On the Power of Coercion Abstraction

Julien Cretin Didier Rémy

INRIA

January 26, 2012

Why study coercions?

People have often used similar mechanisms, called coercions or type conversions, to explain non-trivial type system features.

Why study coercions?

People have often used similar mechanisms, called coercions or type conversions, to explain non-trivial type system features.

These techniques have a lot in common, but also differ in some details.
Can we understand them as several instances of the same framework and use it to more easily design new type system features?

Why study coercions?

People have often used similar mechanisms, called coercions or type conversions, to explain non-trivial type system features.

These techniques have a lot in common, but also differ in some details.
Can we understand them as several instances of the same framework and use it to more easily design new type system features?

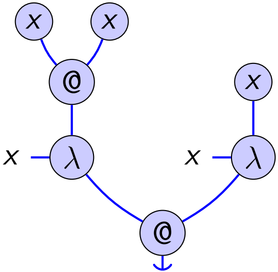
In this work, we restrict to *erasable* coercions (*i.e.* coercions without computational content).

Intuition: Goal

Let's design a type system to type the following untyped lambda term:

$$(\lambda x.x\,x)\,(\lambda x.x)$$

We can graphically represent it bottom-up like that:



Intuition: Typing rules

The type system necessarily gives typing rules for the untyped constructs:

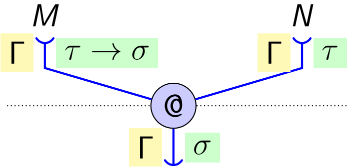
- variable: x
- abstraction: $\lambda x.\mathcal{M}$
- application: $\mathcal{M}\,\mathcal{N}$

We choose *simple types* for illustration.

Intuition: Graphical typing rules

We can annotate the graphical untyped constructs to obtain their graphical typing rule:

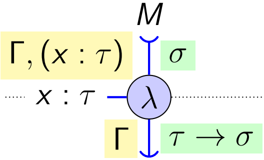
$$\frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash M\ N : \sigma}$$



Intuition: Graphical typing rules

We can annotate the graphical untyped constructs to obtain their graphical typing rule:

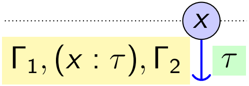
$$\frac{\Gamma, (x : \tau) \vdash M : \sigma}{\Gamma \vdash \lambda(x : \tau)\ M : \tau \to \sigma}$$



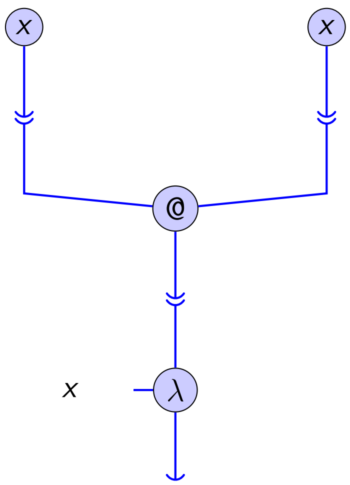
Intuition: Graphical typing rules

We can annotate the graphical untyped constructs to obtain their graphical typing rule:

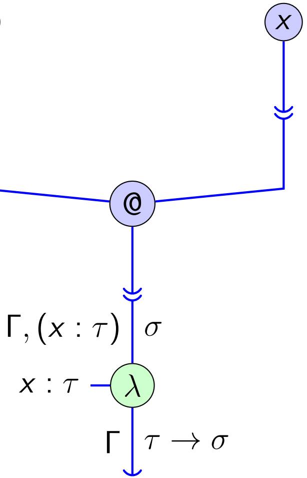
$$\frac{}{\Gamma_1, (x : \tau), \Gamma_2 \vdash x : \tau}$$



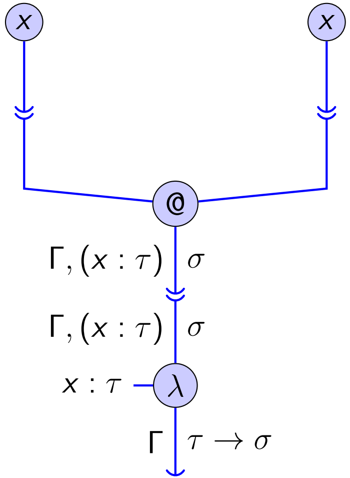
Intuition: Simply-typed lambda calculus



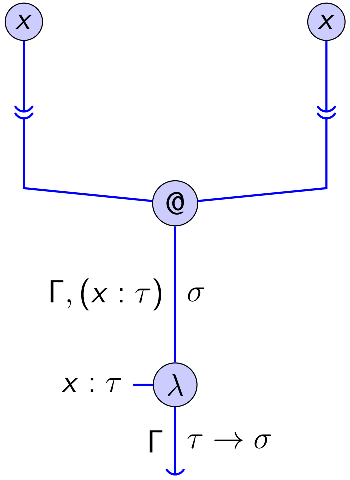
Intuition: Simply-typed lambda calculus



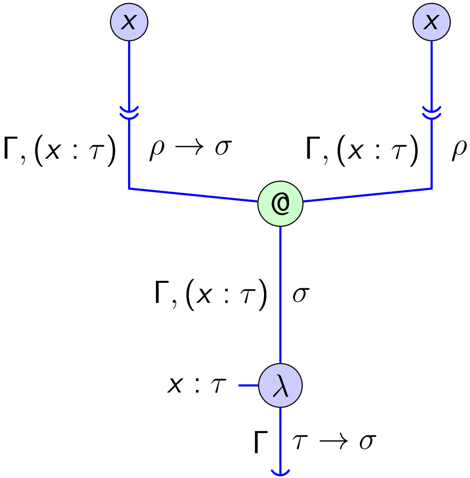
Intuition: Simply-typed lambda calculus



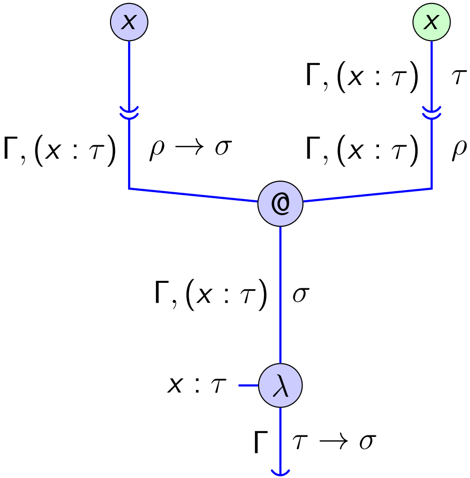
Intuition: Simply-typed lambda calculus



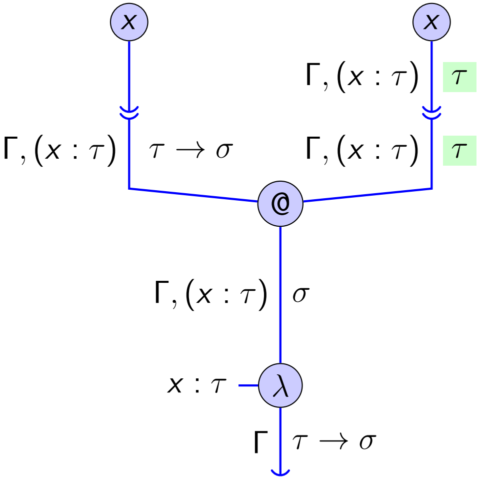
Intuition: Simply-typed lambda calculus



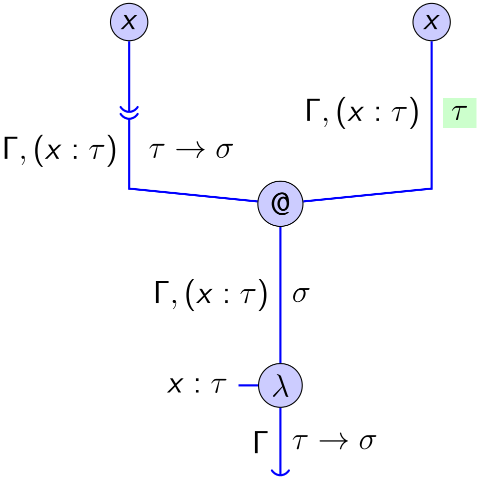
Intuition: Simply-typed lambda calculus



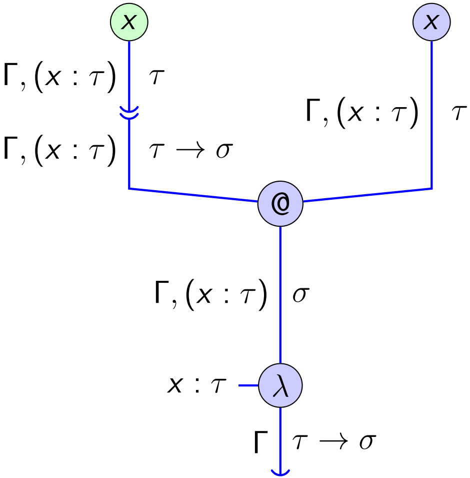
Intuition: Simply-typed lambda calculus



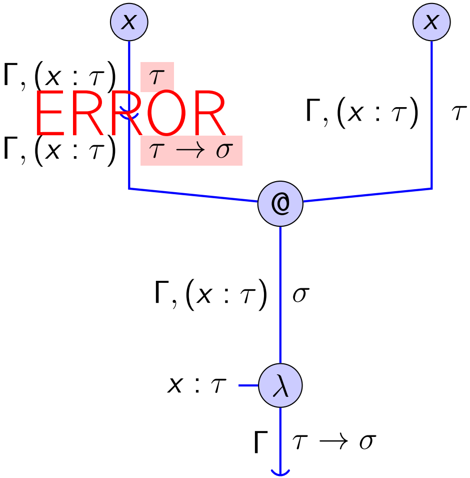
Intuition: Simply-typed lambda calculus



Intuition: Simply-typed lambda calculus



Intuition: Simply-typed lambda calculus



Intuition: Type system features

Terms should be allowed to have several types.

Intuition: Type system features

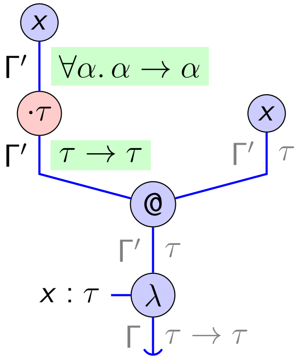
Terms should be allowed to have several types.

Several type system features can represent multiple types:

- intersection types,
- *polymorphism*,
- subtyping, or
- dependent types.

We choose *polymorphism* for illustration.

Intuition: ∀-elim



Polymorphism elimination can be seen as a coercion (which is an erasable type conversion):

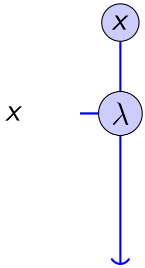
$$\frac{\Gamma' \vdash x : \forall\alpha.\,\alpha \to \alpha}{\Gamma' \vdash x\,\tau : \tau \to \tau}$$

With $\tau \triangleq \forall\alpha.\,\alpha \to \alpha$ and $\Gamma' \triangleq \Gamma, (x : \tau)$.

Intuition: ∀-intro

Polymorphism introduction may extend the environment: so coercions may in fact change the whole typing, not just types!

Type system features are typing conversions.



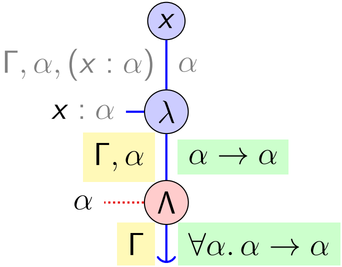
Untyped term:

$$\lambda x.x$$

Intuition: ∀-intro

Polymorphism introduction may extend the environment: so coercions may in fact change the whole typing, not just types!

Type system features are typing conversions.



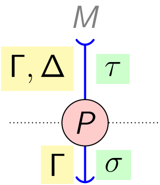
Typing derivation:

$$\dfrac{\dfrac{\Gamma, \alpha, (x : \alpha) \vdash x : \alpha}{\Gamma, \alpha \vdash \lambda(x : \alpha)\, x : \alpha \to \alpha}}{\Gamma \vdash \Lambda\alpha\ \lambda(x : \alpha)\, x : \forall \alpha. \alpha \to \alpha}$$

We can now pass this term to $(\lambda x. x\, x)$ as wanted.

Coercions

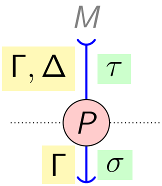
A one-node coercion P, drawn in red, is a one-node *erasable retyping* context.



- *retyping*: $\dfrac{\Gamma, \Delta \vdash M : \tau}{\Gamma \vdash P[M] : \sigma}$ where M and $P[M]$ are explicitly-typed version of the same implicit term.

Coercions

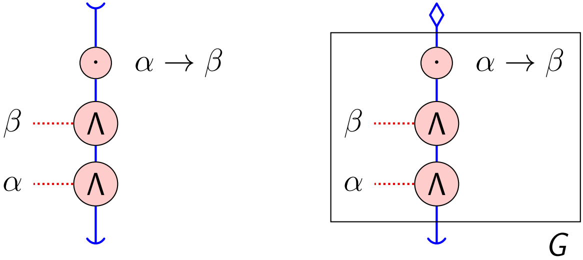
A one-node coercion P, drawn in red, is a one-node *erasable retyping* context.



- *retyping*: $\dfrac{\Gamma, \Delta \vdash M : \tau}{\Gamma \vdash P[M] : \sigma}$ where M and $P[M]$ are explicitly-typed version of the same implicit term.
- *erasable*: P doesn't modify or block the reduction. It is purely static.

Coercions

A coercion G is a sequence of one-node coercions.

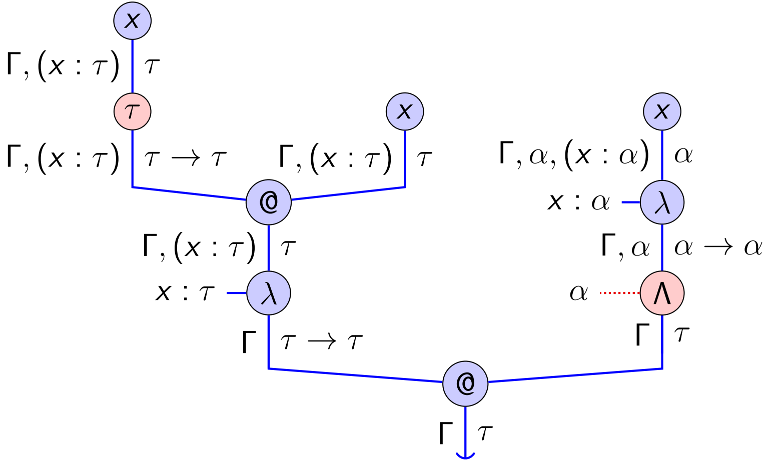


We fill the hole with a diamond:

$$G = \Lambda\alpha\ \Lambda\beta\ \Diamond\ (\alpha \to \beta)$$

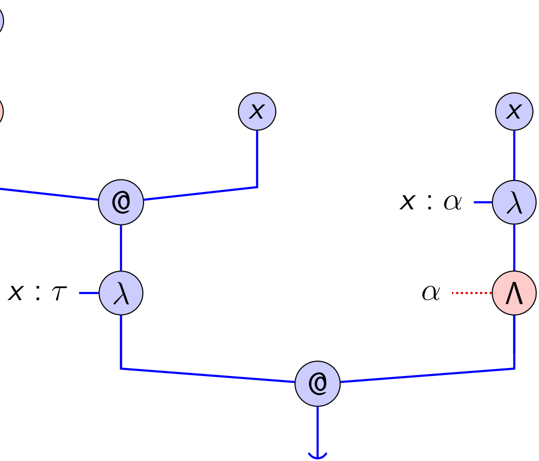
Erasability

The erasing function $\lfloor \cdot \rfloor$ keeps the blue parts and removes *both* the annotations and the red nodes.



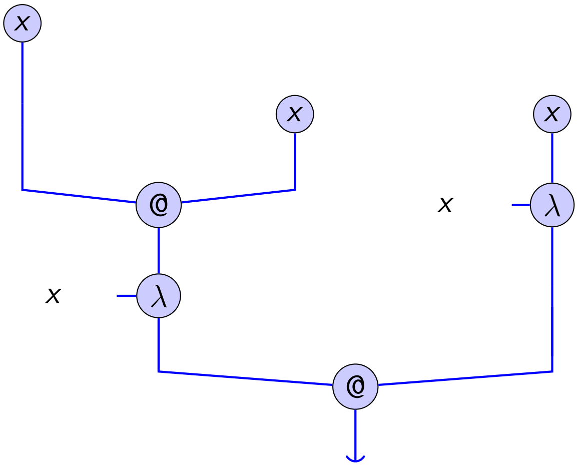
Erasability

The erasing function $\lfloor \cdot \rfloor$ keeps the blue parts and removes *both* the annotations and the red nodes.



Erasability

The erasing function $\lfloor \cdot \rfloor$ keeps the blue parts and removes *both* the annotations and the red nodes.



Bisimulation

The reduction is labelled:

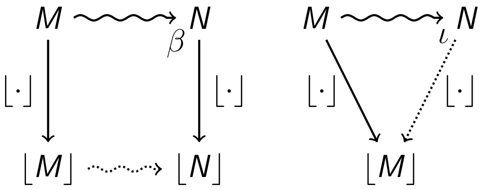
- β-reduction involves only blue nodes
- ι-reduction involves at least one red node

Bisimulation

The reduction is labelled:

- β-reduction involves only blue nodes
- ι-reduction involves at least one red node

We want a bisimulation up to ι-steps:



Forward simulation

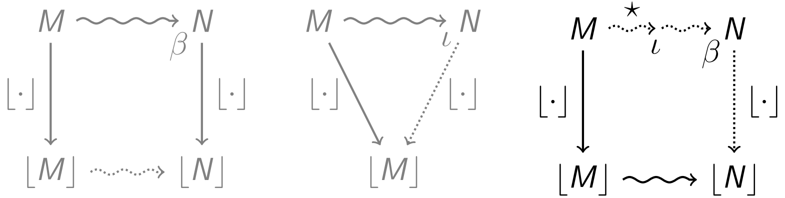
The forward simulation tells that coercions do not contribute to computation.

Bisimulation

The reduction is labelled:

- β-reduction involves only blue nodes
- ι-reduction involves at least one red node

We want a bisimulation up to ι-steps:



Forward simulation

Backward simulation

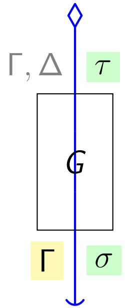
The forward simulation tells that coercions do not contribute to computation.

The backward simulation tells that coercions cannot block the computation. (Thus, values remain values after erasure.)

Coercion judgments

We give the following judgment for coercions:

$$\Gamma \vdash G : \tau \triangleright \sigma$$



System F

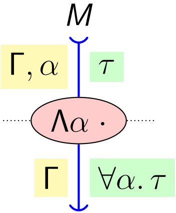
$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau$$

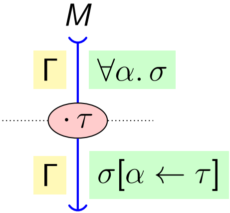
$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N$$

$$\mid \Lambda\alpha\ M \mid M\,\tau$$

$$G ::= \Lambda\alpha\ G \mid G\,\tau$$

Polymorphism: $(\Lambda\alpha\ M)\,\tau \rightsquigarrow_{\iota} M[\alpha \leftarrow \tau]$

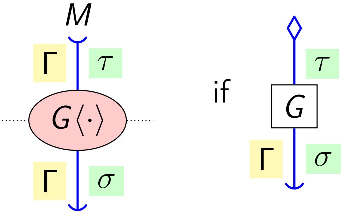




System F_η

$$\begin{aligned} \tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha.\tau \\ M, N &::= x \mid \lambda(x:\tau)\ M \mid M\ N \\ &\quad \mid \Lambda\alpha\ M \mid M\,\tau \mid G\langle M \rangle \\ G &::= \Lambda\alpha\ G \mid G\,\tau \mid G_1\langle G_2 \rangle \end{aligned}$$

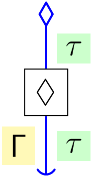
Coercion application: (we want $G\langle M \rangle \rightsquigarrow^{\star}_{\iota} G[\Diamond \leftarrow M]$)



System F_η

$$\begin{aligned} \tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \\ M, N &::= x \mid \lambda(x : \tau)\, M \mid M\, N \\ &\mid\ \Lambda\alpha\, M \mid M\, \tau \mid G\langle M \rangle \\ G &::= \Lambda\alpha\, G \mid G\, \tau \mid G_1 \langle G_2 \rangle \\ &\mid\ \Diamond^{\tau} \end{aligned}$$

Reflexivity: $\Diamond^{\tau}\langle M \rangle \rightsquigarrow_{\iota} M$

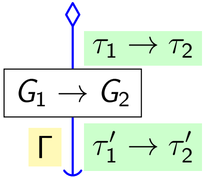


System F_η

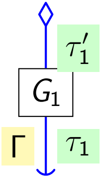
$$
\begin{aligned}
\tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \\
M, N &::= x \mid \lambda(x : \tau)\ M \mid M\ N \\
&\quad\mid\ \Lambda\alpha\ M \ \mid\ M\,\tau \ \mid\ G\langle M \rangle \\
G &::= \Lambda\alpha\ G \ \mid\ G\,\tau \ \mid\ G_1\langle G_2 \rangle \\
&\quad\mid\ \Diamond^\tau \ \mid\ G_1 \xrightarrow{\tau} G_2
\end{aligned}
$$

Arrow congruence (subtyping):

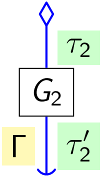
$$(G_1 \xrightarrow{\tau_1'} G_2)\langle \lambda(x : \tau_1)\ M \rangle \rightsquigarrow_\iota \lambda(x : \tau_1')\ G_2\langle M[x \leftarrow G_1\langle x \rangle] \rangle$$



if



and

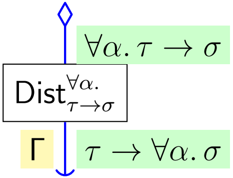


System F_η

$$
\begin{array}{rcl}
\tau, \sigma & ::= & \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \\
M, N & ::= & x \mid \lambda(x : \tau)\ M \mid M\ N \\
 & \mid & \Lambda\alpha\ M \mid M\,\tau \mid G\langle M \rangle \\
G & ::= & \Lambda\alpha\ G \mid G\,\tau \mid G_1\langle G_2 \rangle \\
 & \mid & \Diamond^\tau \mid G_1 \xrightarrow{\tau} G_2 \mid \mathsf{Dist}^{\forall\alpha.}_{\tau\to\sigma}
\end{array}
$$

It permutes $\Lambda\alpha$ and $\lambda(x : \tau)$

$$\mathsf{Dist}^{\forall\alpha.}_{\tau'\to\sigma'}\langle \Lambda\alpha\ \lambda(x : \tau)\ M \rangle \rightsquigarrow_\iota \lambda(x : \tau)\ \Lambda\alpha\ M$$



with $\alpha \notin ftv(\tau)$

System F_η

$$
\begin{aligned}
\tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \\
M, N &::= x \mid \lambda(x : \tau)\, M \mid M\, N \\
&\quad \mid \ \Lambda\alpha\, M \ \mid \ M\,\tau \ \mid \ G\langle M \rangle \\
G &::= \Lambda\alpha\, G \ \mid \ G\,\tau \ \mid \ G_1\langle G_2 \rangle \\
&\quad \mid \ \Diamond^\tau \ \mid \ G_1 \xrightarrow{\tau} G_2 \ \mid \ \mathsf{Dist}^{\forall\alpha.}_{\tau\to\sigma}
\end{aligned}
$$

We now have described F_η (using an explicit variant of Mitchell's presentation).

F_η models subtyping which is at the essence of $\mathsf{F}_{<:}$, but it is not sufficient to model $\mathsf{F}_{<:}$ itself.

We add coercion abstraction for that purpose.

System F_ι

$$
\begin{aligned}
\tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \\
M, N &::= x \mid \lambda(x : \tau)\ M \mid M\ N \\
&\quad \mid \ \Lambda\alpha\ M \ \mid \ M\,\tau \ \mid \ G\langle M \rangle \\
G &::= \Lambda\alpha\ G \ \mid \ G\,\tau \ \mid \ G_1\langle G_2 \rangle \\
&\quad \mid \ \Diamond^\tau \ \mid \ G_1 \xrightarrow{\tau} G_2 \ \mid \ \mathsf{Dist}^{\forall\alpha.}_{\tau\to\sigma}
\end{aligned}
$$

$$\varphi ::= \tau \triangleright \sigma$$

System F_ι

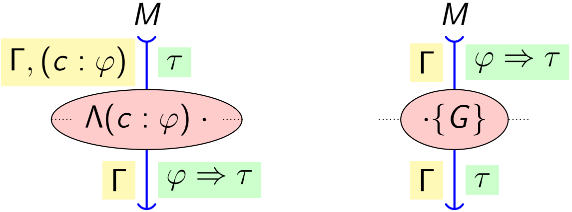
$$
\begin{aligned}
\tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \triangleright \sigma \\
M, N &::= x \mid \lambda (x : \tau)\ M \mid M\ N \\
&\quad \mid \Lambda \alpha\ M \mid M\ \tau \mid G \langle M \rangle \mid \Lambda (c : \varphi)\ M \mid M \{ G \} \\
G &::= \Lambda \alpha\ G \mid G\ \tau \mid G_1 \langle G_2 \rangle \mid \Lambda (c : \varphi)\ G \mid G \{ G' \} \\
&\quad \mid \Diamond^{\tau} \mid G_1 \xrightarrow{\tau} G_2 \mid \mathsf{Dist}^{\forall \alpha.}_{\tau \to \sigma}
\end{aligned}
$$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall\alpha.\tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \triangleright \sigma$$

$$\begin{aligned} M, N &::= x \mid \lambda(x:\tau)\ M \mid M\ N \\ &\mid \Lambda\alpha\ M \mid M\,\tau \mid G\langle M\rangle \mid \Lambda(c:\varphi)\ M \mid M\{G\} \\ G &::= \Lambda\alpha\ G \mid G\,\tau \mid G_1\langle G_2\rangle \mid \Lambda(c:\varphi)\ G \mid G\{G'\} \\ &\mid \Diamond^\tau \mid G_1 \stackrel{\tau}{\to} G_2 \mid \mathrm{Dist}^{\forall\alpha.}_{\tau\to\sigma} \end{aligned}$$

Coercion abstraction: $(\Lambda(c:\varphi)\ M)\{G\} \rightsquigarrow_\iota M[c \leftarrow G]$



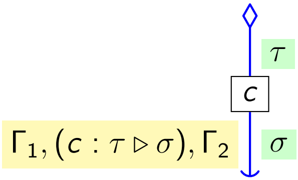
with $\Gamma \vdash G : \varphi$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall\alpha.\tau \mid \varphi \Rightarrow \tau \qquad \varphi ::= \tau \triangleright \sigma$$

$$M, N ::= x \mid \lambda(x:\tau)\ M \mid M\ N$$
$$\mid\ \Lambda\alpha\ M \mid M\ \tau \mid G\langle M\rangle \mid \Lambda(c:\varphi)\ M \mid M\{G\}$$
$$G ::= \Lambda\alpha\ G \mid G\ \tau \mid G_1\langle G_2\rangle \mid \Lambda(c:\varphi)\ G \mid G\{G'\}$$
$$\mid\ \Diamond^{\tau} \mid G_1 \stackrel{\tau}{\to} G_2 \mid \mathrm{Dist}^{\forall\alpha.}_{\tau\to\sigma} \mid c$$

Coercion variable:



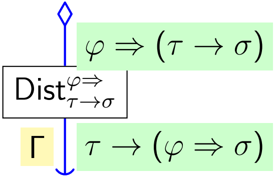
System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad \varphi ::= \tau \rhd \sigma$$

$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N$$
$$\mid\ \Lambda\alpha\ M \mid M\,\tau \mid G\langle M\rangle \mid \Lambda(c : \varphi)\ M \mid M\{G\}$$
$$G ::= \Lambda\alpha\ G \mid G\,\tau \mid G_1\langle G_2\rangle \mid \Lambda(c : \varphi)\ G \mid G\{G'\}$$
$$\mid\ \Diamond^\tau \mid G_1 \xrightarrow{\tau} G_2 \mid \mathrm{Dist}^{\forall\alpha.}_{\tau\to\sigma} \mid c \mid \mathrm{Dist}^{\varphi\Rightarrow}_{\tau\to\sigma}$$

It permutes $\Lambda(c : \varphi)$ and $\lambda(x : \tau)$

$$\mathrm{Dist}^{\varphi'\Rightarrow}_{\tau'\to\sigma'}\langle\Lambda(c : \varphi)\ \lambda(x : \tau)\ M\rangle \leadsto_\iota \lambda(x : \tau)\ \Lambda(c : \varphi)\ M$$



Properties of F_ι

F_ι is well-behaved: it satisfies preservation, progress, confluence, and normalization.

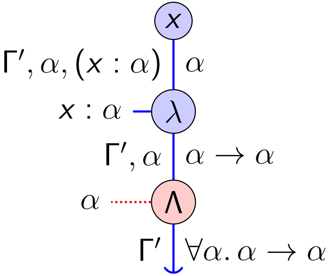
Properties of F_ι

F_ι is well-behaved: it satisfies preservation, progress, confluence, and normalization.

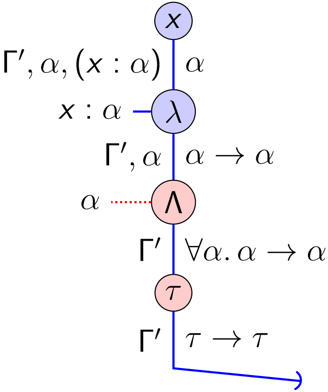
However, it is not a coercion language: it obeys the forward simulation but not the backward simulation.

The backward simulation is necessary for values to correspond before and after erasure: *types should not block the computation.*

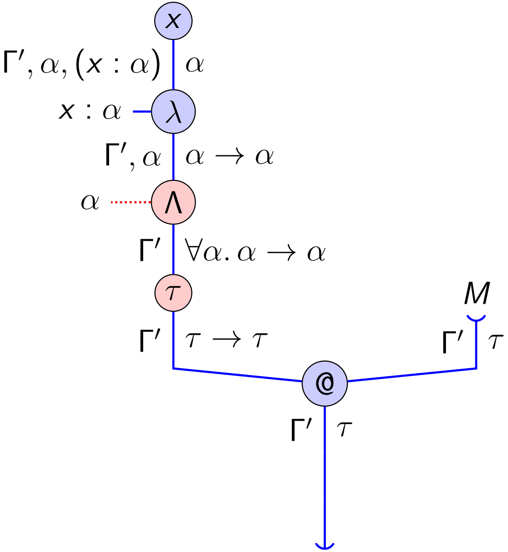
Losing backward simulation



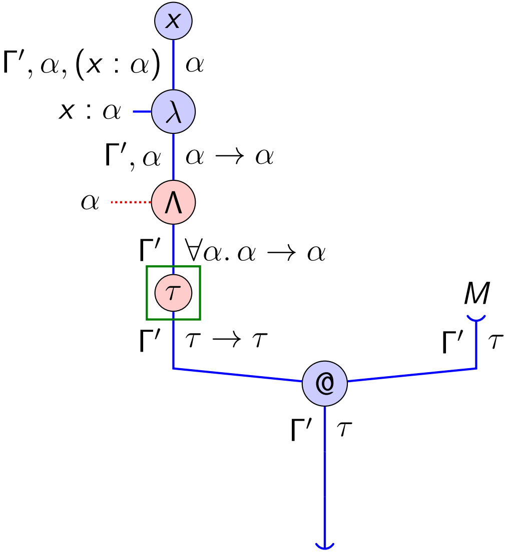
Losing backward simulation



Losing backward simulation

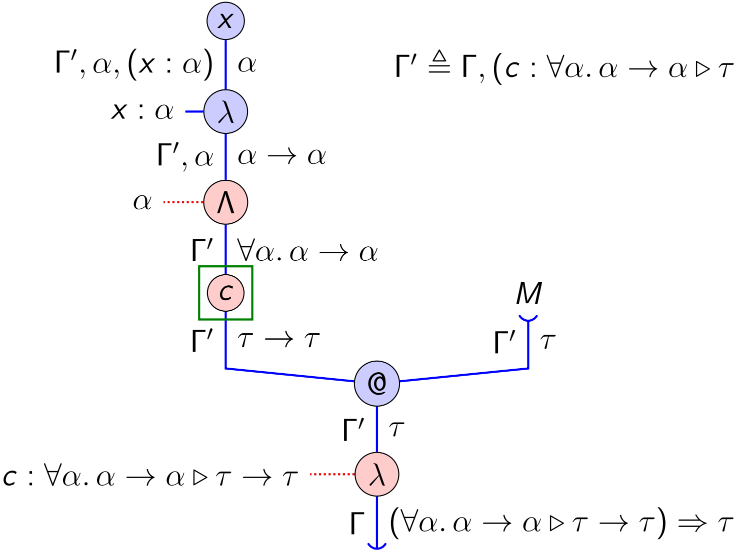


Losing backward simulation



Losing backward simulation

$\Gamma' \triangleq \Gamma, (c : \forall\alpha.\, \alpha \to \alpha \triangleright \tau \to \tau)$



A default solution

One solution is to use weak reduction and value restriction on coercion abstraction.

However, it delays error detection. We could type any pure lambda term by abstracting over an incoherent set of coercions like $U \triangleright (U \rightarrow U)$ and $(U \rightarrow U) \triangleright U$.

System F_ι^p

MLF and $F_{<:}$ have some coercion abstraction because of bounded polymorphism.

System F_ι^p

MLF and $F_{<:}$ have some coercion abstraction because of bounded polymorphism.

$F_{<:}$	MLF
$\Lambda(\alpha \leq \tau)M$	$\Lambda(\alpha \geq \tau)M$

System F_ι^p

MLF and $F_{<:}$ have some coercion abstraction because of bounded polymorphism.

$F_{<:}$	MLF
$\Lambda(\alpha \leq \tau)M$	$\Lambda(\alpha \geq \tau)M$
$\Lambda\alpha\ \Lambda(c : \alpha \triangleright \tau)\ M$	$\Lambda\alpha\ \Lambda(c : \tau \triangleright \alpha)\ M$

System F_ι^p

MLF and $F_{<:}$ have some coercion abstraction because of bounded polymorphism.

$F_{<:}$	MLF
$\Lambda(\alpha \leq \tau) M$	$\Lambda(\alpha \geq \tau) M$
$\Lambda\alpha\ \Lambda(c : \alpha \triangleright \tau)\ M$	$\Lambda\alpha\ \Lambda(c : \tau \triangleright \alpha)\ M$
$\Lambda(\alpha \triangleright c : \tau)\ M$	$\Lambda(\alpha \triangleleft c : \tau)\ M$

From F_ι, we replace unrestricted coercion abstraction with these two features and call the result F_ι^p. We gain backward simulation and the previous example is ill-formed.

F_ι^p is a coercion language (soundness, normalization, confluence, bisimulation with its erasure).

Result: F_ι^p subsumes $F_{<:}$, F_η, and MLF

		Languages				
		F				
Features	$\forall^=$	$\surd$				

- $\forall^=$ is simple polymorphism

Result: F_ι^p subsumes $F_{<:}$, F_η, and MLF

		Languages				
		F	F_η			
Features	$\forall^=$	$\surd$	$\surd$			
	$\xrightarrow{\eta}$		$\surd$			

- $\forall^=$ is simple polymorphism
- $\xrightarrow{\eta}$ is subtyping i.e. the η-expansion for arrow

Result: F_ι^p subsumes $F_{<:}$, F_η, and MLF

		Languages				
		F	F_η	MLF		
Features	$\forall^=$	$\surd$	$\surd$	$\surd$		
	$\xrightarrow{\eta}$		$\surd$			
	$\forall^\geq$			$\surd$		

- $\forall^=$ is simple polymorphism
- $\xrightarrow{\eta}$ is subtyping i.e. the η-expansion for arrow
- $\forall^\geq$ is lower bounded polymorphism (includes $\forall^=$)

Result: F_ι^p subsumes $F_{<:}$, F_η, and MLF

		Languages				
		F	F_η	MLF	$F_{<:}$	
Features	$\forall^=$	$\surd$	$\surd$	$\surd$	$\surd$	
	$\xrightarrow{\eta}$		$\surd$		$\surd$	
	$\forall^\geq$			$\surd$		
	$\forall^\leq$				$\surd$	

- $\forall^=$ is simple polymorphism
- $\xrightarrow{\eta}$ is subtyping i.e. the η-expansion for arrow
- $\forall^\geq$ is lower bounded polymorphism (includes $\forall^=$)
- $\forall^\leq$ is upper bounded polymorphism (includes $\forall^=$)

Result: F_ι^p subsumes $F_{<:}$, F_η, and MLF

	Languages				
Features	F	F_η	MLF	$F^+_{<:}$	
$\forall^=$	$\surd$	$\surd$	$\surd$	$\surd$	
$\xrightarrow{\eta}$		$\surd$		$\surd$	
$\forall^\geq$			$\surd$		
$\forall^\leq$				$\surd$	

- $\forall^=$ is simple polymorphism
- $\xrightarrow{\eta}$ is subtyping i.e. the η-expansion for arrow
- $\forall^\geq$ is lower bounded polymorphism (includes $\forall^=$)
- $\forall^\leq$ is upper bounded polymorphism (includes $\forall^=$)

$F^+_{<:}$, the combination of $\forall^\leq$ and $\xrightarrow{\eta}$, also contains deep instantiation and distributivity which are absent from $F_{<:}$.

Result: F_ι^p subsumes $F_{<:}$, F_η, and MLF

		Languages				
		F	F_η	MLF	$F_{<:}^+$	F_ι^p
Features	$\forall^=$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
	$\xrightarrow{\eta}$		$\surd$		$\surd$	$\surd$
	$\forall^\geq$			$\surd$		$\surd$
	$\forall^\leq$				$\surd$	$\surd$

- $\forall^=$ is simple polymorphism
- $\xrightarrow{\eta}$ is subtyping i.e. the η-expansion for arrow
- $\forall^\geq$ is lower bounded polymorphism (includes $\forall^=$)
- $\forall^\leq$ is upper bounded polymorphism (includes $\forall^=$)

$F_{<:}^+$, the combination of $\forall^\leq$ and $\xrightarrow{\eta}$, also contains deep instantiation and distributivity which are absent from $F_{<:}$.

Future work

- See if other type system features can be expressed as coercions:
  - recursive types
  - intersection types
  - existential types
  - linear types
  - type operators
  - dependent types, etc.

Future work

- See if other type system features can be expressed as coercions:
  - recursive types
  - intersection types
  - existential types
  - linear types
  - type operators
  - dependent types, etc.
- A coercion abstraction less restricted than bounded polymorphism.

Future work

- See if other type system features can be expressed as coercions:
  - recursive types
  - intersection types
  - existential types
  - linear types
  - type operators
  - dependent types, etc.
- A coercion abstraction less restricted than bounded polymorphism.
- Looking at non erasable coercions.

Future work

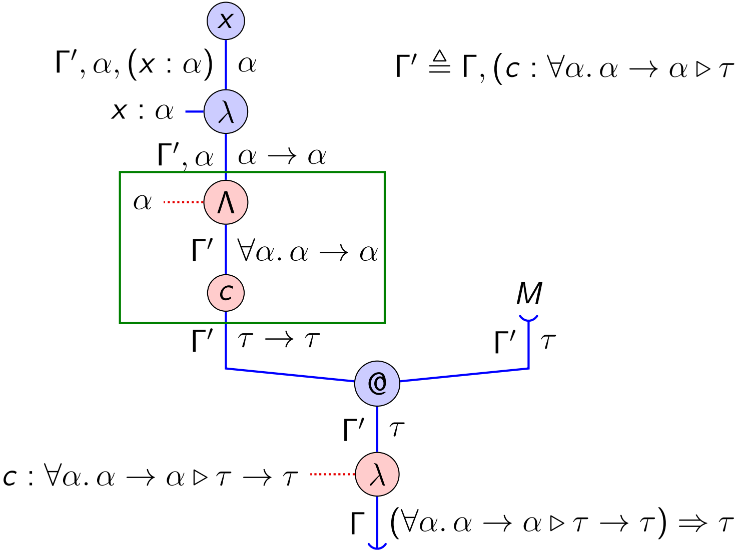
- See if other type system features can be expressed as coercions:
  - recursive types
  - intersection types
  - existential types
  - linear types
  - type operators
  - dependent types, etc.
- A coercion abstraction less restricted than bounded polymorphism.
- Looking at non erasable coercions.

Thank you!

Extra slides

Push

$\Gamma' \triangleq \Gamma, (c : \forall\alpha.\, \alpha \to \alpha \triangleright \tau \to \tau)$



Push

RedPushArrow

$$G\langle\lambda(x:\tau)\ M\rangle\ N \rightsquigarrow_\iota$$
$$(\lambda(x:\tau')\ (\mathtt{Right}\ G)\langle M[x \leftarrow (\mathtt{Left}\ G)\langle x\rangle]\rangle)\ N$$

RedLeftArrow

$$\mathtt{Left}\,(G_1 \xrightarrow{\tau} G_2) \rightsquigarrow_\iota G_1$$

RedRightArrow

$$\mathtt{Right}\,(G_1 \xrightarrow{\tau} G_2) \rightsquigarrow_\iota G_2$$

$$\Lambda(c_{app} : U \triangleright (U \rightarrow U))\,\Lambda(c_{lam} : (U \rightarrow U) \triangleright U)\,M$$

System $F_{<:}$

Orthogonal features should easily and fully compose.
When combining upper bounded polymorphism and subtyping we naturally get an *intermediate language* more expressive than the most expressive version of $F_{<:}$.

$$\frac{\qquad\qquad \qquad \Gamma, \alpha <: \tau' \vdash \sigma <: \sigma'}{\Gamma \vdash \forall(\alpha <: \tau)\ \sigma <: \forall(\alpha <: \tau')\ \sigma'}$$

Depending on the variant, the first premise may be:

Kernel-Fsub
$\tau' = \tau$

System $F_{<:}$

Orthogonal features should easily and fully compose.
When combining upper bounded polymorphism and subtyping we naturally get an *intermediate language* more expressive than the most expressive version of $F_{<:}$.

$$\frac{\qquad\qquad \quad \Gamma, \alpha <: \tau' \vdash \sigma <: \sigma'}{\Gamma \vdash \forall(\alpha <: \tau)\ \sigma <: \forall(\alpha <: \tau')\ \sigma'}$$

Depending on the variant, the first premise may be:

Kernel-Fsub
$\tau' = \tau$

Full-Fsub
$\Gamma \vdash \tau' <: \tau$

System $\mathsf{F}_{<:}$

Orthogonal features should easily and fully compose.
When combining upper bounded polymorphism and subtyping we naturally get an *intermediate language* more expressive than the most expressive version of $\mathsf{F}_{<:}$.

$$\frac{\qquad\qquad \quad \Gamma, \alpha <: \tau' \vdash \sigma <: \sigma'}{\Gamma \vdash \forall(\alpha <: \tau)\, \sigma <: \forall(\alpha <: \tau')\, \sigma'}$$

Depending on the variant, the first premise may be:

Kernel-Fsub
$\tau' = \tau$

Full-Fsub
$\Gamma \vdash \tau' <: \tau$

F-Bounded
$\Gamma, \alpha <: \tau' \vdash \alpha <: \tau$

System $F_{<:}$

Orthogonal features should easily and fully compose.
When combining upper bounded polymorphism and subtyping we naturally get an *intermediate language* more expressive than the most expressive version of $F_{<:}$.

The typing rule of $F_{\mu<:}$ is derivable in F_ι^p using the following typing rules (absent from $F_{\mu<:}$):

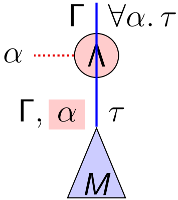
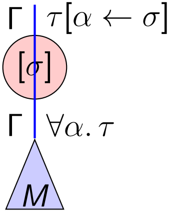
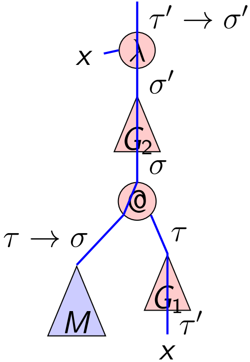
$$\frac{\Gamma, (\alpha \triangleright c : \tau) \vdash G : \rho \triangleright \sigma \qquad \Gamma \vdash \rho}{\Gamma \vdash \lambda(\alpha \triangleright c : \tau)\, G : \rho \triangleright \forall(\alpha \triangleright \tau) \Rightarrow \sigma}$$

$$\frac{\Gamma \vdash G : \rho \triangleright \forall(\alpha \triangleright \tau) \Rightarrow \tau' \qquad \Gamma \vdash G' : \sigma \triangleright \tau[\alpha \leftarrow \sigma]}{\Gamma \vdash G\{\sigma \triangleright G'\} : \rho \triangleright \tau'[\alpha \leftarrow \sigma]}$$

Full distrib

$$\dfrac{\dfrac{\dfrac{\dfrac{\alpha \vdash \Diamond\,\alpha : \forall\alpha.\,\tau \triangleright \tau}{\alpha \vdash (\Diamond\,\alpha) \to \boxed{\Diamond} : \tau \to \sigma \triangleright (\forall\alpha.\,\tau) \to \sigma}}{\alpha \vdash ((\Diamond\,\alpha) \to \Diamond)\langle \boxed{\Diamond\,\alpha} \rangle : \forall\alpha.\,\tau \to \sigma \triangleright (\forall\alpha.\,\tau) \to \sigma}}{\vdash \boxed{\Lambda\alpha}\,((\Diamond\,\alpha) \to \Diamond)\langle \Diamond\,\alpha \rangle : \forall\alpha.\,\tau \to \sigma \triangleright \forall\alpha.\,(\forall\alpha.\,\tau) \to \sigma}}{\vdash \boxed{\mathsf{Dist}}\,\langle \Lambda\alpha\,((\Diamond\,\alpha) \to \Diamond)\langle \Diamond\,\alpha \rangle \rangle : \forall\alpha.\,\tau \to \sigma \triangleright (\forall\alpha.\,\tau) \to \forall\alpha.\,\sigma}$$

System F_η examples

<table>
<tr><th>generalization</th><th>instantiation</th><th>η-expansion</th></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
<tr><td>$\Lambda\alpha\ M$</td><td>$M\,\sigma$</td><td>$\lambda(x : \tau')\ G_2[M\ (G_1[x])]$</td></tr>
</table>

Pure Lambda Calculus

$$\begin{array}{rcll} x, y & & & \text{Variables} \\ \mathcal{M} & ::= & x \mid \lambda x.\mathcal{M} \mid \mathcal{M}\,\mathcal{M} & \text{Terms} \\ \mathcal{C} & ::= & \lambda x.[] \mid []\,\mathcal{M} \mid \mathcal{M}\,[] & \text{Reduction contexts} \end{array}$$

RedContext

$$\frac{\mathcal{M} \rightsquigarrow \mathcal{M}'}{\mathcal{C}[\mathcal{M}] \rightsquigarrow \mathcal{C}[\mathcal{M}']}$$

RedBeta

$$(\lambda x.\mathcal{M})\,\mathcal{M}' \rightsquigarrow \mathcal{M}[x \leftarrow \mathcal{M}']$$

Simply-typed lambda calculus

$$
\begin{array}{rcll}
x, y & & & \text{Term variables} \\
\tau, \sigma & ::= & \tau \to \sigma & \text{Types} \\
M, N & ::= & x \mid \lambda(x:\tau)\ M \mid M\ N & \text{Terms} \\
C & ::= & \lambda(x:\tau)\ [] \mid []\ M \mid M\ [] & \text{Reduction contexts}
\end{array}
$$

TermVar

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}$$

TermTermLam

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda(x : \tau)\ M : \tau \to \sigma}$$

TermTermApp

$$\frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash M\ N : \sigma}$$

RedContextBeta

$$\frac{M \rightsquigarrow_\beta N}{C[M] \rightsquigarrow_\beta C[N]}$$

RedTerm

$$(\lambda(x : \tau)\ M)\ N \rightsquigarrow_\beta M[x \leftarrow N]$$

System F: Polymorphism as coercions

The necessary simply-typed lambda calculus is in grey.

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \qquad \text{Types}$$

$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N \mid P[M] \qquad \text{Terms}$$

$$P ::= \Lambda\alpha\ [] \mid []\ \tau \qquad \text{One-node coercions}$$

TermTypeLam

$$\frac{\Gamma, \alpha \vdash M : \tau}{\Gamma \vdash \Lambda\alpha\ M : \forall \alpha . \tau}$$

TermTypeApp

$$\frac{\Gamma \vdash M : \forall \alpha . \tau \qquad \Gamma \vdash \sigma}{\Gamma \vdash M\,\sigma : \tau[\alpha \leftarrow \sigma]}$$

RedType

$$(\Lambda\alpha\ M)\,\tau \leadsto_{\iota} M[\alpha \leftarrow \tau]$$

System F: Polymorphism as coercions

$$
\begin{array}{rcll}
\alpha, \beta & & & \text{Type variables} \\
\tau, \sigma & ::= & \ldots \mid \alpha \mid \forall \alpha . \tau & \text{Types} \\
M, N & ::= & \ldots \mid P[M] & \text{Terms} \\
P & ::= & \Lambda\alpha\,[] \mid []\,\tau & \text{Coercion contexts} \\
C & ::= & \ldots \mid P & \text{Reduction contexts}
\end{array}
$$

TermTypeLam

$$\frac{\Gamma, \alpha \vdash M : \tau}{\Gamma \vdash \Lambda\alpha\; M : \forall \alpha .\, \tau}$$

TermTypeApp

$$\frac{\Gamma \vdash M : \forall \alpha .\, \tau}{\Gamma \vdash M\, \sigma : \tau[\alpha \leftarrow \sigma]}$$

RedContextIota

$$\frac{M \rightsquigarrow_\iota N}{C[M] \rightsquigarrow_\iota C[N]}$$

RedType

$$(\Lambda\alpha\; M)\, \tau \rightsquigarrow_\iota M[\alpha \leftarrow \tau]$$

System F_η: Subtyping as coercions

System F_η is the closure of System F by η-reduction.

$$\frac{\Gamma \vdash \mathcal{M} : \tau \qquad \mathcal{M} \leadsto_\eta \mathcal{M}'}{\Gamma \vdash \mathcal{M}' : \tau}$$

System F_η: Subtyping as coercions

System F_η is the closure of System F by η-reduction.

$$\frac{\Gamma \vdash \mathcal{M} : \tau \qquad \mathcal{M} \rightsquigarrow_\eta \mathcal{M}'}{\Gamma \vdash \mathcal{M}' : \tau}$$

There are two presentations of F_η with coercions:

- A lambda-term version: the one we have seen so far, where judgments are $\Gamma \vdash G : (\Delta \cdot \tau) \triangleright \sigma$.
  The syntax is simple but typing is involved because coercions may bind.
- A proof-term version where judgments take the form $\Gamma \vdash G : \tau \triangleright \sigma$.
  Typing is simpler but the coercion constructs are less atomic and numerous.

We chose a mix presentation to get the best of both.

System F_ι^p

$$
\begin{array}{rcll}
c & & & \text{Coercion variables} \\
\diamondvert & ::= & \triangleleft \mid \triangleright & \text{Bounds} \\
\tau, \sigma & ::= & \ldots \mid \forall(\alpha \diamondvert \tau) \Rightarrow \sigma & \text{Types} \\
P & ::= & \ldots \mid \lambda(\alpha \diamondvert c : \tau)\, M \mid M\{\tau \diamondvert G\} & \text{One-node coercions} \\
G & ::= & \ldots \mid \mathrm{Dist}_{\tau \to \sigma}^{\forall \alpha \diamondvert \rho \Rightarrow} & \text{Coercions}
\end{array}
$$

TermTCoerLam

$$
\frac{\Gamma, \alpha \diamondvert c : \tau \vdash M : \sigma}{\Gamma \vdash \lambda(\alpha \diamondvert c : \tau)\, M : \forall(\alpha \diamondvert \tau) \Rightarrow \sigma}
$$

TermTCoerApp

$$
\frac{\Gamma \vdash M : \forall(\alpha \diamondvert \tau) \Rightarrow \tau' \qquad \Gamma \vdash G : \sigma \diamondvert \tau[\alpha \leftarrow \sigma]}{\Gamma \vdash M\{\sigma \diamondvert G\} : \tau'[\alpha \leftarrow \sigma]}
$$

RedCoer

$$
(\lambda(\alpha \diamondvert c : \tau)\, M)\{\sigma \diamondvert G\} \leadsto_\iota M[\alpha \leftarrow \sigma][c \leftarrow G]
$$

System F_ι^p

$$
\begin{array}{rcll}
c & & & \text{Coercion variables} \\
\Diamond & ::= & \triangleleft \mid \triangleright & \text{Bounds} \\
\tau, \sigma & ::= & \ldots \mid \forall(\alpha \Diamond \tau) \Rightarrow \sigma & \text{Types} \\
P & ::= & \ldots \mid \lambda(\alpha \Diamond c : \tau)\, M \mid M\{\tau \Diamond G\} & \text{One-node coercions} \\
G & ::= & \ldots \mid \mathrm{Dist}_{\tau \to \sigma}^{\forall \alpha \Diamond \rho \Rightarrow} & \text{Coercions}
\end{array}
$$

CoerDistTCoerArrow

$$
\frac{\Gamma \vdash \tau \qquad \Gamma, \alpha \vdash \rho \qquad \Gamma, \alpha \vdash \sigma}{\Gamma \vdash \mathrm{Dist}_{\tau \to \sigma}^{\forall \alpha \Diamond \rho \Rightarrow} : (\forall(\alpha \Diamond \rho) \Rightarrow \tau \to \sigma) \triangleright (\tau \to \forall(\alpha \Diamond \rho) \Rightarrow \sigma)}
$$

RedCoerDistCoerArrow

$$
\mathrm{Dist}_{\tau' \to \sigma'}^{\forall \alpha \Diamond \rho' \Rightarrow} \langle \lambda(\alpha \Diamond c : \rho)\, \lambda(x : \tau)\ M \rangle \rightsquigarrow_\iota \lambda(x : \tau)\, \lambda(\alpha \Diamond c : \rho)\, M
$$

Erasing function

The erasing function removes type annotations, abstractions, and applications.

$$\begin{aligned} \lfloor x \rfloor &= x \\ \lfloor \lambda(x : \tau)\ M \rfloor &= \lambda x . \lfloor M \rfloor \\ \lfloor M\ N \rfloor &= \lfloor M \rfloor\ \lfloor N \rfloor \\ \lfloor P[M] \rfloor &= \lfloor M \rfloor \end{aligned}$$

Erasing function

The erasing function removes type annotations, abstractions, and applications.

$$\lfloor x \rfloor = x$$
$$\lfloor \lambda(x : \tau)\ M \rfloor = \lambda x. \lfloor M \rfloor$$
$$\lfloor M\ N \rfloor = \lfloor M \rfloor\ \lfloor N \rfloor$$
$$\lfloor P[M] \rfloor = \lfloor M \rfloor$$

The unfolding of the last line is:

$$\lfloor \Lambda\alpha\ M \rfloor = \lfloor M \rfloor$$
$$\lfloor M\, \sigma \rfloor = \lfloor M \rfloor$$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall \alpha. \tau \mid \varphi \Rightarrow \tau \qquad \varphi ::= \tau \triangleright \sigma$$

$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N$$
$$\mid \Lambda\alpha\ M \mid M\ \tau$$
$$G ::= \Lambda\alpha\ G \mid G\ \tau$$

Polymorphism:

TermTypeLam
$$\frac{\Gamma, \alpha \vdash M : \tau}{\Gamma \vdash \Lambda\alpha\ M : \forall \alpha. \tau}$$

TermTypeApp
$$\frac{\Gamma \vdash M : \forall \alpha. \tau \qquad \Gamma \vdash \sigma}{\Gamma \vdash M\ \sigma : \tau[\alpha \leftarrow \sigma]}$$

RedType
$$(\Lambda\alpha\ M)\ \tau \rightsquigarrow_{\iota} M[\alpha \leftarrow \tau]$$

System F_ι

$$\begin{aligned} \tau, \sigma \; &::= \; \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \triangleright \sigma \\ M, N \; &::= \; x \mid \lambda (x : \tau) \; M \mid M \; N \\ &\mid \; \Lambda \alpha \; M \mid M \, \tau \mid G \langle M \rangle \\ G \; &::= \; \Lambda \alpha \; G \mid G \, \tau \mid G_1 \langle G_2 \rangle \end{aligned}$$

Coercion application:

TermCoer

$$\frac{\Gamma \vdash G : \tau \triangleright \sigma \qquad \Gamma \vdash M : \tau}{\Gamma \vdash G \langle M \rangle : \sigma}$$

System F_ι

$$
\begin{aligned}
\tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \rhd \sigma \\
M, N &::= x \mid \lambda(x : \tau)\ M \mid M\ N \\
&\quad\mid\ \Lambda\alpha\ M \ \mid\ M\,\tau \ \mid\ G\langle M \rangle \\
G &::= \Lambda\alpha\ G \ \mid\ G\,\tau \ \mid\ G_1\langle G_2 \rangle \\
&\quad\mid\ \Diamond^\tau
\end{aligned}
$$

Reflexivity:

CoerDot

$$\frac{\Gamma \vdash \tau}{\Gamma \vdash \Diamond^\tau : \tau \rhd \tau}$$

RedCoerDot

$$\Diamond^\tau \langle M \rangle \leadsto_\iota M$$

System F_ι

$$
\begin{aligned}
\tau, \sigma \; &::= \; \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \triangleright \sigma \\
M, N \; &::= \; x \mid \lambda(x:\tau)\, M \mid M\, N \\
&\quad \mid \; \Lambda\alpha\, M \mid M\, \tau \mid G\langle M \rangle \\
G \; &::= \; \Lambda\alpha\, G \mid G\, \tau \mid G_1\langle G_2 \rangle \\
&\quad \mid \; \Diamond^{\tau}
\end{aligned}
$$

One-node coercion injection:

P on M

$$\frac{\Gamma, \Delta \vdash M : \tau}{\Gamma \vdash P[M] : \sigma}$$

P on G

$$\frac{\Gamma, \Delta \vdash G : \rho \triangleright \tau \qquad \Gamma \vdash \rho}{\Gamma \vdash P[G] : \rho \triangleright \sigma}$$

RedCoerFill

$$(P[G])\langle M \rangle \rightsquigarrow_{\iota} P[G\langle M \rangle]$$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad \varphi ::= \tau \triangleright \sigma$$

$$M, N ::= x \mid \lambda (x : \tau)\ M \mid M\ N \mid \Lambda \alpha\ M \mid M\ \tau \mid G \langle M \rangle$$

$$G ::= \Lambda \alpha\ G \mid G\ \tau \mid G_1 \langle G_2 \rangle \mid \Diamond^\tau \mid G_1 \xrightarrow{\tau} G_2$$

Arrow congruence (subtyping):

CoerArrow

$$\frac{\Gamma \vdash G_1 : \tau_1 \triangleright \tau_1' \qquad \Gamma \vdash G_2 : \tau_2 \triangleright \tau_2'}{\Gamma \vdash G_1 \xrightarrow{\tau_1} G_2 : (\tau_1' \to \tau_2) \triangleright (\tau_1 \to \tau_2')}$$

RedCoerArrow

$$(G_1 \xrightarrow{\tau_1} G_2) \langle \lambda (x : \tau_1')\ M \rangle \leadsto_\iota \lambda (x : \tau_1)\ G_2 \langle M [x \leftarrow G_1 \langle x \rangle] \rangle$$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall\alpha.\tau \mid \varphi \Rightarrow \tau \qquad \varphi ::= \tau \triangleright \sigma$$

$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N$$
$$\mid \Lambda\alpha\ M \mid M\,\tau \mid G\langle M\rangle$$
$$G ::= \Lambda\alpha\ G \mid G\,\tau \mid G_1\langle G_2\rangle$$
$$\mid \Diamond^\tau \mid G_1 \xrightarrow{\tau} G_2 \mid \mathsf{Dist}^{\forall\alpha.}_{\tau\to\sigma}$$

It permutes $\Lambda\alpha$ and $\lambda(x : \tau)$

CoerDistTypeArrow

$$\frac{\Gamma \vdash \tau \quad (\textit{i.e. } \alpha \notin \mathit{ftv}(\tau)) \qquad \Gamma, \alpha \vdash \sigma}{\Gamma \vdash \mathsf{Dist}^{\forall\alpha.}_{\tau\to\sigma} : (\forall\alpha.\,\tau \to \sigma) \triangleright (\tau \to \forall\alpha.\,\sigma)}$$

RedCoerDistTypeArrow

$$\mathsf{Dist}^{\forall\alpha.}_{\tau'\to\sigma'}\langle\Lambda\alpha\ \lambda(x : \tau)\ M\rangle \rightsquigarrow_\iota \lambda(x : \tau)\ \Lambda\alpha\ M$$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad \varphi ::= \tau \triangleright \sigma$$

$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N$$
$$\mid\ \Lambda \alpha\ M \mid M\ \tau \mid G\langle M \rangle \mid \Lambda(c : \varphi)\ M \mid M\{G\}$$
$$G ::= \Lambda \alpha\ G \mid G\ \tau \mid G_1 \langle G_2 \rangle \mid \Lambda(c : \varphi)\ G \mid G\{G'\}$$
$$\mid\ \Diamond^\tau \mid G_1 \xrightarrow{\tau} G_2 \mid \mathsf{Dist}^{\forall \alpha.}_{\tau \to \sigma}$$

Coercion abstraction:

TermCoerLam
$$\frac{\Gamma, (c : \varphi) \vdash M : \tau}{\Gamma \vdash \Lambda(c : \varphi)\ M : \varphi \Rightarrow \tau}$$

TermCoerApp
$$\frac{\Gamma \vdash G : \varphi \qquad \Gamma \vdash M : \varphi \Rightarrow \tau}{\Gamma \vdash M\{G\} : \tau}$$

RedCoer
$$(\lambda(c : \varphi)\ M)\{G\} \rightsquigarrow_\iota M[c \leftarrow G]$$

System F_ι

$$
\begin{aligned}
\tau, \sigma &::= \tau \to \sigma \mid \alpha \mid \forall \alpha . \tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \triangleright \sigma \\
M, N &::= x \mid \lambda (x : \tau)\ M \mid M\ N \\
&\quad \mid\ \Lambda \alpha\ M \mid M\ \tau \mid G \langle M \rangle \mid \Lambda (c : \varphi)\ M \mid M \{ G \} \\
G &::= \Lambda \alpha\ G \mid G\ \tau \mid G_1 \langle G_2 \rangle \mid \Lambda (c : \varphi)\ G \mid G \{ G' \} \\
&\quad \mid\ \Diamond^{\tau} \mid G_1 \xrightarrow{\tau} G_2 \mid \mathrm{Dist}^{\forall \alpha .}_{\tau \to \sigma} \mid c
\end{aligned}
$$

Coercion variable:

$$
\text{CoerVar} \quad \frac{\Gamma \vdash ok \qquad c : \varphi \in \Gamma}{\Gamma \vdash c : \varphi}
$$

System F_ι

$$\tau, \sigma ::= \tau \to \sigma \mid \alpha \mid \forall\alpha.\tau \mid \varphi \Rightarrow \tau \qquad\qquad \varphi ::= \tau \triangleright \sigma$$

$$M, N ::= x \mid \lambda(x : \tau)\ M \mid M\ N$$
$$\mid\ \Lambda\alpha\ M \mid M\,\tau \mid G\langle M\rangle \mid \Lambda(c : \varphi)\ M \mid M\{G\}$$
$$G ::= \Lambda\alpha\ G \mid G\,\tau \mid G_1\langle G_2\rangle \mid \Lambda(c : \varphi)\ G \mid G\{G'\}$$
$$\mid\ \Diamond^{\tau} \mid G_1 \xrightarrow{\tau} G_2 \mid \mathrm{Dist}^{\forall\alpha.}_{\tau\to\sigma} \mid c \mid \mathrm{Dist}^{\varphi\Rightarrow}_{\tau\to\sigma}$$

It permutes $\Lambda(c : \varphi)$ and $\lambda(x : \tau)$

CoerDistCoerArrow

$$\frac{\Gamma \vdash \tau \qquad \Gamma \vdash \varphi \qquad \Gamma \vdash \sigma}{\Gamma \vdash \mathrm{Dist}^{\varphi\Rightarrow}_{\tau\to\sigma} : (\varphi \Rightarrow (\tau \to \sigma)) \triangleright (\tau \to (\varphi \Rightarrow \sigma))}$$

RedCoerDistCoerArrow

$$\mathrm{Dist}^{\varphi'\Rightarrow}_{\tau'\to\sigma'}\langle\Lambda(c : \varphi)\ \lambda(x : \tau)\ M\rangle \rightsquigarrow_{\iota} \lambda(x : \tau)\ \Lambda(c : \varphi)\ M$$