Types
for
Information-Flow Control
in
Functional Programming
Languages

Willard Rafnsson +3
IT University of Copenhagen

Information Flow Control

Prove absence of *undesired flows* of information in programs.

- {JS , App} does not leak {credit card nr, GPS} to ad provider

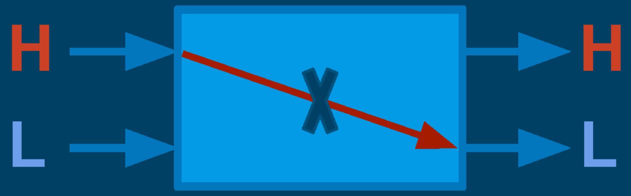
Well developed (40+ years) (but hard)

- OS (seL4), voting system (Civitas), web framework (Swift), email client (JPMail)

Information Flow Control

Noninterference

Property



"H inputs do not interfere with L outputs."

Property of behavior — sets of traces

Many flavors — confidentiality / integrity

Enforced — analysis / transformation

Flows

explicit:

```
low := high;
```

implicit:

```
if (high mod 2 == 1) {
    low := 1;
} else {
    low := 0;
}
```

Type systems for IFC check & disallow this.
Several typed **imperative** IFC languages:
Jif, Paragon, Joana, SPARK, JSFlow

Functional Programming Languages

Different Approaches

How do you track information flows in functional programs?

Two mainstream styles:

Flow Caml vs. lio

Label everything
FG (Fine-Grained)

Label I/O interface
CG (Coarse-)

Two extremes.

Expressive Power?

Main result: Both styles *equally expressive!*

Proof: semantics- and type-preserving translation. (à la Abadi et al. *A Core Calculus of Dependency*).

- FG translates to CG.
- CG translates to FG. **QED**

Today's goal

Teach you IFC for FP (rest is extra)

Outline

FG, CG, translations (touch on that)

- Values have information.
- Deconstructing a value transfers its information to the result.

$e\ e \downarrow v$

Recall: a **function** is a value. application deconstructs it...

FG

Flow Caml
(DCC ← SLam Calculus)

e	::=	$x \mid \lambda x\,.\,e \mid e\ e \mid$	λ- calculus +
		$(e, e) \mid \mathsf{fst}(e) \mid \mathsf{snd}(e) \mid$	tuples +
		$\mathsf{inl}(e) \mid \mathsf{inr}(e) \mid \mathsf{case}(e, x.e, x.e) \mid$	sums +
		$\mathsf{new}\ e \mid !e \mid e := e \mid ()$	effects + unit

τ	::=	A^{ℓ}	types (labeled)
A	::=	$\mathsf{unit} \mid \tau \rightarrow^{\ell} \tau \mid \tau \times \tau \mid \tau + \tau \mid \mathsf{ref}\ \tau$	

labels on
unit and τ × τ
are redundant.

Examples

$\Gamma \vdash_{pc} e : \tau$ judgement

structural label: ℓ in A^{ℓ} (in τ).

$(\text{unit}^{L} + \text{unit}^{L})^{H}$ (think "bool^{H}")

reveals information about
structure of sum (inl or inr) (thus H)

propagation of labels.

$\text{bool}^{H} \wedge \text{bool}^{L}$

reveals information about
either side of $\wedge$ (thus, result type bool^{H})

control label: pc in type judgement.

$e_V : (\text{unit}^{L} + \text{unit}^{L})^{H}$
$e_R : (\text{ref nat}^{L})^{L}$
$e = \text{case}(e_V,\ \_.()\ ,\ \_.e_R := 42\ )$

result always (). but,
e reveals whether e_V is inl or inr, via.
(non-)presence of write on e_R.

FG tracks this *control-flow* information, in pc
(raise pc when typing left/right, by label of e_V).
Lower bound on write effects (in e)
(output must be allowed to learn about pc)

FG

Examples

effect label: ℓ in $(\to^{\ell})$.

$e_V : (\text{unit}^L + \text{unit}^L)^H$
$e = \lambda\, x_F.\ \text{case}(x_V,\ \_.() ,\ \_.(x_F\ ()\ )\ )$

If x_F maps unit^L to unit^L, then e will map such x_F to unit^H (that's OK). But, now consider

$e_R : (\text{ref nat}^L)^L$
$e_F : \lambda\_ .\ (e_R := 42)$

$e\ e_F$ conditionally applies e_F. Thus, $e_R := 42$ in e_F leaks e_V (which is H) to L.

$\Gamma \vdash_{pc} e : \tau$ judgement

FG tracks this control-flow information, in $(\to^{\ell})$.
Lower bound on write effects (in function body)

$e_F : (\text{unit}^L \to^L \text{unit}^L)^L$

FG rejects the above example w/ this type.

How to fix: (i.e. why FG rejected this):

- Make e_V L , or make e_F -stuff H
- Remove the assignment

note: that functions of type $(\tau \to^L \tau)^H$ *can* be constructed, passed around, but never applied, as that would leak its own identity.

Look, a
type system!
moving along

$$\frac{}{\Gamma, x:\tau \vdash_{pc} x:\tau}\ \text{FG-var} \qquad \frac{\Gamma, x:\tau_1 \vdash_{\ell_e} e:\tau_2}{\Gamma \vdash_{pc} \lambda x.e : (\tau_1 \xrightarrow{\ell_e} \tau_2)^{\perp}}\ \text{FG-lam}$$

$$\frac{\Gamma \vdash_{pc} e_1 : (\tau_1 \xrightarrow{\ell_e} \tau_2)^{\ell} \quad \Gamma \vdash_{pc} e_2 : \tau_1 \quad \mathcal{L} \vdash \tau_2 \searrow \ell \quad \mathcal{L} \vdash pc \sqcup \ell \sqsubseteq \ell_e}{\Gamma \vdash_{pc} e_1\ e_2 : \tau_2}\ \text{FG-app}$$

$$\frac{\Gamma \vdash_{pc} e_1 : \tau_1 \quad \Gamma \vdash_{pc} e_2 : \tau_2}{\Gamma \vdash_{pc} (e_1, e_2) : (\tau_1 \times \tau_2)^{\perp}}\ \text{FG-prod} \qquad \frac{\Gamma \vdash_{pc} e : (\tau_1 \times \tau_2)^{\ell} \quad \mathcal{L} \vdash \tau_1 \searrow \ell}{\Gamma \vdash_{pc} \mathsf{fst}(e) : \tau_1}\ \text{FG-fst}$$

$$\frac{\Gamma \vdash_{pc} e : \tau_1}{\Gamma \vdash_{pc} \mathsf{inl}(e) : (\tau_1 + \tau_2)^{\perp}}\ \text{FG-inl}$$

$$\frac{\Gamma \vdash_{pc} e : (\tau_1 + \tau_2)^{\ell} \quad \Gamma, x:\tau_1 \vdash_{pc \sqcup \ell} e_1 : \tau \quad \Gamma, y:\tau_2 \vdash_{pc \sqcup \ell} e_2 : \tau \quad \mathcal{L} \vdash \tau \searrow \ell}{\Gamma \vdash_{pc} \mathsf{case}(e, x.e_1, y.e_2) : \tau}\ \text{FG-case}$$

$$\frac{\Gamma \vdash_{pc'} e : \tau' \quad \mathcal{L} \vdash pc \sqsubseteq pc' \quad \mathcal{L} \vdash \tau' <: \tau}{\Gamma \vdash_{pc} e : \tau}\ \text{FG-sub} \qquad \frac{\Gamma \vdash_{pc} e : \tau \quad \mathcal{L} \vdash \tau \searrow pc}{\Gamma \vdash_{pc} \mathsf{new}\ e : (\mathsf{ref}\ \tau)^{\perp}}\ \text{FG-ref}$$

$$\frac{\Gamma \vdash_{pc} e : (\mathsf{ref}\ \tau)^{\ell} \quad \mathcal{L} \vdash \tau <: \tau' \quad \mathcal{L} \vdash \tau' \searrow \ell}{\Gamma \vdash_{pc} !e : \tau'}\ \text{FG-deref}$$

$$\frac{\Gamma \vdash_{pc} e_1 : (\mathsf{ref}\ \tau)^{\ell} \quad \Gamma \vdash_{pc} e_2 : \tau \quad \mathcal{L} \vdash \tau \searrow (pc \sqcup \ell)}{\Gamma \vdash_{pc} e_1 := e_2 : \mathsf{unit}}\ \text{FG-assign} \qquad \frac{}{\Gamma \vdash_{pc} () : \mathsf{unit}^{\perp}}\ \text{FG-unitI}$$

FG

Types

CG

LIO
(Static fragment of HLIO)

e ::=	x | $\lambda x . e$ | $e\ e$ |	λ- calculus +
	(e, e) | fst(e) | snd(e) |	tuples +
	inl(e) | inr(e) | case(e, $x.e$, $x.e$) |	sums +
	new e | !e | e := e | () |	effects + unit
	lb(e) | unLb(e) | toLb(e) |	labeling +
	ret (e) | bind(e, $x.e$)	label monad

τ ::=	unit | $\tau \rightarrow \tau$ | $\tau \times \tau$ | $\tau + \tau$ | ref ℓ τ |	
	Labeled ℓ τ | ©ℓ_1 ℓ_2 τ	types

Examples

Γ |-- $e : \tau$ judgement

Labeled ℓ τ	like τ^{ℓ} before
ref ℓ τ	stores Labeled ℓ τ (only)
$\mathbb{C}\ \ell_1\ \ell_2\ \tau$	suspended *effectful* computation
input	(computations learns info)
unlabel(e)	(don't learn on read, but on unLabel)
output	(computation releases info)
Label(e)	(or write to ref. that's not strictly needed, though)

ℓ_1 is *pc* ; lower bound on output.
ℓ_2 is *taint* ; upper bound of all unLb

CG only tracks flows through bind.

references: consider the program

e_R = new 0
e_V = label(4)
e_A = x_R := x_V
e = bind(e_R , x_R.bind(ret(e_V) , x_V. e_A)

Consider typing

e_R : $\mathbb{C}$ L L (ref L nat)
e_V : Labeled H nat

Then e does not typecheck; e_A writes (Labeled H nat) to (ref L nat). CG checks this.

CG

Examples

Γ |-- $e : \tau$ judgement

bind: consider instead the program

$e_A = e_{RO}$:= label(4)
e = bind(!e_{RI} , x_I.bind(unlabel(x_I) , $x_\bot$. e_A)

Consider typing

e_{RI} : ref H nat
e_{RO} : ref L nat

CG rejects e w/ these types;
unlabel raises *pc* of e_A to H, making e_A untypeable.

CG tracks this flow through bind.

How to fix (i.e. why CG rejected it): Either

- Make e_{RI} and e_{RO} both H or both L
- Don't unlabel x_I
- Use toLabeled(·) around the part using x_I

That last option added to deal with *label creep*.
(label creep: label monotonely increasing through a bind-chain). Can now do H sub-computation.

Types

Look,
another one!
moving along

(All rules of the simply typed lambda-calculus pertaining to the types b, $\tau \to \tau, \tau \times \tau, \tau + \tau$, and unit are included.)

$$\frac{\Gamma \vdash e_1 : \mathbb{C}\ \ell_1\ \ell_2\ \tau \qquad \Gamma, x:\tau \vdash e_2 : \mathbb{C}\ \ell_3\ \ell_4\ \tau' \qquad \ell \sqsubseteq \ell_1 \qquad \ell \sqsubseteq \ell_3 \qquad \ell_2 \sqsubseteq \ell_3 \qquad \ell_2 \sqsubseteq \ell_4}{\Gamma \vdash \mathsf{bind}(e_1, x.e_2) : \mathbb{C}\ \ell\ \ell_4\ \tau'}\ \text{CG-bind}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{ret}(e) : \mathbb{C}\ \top\ \bot\ \tau}\ \text{CG-ret} \qquad \frac{\Gamma \vdash e : \tau' \qquad \mathcal{L} \vdash \tau' <: \tau}{\Gamma \vdash e : \tau}\ \text{CG-sub} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{Lb}(e) : \mathsf{Labeled}\ \ell\ \tau}\ \text{CG-label}$$

$$\frac{\Gamma \vdash e : \mathsf{Labeled}\ \ell\ \tau}{\Gamma \vdash \mathsf{unlabel}(e) : \mathbb{C}\ \top\ \ell\ \tau}\ \text{CG-unlabel} \qquad \frac{\Gamma \vdash e : \mathsf{Labeled}\ \ell\ \tau}{\Gamma \vdash \mathsf{new}\ e : \mathbb{C}\ \ell\ \bot\ (\mathsf{ref}\ \ell\ \tau)}\ \text{CG-ref} \qquad \frac{\Gamma \vdash e : \mathsf{ref}\ \ell'\ \tau}{\Gamma \vdash\ !e : \mathbb{C}\ \top\ \bot\ (\mathsf{Labeled}\ \ell'\tau)}\ \text{CG-deref}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{ref}\ \ell\ \tau \qquad \Gamma \vdash e_2 : \mathsf{Labeled}\ \ell\ \tau}{\Gamma \vdash e_1 := e_2 : \mathbb{C}\ \ell\ \bot\ \mathsf{unit}}\ \text{CG-assign} \qquad \frac{\Gamma \vdash e : \mathbb{C}\ \ell\ \ell'\ \tau}{\Gamma \vdash \mathsf{toLabeled}(e) : \mathbb{C}\ \ell\ \bot\ (\mathsf{Labeled}\ \ell'\ \tau)}\ \text{CG-toLabeled}$$

Subtyping judgment: $\boxed{\mathcal{L} \vdash \tau <: \tau'}$

$$\frac{\mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \ell \sqsubseteq \ell'}{\mathcal{L} \vdash \mathsf{Labeled}\ \ell\ \tau <: \mathsf{Labeled}\ \ell'\ \tau'}\ \text{CGsub-labeled} \qquad \frac{\mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \ell_1' \sqsubseteq \ell_1 \qquad \mathcal{L} \vdash \ell_2 \sqsubseteq \ell_2'}{\mathcal{L} \vdash \mathbb{C}\ \ell_1\ \ell_2\ \tau <: \mathbb{C}\ \ell_1'\ \ell_2'\ \tau'}\ \text{CGsub-monad}$$

Equivalence

FG *seems* more expressive;
it tracks flows at finer granularity.

But what if you structure code as *many tiny computations* in CG?

Theorem: e well-typed in CG translates to $[\![e]\!]$ well-typed in FG w/ *same behavior*, and vice versa

Proof: semantics- and type-preserving translation (à la DCC)

(sketch follows)

Equivalence

CG → FG

$\Gamma \mid\!\!-\!\!- \; e : \tau \;\sim\!> [\![\Gamma]\!] \mid\!\!-\!\!- \; [\![e]\!] : [\![\tau]\!]$?

Equivalence

CG → FG

Γ |-- $e : \tau$ ~> $[\![\Gamma]\!]$ $|\text{--}_{\top}$ $[\![e]\!] : [\![\tau]\!]$

All CG-computation is suspended

Equivalence

CG → FG $\Gamma \vdash e : \tau \rightsquigarrow [\![\Gamma]\!] \vdash_{\top} [\![e]\!] : [\![\tau]\!]$

Types

First four trivial (CG unlabeled)

Labeled ℓ τ Label [[τ]] with ℓ, how? Already labeled.
Coding trick.
We ensure expr. always eval to inl.

ref ℓ τ Label [[τ]] with ℓ, same trick ^

$\mathbb{C}$ ℓ_1 ℓ_2 τ $\mathbb{C}$ is suspended. so, map it to a *thunk*.
ℓ_1 is bound on writes. Put that on →.
ℓ_2 is value label. Label [[τ]] with ℓ ^

$$
\begin{aligned}
[\![\mathsf{b}]\!] &= \mathsf{b}^{\perp} \\
[\![\tau_1 \to \tau_2]\!] &= ([\![\tau_1]\!] \xrightarrow{\top} [\![\tau_2]\!])^{\perp} \\
[\![\tau_1 \times \tau_2]\!] &= ([\![\tau_1]\!] \times [\![\tau_2]\!])^{\perp} \\
[\![\tau_1 + \tau_2]\!] &= ([\![\tau_1]\!] + [\![\tau_2]\!])^{\perp} \\
[\![\mathsf{ref}\ \ell\ \tau]\!] &= (\mathsf{ref}\ ([\![\tau]\!] + \mathsf{unit})^{\ell})^{\perp} \\
[\![\mathbb{C}\ \ell_1\ \ell_2\ \tau]\!] &= (\mathsf{unit} \xrightarrow{\ell_1} ([\![\tau]\!] + \mathsf{unit})^{\ell_2})^{\perp} \\
[\![\mathsf{Labeled}\ \ell\ \tau]\!] &= ([\![\tau]\!] + \mathsf{unit})^{\ell}
\end{aligned}
$$

Equivalence

CG → FG $\Gamma \vdash e : \tau \rightsquigarrow [\![\Gamma]\!] \vdash_{\top} [\![e]\!] : [\![\tau]\!]$

Expressions

Translates a well-typed CG-expression (i.e. proof of $\Gamma \vdash e : \tau$),
into a well-typed FG-expression (i.e. proof of $[\![\Gamma]\!] \vdash_{\top} [\![e]\!] : [\![\tau]\!]$).

$$\frac{\Gamma \vdash e_1 : \mathsf{CG}\ \ell_i\ \ell\ \tau \qquad \Gamma, x : \tau \vdash e_2 : \mathsf{CG}\ \ell\ \ell_o\ \tau'}{\Gamma \vdash \mathsf{bind}(e_1, x.e_2) : \mathsf{CG}\ \ell_i\ \ell_o\ \tau'} \rightsquigarrow$$

$$\frac{[\![\Gamma]\!] \vdash_{\top} e_1' : (\mathsf{unit} \xrightarrow{\ell_i} ([\![\tau]\!] + \mathsf{unit})^{\ell})^{\perp} \qquad [\![\Gamma]\!], x : [\![\tau]\!] \vdash_{\top} e_2' : (\mathsf{unit} \xrightarrow{\ell} ([\![\tau']\!] + \mathsf{unit})^{\ell_o})^{\perp}}{[\![\Gamma]\!] \vdash_{\top} \lambda\_.\mathsf{case}(e_1'(), x.e_2'(), y.\mathsf{inr}()) : (\mathsf{unit} \xrightarrow{\ell_i} ([\![\tau']\!] + \mathsf{unit})^{\ell_o})^{\perp}}$$

Equivalence

FG → CG

$\Gamma \vdash_{pc} e : \tau \rightsquigarrow (\!|\Gamma|\!) \vdash (\!|e|\!) : \quad (\!|\tau|\!) \quad ?$

Equivalence

FG → CG

$$\Gamma \vdash_{pc} e : \tau \rightsquigarrow (\!|\Gamma|\!) \vdash (\!|e|\!) : \mathbb{C}\; pc \perp (\!|\tau|\!)$$

Track FG-pc
With CG-monad

Equivalence

FG → CG $\Gamma \vdash_{pc} e : \tau \rightsquigarrow (\!|\Gamma|\!) \vdash (\!|e|\!) : \mathbb{C}\ pc\ \bot\ (\!|\tau|\!)$

Types

All cases trivial (use Labeled), save for one.

$\tau_1 \xrightarrow{\ell_e} \tau_2$ Function that returns a monadic computation.
monad confines I/O of original f.
ℓ_e is lower bound on writes (like pc).
Put that in $\mathbb{C}$.

$(\!|e|\!)$ boxes everything with toLabeled, hence the $\bot$.

$$
\begin{array}{lcll}
(\!|\mathsf{b}|\!) & = & \mathsf{b} & \\
(\!|\mathsf{unit}|\!) & = & \mathsf{unit} & \\
(\!|\tau_1 \xrightarrow{\ell_e} \tau_2|\!) & = & (\!|\tau_1|\!) \to \mathbb{C}\ \ell_e\ \bot\ (\!|\tau_2|\!) & \\
(\!|\tau_1 \times \tau_2|\!) & = & (\!|\tau_1|\!) \times (\!|\tau_2|\!) & \\
(\!|\tau_1 + \tau_2|\!) & = & (\!|\tau_1|\!) + (\!|\tau_2|\!) & \\
(\!|\mathsf{ref}\ \tau|\!) & = & \mathsf{ref}\ \ell\ (\!|\mathsf{A}|\!) & \text{when } \tau = \mathsf{A}^{\ell} \\
(\!|\mathsf{A}^{\ell}|\!) & = & \mathsf{Labeled}\ \ell\ (\!|\mathsf{A}|\!) &
\end{array}
$$

Equivalence

FG → CG

$$\Gamma \mid\!\!\text{--}_{pc}\ e : \tau \rightsquigarrow (\!|\Gamma|\!) \mid\!\!\text{--}\ (\!|e|\!) : \mathbb{C}\ pc \perp (\!|\tau|\!)$$

Expressions

$$\frac{\Gamma \vdash_{pc} e_{\mathtt{F}} : (\tau_{\mathtt{I}} \xrightarrow{\ell_e} \tau_{\mathtt{O}})^{\ell} \qquad \Gamma \vdash_{pc} e_{\mathtt{I}} : \tau_{\mathtt{I}} \qquad \mathcal{L} \vdash \tau_{\mathtt{O}} \searrow \ell \qquad \mathcal{L} \vdash pc \sqcup \ell \sqsubseteq \ell_e}{\Gamma \vdash_{pc} e_{\mathtt{F}}\ e_{\mathtt{I}} : \tau_{\mathtt{O}}}\ \text{FG-app} \rightsquigarrow$$

$$\frac{?}{(\!|\Gamma|\!) \vdash (\!|e_{\mathtt{F}}\ e_{\mathtt{I}}|\!) : \mathsf{CG}\ \ell_e \perp (\!|\tau_{\mathtt{O}}|\!)}\ \text{CG-toLabeled}$$

Huge derivation

$$\begin{aligned}
(\!|e_{\mathtt{F}}|\!) &: \mathsf{CG}\ pc \perp (\!|(\tau_{\mathtt{I}} \xrightarrow{\ell_e} \tau_{\mathtt{O}})^{\ell}|\!) \\
&\ \mathsf{CG}\ pc \perp (\ \mathsf{Labeled}\ \ell\ ((\!|\tau_{\mathtt{I}}|\!) \to \mathsf{CG}\ \ell_e \perp (\!|\tau_{\mathtt{O}}|\!))\ ) \\
(\!|e_{\mathtt{I}}|\!) &: \mathsf{CG}\ pc \perp (\!|\tau_{\mathtt{I}}|\!) \\
(\!|e_{\mathtt{F}}\ e_{\mathtt{I}}|\!) &: \mathsf{CG}\ \ell_e \perp (\!|\tau_{\mathtt{O}}|\!)
\end{aligned}$$

$$(\!|e_{\mathtt{F}}\ e_{\mathtt{I}}|\!) = \mathsf{toLabeled}(\ \mathsf{bind}(\ \mathsf{bind}(\ (\!|e_{\mathtt{F}}|\!)\ ,\ x_{\mathtt{LF}}.\mathsf{bind}(\ (\!|e_{\mathtt{I}}|\!)\ ,\ x_{\mathtt{I}}.\mathsf{bind}(\ \mathsf{unlabel}(x_{\mathtt{LF}})\ ,\ x_{\mathtt{F}}.x_{\mathtt{F}}\ x_{\mathtt{I}})))\ ,\ x_{\mathtt{LO}}.\mathsf{unlabel}(x_{\mathtt{LO}}))\ )$$

CG↔FG, Semantics

Last (crucial) step:

- Type Soundness
- Semantics-Preservation

Co-author (Vineet Rajani) did this in a follow-up paper!

- Step-indexed Kripke *logical relation*
  (based on a model of types
  in incremental computational complexity).

Summary

- IFC in FP crash-course
- FG and CG different granularity
- FG and CG equally expressive

  Proof: semantics- and type-preserving translation.

  logical relations.
- Suffices to use CG

Acknowledgements

Willard Rafnsson

Iulia Bastys

Vineet Rajani

Deepak Garg

SIGLOG News 2017

Type Systems for Information Flow Control: The Question of Granularity

CSF 2018

Types for Information Flow Control: Labeling Granularity and Semantic Models

Vineet Rajani
Max Planck Institute for Software Systems
Saarland Informatics Campus

Deepak Garg
Max Planck Institute for Software Systems
Saarland Informatics Campus

Appendix

FG

Types

Typing judgment: $\boxed{\Gamma \vdash_{pc} e : \tau}$

$$\frac{}{\Gamma, x : \tau \vdash_{pc} x : \tau}\ \text{FG-var} \qquad \frac{\Gamma, x : \tau_1 \vdash_{\ell_e} e : \tau_2}{\Gamma \vdash_{pc} \lambda x.e : (\tau_1 \xrightarrow{\ell_e} \tau_2)^{\perp}}\ \text{FG-lam}$$

$$\frac{\Gamma \vdash_{pc} e_1 : (\tau_1 \xrightarrow{\ell_e} \tau_2)^{\ell} \qquad \Gamma \vdash_{pc} e_2 : \tau_1 \qquad \mathcal{L} \vdash \tau_2 \searrow \ell \qquad \mathcal{L} \vdash pc \sqcup \ell \sqsubseteq \ell_e}{\Gamma \vdash_{pc} e_1\ e_2 : \tau_2}\ \text{FG-app}$$

$$\frac{\Gamma \vdash_{pc} e_1 : \tau_1 \qquad \Gamma \vdash_{pc} e_2 : \tau_2}{\Gamma \vdash_{pc} (e_1, e_2) : (\tau_1 \times \tau_2)^{\perp}}\ \text{FG-prod} \qquad \frac{\Gamma \vdash_{pc} e : (\tau_1 \times \tau_2)^{\ell} \qquad \mathcal{L} \vdash \tau_1 \searrow \ell}{\Gamma \vdash_{pc} \mathsf{fst}(e) : \tau_1}\ \text{FG-fst}$$

$$\frac{\Gamma \vdash_{pc} e : \tau_1}{\Gamma \vdash_{pc} \mathsf{inl}(e) : (\tau_1 + \tau_2)^{\perp}}\ \text{FG-inl}$$

$$\frac{\Gamma \vdash_{pc} e : (\tau_1 + \tau_2)^{\ell} \qquad \Gamma, x : \tau_1 \vdash_{pc \sqcup \ell} e_1 : \tau \qquad \Gamma, y : \tau_2 \vdash_{pc \sqcup \ell} e_2 : \tau \qquad \mathcal{L} \vdash \tau \searrow \ell}{\Gamma \vdash_{pc} \mathsf{case}(e, x.e_1, y.e_2) : \tau}\ \text{FG-case}$$

$$\frac{\Gamma \vdash_{pc'} e : \tau' \qquad \mathcal{L} \vdash pc \sqsubseteq pc' \qquad \mathcal{L} \vdash \tau' <: \tau}{\Gamma \vdash_{pc} e : \tau}\ \text{FG-sub} \qquad \frac{\Gamma \vdash_{pc} e : \tau \qquad \mathcal{L} \vdash \tau \searrow pc}{\Gamma \vdash_{pc} \mathsf{new}\ e : (\mathsf{ref}\ \tau)^{\perp}}\ \text{FG-ref}$$

$$\frac{\Gamma \vdash_{pc} e : (\mathsf{ref}\ \tau)^{\ell} \qquad \mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \tau' \searrow \ell}{\Gamma \vdash_{pc} !e : \tau'}\ \text{FG-deref}$$

$$\frac{\Gamma \vdash_{pc} e_1 : (\mathsf{ref}\ \tau)^{\ell} \qquad \Gamma \vdash_{pc} e_2 : \tau \qquad \mathcal{L} \vdash \tau \searrow (pc \sqcup \ell)}{\Gamma \vdash_{pc} e_1 := e_2 : \mathsf{unit}}\ \text{FG-assign} \qquad \frac{}{\Gamma \vdash_{pc} () : \mathsf{unit}^{\perp}}\ \text{FG-unitI}$$

FG

Types

Typing judgment: $\boxed{\Gamma \vdash_{pc} e : \tau}$

$$\frac{}{\Gamma, x : \tau \vdash_{pc} x : \tau} \text{ FG-var}$$

$$\frac{}{\Gamma \vdash_{pc} () : \mathsf{unit}^{\perp}} \text{ FG-unitI}$$

FG

Types

Always safe to
- **up-classify** information
- **weaken** the guarantee (*)

Rule FG-sub: ↓
- raise τ
- lower *pc* (*)

Raise τ means
- raise structural ℓ
- lower τ_I and ℓ in $\tau_I \to^{\ell} \tau_O$ (*)

Typing judgment: $\boxed{\Gamma \vdash_{pc} e : \tau}$

$$\frac{\Gamma \vdash_{pc'} e : \tau' \qquad \mathcal{L} \vdash pc \sqsubseteq pc' \qquad \mathcal{L} \vdash \tau' <: \tau}{\Gamma \vdash_{pc} e : \tau} \text{ FG-sub}$$

FG

Types

Always safe to
- **up-classify** &
- **weaken**

$A^{\ell} \searrow \ell'$ just means $\ell \sqsubseteq \ell'$

(prevent leaking context into result)

Typing judgment: $\boxed{\Gamma \vdash_{pc} e : \tau}$

$$\frac{\Gamma \vdash_{pc} e : \tau_1}{\Gamma \vdash_{pc} \mathsf{inl}(e) : (\tau_1 + \tau_2)^{\perp}} \text{ FG-inl}$$

$$\frac{\Gamma \vdash_{pc} e : (\tau_1 + \tau_2)^{\ell} \quad \Gamma, x : \tau_1 \vdash_{pc \sqcup \ell} e_1 : \tau \quad \Gamma, y : \tau_2 \vdash_{pc \sqcup \ell} e_2 : \tau \quad \mathcal{L} \vdash \tau \searrow \ell}{\Gamma \vdash_{pc} \mathsf{case}(e, x.e_1, y.e_2) : \tau} \text{ FG-case}$$

FG

Types

Always safe to
- **up-classify** &
- **weaken**

$A^{\ell} \searrow \ell' \Leftrightarrow \ell \sqsubseteq \ell'$

FG-lam, why ■ :
- no evaluation; only checking that ℓ_e bounds effects below.
- FG-case , FG-app put pc into structural info.

Typing judgment: $\boxed{\Gamma \vdash_{pc} e : \tau}$

$$\frac{\Gamma, x : \tau_1 \vdash_{\ell_e} e : \tau_2}{\Gamma \vdash_{pc} \lambda x.e : (\tau_1 \xrightarrow{\ell_e} \tau_2)^{\perp}} \text{ FG-lam}$$

$$\frac{\Gamma \vdash_{pc} e_1 : (\tau_1 \xrightarrow{\ell_e} \tau_2)^{\ell} \quad \Gamma \vdash_{pc} e_2 : \tau_1 \quad \mathcal{L} \vdash \tau_2 \searrow \ell \quad \mathcal{L} \vdash pc \sqcup \ell \sqsubseteq \ell_e}{\Gamma \vdash_{pc} e_1\ e_2 : \tau_2} \text{ FG-app}$$

CG

Types

Typing judgment: $\boxed{\Gamma \vdash e : \tau}$

(All rules of the simply typed lambda-calculus pertaining to the types $\mathsf{b}, \tau \to \tau, \tau \times \tau, \tau + \tau$, and unit are included.)

$$\frac{\Gamma \vdash e_1 : \mathbb{C}\; \ell_1\; \ell_2\; \tau \qquad \Gamma, x : \tau \vdash e_2 : \mathbb{C}\; \ell_3\; \ell_4\; \tau' \qquad \ell \sqsubseteq \ell_1 \qquad \ell \sqsubseteq \ell_3 \qquad \ell_2 \sqsubseteq \ell_3 \qquad \ell_2 \sqsubseteq \ell_4}{\Gamma \vdash \mathsf{bind}(e_1, x.e_2) : \mathbb{C}\; \ell\; \ell_4\; \tau'}\;\text{CG-bind}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{ret}(e) : \mathbb{C}\; \top\; \bot\; \tau}\;\text{CG-ret} \qquad \frac{\Gamma \vdash e : \tau' \qquad \mathcal{L} \vdash \tau' <: \tau}{\Gamma \vdash e : \tau}\;\text{CG-sub} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{Lb}(e) : \mathsf{Labeled}\; \ell\; \tau}\;\text{CG-label}$$

$$\frac{\Gamma \vdash e : \mathsf{Labeled}\; \ell\; \tau}{\Gamma \vdash \mathsf{unlabel}(e) : \mathbb{C}\; \top\; \ell\; \tau}\;\text{CG-unlabel} \qquad \frac{\Gamma \vdash e : \mathsf{Labeled}\; \ell\; \tau}{\Gamma \vdash \mathsf{new}\; e : \mathbb{C}\; \ell\; \bot\; (\mathsf{ref}\; \ell\; \tau)}\;\text{CG-ref} \qquad \frac{\Gamma \vdash e : \mathsf{ref}\; \ell'\; \tau}{\Gamma \vdash !e : \mathbb{C}\; \top\; \bot\; (\mathsf{Labeled}\; \ell'\tau)}\;\text{CG-deref}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{ref}\; \ell\; \tau \qquad \Gamma \vdash e_2 : \mathsf{Labeled}\; \ell\; \tau}{\Gamma \vdash e_1 := e_2 : \mathbb{C}\; \ell\; \bot\; \mathsf{unit}}\;\text{CG-assign} \qquad \frac{\Gamma \vdash e : \mathbb{C}\; \ell\; \ell'\; \tau}{\Gamma \vdash \mathsf{toLabeled}(e) : \mathbb{C}\; \ell\; \bot\; (\mathsf{Labeled}\; \ell'\; \tau)}\;\text{CG-toLabeled}$$

Subtyping judgment: $\boxed{\mathcal{L} \vdash \tau <: \tau'}$

$$\frac{\mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \ell \sqsubseteq \ell'}{\mathcal{L} \vdash \mathsf{Labeled}\; \ell\; \tau <: \mathsf{Labeled}\; \ell'\; \tau'}\;\text{CGsub-labeled} \qquad \frac{\mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \ell_1' \sqsubseteq \ell_1 \qquad \mathcal{L} \vdash \ell_2 \sqsubseteq \ell_2'}{\mathcal{L} \vdash \mathbb{C}\; \ell_1\; \ell_2\; \tau <: \mathbb{C}\; \ell_1'\; \ell_2'\; \tau'}\;\text{CGsub-monad}$$

CG

Types

Typing judgment: $\boxed{\Gamma \vdash e : \tau}$

$$\frac{\Gamma \vdash e_1 : \mathbb{C}\;\ell_1\;\ell_2\;\tau \qquad \Gamma, x : \tau \vdash e_2 : \mathbb{C}\;\ell_3\;\ell_4\;\tau' \qquad \ell \sqsubseteq \ell_1 \qquad \ell \sqsubseteq \ell_3 \qquad \ell_2 \sqsubseteq \ell_3 \qquad \ell_2 \sqsubseteq \ell_4}{\Gamma \vdash \mathsf{bind}(e_1, x.e_2) : \mathbb{C}\;\ell\;\ell_4\;\tau'}\ \text{CG-bind}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{ret}(e) : \mathbb{C}\;\top\;\bot\;\tau}\ \text{CG-ret}$$

CG

Types

Typing judgment: $\boxed{\Gamma \vdash e : \tau}$

$$\frac{\Gamma \vdash e : \tau' \qquad \mathcal{L} \vdash \tau' <: \tau}{\Gamma \vdash e : \tau} \text{ CG-sub}$$

Subtyping judgment: $\boxed{\mathcal{L} \vdash \tau <: \tau'}$

$$\frac{\mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \ell \sqsubseteq \ell'}{\mathcal{L} \vdash \mathsf{Labeled}\ \ell\ \tau <: \mathsf{Labeled}\ \ell'\ \tau'} \text{ CGsub-labeled}$$

$$\frac{\mathcal{L} \vdash \tau <: \tau' \qquad \mathcal{L} \vdash \ell_1' \sqsubseteq \ell_1 \qquad \mathcal{L} \vdash \ell_2 \sqsubseteq \ell_2'}{\mathcal{L} \vdash \mathbb{C}\ \ell_1\ \ell_2\ \tau <: \mathbb{C}\ \ell_1'\ \ell_2'\ \tau'} \text{ CGsub-monad}$$

up-classify
& **weaken**

Types

Typing judgment: $\boxed{\Gamma \vdash e : \tau}$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{Lb}(e) : \mathsf{Labeled}\ \ell\ \tau}\ \text{CG-label}$$

$$\frac{\Gamma \vdash e : \mathsf{Labeled}\ \ell\ \tau}{\Gamma \vdash \mathsf{unlabel}(e) : \mathbb{C}\ \top\ \ell\ \tau}\ \text{CG-unlabel}$$

$$\frac{\Gamma \vdash e : \mathbb{C}\ \ell\ \ell'\ \tau}{\Gamma \vdash \mathsf{toLabeled}(e) : \mathbb{C}\ \ell\ \bot\ (\mathsf{Labeled}\ \ell'\ \tau)}\ \text{CG-toLabeled}$$

Types

Typing judgment: $\boxed{\Gamma \vdash e : \tau}$

$$\frac{\Gamma \vdash e : \textsf{Labeled}\ \ell\ \tau}{\Gamma \vdash \textsf{new}\ e : \mathbb{C}\ \ell\ \bot\ (\textsf{ref}\ \ell\ \tau)}\ \text{CG-ref}$$

$$\frac{\Gamma \vdash e : \textsf{ref}\ \ell'\ \tau}{\Gamma \vdash !e : \mathbb{C}\ \top\ \bot\ (\textsf{Labeled}\ \ell'\tau)}\ \text{CG-deref}$$

$$\frac{\Gamma \vdash e_1 : \textsf{ref}\ \ell\ \tau \qquad \Gamma \vdash e_2 : \textsf{Labeled}\ \ell\ \tau}{\Gamma \vdash e_1 := e_2 : \mathbb{C}\ \ell\ \bot\ \textsf{unit}}\ \text{CG-assign}$$