Theorem Proving in Haskell

Niki Vazou
IMDEA Software

Theorem Proving in Haskell

e.g., theorem:

$$\phi \equiv (\exists x. \forall y. (p\ x\ y)) \Rightarrow (\forall y. \exists x. (p\ x\ y))$$

Encode Theorem as Haskell Type,

then Proof is a Haskell Expression.

Theorem Proving in Haskell

I. Refinement Types

II. Theorems as Types

III. Proofs as Programs

I. Refinement Types

42 :: Int

42 :: {v:Int | v == 42}

Base type

Refinement

```
2 :: {v:Int | v == 2}
42 :: {v:Int | v == 42}
div :: Int -> {v:Int | v /= 0} -> Int
```

div 42 2 :: ??

2 :: {v:Int | v == 2}

42 :: {v:Int | v == 42}

div :: Int -> {v:Int | v /= 0} -> Int

$$\frac{42 :: \mathrm{Int} \qquad 2 :: \{v{:}\mathrm{Int} \mid v \mathrel{/=} 0\}}{\mathrm{div}\ 42\ 2 :: ??}$$

2 :: {v:Int | v == 2}

42 :: {v:Int | v == 42}

div :: Int -> {v:Int | v /= 0} -> Int

We know

We want

$$\frac{2 :: \{v{:}Int \mid v == 2\} \Rightarrow 2 :: \{v{:}Int \mid v \mathrel{/=} 0\}}{2 :: \{v{:}Int \mid v \mathrel{/=} 0\}}$$

That is subtyping!

t_1 <: t_2 iff. $\forall$e. e::t_1 $\Rightarrow$ e::t_2

$$\frac{\{v{:}Int \mid v == 2\} <: \{v{:}Int \mid v \mathrel{/}= 0\}}{2 :: \{v{:}Int \mid v == 2\} \Rightarrow 2 :: \{v{:}Int \mid v \mathrel{/}= 0\}}$$

$$\frac{}{\{v{:}\mathrm{Int} \mid v == 2\} <: \{v{:}\mathrm{Int} \mid v \;/\!= 0\}}$$

Implication implies subtyping

$$\frac{\forall\ v.\ v == 2 \Rightarrow v\ /= 0}{\{v{:}Int\ \mid\ v == 2\} <: \{v{:}Int \mid v\ /= 0\}}$$

Putting it all together

$$\dfrac{\dfrac{42 :: \text{Int} \qquad \dfrac{2 :: \{v{:}\text{Int} \mid v == 2\} \quad \dfrac{\forall v.\ v == 2 \Rightarrow v \mathrel{/}= 0}{\{v{:}\text{Int} \mid v == 2\} <: \{v{:}\text{Int} \mid v \mathrel{/}= 0\}}}{2 :: \{v{:}\text{Int} \mid v \mathrel{/}= 0\}}}{}}{\text{div } 42\ 2 :: \text{Int}}$$

Specifications

div :: Int -> {v:Int | v /= 0} -> Int

42 :: {v:Int | v == 42}

2 :: {v:Int | v == 2}

Observe: Implications have the language of specifications

$$\dfrac{42 :: \mathrm{Int} \qquad \dfrac{2 :: \{v:\mathrm{Int} \mid \boxed{v == 2}\} \qquad \dfrac{\forall\, v.\ \boxed{v == 2} \Rightarrow \boxed{v \mathbin{/=} 0}}{\{v:\mathrm{Int} \mid \boxed{v == 2}\} <: \{v:\mathrm{Int} \mid \boxed{v \mathbin{/=} 0}\}}}{2 :: \{v:\mathrm{Int} \mid \boxed{v \mathbin{/=} 0}\}}}{\mathrm{div}\ 42\ 2 :: \mathrm{Int}}$$

Specifications

$$\mathrm{div} :: \mathrm{Int} \to \{v:\mathrm{Int} \mid \boxed{v \mathbin{/=} 0}\} \to \mathrm{Int}$$

$$42 :: \{v:\mathrm{Int} \mid v == 42\}$$

$$2 :: \{v:\mathrm{Int} \mid \boxed{v == 2}\}$$

Let p be the language of refinements

Implications

$\forall \overline{x} . p \Rightarrow p$

$$\dfrac{42 :: \text{Int} \qquad \dfrac{2 :: \{v{:}\text{Int} \mid \boxed{v == 2}\} \quad \dfrac{\forall v. \boxed{v == 2} \Rightarrow \boxed{v \mathrel{/=} 0}}{\{v{:}\text{Int} \mid \boxed{v == 2}\} <: \{v{:}\text{Int} \mid \boxed{v \mathrel{/=} 0}\}}}{2 :: \{v{:}\text{Int} \mid \boxed{v \mathrel{/=} 0}\}}}{\text{div } 42 \ 2 :: \text{Int}}$$

Specifications

over p

$\text{div} :: \text{Int} \rightarrow \{v{:}\text{Int} \mid \boxed{v \mathrel{/=} 0}\} \rightarrow \text{Int}$

$42 :: \{v{:}\text{Int} \mid v == 42\}$

$2 :: \{v{:}\text{Int} \mid \boxed{v == 2}\}$

Let p be the language of refinements

Implications

$\forall \overline{x} \,.\, p \Rightarrow p$

$$\dfrac{42 :: \mathrm{Int} \qquad \dfrac{\dfrac{\forall v.\ \boxed{v == 2} \Rightarrow \boxed{v \mathrel{/=} 0}}{2 :: \{v{:}\mathrm{Int} \mid \boxed{v == 2}\} \quad \{v{:}\mathrm{Int} \mid \boxed{v == 2}\} <: \{v{:}\mathrm{Int} \mid \boxed{v \mathrel{/=} 0}\}}}{2 :: \{v{:}\mathrm{Int} \mid \boxed{v \mathrel{/=} 0}\}}}{\boxed{\Gamma \vdash e :: t}}$$

Specifications

over p

div :: Int -> {v:Int | v /= 0} -> Int

42 :: {v:Int | v == 42}

2 :: {v:Int | v == 2}

Let p be the language of refinements

Implications

$\forall\ \overline{x}\ .\ p \Rightarrow p$

$$\forall\ v.\ v == 2 \Rightarrow v \mathbin{/=} 0$$

$$\dfrac{\forall\ v.\ v == 2 \Rightarrow v \mathbin{/=} 0}{2 :: \{v{:}Int \mid v == 2\}\quad \{v{:}Int \mid v == 2\} <: \{v{:}Int \mid v \mathbin{/=} 0\}}$$

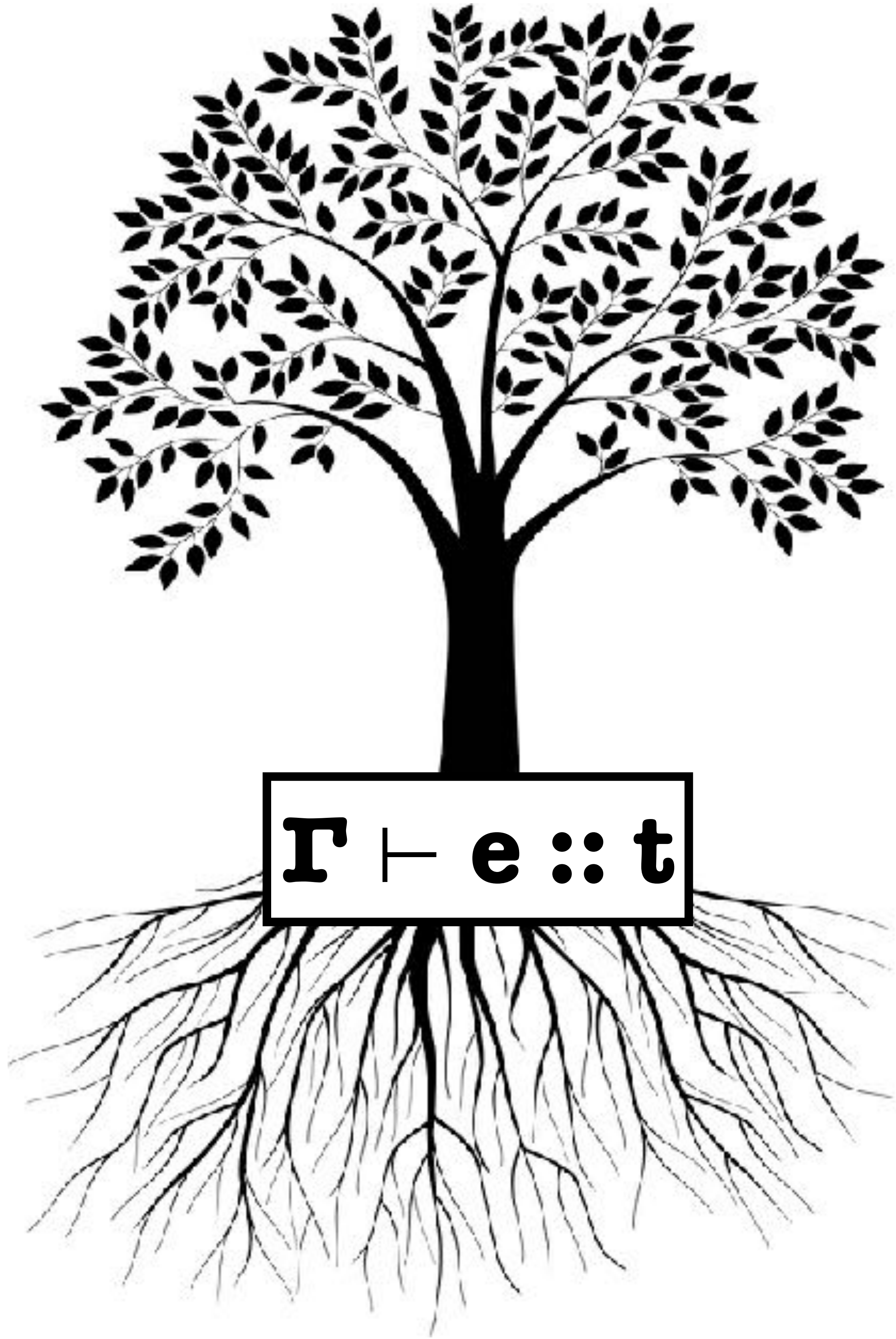
$$\dfrac{42 :: Int \qquad\qquad 2 :: \{v{:}Int \mid v \mathbin{/=} 0\}}{\boxed{\boldsymbol{\Gamma \vdash e :: t}}}$$

Specifications

over p

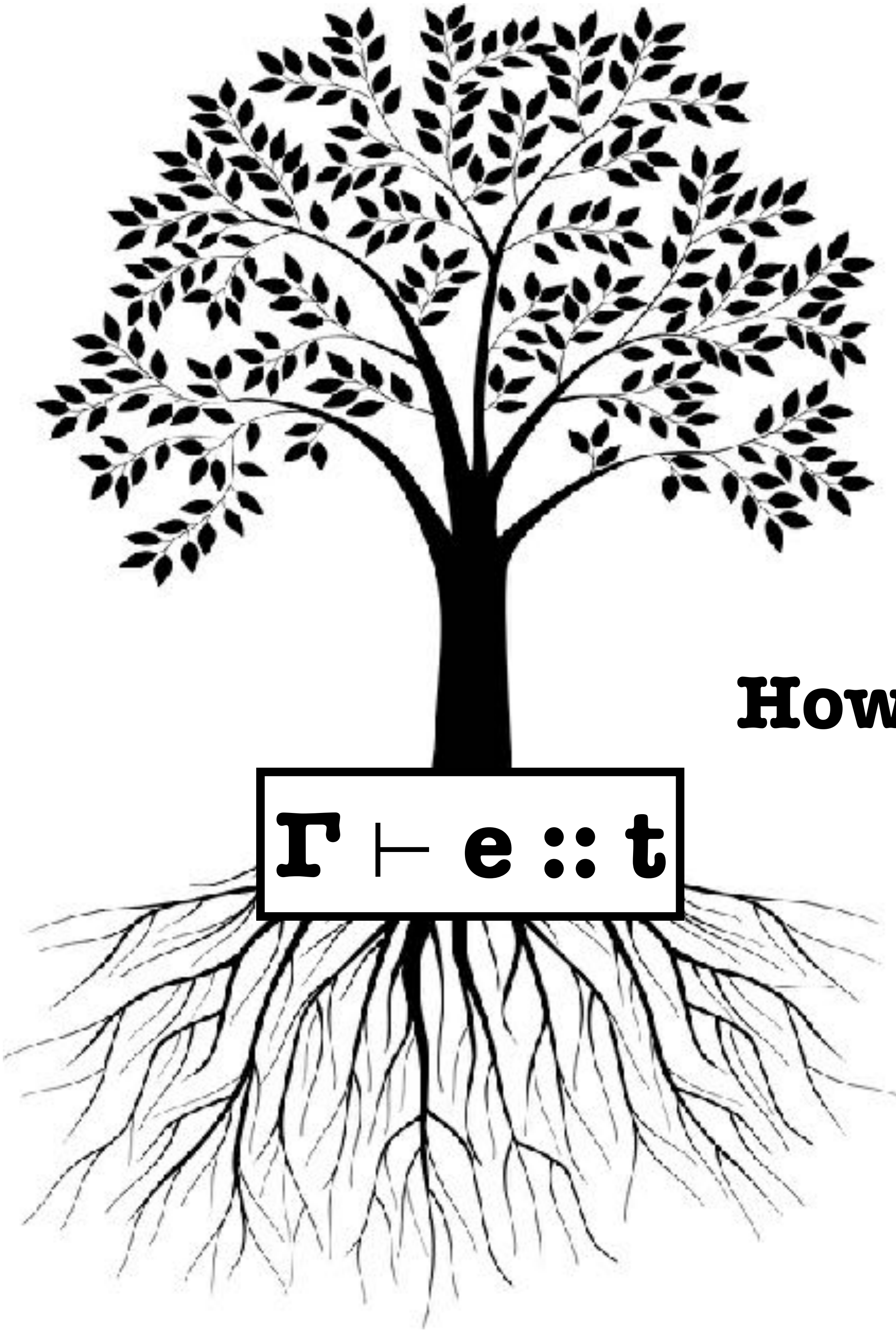
Implications

$\forall\, \overline{x}\,.\, p \Rightarrow p$



Specifications

over p



Implications

$\forall \overline{x} . p \Rightarrow p$

For decidable implications,
p should be quantifier free.

How to express quantified theorems?

Specifications

over p

Theorem Proving in Haskell

I. Refinement Types

II. Theorems as Types

II. Theorems as Types
(a.k.a. Curry Howard Correspondence)

	Logical Formula
Native Term	e
Conjunction	$\varphi_1 \wedge \varphi_2$
Disjunction	$\varphi_1 \vee \varphi_2$
Implication	$\varphi_1 \Rightarrow \varphi_2$
Forall	$\forall\ x.\ \varphi$
Exists	$\exists\ x.\ \varphi$

	Logical Formula	Refinement Type
Native Term	e	
Conjunction	$\varphi_1 \land \varphi_2$	
Disjunction	$\varphi_1 \lor \varphi_2$	
Implication	$\varphi_1 \Rightarrow \varphi_2$	
Forall	$\forall\ x.\ \varphi$	
Exists	$\exists\ x.\ \varphi$	

	Logical Formula	Refinement Type
Native Term	e	??
Conjunction	$\varphi_1 \land \varphi_2$	
Disjunction	$\varphi_1 \lor \varphi_2$	
Implication	$\varphi_1 \Rightarrow \varphi_2$	
Forall	$\forall x.\ \varphi$	
Exists	$\exists x.\ \varphi$	

Native Term e as a Refinement Type

$$\{v{:}() \mid e\}$$

Some unit value

so that e is true.

	Logical Formula	**Refinement Type**
Native Term	e	{v:() | e }
e.g.,	42 <= 42	{v:() | 42 <= 42 }
	x == y	{v:() | x == y }
	p x y	{v:() | p x y }

	Logical Formula	Refinement Type
Native Term	e	$\{v:() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	??
Disjunction	$\varphi_1 \vee \varphi_2$	
Implication	$\varphi_1 \Rightarrow \varphi_2$	
Forall	$\forall x.\ \varphi$	
Exists	$\exists x.\ \varphi$	

Conjunction $\varphi_1 \wedge \varphi_2$ as a Refinement Type

$$(\varphi_1, \varphi_2)$$

Conjunction $\varphi_1 \wedge \varphi_2$ as a Refinement Type

$$\frac{\Gamma \vdash \texttt{e :: } (\varphi_1, \varphi_2)}{\Gamma \vdash \texttt{case e of \{(x,y) -> x\} :: } \varphi_1}$$

(i.e., ∧-L-Elimination)

Conjunction $\varphi_1 \wedge \varphi_2$ as a Refinement Type

$$\frac{\Gamma \vdash \texttt{e :: } (\varphi_1, \varphi_2)}{\Gamma \vdash \texttt{case e of \{(x,y) -> y\} :: } \varphi_2}$$

(i.e., ∧-R-Elimination)

Conjunction $\varphi_1 \wedge \varphi_2$ as a Refinement Type

$$\frac{\Gamma \vdash e_1 :: \varphi_1 \qquad \Gamma \vdash e_2 :: \varphi_2}{\Gamma \vdash (e_1, e_2) :: (\varphi_1, \varphi_2)}$$

(i.e., $\wedge$-Introduction)

	Logical Formula	Refinement Type
Native Term	e	$\{v{:}() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \vee \varphi_2$	
Implication	$\varphi_1 \Rightarrow \varphi_2$	
Forall	$\forall x. \varphi$	
Exists	$\exists x. \varphi$	

	Logical Formula	**Refinement Type**
Native Term	e	$\{v{:}() \mid e\}$
Conjunction	$\varphi_1 \land \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \lor \varphi_2$	**??**
Implication	$\varphi_1 \Rightarrow \varphi_2$	
Forall	$\forall\ x.\ \varphi$	
Exists	$\exists\ x.\ \varphi$	

Disjunction $\varphi_1 \vee \varphi_2$ as a Refinement Type

$$\text{Either } \varphi_1 \ \varphi_2$$

```
data Either a b = Left a | Right b
```

	Logical Formula	**Refinement Type**
Native Term	e	$\{v:() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \vee \varphi_2$	Either φ_1 φ_2
Implication	$\varphi_1 \Rightarrow \varphi_2$	
Forall	$\forall$ x. φ	
Exists	$\exists$ x. φ	

	Logical Formula	**Refinement Type**
Native Term	e	$\{v:() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \vee \varphi_2$	Either φ_1 φ_2
Implication	$\varphi_1 \Rightarrow \varphi_2$	**??**
Forall	$\forall x.\ \varphi$	
Exists	$\exists x.\ \varphi$	

Implication $\varphi_1 \Rightarrow \varphi_2$ as a Refinement Type

$$\varphi_1 \rightarrow \varphi_2$$

Implication $\varphi_1 \Rightarrow \varphi_2$ as a Refinement Type

$$\frac{\Gamma,\ \texttt{x}:\ \varphi_1 \vdash \texttt{e} :: \varphi_2}{\Gamma \vdash \texttt{\textbackslash x. e} :: \varphi_1 \rightarrow \varphi_2}$$

(i.e., $\Rightarrow$-Introduction)

Implication $\varphi_1 \Rightarrow \varphi_2$ as a Refinement Type

$$\frac{\Gamma \vdash e :: \varphi_1 \to \varphi_2 \qquad \Gamma \vdash e_x :: \varphi_1}{\Gamma \vdash e\ e_x :: \varphi_2}$$

(i.e., $\Rightarrow$-Elimination)

	Logical Formula	Refinement Type
Native Term	e	$\{v{:}() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \vee \varphi_2$	Either φ_1 φ_2
Implication	$\varphi_1 \Rightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$
Forall	$\forall x.\ \varphi$	??
Exists	$\exists x.\ \varphi$	

$\forall$ x. φ as a Refinement Type

$$x{:}\varphi_x \rightarrow \varphi$$

$\forall$ x. φ as a Refinement Type

$$\frac{\Gamma,\ x : \varphi_x \vdash e :: \varphi}{\Gamma \vdash \backslash x.\ e :: x{:}\varphi_x \rightarrow \varphi}$$

(i.e., $\forall$-Introduction)

$\forall$ x. φ as a Refinement Type

$$\frac{\Gamma \vdash e :: x{:}\varphi_x \to \varphi \qquad \Gamma \vdash e_x :: \varphi_x}{\Gamma \vdash e\ e_x :: \varphi[e_x/x]}$$

(i.e., $\forall$-Elimination)

	Logical Formula	**Refinement Type**
Native Term	e	$\{v{:}() \mid e\}$
Conjunction	$\varphi_1 \land \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \lor \varphi_2$	Either φ_1 φ_2
Implication	$\varphi_1 \Rightarrow \varphi_2$	$\varphi_1 \rightarrow \varphi_2$
Forall	$\forall$ x. φ	$x{:}\varphi_x \rightarrow \varphi$
Exists	$\exists$ x. φ	??

$\exists$ x. φ as a Refinement Type

$$(\mathtt{x}{:}\varphi_{\mathtt{x}},\ \varphi)$$

$\exists$ x. φ as a Refinement Type

$$\frac{\Gamma \vdash e_x :: \varphi_x \qquad \Gamma, x : \varphi_x \vdash e :: \varphi}{\Gamma \vdash (e_x, e) :: (x{:}\varphi_x, \varphi)}$$

(i.e., $\exists$-Introduction)

$\exists$ x. φ as a Refinement Type

$$\frac{\begin{array}{c}\Gamma \vdash e :: (x{:}\varphi_x,\ \varphi) \\ \Gamma,\ x{:}\varphi_x\ ,\ x_\varphi{:}\varphi \vdash e_0 :: \varphi_0\end{array}}{\Gamma \vdash \texttt{case } e \texttt{ of } \{(x, x_\varphi) \texttt{ -> } e_0\} :: \varphi_0}$$

(i.e., $\exists$-Elimination)

	Logical Formula	Refinement Type
Native Terms	e	$\{v:() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \vee \varphi_2$	Either φ_1 φ_2
Implication	$\varphi_1 \Rightarrow \varphi_2$	$\varphi_1 \to \varphi_2$
Forall	$\forall$ x. φ	$x{:}\varphi_x \to \varphi$
Exists	$\exists$ x. φ	$(x{:}\varphi_x, \varphi)$

	Logical Formula	**Refinement Type**
Native Terms	e	$\{v{:}() \mid e\}$
Conjunction	$\varphi_1 \wedge \varphi_2$	(φ_1, φ_2)
Disjunction	$\varphi_1 \vee \varphi_2$	Either φ_1 φ_2
Implication	$\varphi_1 \Rightarrow \varphi_2$	$\varphi_1 \to \varphi_2$
Forall	$\forall\, x.\ \varphi$	$x{:}\varphi_x \to \varphi$
Exists	$\exists\, x.\ \varphi$	$(x{:}\varphi_x, \varphi)$

Example:

$$(\exists\, x.\ \forall\, y.\ p\ x\ y) \Rightarrow (\forall\, y.\ \exists\, x.\ p\ x\ y)$$

Example:

$$\boxed{(\exists\, x.\ \forall\, y.\ p\ x\ y) \Rightarrow (\forall\, y.\ \exists\, x.\ p\ x\ y)}$$

$$\boxed{\rightarrow}$$

Example:

$$\boxed{(\exists\, x.\ \forall\, y.\ p\ x\ y)} \Rightarrow (\forall\, y.\ \exists\, x.\ p\ x\ y)$$

$$\boxed{(x{:}a, \qquad\qquad\qquad\qquad)} \to$$

Example:

$$(\exists\, x.\ \boxed{\forall\, y.\ p\ x\ y}) \Rightarrow (\forall\, y.\ \exists\, x.\ p\ x\ y)$$

$$(x{:}a,\ \boxed{y{:}a \rightarrow \qquad\qquad\qquad\qquad}) \rightarrow$$

Example:

$$(\exists\, x.\ \forall\, y.\ \boxed{p\ x\ y}) \Rightarrow (\forall\, y.\ \exists\, x.\ p\ x\ y)$$

$$(x{:}a,\ y{:}a \rightarrow \boxed{\{v{:}\ ()\ \mid\ p\ x\ y\}}) \rightarrow$$

Example:

$$(\exists\, x.\ \forall\, y.\ p\ x\ y) \Rightarrow (\boxed{\forall\, y.\ \exists\, x.\ p\ x\ y})$$

$$(x{:}a,\ y{:}a \rightarrow \{v{:}\ ()\ \mid\ p\ x\ y\})\rightarrow \boxed{y{:}a \rightarrow \quad\quad\quad\quad\quad\quad}$$

Example:

$$(\exists\, x.\ \forall\, y.\ p\ x\ y) \Rightarrow (\forall\, y.\ \boxed{\exists\, x.\ p\ x\ y})$$

$$(x{:}a,\ y{:}a \to \{v{:}\ ()\ \mid\ p\ x\ y\}) \to y{:}a \to \boxed{(x{:}a, \qquad\qquad\qquad}\,)$$

Example:

$$(\exists\, x.\ \forall\, y.\ p\ x\ y) \Rightarrow (\forall\, y.\ \exists\, x.\ \boxed{p\ x\ y})$$

$$(x{:}a,\ y{:}a \rightarrow \{v{:}\ ()\ \mid\ p\ x\ y\}\ ) \rightarrow y{:}a \rightarrow (x{:}a,\ \boxed{\{v{:}()\ \mid\ p\ x\ y\}})$$

Example:

$$(\exists\, x.\ \forall\, y.\ p\ x\ y) \Rightarrow (\forall\, y.\ \exists\, x.\ p\ x\ y)$$

$$(x{:}a,\ y{:}a \to \{v{:}\ ()\ |\ p\ x\ y\}) \to y{:}a \to (x{:}a,\ \{v{:}()\ |\ p\ x\ y\})$$

Proof?

Theorem Proving in Haskell

I. Refinement Types

II. Theorems as Types

III. Proofs as Programs

III. Proofs as Programs

DEMO

Theorem Proving in Haskell

I. Refinement Types

II. Theorems as Types

III. Proofs as Programs

Looking for a PostDoc @IMDEA, Madrid

Theorem Proving in Haskell

I. Refinement Types

II. Theorems as Types

III. Proofs as Programs

Thanks!