QuB

A Resource Aware Functional Programming Language

Apoorv Ingle

The University of Kansas

Table of Contents

Hard problems in programming

Naming variables

Introduction and Motivation

Hard problems in programming

Resource management

in evolving production code

Resources: Files, database connections, entity with a shared state

Resource Management: File Handling

- Modified File Handling API in Haskell

$$\begin{aligned}
&\texttt{openFile} :: \texttt{FilePath} \to \texttt{IO FileHandle}\\
&\texttt{closeFile} :: \texttt{FileHandle} \to \texttt{IO}\ ()\\
&\texttt{readLine} :: \texttt{FileHandle} \to \texttt{IO}\ (\texttt{String}, \texttt{FileHandle})\\
&\texttt{writeFile} :: \texttt{String} \to \texttt{FileHandle}\\
&\qquad\qquad \to \texttt{IO}\ ((), \texttt{FileHandle})\\
&\texttt{upper} :: \texttt{String} \to \texttt{String}
\end{aligned}$$

Resource Management: File Handling

- File Handling in Haskell

```
do f      ← openFile "sample.txt"
   (s, f)   ← readLine f
   let c = upper s
   ((), f) ← writeLine f c
         .
         .
         .
   ()    ← closeFile f
```

Resource Management: File Handling

- File Handling in Haskell Gone Wrong (Part I)

```
do f      ← openFile "sample.txt"
   (s, f)  ← readLine f
   let c = upper s
   ((), f) ← writeLine f c
        .
        .
        .
   ()  ← closeFile f
        .
        .
        .
   ()  ← closeFile f
   return c
```

Resource Management: File Handling

- File Handling in Haskell Gone Wrong (Part I)

```
do f  ← openFile "sample.txt"
   (s, f)  ← readLine f
   let c = upper s
   ((), f) ← writeLine f c
         .
         .
         .
  ()  ← closeFile f
         .
         .
         .
  ()  ← closeFile f
   return c
```

- File is closed twice: Run time crash

Resource Management: File Handling

- File Handling in Haskell Gone Wrong (Part II)

```
do f  ← openFile "sample.txt"
   (s, f)  ← readLine f
   let c = upper s
   ((), f) ← writeLine f c
       .
       .
       .
   return c
```

Resource Management: File Handling

- File Handling in Haskell Gone Wrong (Part II)

```
do f  ← openFile "sample.txt"
   (s, f)  ← readLine f
   let c = upper s
   ((), f) ← writeLine f c
       .
       .
       .
   return c /*File not closed!!*/
```

- File not closed: Memory leak

Resource Management: Exception Handling

- `MonadError`[4] in Haskell

```
class Monad m ⇒ MonadError e m | m → e where
    throwError :: e → m a
    catchError :: m a → (e → m a) → m a
```

- `throwError` starts exception processing
- `catchError` exception handler

Resource Management: Exception Handling

- Using `MonadError` in Haskell

```
do f ← openFile "sample.txt"
   ((s, f)  ← readLine f
   let c = upper s
   ()  ← closeFile f
   return $ Right c)
       `catchError` (\_ →
               return $ Left "Error in reading file")
```

- Exception may cause memory leak

Well typed programs do not go wrong.

— R. Milner

Well typed programs do not go wrong.

— R. Milner

~~Lights~~Types will guide you home

— Coldplay

Contributions

- Design and implement QuB type system
  - Resources as first class citizens
  - Program objects are restricted or unrestricted
  - Functions that share resources with their arguments or are separate.
- Formalizing and proving important properties of QuB
- QuB is logic of ***BI*** with steroids
  - Environments as graphs
- Working examples

Background Work: Simply Typed Lambda Calculus (STLC)

$$\lambda x.M \begin{cases} \text{Abstract over computation} \\ \text{Define functions} \end{cases}$$

$$MN \begin{cases} \text{Do the computation} \\ \text{Use functions} \end{cases}$$

Background Work: Simply Typed Lambda Calculus (STLC)

$$\lambda x.M \begin{cases} \text{Abstract over computation} \\ \text{Define functions} \end{cases}$$

$$MN \begin{cases} \text{Do the computation} \\ \text{Use functions} \end{cases}$$

$$\frac{\Gamma_x, x : \tau \vdash M : \tau'}{\Gamma \vdash \lambda x.M : \tau \to \tau'}\,[\to \mathrm{I}] \qquad \frac{\Gamma \vdash M : \tau \to \tau' \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \tau'}\,[\to \mathrm{E}]$$

Background Work: Simply Typed Lambda Calculus (STLC)

$$\lambda x.M \begin{cases} \text{Abstract over computation} \\ \text{Define functions} \end{cases}$$

$$MN \begin{cases} \text{Do the computation} \\ \text{Use functions} \end{cases}$$

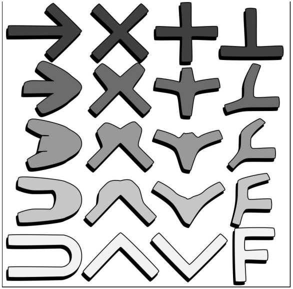
$$\frac{\Gamma_x, x : \tau \vdash M : \tau'}{\Gamma \vdash \lambda x.M : \tau \to \tau'} [\to \mathrm{I}] \qquad \frac{\Gamma \vdash M : \tau \to \tau' \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \tau'} [\to \mathrm{E}]$$

Hinley-Milner (***HM***) type system ensures sane programs

Background Work: Curry-Howard Correspondence

- Types are Propostions
- Programs are Proofs

HM type system ≡ Second Order Intuitionistic Propositional Logic



LC'90 The Curry-Howard homeomorphism

Source: `http://lucacardelli.name/Artifacts/Drawings/CurryHoward/CurryHoward.pdf`

Background Work: Second Order Intuitionistic Propositional Logic

$\boxed{\text{Language}}$

$$\text{Propostions \& connectives} \quad A, B, C ::= x \mid A \supset B \mid \forall x.B \mid \ldots$$
$$\text{Context} \quad \Gamma, \Delta ::= \epsilon \mid \Gamma, A$$

$\boxed{\text{Logic Rules}}$

$$\frac{}{A \vdash A}\,[\mathrm{Ax}]$$

$$\frac{\Gamma \vdash B \qquad x \notin \Gamma}{\forall x.B}\,[\forall \mathrm{I}] \qquad \frac{\Gamma \vdash \forall x.B \qquad \Gamma \vdash A}{B[x/A]}\,[\forall \mathrm{E}]$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\,[\supset \mathrm{I}] \qquad \frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\,[\supset \mathrm{E}]$$

Background Work: Second Order Intuitionistic Propositional Logic

Propostions are truth values not resources

$\boxed{\text{Language}}$

$$\text{Propostions \& connectives} \quad A, B, C ::= x \mid A \supset B \mid \forall x.B \mid \ldots$$
$$\text{Context} \quad \Gamma, \Delta ::= \epsilon \mid \Gamma, A$$

$\boxed{\text{Logic Rules}}$

$$\frac{}{A \vdash A}\,[\mathrm{Ax}]$$

$$\frac{\Gamma \vdash B \qquad x \notin \Gamma}{\forall x.B}\,[\forall\mathrm{I}] \qquad \frac{\Gamma \vdash \forall x.B \qquad \Gamma \vdash A}{B[x/A]}\,[\forall\mathrm{E}]$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B}\,[\supset\mathrm{I}] \qquad \frac{\Gamma \vdash A \supset B \qquad \Gamma \vdash A}{\Gamma \vdash B}\,[\supset\mathrm{E}]$$

Background Work: Substructural Logic

- Structural rules implicit in intuistionistic propositional logics

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\,[\text{WKN}] \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\,[\text{CTR}] \qquad \frac{\Gamma, \Delta \vdash B}{\Delta, \Gamma \vdash B}\,[\text{EXCH}]$$

Background Work: Substructural Logic

- Structural rules implicit in intuistionistic propositional logics

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ [\text{WKN}] \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ [\text{CTR}] \qquad \frac{\Gamma, \Delta \vdash B}{\Delta, \Gamma \vdash B}\ [\text{EXCH}]$$

- Control the use of [WKN] and [CTR]

Propostions now behave like resources

Background Work: Substructural Logic

System	Who	Restrictions
Linear Logic[1]	Girard	[WKN] [CTRN]
Lambek Logic[3]	Lambek	[EXCH]
Logic of Bunched Implications[6]	O'Hearn and Pym	[WKN] [CTRN]
⋮	⋮	⋮

Background Work: Qualified Types

$$P \mid \Gamma \vdash M : \sigma$$

"Type of M is σ
when predicates in P are satisfied
and Γ specifies the free variables in M"[2]

General framework, incorporates predicates into type language

Background Work: Quill

Quill[5]: Qualified types + linear logic

Predicates:

- `Un` τ If τ does not have resources or can be copied or dropped easily.
- `Fun` τ If τ is a function type
- $\tau \geq \tau'$ If τ less restricting than τ'

Background Work: Quill

Quill[5]: Qualified types + linear logic

Qualifying Types:

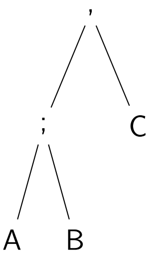
- Unrestricted Types: `Un Int`, `Un` Bool
- Restricted or Linear Types: FileHandle
- Function Types: `Fun` (`Int` → `Int`), `Fun` (`String` → `String`)

Background Work: Logic of Bunched Implications (*BI*)

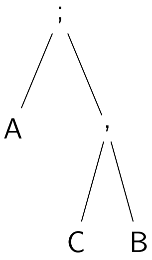
- Contexts are lists or sets

$$\Gamma, A, B$$

- In logic of ***BI***, contexts are trees and called are bunches
- Two connective used to combine bunches: $A; B$ or A, B



$\Gamma = (A; B), C$



$\Gamma = A; (B, C)$

Background Work: Logic of *BI*

Controling structural rules based on context

- Contraction

$$\begin{array}{cc} A; B \vdash A & A; B \vdash B \\ A, B \nvdash A & A, B \nvdash B \end{array}$$

- Weakening

$$\begin{array}{cc} A; B \vdash A; B; B & A; B \vdash A; A; B \\ A, B \nvdash A, B, B & A, B \nvdash A, A, B \end{array}$$

Interpretation:

- Propostions connected with , are separate
- Propostions connected with ; are in sharing

Background Work: Logic of *BI*

(Absense of) Structural rules and logical connecitves:

- Meaning of conjunction

$$A, B \vdash A \otimes B \qquad\qquad A; B \vdash A \,\&\, B$$

- Meaning of implication

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \mathrel{-\!\!*} B}\,[\mathrel{-\!\!*}\mathrm{I}] \qquad\qquad \frac{\Gamma; A \vdash B}{\Gamma \vdash A \twoheadrightarrow B}\,[\twoheadrightarrow\mathrm{I}]$$

Coffee Shop

1 cup coffee costs \$2

$\otimes$

$\vdash$

Background work: Logic of ***BI***

Coffee Shop

1 cup coffee costs \$2

$\otimes$

$\vdash$

$\&$

$\nvdash$

QuB

- Quill: Qualified types + linear logic
- QuB: Qualified types + logic of bunched implications

QuB: Types and Predicates

$$
\begin{array}{rl}
\text{Types} & \tau, \upsilon, \phi ::= t \mid \iota \mid \tau \to \tau \\
\text{where} & \to \in \{ \overset{!}{\multimap}, \multimap, \overset{!}{\twoheadrightarrow}, \twoheadrightarrow \} \\
\text{Predicates} & \pi, \omega ::= \mathtt{Un}\ \tau \mid \mathtt{SeFun}\ \tau \mid \mathtt{ShFun}\ \tau \mid \tau \geq \tau' \\
\text{Qualified Types} & \rho ::= \tau \mid \pi \Rightarrow \rho \\
\text{Type schemes} & \sigma ::= \rho \mid \forall t.\sigma
\end{array}
$$

- $\mathtt{SeFun}\ \tau$: τ is a function that is separate from its argument
- $\mathtt{ShFun}\ \tau$: τ is a function that is in sharing with its argument
- $\mathtt{Un}\ \tau$: τ does not have resources or they can be copied/dropped easily

$$
\begin{aligned}
\text{Types} \quad & \tau, \upsilon, \phi ::= t \mid \iota \mid \tau \to \tau \\
& \quad \text{where} \qquad \to \in \{\overset{!}{\mathrel{-\!\!*}}, \mathrel{-\!\!*}, \overset{!}{\twoheadrightarrow}, \twoheadrightarrow\} \\
\text{Predicates} \quad & \pi, \omega ::= \texttt{Un}\ \tau \mid \texttt{SeFun}\ \tau \mid \texttt{ShFun}\ \tau \mid \tau \geq \tau' \\
\text{Qualified Types} \quad & \rho ::= \tau \mid \pi \Rightarrow \rho \\
\text{Type schemes} \quad & \sigma ::= \rho \mid \forall t.\sigma
\end{aligned}
$$

- $\mathrel{-\!\!*}$: Function type that is separate with its argument
- $\twoheadrightarrow$: Function type that is in sharing with its argument
- $\overset{!}{\mathrel{-\!\!*}}$, $\overset{!}{\twoheadrightarrow}$: unrestriced versions of $\mathrel{-\!\!*}$ and $\twoheadrightarrow$

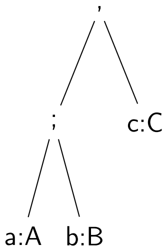
QuB: Language

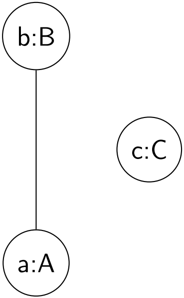
Term Variables	$x, y, z \in \text{Var}$
Expressions	$M, N ::= x \mid \lambda^{\multimap\!\!\ast} x.M \mid \lambda^{\twoheadrightarrow} x.M \mid MN \mid \texttt{let}\ x = M\ \texttt{in}\ N$

- $\lambda^{\multimap\!\!\ast} x.M$: Introduces $\multimap\!\!\ast$ function
- $\lambda^{\twoheadrightarrow} x.N$: Introduces $\twoheadrightarrow$ function

QuB: Typing Environmnet

- Logic of ***BI***: Contexts are trees
- QuB: Contexts are flattened sharing graphs





- Sharing relation Ψ

$$\forall_{x \in \mathtt{dom}(\Psi)}\ x \in \Psi(x) \qquad \text{(reflexive)}$$

$$\forall_{x,y \in \mathtt{dom}(\Psi)}\ \text{if } y \in \Psi(x) \text{ then } x \in \Psi(y) \qquad \text{(symmetric)}$$

$$\forall_{x,y,z \in \mathtt{dom}(\Psi)}\ \text{if } y \in \Psi(x) \text{ and } z \in \Psi(y)\ \not\Rightarrow z \in \Psi(x) \qquad \text{(non-transitive)}$$

QuB: Typing Environment

"x of type σ is in sharing with $\vec{y}$"

$$(x, \sigma, \vec{y}) \in \Gamma$$

Typing Context $\quad \Gamma, \Delta ::= \epsilon \mid \Gamma, x^{\vec{y}} : \sigma$

$$\begin{aligned}
\texttt{Vars}(\Gamma, x^{\vec{y}} : \tau) &= \texttt{Vars}(\Gamma) \cup \{x\} \\
\texttt{Shared}(\Gamma, x^{\vec{y}} : \tau) &= \texttt{Shared}(\Gamma) \cup \{\vec{y}\} \\
\texttt{Used}(\Gamma) &= \texttt{Vars}(\Gamma) \cup \texttt{Shared}(\Gamma)
\end{aligned}$$

$$(\Gamma, x^{\vec{y}} : \tau)^{[a \mapsto \vec{b}]} = \begin{cases} a \notin \vec{y} & (\Gamma^{[a \mapsto \vec{b}]}, x^{\vec{y}} : \tau) \\ a \in \vec{y} & (\Gamma^{[a \mapsto \vec{b}]}, x^{(\vec{y} \setminus a) \cup \vec{b}} : \tau) \end{cases}$$

$$\Gamma^{[\vec{a} \mapsto \vec{b}]} = (\ldots((\Gamma^{[a_1 \mapsto \vec{b}]})^{[a_2 \mapsto \vec{b}]})^{\cdots})^{[a_n \mapsto \vec{b}]}$$

$$\begin{aligned}
\Gamma \circledast \Gamma' &= \Gamma \sqcup \Gamma' && \texttt{if } \texttt{Vars}(\Gamma) \# \texttt{Used}(\Gamma') \wedge \texttt{Vars}(\Gamma') \# \texttt{Used}(\Gamma) \\
\Gamma \oplus \Gamma' &= \Gamma \sqcup \Gamma' && \texttt{if } \texttt{Used}(\Gamma) = \texttt{Used}(\Gamma')
\end{aligned}$$

QuB: Typing Rules

$\boxed{\text{Structural Rules}}$

$$\frac{}{P \mid x^{\vec{y}} : \sigma \vdash x : \sigma} \text{[ID]}$$

$$\frac{P \mid \Gamma \circledast \Delta \circledast \Delta \vdash M : \sigma \qquad P \vdash \Delta \ \mathbf{un}}{P \mid \Gamma \circledast \Delta \vdash M : \sigma} \text{[CTR-UN]} \qquad \frac{P \mid \Gamma \oplus \Delta \oplus \Delta \vdash M : \sigma}{P \mid \Gamma \oplus \Delta \vdash M : \sigma} \text{[CTR-SH]}$$

$$\frac{P \mid \Gamma \vdash M : \sigma \qquad P \vdash \Delta \ \mathbf{un}}{P \mid \Gamma \circledast \Delta \vdash M : \sigma} \text{[WKN-UN]} \qquad \frac{P \mid \Gamma \vdash M : \sigma}{P \mid \Gamma \oplus \Delta \vdash M : \sigma} \text{[WKN-SH]}$$

QuB: Typing Rules

Connective Rules

$$\frac{P \mid \Gamma \vdash M : \sigma \qquad P' \mid \Gamma'_x, x : \sigma \vdash N : \tau}{P \cup P' \mid \Gamma \sqcup \Gamma' \vdash (\texttt{let}\ x = M\ \texttt{in}\ N) : \tau}\ [\text{LET}]$$

$$\frac{P \mid \Gamma \vdash M : \sigma \qquad t \notin \texttt{fvs}(\Gamma) \cup \texttt{fvs}(P)}{P \mid \Gamma \vdash M : \forall t.\sigma}\ [\forall\ \text{I}] \qquad \frac{P \mid \Gamma \vdash M : \forall t.\sigma}{P \mid \Gamma \vdash M : [\tau \backslash t]\sigma}\ [\forall\ \text{E}]$$

$$\frac{P, \pi \mid \Gamma \vdash M : \rho}{P \mid \Gamma \vdash M : \pi \Rightarrow \rho}\ [\Rightarrow\ \text{I}] \qquad \frac{P \mid \Gamma \vdash M : \pi \Rightarrow \rho \qquad P \vdash \pi}{P \mid \Gamma \vdash M : \rho}\ [\Rightarrow\ \text{E}]$$

$$\frac{\begin{array}{c} P \Rightarrow \texttt{ShFun}\ \phi \qquad P \vdash \Gamma \geq \phi \\ P \mid \Gamma^{[\varnothing \mapsto \{x\}]}, x^{\mathsf{Vars}(\Gamma)} : \tau \vdash M : \tau' \end{array}}{P \mid \Gamma \vdash \lambda^{\twoheadrightarrow} x.M : \phi\tau\tau'}\ [\twoheadrightarrow\ \text{I}] \qquad \frac{\begin{array}{c} P \Rightarrow \texttt{ShFun}\ \phi \\ P \mid \Gamma \vdash M : \phi\tau\tau' \qquad P \mid \Gamma' \vdash N : \tau \end{array}}{P \mid \Gamma \oplus \Gamma' \vdash MN : \tau'}\ [\twoheadrightarrow\ \text{E}]$$

$$\frac{\begin{array}{c} P \Rightarrow \texttt{SeFun}\ \phi \qquad P \vdash \Gamma \geq \phi \\ P \mid \Gamma, x^{\varnothing} : \tau \vdash M : \tau' \end{array}}{P \mid \Gamma \vdash \lambda^{-\!\!*} x.M : \phi\tau\tau'}\ [-\!\!*\ \text{I}] \qquad \frac{\begin{array}{c} P \Rightarrow \texttt{SeFun}\ \phi \\ P \mid \Gamma \vdash M : \phi\tau\tau' \qquad P \mid \Gamma' \vdash N : \tau \end{array}}{P \mid \Gamma \circledast \Gamma' \vdash MN : \tau'}\ [-\!\!*\ \text{E}]$$

QuB: Typing Terms

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\varnothing \mid y^{\varnothing} : \tau' \vdash y : \tau'}\,[\mathrm{ID}]
\qquad
\dfrac{
\dfrac{}{\varnothing \mid x^{\varnothing} : \tau \vdash x : \tau}\,[\mathrm{ID}]
\qquad
\dfrac{}{\varnothing \mid f^{\varnothing} : \tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon \vdash f : \tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon}\,[\mathrm{ID}]
}{\varnothing \mid x^{\varnothing} : \tau \circledast f^{\varnothing} : \tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon \vdash fx : (\tau' \mathrel{-\!\!*} \upsilon)}\,[\mathrel{-\!\!*}\mathrm{E}]
}{\varnothing \mid y^{\varnothing} : \tau' \circledast x^{\varnothing} : \tau \circledast f^{\varnothing} : \tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon \vdash fxy : \upsilon}\,[\mathrel{-\!\!*}\mathrm{E}]
}{\varnothing \mid x^{\varnothing} : \tau \circledast y^{\varnothing} : \tau' \circledast f^{\varnothing} : \tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon \vdash fxy : \upsilon}\,[\mathit{EXCH}]
}{\varnothing \mid x^{\varnothing} : \tau \circledast y^{\varnothing} : \tau' \vdash \lambda^{-\!*} f.fxy : (\tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon) \mathrel{-\!\!*} \upsilon}\,[\mathrel{-\!\!*}\ \mathrm{I}]
}{\varnothing \mid x^{\varnothing} : \tau \vdash \lambda^{-\!*} y.\lambda^{-\!*} f.fxy : \tau' \mathrel{-\!\!*} (\tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon) \mathrel{-\!\!*} \upsilon}\,[\mathrel{-\!\!*}\ \mathrm{I}]
}{\varnothing \mid I \vdash \lambda^{-\!*} x.\lambda^{-\!*} y.\lambda^{-\!*} f.fxy : \tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} (\tau \mathrel{-\!\!*} \tau' \mathrel{-\!\!*} \upsilon) \mathrel{-\!\!*} \upsilon}\,[\equiv]
$$

QuB: Typing Terms

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\varnothing \mid y^{\boldsymbol{xf}} : \tau' \vdash y : \tau'}\,[\text{ID}]
\qquad
\dfrac{
\dfrac{}{\varnothing \mid x^{\boldsymbol{fy}} : \tau \vdash x : \tau}\,[\text{ID}]
\qquad
\dfrac{}{\varnothing \mid \boldsymbol{f}^{\boldsymbol{xy}} : \tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon \vdash \boldsymbol{f} : \tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon}\,[\text{ID}]
}{\varnothing \mid x^{\boldsymbol{xy}} : \tau \oplus \boldsymbol{f}^{\boldsymbol{xy}} : \tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon \vdash fx : (\tau' \twoheadrightarrow \upsilon)}\,[\mathrel{-\!\!*}\ E]
}{\varnothing \mid y^{\boldsymbol{xf}} : \tau' \oplus x^{\boldsymbol{yf}} : \tau \oplus f^{\boldsymbol{xy}} : \tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon \vdash fxy : \upsilon}\,[\twoheadrightarrow E]
}{\varnothing \mid x^{\boldsymbol{yf}} : \tau \oplus y^{\boldsymbol{xf}} : \tau' \oplus f^{\boldsymbol{xy}} : \tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon \vdash fxy : \upsilon}\,[EXCH]
}{\varnothing \mid x^{\boldsymbol{y}} : \tau \oplus y^{\boldsymbol{x}} : \tau' \vdash \lambda^{\twoheadrightarrow} f.fxy : (\tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon) \twoheadrightarrow \upsilon}\,[\twoheadrightarrow I]
}{\varnothing \mid x^{\varnothing} : \tau \vdash \lambda^{\twoheadrightarrow} y.\lambda^{\twoheadrightarrow} f.fxy : \tau' \twoheadrightarrow (\tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon) \twoheadrightarrow \upsilon}\,[\twoheadrightarrow I]
}{\varnothing \mid \boldsymbol{I} \vdash \lambda^{\mathrel{-\!\!*}} x.\lambda^{\twoheadrightarrow} y.\lambda^{\twoheadrightarrow} f.fxy : \tau \mathrel{-\!\!*} \tau' \twoheadrightarrow (\tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon) \twoheadrightarrow \upsilon}\,[\mathrel{-\!\!*}\ \text{I}]
$$

QuB: Syntax Directed Typing Rules

Term structure $\leftrightarrow$ Typing Rule

QuB: Syntax Directed Typing Rules

$$\frac{P \vdash \Gamma_{\vec{y}}\ \mathtt{un} \qquad (P \Rightarrow \tau) \sqsubseteq \sigma}{P \mid \Gamma, x^{\vec{y}} : \sigma \vdash^{s} x : \tau}\ [\mathrm{VAR}^{s}]$$

$$\frac{\begin{array}{c} Q \mid (\Gamma'_x \oplus \Gamma''_x) \circledast \Delta \vdash^{s} M : \upsilon \qquad P \vdash \Delta\ \mathtt{un} \\ P \mid (\Gamma_x, x^{\varnothing} : \sigma) \oplus \Gamma''_x \circledast \Delta \vdash^{s} N : \tau \qquad \sigma = \mathtt{Gen}(\{\Gamma' \oplus \Gamma''_x \circledast \Delta\}, Q \Rightarrow \upsilon) \end{array}}{P, Q \mid (\Gamma \circledast \Gamma') \oplus \Gamma'' \circledast \Delta \vdash^{s} (\mathtt{let}\ x = M\ \mathtt{in}\ N) : \tau}\ [\mathrm{Let}^{s}]$$

$$[\mathrm{VAR}^{s}] \equiv [\mathrm{ID}] + [\forall\mathrm{E}] + [\Rightarrow\mathrm{E}]$$

$$[\mathrm{LET}^{s}] \equiv [\mathrm{LET}] + [\forall\mathrm{I}] + [\Rightarrow\mathrm{I}]$$

QuB: Syntax Directed Typing Rules

$$\frac{P \Rightarrow \mathtt{SeFun}\ \phi \qquad P \vdash \Gamma \geq \phi \qquad P \mid \Gamma \circledast x^{\varnothing} : \tau \vdash^{s} M : \upsilon}{P \mid \Gamma \vdash^{s} \lambda^{\multimap} x.M : \phi\tau\upsilon}\ [\multimap \mathsf{I}^{s}]$$

$$\frac{P \Rightarrow \mathtt{ShFun}\ \phi \qquad P \vdash \Gamma \geq \phi \qquad P \mid \Gamma^{[\varnothing \mapsto \{x\}]} \oplus x^{\mathtt{Vars}(\Gamma)} : \tau \vdash^{s} M : \upsilon}{P \mid \Gamma \vdash^{s} \lambda^{\twoheadrightarrow} x.M : \phi\tau\upsilon}\ [\twoheadrightarrow \mathsf{I}^{s}]$$

$$\frac{P \mid \Gamma \circledast \Delta \vdash^{s} M : \phi\upsilon\tau \qquad P \mid \Gamma' \circledast \Delta \vdash^{s} N : \upsilon \qquad P \vdash \Delta\ \mathtt{un} \qquad (\Gamma \mathbin{\tilde{\oplus}} \Gamma' \wedge (P \Rightarrow \mathtt{ShFun}\ \phi)) \vee (\Gamma \mathbin{\tilde{\circledast}} \Gamma' \wedge (P \Rightarrow \mathtt{SeFun}\ \phi))}{P \mid \Gamma \sqcup \Gamma' \circledast \Delta \vdash^{s} MN : \tau}\ [\mathsf{App}^{s}]$$

$\boxed{\text{Context Operations}}$

$$\Gamma \mathbin{\tilde{\circledast}} \Delta = \mathtt{Used}(\Gamma) \mathbin{\#} \mathtt{Vars}(\Delta) \wedge \mathtt{Used}(\Delta) \mathbin{\#} \mathtt{Vars}(\Gamma)$$

$$\Gamma \mathbin{\tilde{\oplus}} \Delta = \mathtt{Used}(\Gamma) = \mathtt{Used}(\Delta)$$

$$[\twoheadrightarrow \mathsf{I}] \equiv [\twoheadrightarrow \mathsf{I}] \qquad [\multimap \mathsf{I}] \equiv [\multimap \mathsf{I}]$$

$$[\mathsf{APP}^{s}] \equiv [\twoheadrightarrow \mathsf{E}] + [\multimap \mathsf{E}]$$

QuB: Soundess and Completeness of $\vdash^s$

Theorem (Soundness of $\vdash^s$)

If $P \mid \Gamma \vdash^s M : \tau$ *then* $P \mid \Gamma \vdash M : \tau$

Theorem (Completeness of $\vdash^s$)

If $P \mid \Gamma \vdash M : \sigma$ *then* $\exists Q, \tau$ *such that* $Q \mid \Gamma \vdash^s M : \tau$ *and* $(P \mid \sigma) \sqsubseteq \mathit{Gen}(\Gamma, Q \Rightarrow \tau)$

$\boxed{\text{Original Type System}} \equiv \boxed{\text{Syntax Directed Typing Rules}}$

$\boxed{\text{Proof in Original Type System}} \equiv \boxed{\text{Proof in Syntax Directed Typing Rules}}$

$$\boxed{\mathcal{M}(S, \Psi, \Gamma \vdash M : \tau) = P, S', \Sigma, \Psi'}$$

$$\begin{aligned}
& \mathcal{M}(S, \Psi, \Gamma \vdash x : \tau) = ([\vec{u}/\vec{t}]P), S' \circ S, \{x\}, \Psi \\
\text{where} \quad & (x : \forall \vec{t}.P \Rightarrow \upsilon) \in S\Gamma \\
& S' = \mathcal{U}([\vec{u}/\vec{t}]\upsilon, S\tau)
\end{aligned}$$

$$\begin{aligned}
& \mathcal{M}(S, \Psi, \Gamma \vdash \lambda^{\twoheadrightarrow} x.M : \tau) = \{P \cup Q\}, S', \Sigma \backslash x, \Psi'' \\
\text{where} \quad & P; S'; \Sigma; \Psi' = \mathcal{M}(\mathcal{U}(\tau, u_1 u_2 u_3) \circ S, \Psi, \Gamma, x : u_2 \vdash M : u_3) \\
& \Psi'' = \{\forall_{y \in \mathtt{dom}(\Psi')}.\Psi'(y) + x\} \cup \{(x, \{y \mid y \in \mathtt{dom}(\Gamma)\})\} \\
& Q = \{\mathtt{ShFun}\ u_1\} \cup \mathtt{Leq}(u_1, \Gamma|_\Sigma) \cup \mathtt{Weaken}(x, u_2, \Sigma, \Psi'')
\end{aligned}$$

$$\begin{aligned}
& \mathcal{M}(S, \Psi, \Gamma \vdash \lambda^{\rightarrowtail} x.M : \tau) = \{P \cup Q\}, S', \Sigma \backslash x, \Psi'' \\
\text{where} \quad & P; S'; \Sigma; \Psi' = \mathcal{M}(\mathcal{U}(\tau, u_1 u_2 u_3) \circ S, X; \Gamma, x : u_2 \vdash M : u_3) \\
& \Psi'' = \Psi' \cup \{(x, \{x\})\} \\
& Q = \{\mathtt{SeFun}\ u_1\} \cup \mathtt{Leq}(u_1, \Gamma\ |_\Sigma) \cup \mathtt{Weaken}(x, u_2, \Sigma, \Psi'')
\end{aligned}$$

QuB: Algorithm $\mathcal{M}$

$$\boxed{\mathcal{M}(S, \Psi, \Gamma \vdash M : \tau) = P, S', \Sigma, \Psi'}$$

$$
\begin{aligned}
\mathcal{M}(S, \Psi, \Gamma \vdash MN : \tau) &= \{P \cup P' \cup Q\}, R', \Sigma \cup \Sigma', \Psi'' \\
\text{where} \quad P; R; \Sigma; \Psi' &= \mathcal{M}(S, \Psi, \Gamma \vdash M : u_1 u_2 \tau) \\
P'; R'; \Sigma'; \Psi'' &= \mathcal{M}(SR, \Psi', S\Gamma \vdash N : u_2) \\
&\text{if } \mathcal{C}(\Gamma, \Psi'', \Sigma) = \mathcal{C}(\Gamma, \Psi'', \Sigma') \\
&\quad \text{then } Q = \{\mathtt{ShFun}\ u_1\} \\
&\quad \text{else if } (\Sigma \# \mathcal{C}(R\Gamma, \Psi'', \Sigma') \text{ and } \Sigma' \# \mathcal{C}(R\Gamma, \Psi'', \Sigma)) \\
&\qquad \text{then } Q = \{\mathtt{SeFun}\ u_1\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{M}(S, \Psi, \Gamma \vdash \mathtt{let}\ x = M\ \mathtt{in}\ N : \tau) &= (P \cup Q), R', \Sigma \cup \{\Sigma' \backslash x\}, \Psi'' \\
\text{where} \quad P; R; \Sigma; \Psi' &= \mathcal{M}(S, \Psi, \Gamma \vdash M : u_1) \\
\sigma &= \mathtt{GenI}(R\Gamma; R(P \Rightarrow u_1)) \\
P'; R'; \Sigma'; \Psi'' &= \mathcal{M}(R, \Psi', \Gamma, x : \sigma \vdash N : \tau) \\
Q &= \mathtt{Un}(\Gamma|_{\Sigma \cap \Sigma'}) \cup \mathtt{Weaken}(x, \sigma, \Sigma', \Psi'')
\end{aligned}
$$

QuB: Soundess of Algorithm $\mathcal{M}$

Theorem (Soundness of $\mathcal{M}$)

if $\mathcal{M}(S, \Psi, \Gamma \vdash M : \tau) = P, S', \Sigma, \Psi'$ *then* $S'P \mid S'(\Gamma|_{\Sigma}) \vdash M : S'\tau$

$$\boxed{\text{Algorithm M}} \rightarrow \boxed{\text{Type system}}$$

Examples: Basic Structures

- Multiplicative Product

$$\tau \otimes \tau' = \tau \twoheadrightarrow \tau' \twoheadrightarrow (\tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon) \twoheadrightarrow \upsilon$$
$$(,) = \lambda^{\twoheadrightarrow} x.\lambda^{\twoheadrightarrow} y.\lambda^{\twoheadrightarrow} f.fxy$$

- Additive Product

$$\tau \,\&\, \tau' = \tau \twoheadrightarrow \tau' \twoheadrightarrow (\tau \twoheadrightarrow \tau' \twoheadrightarrow \upsilon) \twoheadrightarrow \upsilon$$
$$(;) = \lambda^{\twoheadrightarrow} x.\lambda^{\twoheadrightarrow} y.\lambda^{\twoheadrightarrow} f.fxy$$

- Sums

$$\tau \oplus \tau' = (\tau \to \upsilon) \to (\tau' \to \upsilon) \to \upsilon$$
$$\texttt{case } c \texttt{ of } \{f; g\} = \lambda^{\twoheadrightarrow} c.\lambda^{\twoheadrightarrow} f.\lambda^{\twoheadrightarrow} g.cfg$$

$$\texttt{inl} : \tau \twoheadrightarrow (\tau \oplus \tau') \qquad \texttt{inr} : \tau' \twoheadrightarrow (\tau \oplus \tau')$$
$$\texttt{inl} = \lambda^{\twoheadrightarrow} x.\lambda^{\twoheadrightarrow} f.\lambda^{\twoheadrightarrow} g.fx \qquad \texttt{inr} = \lambda^{\twoheadrightarrow} y.\lambda^{\twoheadrightarrow} f.\lambda^{\twoheadrightarrow} g.gy$$

QuB: Extension

- User defined types and type classes

QuB: Extension

- User defined types and type classes
- Kind System with type constructors

$$
\begin{array}{rl}
\text{Type Variables} & t, u \in \text{Type Variables} \\
\text{Kinds} & \kappa ::= \star \mid \kappa' \to \kappa \\
\text{Types} & \tau^{\kappa} ::= t^{\kappa} \mid T^{\kappa} \mid \tau^{\kappa' \to \kappa} \tau^{\kappa'} \\
\text{Type Constructors} & T^{\kappa} \in \mathcal{T}^{\kappa} \text{ where } \quad \{\otimes, \&, \oplus, \overset{!}{\twoheadrightarrow}, \twoheadrightarrow, \overset{!}{\twoheadrightarrow}, \twoheadrightarrow\} \subseteq \mathcal{T}^{\star \to \star \to} \\
\text{Predicates} & \pi, \omega ::= \mathtt{Un}\ \tau \mid \mathtt{SeFun}\ \tau \mid \mathtt{ShFun}\ \tau \mid \tau \geq \tau' \\
\text{Qualified Types} & \rho ::= \tau^{\star} \mid \pi \Rightarrow \rho \\
\text{Type schemes} & \sigma ::= \rho \mid \forall t.\sigma
\end{array}
$$

Conclusion and Future Work

What next?

Thank You!

Q & A

References I

[1] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[2] Mark P. Jones. A theory of qualified types. *Science of Computer Programming*, 22(3):231 – 256, 1994.

[3] Joachim Lambek. The mathematics of sentence structure. 65(3):154–170, 1958.

[4] Sheng Liang, Paul Hudak, and Mark Jones. Monad transformers and modular interpreters. In *Proceedings of the 22Nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '95, pages 333–343. ACM, 1995.

[5] J. Garrett Morris. The best of both worlds: Linear functional programming without compromise. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, ICFP 2016, pages 448–461. ACM, 2016.

References II

[6] Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *The Bulletin of Symbolic Logic*, 5(2):215–244, 1999.