Formally Verifying and Deriving Gradual Type Systems

David Darais
University of Maryland

Topics

Topics

`e : ?`

gradual types

Topics

$e : ?$

gradual types

$\vdash \forall(x).P(x)$

formal verification

Topics

$\mathtt{e} : \mathtt{?}$

gradual types

$\vdash \forall(x).P(x)$

formal verification

$\mathbb{Z} \rightleftarrows \{-,0,+\}$

abstract interpretation

Topics

`e : ?`

gradual types

Abstracting Gradual Typing [Garcia, Clark, Tanter; 2016]

$\mathbb{Z} \rightleftarrows \{-,0,+\}$

abstract interpretation

Topics

Mechanically Verified Calculational Abstract Interpretation (Draft) [Darais, Van Horn; 2015]

$\vdash \forall(x).P(x)$

formal verification

$\mathbb{Z} \rightleftarrows \{-,0,+\}$

abstract interpretation

Topics

`e : ?`

gradual types

$\vdash \forall(x).P(x)$

formal verification

$\mathbb{Z} \rightleftarrows \{-, 0, +\}$

abstract interpretation

Deriving Gradual Type Systems

`e : ?` $\quad \longleftrightarrow \quad$ $\mathbb{Z} \rightleftarrows \{-,0,+\}$

gradual types — *abstract interpretation*

- **Challenge:**
  Gradual type systems are ad-hoc and sometimes wrong

- **Insight:**
  Guide design through abstract interpretation

Precise Types (Static)

Precise Types (Static)

STATICS

DYNAMICS

Precise Types (Static)

STATICS

$\vdash$ `1 + 5 : int` ✓

DYNAMICS

Precise Types (Static)

STATICS

⊢ 1 + 5 : int ✓

⊢ 💩 + 5 : _ ✗

DYNAMICS

Precise Types (Static)

STATICS

$\vdash$ 1 + 5 : int ✓

$\vdash$ 💩 + 5 : _ ✗

DYNAMICS

$\vdash$ 1 + 5 $\Downarrow$ 6 ✓

Precise Types (Static)

STATICS

$\vdash$ 1 + 5 : int ✓

$\vdash$ 💩 + 5 : _ ✗

DYNAMICS

$\vdash$ 1 + 5 $\Downarrow$ 6 ✓

<what’s missing?>

Precise Types (Dynamic)

STATICS

DYNAMICS

Precise Types (Dynamic)

STATICS

⊢ 1 + 5 : 🙈 ✓

DYNAMICS

Precise Types (Dynamic)

STATICS

⊢ 1 + 5 : 🙈 ✓

⊢ 💩 + 5 : 🙈 ✓

DYNAMICS

Precise Types (Dynamic)

STATICS

$\vdash$ 1 + 5 : 🙈 ✓

$\vdash$ 💩 + 5 : 🙈 ✓

DYNAMICS

$\vdash$ 1 + 5 $\Downarrow$ 6 ✓

Precise Types (Dynamic)

STATICS

$\vdash$ 1 + 5 : 🙈 ✓

$\vdash$ 💩 + 5 : 🙈 ✓

DYNAMICS

$\vdash$ 1 + 5 $\Downarrow$ 6 ✓

$\vdash$ 💩 + 5 $\Downarrow$ _ ✗

Gradual Types (Hybrid)

STATICS

DYNAMICS

Gradual Types (Hybrid)

STATICS

⊢ 💩 + 5 : _ ✗

DYNAMICS

Gradual Types (Hybrid)

STATICS

$\vdash$ 💩 + 5 : _ ✗

$\vdash$ (💩 : ?) + 5 : int ✓

DYNAMICS

Gradual Types (Hybrid)

STATICS

⊢ 💩 + 5 : ___ ✗

⊢ (💩 : ?) + 5 : int ✓

DYNAMICS

⊢ (💩 : ?) + 5 ⇓ ___ ✗

Gradual Types (Hybrid)

STATICS

⊢ 💩 + 5 : _ ✗

⊢ (💩:?) + 5 : int ✓

DYNAMICS

⊢ (💩:?) + 5 ⇓ _ ✗

<what’s missing?>

“What Do You Want?”

What Do You Want?

```
if(x){💩}{1} + 3
```

What Do You Want?

```
if(x){💩}{1} + 3
```

<Static Rob> 🤖

💩

What Do You Want?

```
if(x){💩}{1} + 3
```

"I couldn't verify that, in every case, it's safe to put 💩 there"

OR

"There exist some case where it's unsafe to put 💩 there"

<Static Rob> 🤖: 💩

What Do You Want?

```
if(x){💩}{1} + 3
```

💩

<Gradual Rob> 🤖

What Do You Want?

```
if(x){💩}{1} + 3
```

"I couldn't verify that, in some case, it's safe to put 💩 there"

OR

"In every case, it's unsafe to put 💩 there"

💩

<Gradual Rob> 🤖

What Do You Want?

```
if(x){💩}{1} + 3
```

💩

<Dynamic Rob> 🤖

What Do You Want?

```
if(x){💩}{1} + 3
```

“F*** it we’ll do it live! 👯” -Bill O’Reilly

💩

<Dynamic Rob> 🤖

Breakdown

Breakdown

Static		
Static Guarantee		
Verification		
"∀" safety		
"∃" rejection		

Breakdown

Static	Gradual
Static Guarantee	Static Guarantee
Verification	Bug Finding
“$\forall$” safety	“$\exists$” safety
“$\exists$” rejection	“$\forall$” rejection

Breakdown

Static	**Gradual**	**Dynamic**
Static Guarantee	Static Guarantee	lol
Verification	Bug Finding	lol
“∀” safety	“∃” safety	lol
“∃” rejection	“∀” rejection	lol

AGT In A Nutshell

```
τ ∈ type  ⩴ 𝔹 | τ→τ
e ∈  exp  ⩴ b | if(e){e}{e}
            | x | λ(x).e | e(e)
```

AGT In A Nutshell

$$\tau \in \text{type} ::= \mathbb{B} \mid \tau \to \tau$$

$$e \in \text{exp} ::= b \mid \underline{\texttt{if}}(e)\{e\}\{e\}$$

$$\mid x \mid \underline{\lambda}(x).e \mid e(e)$$

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\texttt{if}}(e_1)\{e_2\}\{e_3\} : \tau}\,[\mathbb{B}\text{-E}]$$

AGT In A Nutshell

$$
\begin{aligned}
\tau \in \text{type} &::= \mathbb{B} \mid \tau \to \tau \\
e \in \text{exp} &::= b \mid \underline{\texttt{if}}(e)\{e\}\{e\} \\
&\mid x \mid \underline{\lambda}(x).e \mid e(e)
\end{aligned}
$$

$$
\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\texttt{if}}(e_1)\{e_2\}\{e_3\} : \tau}\;[\mathbb{B}\text{-E}]
$$

$$
\frac{e_1 : \tau_1 \to \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\;[\to\text{-E}]
$$

AGT In A Nutshell

$$\tau \in \text{type}^\sharp ::= \mathbb{B} \mid \tau \to \tau \mid ?$$

$$e \in \text{exp} ::= b \mid \underline{\text{if}}(e)\{e\}\{e\}$$

$$\mid x \mid \underline{\lambda}(x).e \mid e(e)$$

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\text{if}}(e_1)\{e_2\}\{e_3\} : \tau}\,[\mathbb{B}\text{-E}]$$

$$\frac{e_1 : \tau_1 \to \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\,[\to\text{-E}]$$

AGT In A Nutshell

$\tau \in \text{type}^\sharp ::= \mathbb{B} \mid \tau \rightarrow \tau \mid ?$

$e \in \text{exp}^\sharp ::= b \mid \underline{\text{if}}(e)\{e\}\{e\}$

$\mid x \mid \underline{\lambda}(x).e \mid e(e) \mid e :: \tau$

$$\frac{e_1 : \mathbb{B} \quad e_2 : \tau \quad e_3 : \tau}{\underline{\text{if}}(e_1)\{e_2\}\{e_3\} : \tau}\ [\mathbb{B}\text{-E}]$$

$$\frac{e_1 : \tau_1 \rightarrow \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\ [\rightarrow\text{-E}]$$

AGT In A Nutshell

$$\tau \in \text{type}^\sharp ::= \mathbb{B} \mid \tau \to \tau \mid ?$$

$$e \in \text{exp}^\sharp ::= b \mid \underline{\text{if}}(e)\{e\}\{e\}$$

$$\mid x \mid \underline{\lambda}(x).e \mid e(e) \mid e :: \tau$$

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \mathbb{B} \quad e_2 : \tau_2 \quad e_3 : \tau_3}{\underline{\text{if}}(e_1)\{e_2\}\{e_3\} : \tau_2 \vee \tau_3}\,[\mathbb{B}\text{-E}]$$

$$\frac{e_1 : \tau_1 \to \tau_2 \quad e_2 : \tau_1}{e_1(e_2) : \tau_2}\,[\to\text{-E}]$$

AGT In A Nutshell

$$\tau \in \mathrm{type}^\sharp ::= \mathbb{B} \mid \tau \to \tau \mid ?$$

$$\begin{aligned} e \in \mathrm{exp}^\sharp ::= {} & b \mid \underline{\mathtt{if}}(e)\{e\}\{e\} \\ & \mid x \mid \underline{\lambda}(x).e \mid e(e) \mid e :: \tau \end{aligned}$$

$$\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \mathbb{B} \\ e_2 : \tau_2 & \\ e_3 : \tau_3 & \end{array}}{\underline{\mathtt{if}}(e_1)\{e_2\}\{e_3\} : \tau_2 \vee \tau_3}\,[\mathbb{B}\text{-E}]$$

$$\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \tau_{11} \to \tau_{21} \\ e_2 : \tau_2 & \tau_2 \sim \tau_{11} \end{array}}{e_1(e_2) : \tau_{21}}\,[\to\text{-E}]$$

Plausibility

$$\dfrac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \tau_{11} \to \tau_{21} \\ e_2 : \tau_2 & \tau_2 \sim \tau_{11} \end{array}}{e_1(e_2) : \tau_{21}}\ [\to\text{-E}]$$

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}} [\rightarrow\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \rightarrow \tau_{21}$"

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\,[\rightarrow\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \rightarrow \tau_{21}$"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \mathbin{⸬} \tau_2) : \tau_2}\,[⸬\text{-I}]$$

Plausibility

$$\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \tau_{11} \to \tau_{21} \\ e_2 : \tau_2 & \tau_2 \sim \tau_{11} \end{array}}{e_1(e_2) : \tau_{21}} [\to\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \to \tau_{21}$"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \,\text{°°}\, \tau_2) : \tau_2} [\text{°°-I}]$$

"I claim **e** *might* have type τ_2"

Plausibility

$$\frac{e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \qquad e_2 : \tau_2 \quad \tau_2 \sim \tau_{11}}{e_1(e_2) : \tau_{21}}\,[\rightarrow\text{-E}]$$

"It's plausible that e_1 has some arrow type $\tau_{11} \rightarrow \tau_{21}$"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \mathrel{⸬} \tau_2) : \tau_2}\,[⸬\text{-I}]$$

"I claim e *might* have type τ_2"

$$\frac{}{? \sim \tau} \qquad \frac{}{\tau \sim ?}$$

Plausibility

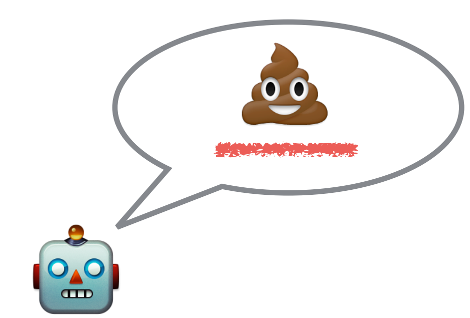
$$\frac{\begin{array}{ll} e_1 : \tau_1 & \tau_1 \sim \tau_{11} \to \tau_{21} \\ e_2 : \tau_2 & \tau_2 \sim \tau_{11} \end{array}}{e_1(e_2) : \tau_{21}}[\to\text{-E}]$$

"It's plausible that **e_1** has some arrow type **$\tau_{11} \to \tau_{21}$**"

$$\frac{e : \tau_1 \quad \tau_1 \sim \tau_2}{(e \mathrel{⦂} \tau_2) : \tau_2}[⦂\text{-I}]$$

"I claim **e** *might* have type **τ_2**"

$$\frac{}{? \sim \tau} \qquad \frac{}{\tau \sim ?}$$



<Gradual Rob>

"If you say so…"

Consistent Equality

$\boxed{g\tau \sim g\tau}$

Consistent Equality

$\boxed{g\tau \sim g\tau}$

"meaning" of a gradual type

$$[\![\_]\!] : \text{type}^\sharp \rightarrow \mathscr{P}(\text{type})$$
$$[\![\mathbb{B}]\!] \coloneqq \{\mathbb{B}\}$$
$$[\![g\tau_1 \rightarrow g\tau_2]\!] \coloneqq \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in [\![g\tau_1]\!] \wedge \tau_2 \in [\![g\tau_2]\!]\}$$
$$[\![?]\!] \coloneqq \{\tau \mid \tau \in \text{type}\}$$

Consistent Equality

$\boxed{g\tau \sim g\tau}$

"meaning" of a gradual type

$[\![\_]\!] : \mathrm{type}^{\sharp} \to \mathscr{P}(\mathrm{type})$
$[\![\mathbb{B}]\!] \coloneqq \{\mathbb{B}\}$
$[\![g\tau_1 \to g\tau_2]\!] \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in [\![g\tau_1]\!] \wedge \tau_2 \in [\![g\tau_2]\!]\}$
$[\![?]\!] \coloneqq \{\tau \mid \tau \in \mathrm{type}\}$

consistent equalities are "plausibilities"

$$\frac{\tau_1 \in [\![g\tau_1]\!] \quad \tau_2 \in [\![g\tau_2]\!] \quad \tau_1 = \tau_2}{g\tau_1 \sim g\tau_2}$$

The Whole AGT Story

- The "meaning" function $[\![\_]\!]$ forms a Galois connection between precise and gradual types.
- Guided by the Galois connection, define consistent equality and derive dynamic and static semantics.
- "Semantics design by abstract interpretation."

Formally Verifying Derived Gradual Type Systems

$e : ?$

gradual types

$\vdash \forall(x).P(x)$

formal verification

$\mathbb{Z} \rightleftarrows \{-,0,+\}$

abstract interpretation

Formally Verifying Derived Gradual Type Systems

$\mathbb{Z} \rightleftarrows \{\texttt{-},\texttt{0},\texttt{+}\} \longleftrightarrow \vdash \forall(x).P(x)$

abstract interpretation *formal verification*

- **Challenge:**
  Galois connections are problematic in formal verification

- **Insight:**
  Isolate the problem with a meta- “specification” effect

Galois Connections

$$\llbracket \_ \rrbracket : \text{type}^\sharp \to \mathscr{P}(\text{type})$$

$$\llbracket \mathbb{B} \rrbracket \coloneqq \{\mathbb{B}\}$$

$$\llbracket g\tau_1 \to g\tau_2 \rrbracket \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \llbracket g\tau_1 \rrbracket \wedge \tau_2 \in \llbracket g\tau_2 \rrbracket\}$$

$$\llbracket ? \rrbracket \coloneqq \{\tau \mid \tau \in \text{type}\}$$

Galois Connections

$\gamma : \text{type}^\sharp \to \mathscr{P}(\text{type})$

$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$

$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \wedge \tau_2 \in \gamma(g\tau_2)\}$

$\gamma(?) \coloneqq \{\tau \mid \tau \in \text{type}\}$

Galois Connections

$$\gamma : \text{type}^\sharp \rightarrow \mathscr{P}(\text{type})$$

$$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$$
$$\gamma(g\tau_1 \rightarrow g\tau_2) \coloneqq \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\}$$
$$\gamma(?) \coloneqq \{\tau \mid \tau \in \text{type}\}$$

$$\alpha : \mathscr{P}(\text{type}) \rightarrow \text{type}^\sharp$$

$$\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$$

Galois Connections

$\gamma : \mathrm{type}^\sharp \to \mathscr{P}(\mathrm{type})$

$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$

$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \wedge \tau_2 \in \gamma(g\tau_2)\}$

$\gamma(?) \coloneqq \{\tau \mid \tau \in \mathrm{type}\}$

$\alpha : \mathscr{P}(\mathrm{type}) \to \mathrm{type}^\sharp$

$\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$

$\eta : \mathrm{type} \to \mathrm{type}^\sharp$

$\eta(\mathbb{B}) \coloneqq \mathbb{B}$

$\eta(\tau_1 \to \tau_2) \coloneqq \eta(\tau_1) \to \eta(\tau_2)$

$$\tau_1 \sqcup \tau_2 \coloneqq \begin{cases} ? & \textit{when } \tau_1 \neq \tau_2 \\ \tau_1 & \textit{when } \tau_1 = \tau_2 \end{cases}$$

Galois Connections

$\gamma : \text{type}^\sharp \to \mathscr{P}(\text{type})$
$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$
$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \wedge \tau_2 \in \gamma(g\tau_2)\}$
$\gamma(?) \coloneqq \{\tau \mid \tau \in \text{type}\}$

Non-constructive	***Constructive***
$\alpha : \mathscr{P}(\text{type}) \to \text{type}^\sharp$ $\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$	$\eta : \text{type} \to \text{type}^\sharp$ $\eta(\mathbb{B}) \coloneqq \mathbb{B}$ $\eta(\tau_1 \to \tau_2) \coloneqq \eta(\tau_1) \to \eta(\tau_2)$ $\tau_1 \sqcup \tau_2 \coloneqq ?$ *when* $\tau_1 \neq \tau_2$ τ_1 *when* $\tau_1 = \tau_2$

Galois Connections

"specification effect"

$$\gamma : \mathrm{type}^\sharp \to \mathscr{P}(\mathrm{type})$$
$$\gamma(\mathbb{B}) \coloneqq \{\mathbb{B}\}$$
$$\gamma(g\tau_1 \to g\tau_2) \coloneqq \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\}$$
$$\gamma(?) \coloneqq \{\tau \mid \tau \in \mathrm{type}\}$$

Non-constructive	***Constructive***
$\alpha : \mathscr{P}(\mathrm{type}) \to \mathrm{type}^\sharp$ $\alpha(\{\tau_1 .. \tau_n\}) \coloneqq \bigsqcup_i \eta(\tau_i)$	$\eta : \mathrm{type} \to \mathrm{type}^\sharp$ $\eta(\mathbb{B}) \coloneqq \mathbb{B}$ $\eta(\tau_1 \to \tau_2) \coloneqq \eta(\tau_1) \to \eta(\tau_2)$ $\tau_1 \sqcup \tau_2 \coloneqq ?$ *when* $\tau_1 \neq \tau_2$ τ_1 *when* $\tau_1 = \tau_2$

In Agda

In Agda

```
data _∈γ[_] : type → type♯ → Set where
  ⟨𝔹⟩ : ⟨𝔹⟩ ∈γ[ ⟨𝔹⟩ ]
  _⟨→⟩_ : ∀ {τ₁♯ τ₂♯ τ₁ τ₂}
    → τ₁ ∈γ[ τ₁♯ ]
    → τ₂ ∈γ[ τ₂♯ ]
    → (τ₁ ⟨→⟩ τ₂) ∈γ[ τ₁♯ ⟨→⟩ τ₂♯ ]
  ⟨?⟩ : ∀ {τ} → τ ∈γ[ ⟨?⟩ ]
```

In Agda

```
data _∈γ[_] : type → type♯ → Set where
  ⟨𝔹⟩ : ⟨𝔹⟩ ∈γ[ ⟨𝔹⟩ ]
  _⟨→⟩_ : ∀ {τ₁♯ τ₂♯ τ₁ τ₂}
    → τ₁ ∈γ[ τ₁♯ ]
    → τ₂ ∈γ[ τ₂♯ ]
    → (τ₁ ⟨→⟩ τ₂) ∈γ[ τ₁♯ ⟨→⟩ τ₂♯ ]
  ⟨?⟩ : ∀ {τ} → τ ∈γ[ ⟨?⟩ ]
```

```
η : type → type♯
η(⟨𝔹⟩) = ⟨𝔹⟩
η(τ₁ ⟨→⟩ τ₂) = η(τ₁) ⟨→⟩ η(τ₂)
```

In Agda

```
data _∈γ[_] : type → type♯ → Set where
  ⟨𝔹⟩ : ⟨𝔹⟩ ∈γ[ ⟨𝔹⟩ ]
  _⟨→⟩_ : ∀ {τ₁♯ τ₂♯ τ₁ τ₂}
    → τ₁ ∈γ[ τ₁♯ ]
    → τ₂ ∈γ[ τ₂♯ ]
    → (τ₁ ⟨→⟩ τ₂) ∈γ[ τ₁♯ ⟨→⟩ τ₂♯ ]
  ⟨?⟩ : ∀ {τ} → τ ∈γ[ ⟨?⟩ ]
```

- OCaml: Datatype
- Math: Inductive Judgment

```
η : type → type♯
η(⟨𝔹⟩) = ⟨𝔹⟩
η(τ₁ ⟨→⟩ τ₂) = η(τ₁) ⟨→⟩ η(τ₂)
```

- OCaml: Function
- Math: Computable Function

Constructive Galois Connections

- Extracting verified computation from proof assistants is based on *constructive logic*

- Problem: classical Galois connections are nonconstructive

- Solution: design a constructive variant of Galois connections and use those

- Bonus: simpler proofs (η is simpler than α)

Formally Verifying Derived Gradual Type Systems

$e : ?$

gradual types

$\vdash \forall(x).P(x)$

formal verification

$\mathbb{Z} \rightleftarrows \{-,0,+\}$

abstract interpretation

Formally Verifying Derived Gradual Type Systems

$e : ?$

gradual types

$\vdash \forall(x).P(x)$

formal verification

$\mathbb{Z} \rightleftarrows \{-,0,+\}$

abstract interpretation

What I Did

- 1. Formally verified gradual type system in AGT
- 2. Simplified some proofs by using η instead of α

“Simplified” How?

$$
\begin{aligned}
&\mathrm{correct}[cod^{\sharp}]/\eta\eta \;:\; \forall\,(\tau \;:\; type) \rightarrow \eta^{t}\cdot(cod\cdot\tau) \;\equiv\; cod^{\sharp}\cdot(\eta^{t}\cdot\tau)\\
&\mathrm{correct}[cod^{\sharp}]/\eta\eta \;\bot \;=\; \mathrm{refl}\\
&\mathrm{correct}[cod^{\sharp}]/\eta\eta \;\langle\mathbb{B}\rangle \;=\; \mathrm{refl}\\
&\mathrm{correct}[cod^{\sharp}]/\eta\eta \;(\tau_1 \;\langle\rightarrow\rangle\; \tau_2) \;=\; \mathrm{refl}
\end{aligned}
$$

vs

$$
\begin{aligned}
&\mathrm{correct}[cod^{\sharp}]/\eta\gamma : \forall\;\tau^{\sharp} \rightarrow (\mathrm{pure}\cdot\eta^{t}) * \cdot ((\mathrm{pure}\cdot cod) * \cdot (\gamma^{ct}\cdot\tau^{\sharp}))\\
&\qquad\qquad\qquad \sqsubseteq \mathrm{pure}\cdot cod^{\sharp}\cdot\tau^{\sharp}\\
&\mathrm{correct}[cod^{\sharp}]/\eta\gamma\;\;\tau^{\sharp} \;=\; \mathrm{extensionality}[\mathcal{P}]\;\;(Q\;\;\tau^{\sharp}) \quad \text{where}\\
&\quad Q \;:\; \forall\;\tau_1^{\sharp}\;\tau_2^{\sharp} \rightarrow \tau_2^{\sharp} \in (\mathrm{pure}\cdot\eta^{t}) * \cdot ((\mathrm{pure}\cdot cod) * \cdot (\gamma^{ct}\cdot\tau_1^{\sharp}))\\
&\qquad\qquad \rightarrow \tau_2^{\sharp} \in \mathrm{pure}\cdot cod^{\sharp}\cdot\tau_1^{\sharp}\\
&\quad Q\;\_\;.\bot\;(\exists\mathcal{P}\;.\bot\;,\;(\exists\mathcal{P}\;.\bot\;,\;\bot\;,\;\bot)\;,\;\bot) \;=\; \bot\\
&\quad Q\;.\top\;\_\;(\exists\mathcal{P}\;\_\;,\;(\exists\mathcal{P}\;\_\;,\;\top\;,\;\_)\;,\;\_) \;=\; \top\\
&\quad Q\;.\langle\mathbb{B}\rangle\;.\bot\;(\exists\mathcal{P}\;.\bot\;,\;(\exists\mathcal{P}\;.\langle\mathbb{B}\rangle\;,\;\langle\mathbb{B}\rangle\;,\;\bot)\;,\;\bot) \;=\; \bot\\
&\quad Q\;(\_\;\langle\rightarrow\rangle\;\tau_2^{\sharp})\;\tau_1^{\sharp}\;(\exists\mathcal{P}\;\tau_1\;,\;(\exists\mathcal{P}\;(\_\;\langle\rightarrow\rangle\;\tau_2)\\
&\qquad\quad ,\;(\_\;\langle\rightarrow\rangle\;\tau_2{\in}\gamma[\tau_2^{\sharp}])\;,\;\tau_1{\sqsubseteq}\tau_2)\;,\;\tau_1^{\sharp}{\sqsubseteq}\eta[\tau_1]) \;=\\
&\qquad \mathrm{complete}^{ct}\;\;\tau_2{\in}\gamma[\tau_2^{\sharp}] \circledcirc \text{respectful-arg}\;\tau_1{\sqsubseteq}\tau_2 \circledcirc \tau_1^{\sharp}{\sqsubseteq}\eta[\tau_1]
\end{aligned}
$$

$\times\, 2$

Going Forward

- I'm interested in applying verified AGT technique to type systems with blame and type polymorphism.
    - Combination is currently an open problem in PL
- I'm interested in verified static analysis frameworks building on constructive Galois connections.

Takeaways

- Gradual type systems are dual to precise ones: allow when success guaranteed vs allow when success plausible.
- If you want to “understand” gradual type systems in the abstract, read the AGT paper [Garcia,Clark,Tanter;2016].
- Designing a gradual type system is fundamentally hard, but there is a method to the madness.
- If you want to use Galois connections in a formal development (Coq/Agda), read the Constructive GCs paper [Darais,Van Horn;2015 Draft].