to C \vdash \lambda x\, y \to ? : A \to B \to C}\,\mathsf{Lam}^2}{\vdash \lambda f\, x\, y \to ? : (A \times B \to C) \to (A \to B \to C)}\,\mathsf{Lam}
$$

$$
\Gamma = f : A \times B \to C, x : A, y : B
$$

Inhabitation, an example

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : A \times B \to C}\,\text{Var} \qquad \dfrac{}{\Gamma \vdash \,? : A \times B}}{\Gamma \vdash f\,? : C}\,\text{App}}{f : A \times B \to C \vdash \lambda x\, y \to f\,? : A \to B \to C}\,\text{Lam}^2}{\vdash \lambda f\, x\, y \to f\,? : (A \times B \to C) \to (A \to B \to C)}\,\text{Lam}
$$

$$
\Gamma = f : A \times B \to C, x : A, y : B
$$

Inhabitation, an example

$$
\dfrac{
  \dfrac{}{\Gamma \vdash f : A \times B \to C}\,\text{Var}
  \qquad
  \dfrac{\dfrac{}{\Gamma \vdash ? : A} \qquad \dfrac{}{\Gamma \vdash ? : B}}{\Gamma \vdash \mathsf{pair}\,?\,? : A \times B}\,\text{Pair}
}{
  \dfrac{\dfrac{\Gamma \vdash f\,(\mathsf{pair}\,?\,?) : C}{f : A \times B \to C \vdash \lambda x\,y \to f\,(\mathsf{pair}\,?\,?) : A \to B \to C}\,\text{Lam}^2}{\vdash \lambda f\,x\,y \to f\,(\mathsf{pair}\,?\,?) : (A \times B \to C) \to (A \to B \to C)}\,\text{Lam}
}\,\text{App}
$$

$$\Gamma = f : A \times B \to C, x : A, y : B$$

Inhabitation, an example

$$
\dfrac{\dfrac{}{\Gamma \vdash f : A \times B \to C}\,\text{Var} \qquad \dfrac{\dfrac{}{\Gamma \vdash x : A}\,\text{Var} \qquad \dfrac{}{\Gamma \vdash ? : B}}{\Gamma \vdash \mathsf{pair}\, x\, ? : A \times B}\,\text{Pair}}{\dfrac{\Gamma \vdash f\,(\mathsf{pair}\, x\, ?) : C}{\dfrac{f : A \times B \to C \vdash \lambda x\, y \to f\,(\mathsf{pair}\, x\, ?) : A \to B \to C}{\vdash \lambda f\, x\, y \to f\,(\mathsf{pair}\, x\, ?) : (A \times B \to C) \to (A \to B \to C)}\,\text{Lam}}\,\text{Lam}^2}\,\text{App}
$$

$$\Gamma = f : A \times B \to C, x : A, y : B$$

Inhabitation, an example

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : A \times B \to C}\,\text{Var} \qquad \dfrac{\dfrac{}{\Gamma \vdash x : A}\,\text{Var} \qquad \dfrac{}{\Gamma \vdash y : B}\,\text{Var}}{\Gamma \vdash \mathsf{pair}\,x\,y : A \times B}\,\text{Pair}}{\Gamma \vdash f\,(\mathsf{pair}\,x\,y) : C}\,\text{App}}{f : A \times B \to C \vdash \lambda x\,y \to f\,(\mathsf{pair}\,x\,y) : A \to B \to C}\,\text{Lam}^2}{\vdash \lambda f\,x\,y \to f\,(\mathsf{pair}\,x\,y) : (A \times B \to C) \to (A \to B \to C)}\,\text{Lam}$$

$$\Gamma = f : A \times B \to C, x : A, y : B$$

Regular expressions

Stand back…

Not the Perl or POSIX regular expressions , but “real” regular expressions:

Stand back…

Not the Perl or POSIX regular expressions , but “real” regular expressions:

Stand back…

Not the Perl or POSIX regular expressions , but “real” regular expressions:

regular expression	r	$::=$	ε	empty string
		$\mid$	a	“character”, $a \in \Sigma$
		$\mid$	$r_1 \diamond r_2$	concatenation
		$\mid$	$r_1 \vee r_2$	alternation
		$\mid$	$r^\star$	Kleene closure

futurice

Matching

We formalise the what-matches -intuition using $\Delta \Vdash r$ relation, which is read as “regular expression r matches string Δ”.

For example we can ask,

$$[x, y, z] \Vdash (x \vee y)^{\star} \diamond z$$

?

Matching

We formalise the what-matches -intuition using $\Delta \Vdash r$ relation, which is read as "regular expression r matches string Δ".

$$\frac{}{\Vdash r_1 \diamond r_2}\ \text{Concat}$$

Matching

We formalise the what-matches -intuition using $\Delta \Vdash r$ relation, which is read as “regular expression r matches string Δ”.

$$\frac{\Vdash r_1}{\Vdash r_1 \diamond r_2}\ \text{Concat}$$

Matching

We formalise the what-matches -intuition using $\Delta \Vdash r$ relation, which is read as "regular expression r matches string Δ".

$$\frac{\Delta_1 \Vdash r_1}{\Vdash r_1 \diamond r_2}\ \text{Concat}$$

Matching

We formalise the what-matches -intuition using $\Delta \Vdash r$ relation, which is read as "regular expression r matches string Δ".

$$\frac{\Delta_1 \Vdash r_1 \qquad \Delta_2 \Vdash r_2}{\Vdash r_1 \diamond r_2}\ \text{Concat}$$

Matching

We formalise the what-matches -intuition using $\Delta \Vdash r$ relation, which is read as “regular expression r matches string Δ”.

$$\frac{\Delta_1 \Vdash r_1 \qquad \Delta_2 \Vdash r_2}{\Delta_1 \mathbin{+\!\!+} \Delta_2 \Vdash r_1 \diamond r_2}\ \text{Concat}$$

And there are more rules

$$\frac{a \in \Sigma}{[a] \Vdash a}\ \text{Atom} \qquad \frac{}{[] \Vdash \varepsilon}\ \text{Eps}$$

$$\frac{\Delta \Vdash r_1}{\Delta \Vdash r_1 \vee r_2}\ \text{Left} \qquad \frac{\Delta \Vdash r_2}{\Delta \Vdash r_1 \vee r_2}\ \text{Right}$$

$$\frac{}{[] \Vdash r^\star}\ \text{Nil} \qquad \frac{\Delta_1 \Vdash r \qquad \Delta_2 \Vdash r^\star}{\Delta_1 +\!\!+ \Delta_2 \Vdash r^\star}\ \text{Cons}$$

A match

$$
\dfrac{
  \dfrac{
    \dfrac{\dfrac{}{[x] \Vdash x}\,\textsf{Atom}}{[x] \Vdash x \vee y}\,\textsf{Left}
    \qquad
    \dfrac{
      \dfrac{\dfrac{}{[y] \Vdash y}\,\textsf{Atom}}{[y] \Vdash x \vee y}\,\textsf{Right}
      \qquad
      \dfrac{}{[] \Vdash (x \vee y)^{\star}}\,\textsf{Nil}
    }{[y] \Vdash (x \vee y)^{\star}}\,\textsf{Cons}
  }{[x, y] \Vdash (x \vee y)^{\star}}\,\textsf{Cons}
  \qquad
  \dfrac{}{[z] \Vdash z}\,\textsf{Atom}
}{[x, y, z] \Vdash (x \vee y)^{\star} \diamond z}\,\textsf{Concat}
$$

REList

$$\frac{\Delta_1 \Vdash r_1 \qquad \Delta_2 \Vdash r_2}{\Delta_1 + \!\!\!+ \, \Delta_2 \Vdash r_1 \diamond r_2}\ \text{Concat} \qquad \frac{\Gamma \vdash x : \mathsf{Int} \qquad \Gamma \vdash y : \mathsf{Bool}}{\Gamma \vdash \mathsf{pair}\ x\ y : \mathsf{Int} \times \mathsf{Bool}}$$

REList

$$\frac{\Delta_1 \Vdash r_1 \qquad \Delta_2 \Vdash r_2}{\Delta_1 \mathbin{+\!\!+} \Delta_2 \Vdash r_1 \diamond r_2}\ \textsf{Concat} \qquad \frac{\Gamma \vdash x : \textsf{Int} \qquad \Gamma \vdash y : \textsf{Bool}}{\Gamma \vdash \textsf{pair}\ x\ y : \textsf{Int} \times \textsf{Bool}}$$

$$\frac{\Gamma \vdash x : \textsf{REList}\ r_1 \qquad \Gamma \vdash y : \textsf{REList}\ r_2}{\Gamma \vdash \textsf{pair}\ x\ y : \textsf{REList}\ (r_1 \diamond r_2)}$$

RElist

$$\frac{\Delta_1 \Vdash r_1 \qquad \Delta_2 \Vdash r_2}{\Delta_1 \mathbin{+\!\!+} \Delta_2 \Vdash r_1 \diamond r_2}\ \text{Concat} \qquad \frac{\Gamma \vdash x : \mathsf{Int} \qquad \Gamma \vdash y : \mathsf{Bool}}{\Gamma \vdash \mathsf{pair}\ x\ y : \mathsf{Int} \times \mathsf{Bool}}$$

$$\frac{\Gamma \vdash x : \mathsf{REList}\ r_1 \qquad \Gamma \vdash y : \mathsf{REList}\ r_2}{\Gamma \vdash \mathsf{pair}\ x\ y : \mathsf{REList}\ (r_1 \diamond r_2)}$$

You only need REList ! No pair, sums, ordinary lists . . .

What's the point?

$$\mathsf{pair}\ (\mathsf{left}\ 1)\ (\mathsf{cons}\ \texttt{'a'}\ (\mathsf{cons}\ \texttt{'b'}\ \mathsf{nil})) : \mathsf{REList}\ ((\mathsf{Int} \vee \mathsf{Bool}) \diamond \mathsf{Char}^{\star})$$

What’s the point?

$$\mathsf{pair}\ (\mathsf{left}\ 1)\ (\mathsf{cons}\ \texttt{'a'}\ (\mathsf{cons}\ \texttt{'b'}\ \mathsf{nil})) : \mathsf{REList}\ ((\mathsf{Int} \vee \mathsf{Bool}) \diamond \mathsf{Char}^{\star})$$

Nicer expression syntax!

$$[\![1, \texttt{'a'}, \texttt{'b'}]\!] : \mathsf{REList}\ ((\mathsf{Int} \vee \mathsf{Bool}) \diamond \mathsf{Char}^{\star})$$

Does it work?

Does it work?

Yes!

Does it work?

Yes!

$\Delta \Vdash r$ is **decidable**

Does it work?

Yes!

$\Delta \Vdash r$ is **decidable**

And I have made a type-checker GHC plugin making these decisions!

find - again

```
find [-H|-L] path... [operand_expression...]
```

$$(\text{-H} \lor \text{-L} \lor \varepsilon) \diamond \text{String}^{\star} \diamond \mathit{operand\text{-}expression}$$

$$\begin{aligned}(s/cat\ &:link\ (s/?\ (s/alt\ :H\ :H\ :L\ :L))\\ &:path\ (s/*\ string?)\\ &:expr\ operand\text{-}expression?)\end{aligned}$$

Lets put regular expressions into types!

$$(\text{-H} \bigvee \text{-L} \bigvee \varepsilon) \diamond \text{String}^{\star} \diamond \textit{operand-expression}$$

futurice

Lets put regular expressions into types!

$$(\text{-H} \vee \text{-L} \vee \varepsilon) \diamond \text{String}^{\star} \diamond \textit{operand-expression}$$

$$\begin{aligned}
&\texttt{type}\ \text{FIND} \\
&\quad = (\text{E} \vee \text{Flag "H"} \vee \text{Flag "L"}) \\
&\quad \diamond\ S\ \text{String} \\
&\quad \diamond\ \text{OPERAND_EXPRESSION}
\end{aligned}$$

futurice

Lets put regular expressions into types!

$$(\text{-H} \vee \text{-L} \vee \varepsilon) \diamond \text{String}^{\star} \diamond \textit{operand-expression}$$

```
type FIND
  = (E ∨ Flag "H" ∨ Flag "L")
  ◇ S String
  ◇ OPERAND_EXPRESSION
```

```
find_ :: REList FIND → IO ()
```

futurice

DEMO

futurice

Conclusion

- This all is cool

- This all is cool
- Which one is “better”?

$$find\_ :: \mathsf{REList\ FIND} \rightarrow \mathsf{IO}\ ()$$

-- or

$$find\_ :: \mathsf{Maybe\ FollowLinks} \rightarrow [\mathsf{Path}] \rightarrow \mathsf{Expr} \rightarrow \mathsf{IO}\ ()$$

- This all is cool
- Which one is "better"?

$$find\_ :: \mathsf{RElist\ FIND} \rightarrow \mathsf{IO}\ ()$$

-- or

$$find\_ :: \mathsf{Maybe\ FollowLinks} \rightarrow [\mathsf{Path}] \rightarrow \mathsf{Expr} \rightarrow \mathsf{IO}\ ()$$

- Missing bit is implementing the rule

$$\frac{\Delta \Vdash r_1 \qquad r_1 \leq r_2}{\Delta \Vdash r_2}\ \text{Sub}$$

so we could use relist syntax but have Haskell types e.g.
$[\![1, \text{'a'}, \text{'b'}]\!] : (\mathsf{Either\ Int\ Bool}, [\mathsf{Char}])$

futurice

Thank you!
Questions?

Oleg Grenrus

@phadej

oleg.fi/gists/

github.com/phadej/kleene-type