Refinement Types
&
Function Extensionality

Work in Progress
Niki Vazou, IMDEA
IFIP WG 2.1

Refinement Types

```
plus1R   :: x:Int → {v:Int | x < v}
plus1R x = x + 1
```

Refinement Types

```
plus1R   :: x:Int → {v:Int | x < v}
plus1R x = x + 1
```

$$\cfrac{\cfrac{\cfrac{\texttt{…,x:Int, v:Int; v = x + 1} \vdash \texttt{x < v}}{\vdots}}{\Gamma\texttt{,x:Int} \vdash \texttt{x + 1 :: \{v:Int | x < v\}}}}{\Gamma \vdash \lambda\texttt{x.x + 1 :: x:Int → \{v:Int | x < v\}}}$$

Refinement Types

```
plus1R   :: x:Int → {v:Int | x < v}
plus1R x = x + 1
```

$$\dfrac{\dfrac{\dfrac{\ldots,x:\mathtt{Int},\ v:\mathtt{Int};\ v = x + 1 \vdash x < v}{\vdots}}{\Gamma,x:\mathtt{Int} \vdash x + 1 :: \{v:\mathtt{Int} \mid x < v\}}}{\underbrace{\Gamma}_{\Gamma} \vdash \underbrace{\lambda x.x + 1 :: x:\mathtt{Int} \to \{v:\mathtt{Int} \mid x < v\}}_{e\ ::\ t}}$$

Refinement Types

```
plus1R   :: x:Int → {v:Int | x < v}
plus1R x = x + 1
```

$$\underbrace{\ldots,x{:}\mathtt{Int},\ v{:}\mathtt{Int};}_{\Sigma}\ \underbrace{v = x + 1}_{\Phi}\ \vdash\ \underbrace{x < v}_{p}$$

$$\vdots$$

Γ ⊢ e :: t **Refinement Typing**

Refinement Types

```
plus1R   :: x:Int → {v:Int | x < v}
plus1R x = x + 1
```

$\Sigma;\Phi \vdash p$ **SMT**

$\Gamma \vdash e :: t$ **Refinement Typing**

Function Equality & Refinement Types

```
plus1L x = 1 + x
plus1R x = x + 1
```

We want to show that plus1R == plus1L

```
lemma :: x:Int → { plus1L x == plus1R x }
lemma x =  ()
```

Function Equality & Refinement Types

```
plus1L x = 1 + x
plus1R x = x + 1
```

We want to show that `plus1R == plus1L`

$$\frac{\dfrac{\ldots,x{:}\mathrm{Int};\varnothing \vdash \mathtt{plus1R}\ x == \mathtt{plus1L}\ x}{\Gamma,x{:}\mathrm{Int} \vdash () :: \{\mathtt{plus1R}\ x == \mathtt{plus1L}\ x\}}}{\Gamma \vdash \lambda x.() :: x{:}\mathrm{Int} \to \{\mathtt{plus1R}\ x == \mathtt{plus1L}\ x\}}$$

Function Equality & Refinement Types

```
plus1L x = 1 + x
plus1R x = x + 1
```

Add an intermediate level that unfolds functions!

$\ldots, x{:}\mathtt{Int}; \varnothing \vdash \mathtt{plus1R}\ x == \mathtt{plus1L}\ x$ ❌

$$\frac{\Gamma, x{:}\mathtt{Int} \vdash () :: \{\mathtt{plus1R}\ x == \mathtt{plus1L}\ x\}}{\Gamma \vdash \lambda x.() :: x{:}\mathtt{Int} \to \{\mathtt{plus1R}\ x == \mathtt{plus1L}\ x\}}$$

Function Equality & Refinement Types

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \Phi =$$

Add an intermediate level that unfolds functions!

$$F;\ldots,x:\mathtt{Int};\varnothing \vdash \texttt{plus1R x == plus1L x} ;\Phi$$

Function Equality & Refinement Types

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \boxed{\texttt{plus1R x = x + 1}} \end{cases} \qquad \phi = \texttt{plus1R x = x + 1}$$

Add an intermediate level that unfolds functions!

$$\texttt{F;…,x:Int;}\varnothing \vdash \boxed{\texttt{plus1R x}} \texttt{ == plus1L x ;}\phi$$

Function Equality & Refinement Types

$$F = \begin{cases} \boxed{\texttt{plus1L x = 1 + x}} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \phi = \begin{array}{l} \texttt{plus1R x = x + 1} \\ \texttt{plus1L x = 1 + x} \end{array}$$

Add an intermediate level that unfolds functions!

$$\boxed{F;\ldots,x{:}\texttt{Int};\varnothing \vdash \texttt{plus1R x ==} \boxed{\texttt{plus1L x}};\phi}$$

Function Equality & Refinement Types

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \Phi = \begin{array}{l} \texttt{plus1R x = x + 1} \\ \texttt{plus1L x = 1 + x} \end{array}$$

$$\overline{\ldots, x{:}\mathrm{Int}; \Phi \vdash \texttt{plus1R x == plus1L x}} \quad ✓$$

$$F; \ldots, x{:}\mathrm{Int}; \varnothing \vdash \texttt{plus1R x == plus1L x} \;; \Phi$$

$$\frac{\Gamma, x{:}\mathrm{Int} \vdash () :: \{\texttt{plus1R x == plus1L x}\}}{\Gamma \vdash \lambda x.() :: x{:}\mathrm{Int} \to \{\texttt{plus1R x == plus1L x}\}}$$

Function Equality & Refinement Types

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \phi = \begin{array}{l} \texttt{plus1R x = x + 1} \\ \texttt{plus1L x = 1 + x} \end{array}$$

$\Sigma;\Phi \vdash p$ **SMT**

$F;\ldots,x{:}\mathtt{Int};\varnothing \vdash \texttt{plus1R x == plus1L x} \;;\phi$

$\Gamma \vdash e :: t$ **Refinement Typing**

Proof By Logical Evaluation (PLE)

Unfolds functions to strengthen SMT environment

✓ terminating
✓ complete
✓ sound

$\Sigma;\Phi \vdash p$ **SMT**

$F;\Sigma;\Phi \vdash p;\Phi$ **PLE**

$\Gamma \vdash e :: t$ **Refinement Typing**

Proof By Logical Evaluation (PLE)

```
plus1L x = 1 + x
plus1R x = x + 1
```

We can now show that plus1R == plus1L

```
lemma :: x:Int → { plus1L x == plus1R x }
lemma x =  ()
```

How about:

```
thm :: { plus1L == plus1R }
thm =  ()
```

Proof By Logical Evaluation (PLE)

```
plus1L x = 1 + x
plus1R x = x + 1
```

```
thm :: { plus1L == plus1R }
thm =  ()
```

$$\Gamma \vdash \texttt{()} :: \{\texttt{plus1R == plus1L}\}$$

Proof By Logical Evaluation (PLE)

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \Phi = \varnothing$$

$$F;\ \ldots;\varnothing \vdash \texttt{plus1R == plus1L}\ ;\Phi$$

$$\Gamma \vdash \texttt{()} :: \{\texttt{plus1R == plus1L}\}$$

Proof By Logical Evaluation (PLE)

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \Phi = \varnothing$$

$\ldots; \varnothing \vdash \texttt{plus1R == plus1L}$

$F; \ldots; \varnothing \vdash \texttt{plus1R == plus1L}\ ; \Phi$

$\Gamma \vdash \texttt{()} :: \{\texttt{plus1R == plus1L}\}$

Proof By Logical Evaluation (PLE) fails on unsaturated functions

$$F = \begin{cases} \texttt{plus1L x = 1 + x} \\ \texttt{plus1R x = x + 1} \end{cases} \qquad \Phi = \varnothing$$

$\ldots;\varnothing \vdash$ `plus1R == plus1L` ❌

$F;\ \ldots;\varnothing \vdash$ `plus1R == plus1L` $;\Phi$

$\Gamma \vdash$ `() :: {plus1R == plus1L}`

Proof By Logical Evaluation (PLE) fails on unsaturated functions

```
lemma :: x:Int → { plus1L x == plus1R x }
lemma x =  ()
```

✓

Link: Function Extensionality

```
thm :: { plus1L == plus1R }
thm =  ()
```

✗

Refinement Types & Function Extensionality

Approach I: Use extensionality as an axiom

Extensionality as an Axiom

```
extensionality :: f:(a → b) → g:(a → b)
               → (x:a → {f x == g x})
               → {f == g}
```

Extensionality as an Axiom

```
extensionality :: f:(a → b) → g:(a → b)
                → (x:a → {f x == g x})
                → {f == g}

lemma :: x:Int -> { plus1L x == plus1R x }
lemma x =  ()

thm :: { plus1L == plus1R }
thm =  extensionality plus1L plus1R lemma
```
✓

Refinement Types & Function Extensionality

Approach I: Use extensionality as an axiom

+ the standard route (Coq, Agda, Dafny, F*)
+ no system modification

- requires explicit FO proofs
- it can be unsound!!!

Unsoundness of Function Extensionality

```
extensionality ::forall a b.
    f:(a → b) → g:(a → b)
  → (x:a → {f x == g x})
  → {f == g}

thm :: { plus1L == plus1R }
thm =  extensionality @{v:Int|d} @{v:Int|r}
                        plus1L plus1R lemma
```

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r} ::
    f:({v:Int|d} → {v:Int|r})
  → g:({v:Int|d} → {v:Int|r})
  → (x:{v:Int|d} → {f x == g x})
  → {f == g}
```

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r} ::
    f:({v:Int|d} → {v:Int|r})
  → g:({v:Int|d} → {v:Int|r})
  → (x:{v:Int|d} → {f x == g x})
  → {f == g}
```

```
plus1L :: {v:Int|d} → {v:Int|r}
```

Unsoundness of Function Extensionality

```
plus1L :: {v:Int| domain plus1L}
        → {v:Int| range plus1L }
----------------------------------------------
plus1L :: {v:Int|d} → {v:Int|r}
```

Functional Subtyping

$$\frac{\Gamma \vdash t'_x <: t_x \qquad \Gamma, x{:}t_x' \vdash t <: t'}{\Gamma \vdash x{:}t_x \rightarrow t <: x{:}t_x' \rightarrow t'}\ \text{TSub}$$

$$\frac{\begin{array}{l}\texttt{plus1L :: \{v:Int| domain plus1L\}} \\ \qquad\quad \texttt{→ \{v:Int| range plus1L \}}\end{array}}{\texttt{plus1L :: \{v:Int|d\} → \{v:Int|r\}}}\ \text{TSub}$$

Functional Subtyping

$$\frac{\Gamma \vdash t'_x <: t_x \qquad \Gamma, x{:}t_x' \vdash t <: t'}{\Gamma \vdash x{:}t_x \rightarrow t <: x{:}t_x' \rightarrow t'} \ \text{TSub}$$

$$\frac{\dfrac{d \Rightarrow \text{domain plus1L} \qquad d \wedge \text{range plus1L} \Rightarrow r}{\texttt{plus1L :: \{v:Int| domain plus1L\}} \rightarrow \texttt{\{v:Int| range plus1L \}}} \ \text{TSub}}{\texttt{plus1L :: \{v:Int|d\}} \rightarrow \texttt{\{v:Int|r\}}}$$

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L ::
    g:({v:Int|d} → {v:Int|r})
  → (x:{v:Int|d} → {plus1L x == g x}
  → {plus1L == g}
```

d ⇒ domain plus1L

d ∧ range plus1L ⇒ r

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L ::
    g:({v:Int|d} → {v:Int|r})
  → (x:{v:Int|d} → {plus1L x == g x}
  → {plus1L == g}
```

$$\frac{\begin{array}{cc} d \Rightarrow \text{domain plus1L} & d \wedge \text{range plus1L} \Rightarrow r \\ d \Rightarrow \text{domain plus1R} & d \wedge \text{range plus1R} \Rightarrow r \end{array}}{\texttt{plus1R :: \{v:Int|d\} → \{v:Int|r\}}}$$

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L
    plus1R ::
    (x:{v:Int|d} → {plus1L x == plus1R x}
  → {plus1L == plus1R}
```

$d \Rightarrow$ domain plus1L
$d \Rightarrow$ domain plus1R

$d \wedge$ range plus1L $\Rightarrow r$
$d \wedge$ range plus1R $\Rightarrow r$

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L
    plus1R ::
    (x:{v:Int|d} → {plus1L x == plus1R x}
  → {plus1L == plus1R}
```

$d \Rightarrow$ domain plus1L

$d \Rightarrow$ domain plus1R

$d \Rightarrow$ domain lemma

$d \wedge$ range plus1L $\Rightarrow r$

$d \wedge$ range plus1R $\Rightarrow r$

$d \wedge$ range lemma $\Rightarrow$ plus1L x == plus1R x

```
lemma :: x:{v:Int|d} → {plus1L x == plus1R x}
```

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L plus1R lemma :: {plus1L == plus1R}
```

d ⇒ domain plus1L
d ⇒ domain plus1R
d ⇒ domain lemma

d ∧ range plus1L ⇒ r
d ∧ range plus1R ⇒ r
d ∧ range lemma ⇒
plus1L x == plus1R x

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L plus1R lemma :: {plus1L == plus1R}
```

d ⇒ domain plus1L
d ⇒ domain plus1R
d ⇒ domain lemma

d ∧ range plus1L ⇒ r
d ∧ range plus1R ⇒ r
d ∧ range lemma ⇒
plus1L x == plus1R x

Note: d only appears in assumptions!

So, if d := false extensionality proofs **ANYTHING**!

Unsoundness of Function Extensionality

```
extensionality @{v:Int|d} @{v:Int|r}
    plus1L plus1R lemma :: {plus1L == plus2 }
```

d ⇒ domain plus1L
d ⇒ domain plus1R
d ⇒ domain lemma

d ∧ range plus1L ⇒ r
d ∧ range plus1R ⇒ r
d ∧ range lemma ⇒
plus1L x == plus2 x

Note: d only appears in assumptions!

So, if d := false extensionality proofs **ANYTHING**!

Cure: default d to true

Cure of Unsoundness of Function Extensionality

```
extensionality :: f:(a → b) → g:(a → b)
               → (x:a → {f x == g x})
               → {f == g}
```

$\boxed{\texttt{a := \{v:Int|d\}}}$

Note: d only appears in assumptions!

because a only appears positively

Cure: default d to true

Cure of Unsoundness of Function Extensionality

```
extensionality :: f:(a → b) → g:(a → b)
                → (x:a → {f x == g x})
                → {f == g}
```

```
a := {v:Int|true}
```

Note: d only appears in assumptions!

because a only appears positively

Cure: instantiate positive variables with true

Refinement Types & Function Extensionality

Approach I: Use extensionality as an axiom

+ the standard route (Coq, Agda, Dafny, F*)
+ no system modification

- requires explicit FO proofs
- it can be unsound!!!

Refinement Types & Function Extensionality

Approach I: Use extensionality as an axiom

Approach II: Define an extensionality “layer”

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

…;∅ ⊢ plus1R == plus2L ❌

F;…;∅ ⊢ plus1R == plus2L;∅

Γ ⊢ () :: {plus1R == plus1L}

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

SMT

PLE

$\Gamma \vdash$ `() :: {plus1R == plus1L}`

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

SMT

PLE

Function Extensinality

$\Gamma \vdash$ `() :: {plus1R == plus1L}`

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

SMT

PLE

$$\frac{\ldots, x{:}\mathtt{Int}; \varnothing \vdash \mathtt{plus1R}\ x == \mathtt{plus1L}\ x}{\ldots; \varnothing \vdash \mathtt{plus1R} == \mathtt{plus1L}}$$

$$\Gamma \vdash () :: \{\mathtt{plus1R} == \mathtt{plus1L}\}$$

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

$$\phi = \begin{array}{l}\texttt{plus1R x = x + 1}\\ \texttt{plus1L x = 1 + x}\end{array}$$

SMT

$\texttt{F;…,x:Int;}\varnothing \vdash \texttt{plus1R x == plus1L x;}\phi$

$$\frac{\texttt{…,x:Int;}\varnothing \vdash \texttt{plus1R x == plus1L x}}{\texttt{…;}\varnothing \vdash \texttt{plus1R == plus1L}}$$

$\Gamma \vdash \texttt{() :: \{plus1R == plus1L\}}$

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

$$\phi = \begin{array}{l}\texttt{plus1R x = x + 1}\\ \texttt{plus1L x = 1 + x}\end{array}$$

$$\ldots, x{:}Int; \phi \vdash \texttt{plus1R x == plus2L x} \quad ✓$$

$$F; \ldots, x{:}Int; \varnothing \vdash \texttt{plus1R x == plus1L x}; \phi$$

$$\frac{\ldots, x{:}Int; \varnothing \vdash \texttt{plus1R x == plus1L x}}{\ldots; \varnothing \vdash \texttt{plus1R == plus1L}}$$

$$\Gamma \vdash () :: \{\texttt{plus1R == plus1L}\}$$

The Function Extensionality Layer

```
thm :: { plus1L == plus1R }
thm =  ()
```

SMT

PLE

Function Extensionality

Refinement Typing

The Function Extensionality Layer

$$\frac{\Sigma, x{:}a; \Phi \vdash \texttt{f x == g x} \qquad \Sigma \vdash \texttt{f,g :: a → b}}{\Sigma; \Phi \vdash \texttt{f == g}} \;=\!R$$

How to “uncover” equalities?

How to “uncover” equalities?

```
thm :: f:(a → b) → g:(a → b) → { id f == id g ⇒ f == g }
thm =  ()
```

$$\dfrac{\dfrac{\dfrac{\Sigma,x{:}a;\Phi,\mathtt{id\ f\ x == id\ g\ x} \vdash \mathtt{f\ x == g\ x}}{\Sigma,x{:}a;\Phi,\mathtt{id\ f == id\ g} \vdash \mathtt{f\ x == g\ x}}\ {=}\mathrm{L}}{\Sigma;\Phi,\mathtt{id\ f == id\ g} \vdash \mathtt{f == g}}\ {=}\mathrm{R}}{\Sigma;\Phi \vdash \mathtt{id\ f == id\ g} \Rightarrow \mathtt{f == g}}\ {\Rightarrow}\mathrm{R}$$

The Function Extensionality Layer

Algorithm

Step I: $\Rightarrow$, $\wedge$ normalization

Step II: =R

Step III: =L

The Function Extensionality Layer

Algorithm

Step I: $\Rightarrow$, $\wedge$ normalization

Step II: =R

Step III: =L

Properties

✓ Soundness

✓ Termination

✗ Completeness

Incompleteness in Presence of ADTs

```
thm :: f:(a → b) → g:(a → b) → {first f == first g ⇒ f == g}
first f (x,y) = (f x,y)
```

!!no pair available!!

$$\dfrac{\dfrac{\dfrac{}{\Sigma,x:a;\Phi,\ \texttt{first f == first g} \vdash \texttt{f x == g x}}}{\Sigma;\Phi,\ \texttt{first f == first g} \vdash \texttt{f == g}}\ {=}\mathrm{R}}{\Sigma;\Phi \vdash \texttt{first f == first g} \Rightarrow \texttt{f == g}}\ {\Rightarrow}\mathrm{R}$$

Incompleteness in Presence of ADTs

```
thm :: f:(a → b) → g:(a → b) → {first f == first g ⇒ f == g}
first f (x,y) = (f x,y)
```

$$\dfrac{\dfrac{\dfrac{\Sigma,\mathtt{x{:}a},\mathtt{y{:}b};\Phi,\ \mathtt{first\ f\ (x,y) == first\ g\ (x,y)} \vdash \mathtt{f\ x == g\ x}}{\Sigma,\mathtt{x{:}a};\Phi,\ \mathtt{first\ f == first\ g} \vdash \mathtt{f\ x == g\ x}}\ {=}\mathrm{L}{+}}{\Sigma;\Phi,\ \mathtt{first\ f == first\ g} \vdash \mathtt{f == g}}\ {=}\mathrm{R}}{\Sigma;\Phi \vdash \mathtt{first\ f == first\ g \Rightarrow f == g}}\ {\Rightarrow}\mathrm{R}$$

Function Extensionality + ADTs

Soundness
(all types has inhabitant)

Termination
(no recursive ADTS)

❓ Completeness

Refinement Types & Function Extensionality

PLE automates **saturated** function equality
For saturation we need extendionality

Approach I: Use extensionality as an axiom

- requires explicit FO proofs
- can be unsound

Approach II: Define an extensionality "layer"

+ smooth interaction with PLE
+ increases automation

? completeness

Thanks!