Formally Verifying and Deriving Gradual Type Systems

David Darais
University of Maryland
Topics
Topics

e : ?

gradual types
Topics

e : ?

gradual types

\( \forall (x) \cdot P(x) \)

formal verification
Topics

e : ?

gradual types

⊢ ∀(x).P(x)

formal verification

\( \mathbb{Z} \leftrightarrow \{-, \theta, +\} \)

abstract interpretation
Topics

e : ?

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

gradual types

\[ \mathbb{Z} \cong \{ - , 0 , + \} \]

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

\[ \forall (x) . P(x) \]

abstract interpretation

formal verification
Topics

e : ?

gradual types

 overhaul types

\( \mathbb{Z} \cong \{ - , \emptyset , + \} \)

abstract interpretation

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

formal verification
Deriving Gradual Type Systems

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

- **Insight:**
  Guide design through abstract interpretation
Precise Types (Static)
Precise Types (Static)
Precise Types (Static)

⊢ 1 + 5 : int ✓

STATICS

DYNAMICS
Precise Types (Static)

⊢ 1 + 5 : int ✓
⊢ 🕦 + 5 : _ ✗

STATICS

DYNAMICS
Precise Types (Static)

\[ \vdash 1 + 5 : \text{int} \quad \checkmark \]
\[ \vdash \text{💩} + 5 : \_ \quad \times \]

\[ \vdash 1 + 5 \downarrow 6 \quad \checkmark \]
Precise Types (Static)

STATICS

\[ \vdash 1 + 5 : \text{int} \quad \checkmark \]
\[ \vdash \text{💩} + 5 : \_ \quad \xmark \]

DYNAMICS

\[ \vdash 1 + 5 \downarrow 6 \quad \checkmark \]

<what's missing?>
Precise Types (Dynamic)
Precise Types (Dynamic)

⊢ 1 + 5 : 🙈 ✓

STATICS

DYNAMICS
Precise Types (Dynamic)

⊢ 1 + 5 : 🧐 ✓

⊢ 🍎 + 5 : 🧐 ✓
Precise Types (Dynamic)

**STATICs**

\[ \vdash 1 + 5 : \text{emoji} \]

\[ \vdash \text{poop emoji} + 5 : \text{monkey emoji} \]

**DYNAMICS**

\[ \vdash 1 + 5 \Downarrow 6 \]
Precise Types (Dynamic)

STATICS

⊢ 1 + 5 : 😊 ✓

⊢ 😳 + 5 : 😏 ✓

DYNAMICS

⊢ 1 + 5 \downarrow 6 ✓

⊢ 😳 + 5 \downarrow _ X
Gradual Types (Hybrid)
Gradual Types (Hybrid)

\[
\begin{align*}
\vdash & \text{ poop} + 5 : \_ X
\end{align*}
\]
Gradual Types (Hybrid)

**Statics**

\[ \vdash \, \text{펑} + 5 : \_ \quad \times \]

\[ \vdash (\text{💩} : \text{‽}) + 5 : \text{int} \quad \checkmark \]

**Dynamics**
Gradual Types (Hybrid)

**STATICS**

\[ \vdash \text{💩} + 5 : \_ \quad \times \]
\[ \vdash (\text{💩} : ?) + 5 : \text{int} \quad \checkmark \]

**DYNAMICS**

\[ \vdash (\text{💩} : ?) + 5 \quad \downarrow \quad \_ \quad \times \]
Gradual Types (Hybrid)

**STATICS**

\[ \Gamma \vdash \text{💩} + 5 : \_ \quad \times \]

\[ \Gamma \vdash (\text{💩} : ?) + 5 : \text{int} \quad \checkmark \]

**DYNAMICS**

\[ \Gamma \vdash (\text{💩} : ?) + 5 \Downarrow \_ \quad \times \]

*what’s missing?>
“What Do You Want?”
What Do You Want?

12: \text{if}(x)\{\text{💩}\}\{1\} + 3
What Do You Want?

12: \texttt{if} (x) \{ \text{😊} \} \{1\} + 3
What Do You Want?

12: \( \text{if}(x)\{\text{💩}\}\{1\} + 3 \)

"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"
What Do You Want?

12: \texttt{if(x)\{\text{ emojis}\}\{1\} + 3}

<Gradual Rob>
What Do You Want?

"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?

12: \texttt{if(x)\{-duration\}\{1\} + 3}

<Dynamic Rob>
What Do You Want?

12: `if(x) {💩} {1} + 3`

“F*** it we’ll do it live! 🦇” -Bill O’Reilly
Breakdown
<table>
<thead>
<tr>
<th>Static Guarantee</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>“∀” safety</td>
<td>“∃” rejection</td>
</tr>
<tr>
<td>Static Guarantee</td>
<td>Gradual Guarantee</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Verification</td>
<td>Bug Finding</td>
</tr>
<tr>
<td>“∀” safety</td>
<td>“∃” safety</td>
</tr>
<tr>
<td>“∃” rejection</td>
<td>“∀” rejection</td>
</tr>
</tbody>
</table>

**Breakdown**

- **Static**
  - Static Guarantee
  - Verification
  - “∀” safety
  - “∃” rejection

- **Gradual**
  - Static Guarantee
  - Bug Finding
  - “∃” safety
  - “∀” rejection
<table>
<thead>
<tr>
<th>Static</th>
<th>Gradual</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Guarantee</td>
<td>Static Guarantee</td>
<td>lol</td>
</tr>
<tr>
<td>Verification</td>
<td>Bug Finding</td>
<td>lol</td>
</tr>
<tr>
<td>“∀” safety</td>
<td>“∃” safety</td>
<td>lol</td>
</tr>
<tr>
<td>“∃” rejection</td>
<td>“∀” rejection</td>
<td>lol</td>
</tr>
</tbody>
</table>
AGT In A Nutshell

\[ \tau \in \text{type} \quad ::= \quad \text{B} \mid \tau \rightarrow \tau \]

\[ e \in \text{exp} \quad ::= \quad \text{b} \mid \text{if}(e)\{e\}\{e\} \]
\[ \mid x \mid \lambda(x).e \mid e(e) \]
AGT In A Nutshell

\[ \tau \in \text{type} \; \equiv \; B \mid \tau \rightarrow \tau \]
\[ e \in \text{exp} \; \equiv \; b \mid \text{if}(e) \{e\}\{e\} \]
\[ \mid x \mid \lambda(x).e \mid e(e) \]

\[ e_1 : B \]
\[ e_2 : \tau \]
\[ e_3 : \tau \]

\[ \frac{}{if(e_1)\{e_2\}\{e_3\} : \tau} \quad [B\text{-}E] \]
AGT In A Nutshell

\[\tau \in \text{type} \equiv \mathbb{B} \mid \tau \to \tau\]
\[e \in \text{exp} \equiv b \mid \text{if}(e)\{e\}\{e\} \mid x \mid \lambda(x).e \mid e(e)\]

\[
\begin{align*}
e_1 : \mathbb{B} \\
e_2 : \tau \\
e_3 : \tau \\
\end{align*}
\]

\[
\begin{align*}
e_1 : \mathbb{B} \\
e_2 : \tau \\
e_3 : \tau \\
\end{align*}
\]

\[
\begin{align*}
e_1 (e_2) : \tau_2 \\
e_2 : \tau_1 \\
\end{align*}
\]

\[
\begin{align*}
e_1 : \tau_1 \to \tau_2 \\
e_2 : \tau_1 \\
\end{align*}
\]

\[
\begin{align*}
e_1 (e_2) : \tau_2 \\
\end{align*}
\]

\[
\begin{align*}
\text{if}(e_1)\{e_2\}\{e_3\} : \tau \\
\end{align*}
\]

\[
\begin{align*}
\text{if}(e_1)\{e_2\}\{e_3\} : \tau \\
\end{align*}
\]

\[
\begin{align*}
\text{if}(e_1)\{e_2\}\{e_3\} : \tau \\
\end{align*}
\]
AGT In A Nutshell

\[ \begin{align*}
\tau \in \text{type} & \quad ::= \ B \mid \tau \rightarrow \tau \mid ? \\
e \in \text{exp} & \quad ::= b \mid \text{if}(e)\{e\}\{e\} \\
 & \quad \mid x \mid \lambda(x).e \mid e(e)
\end{align*} \]

\[
\begin{align*}
e_1 : B \\
e_2 : \tau \\
e_3 : \tau \\
\hline \\
\text{if}(e_1)\{e_2\}\{e_3\} : \tau
\end{align*}
\]

\[
\begin{align*}
e_1 : \tau_1 \rightarrow \tau_2 \\
e_2 : \tau_1 \\
\hline \\
\hline \\
e_1(e_2) : \tau_2
\end{align*}
\]
AGT In A Nutshell

\[ \tau \in \text{type}^* = B \mid \tau \rightarrow \tau \mid ? \]
\[ e \in \text{exp}^* = b \mid \text{if}(e){e}{e} \]
\[ \mid x \mid \lambda(x).e \mid e(e) \mid e^\tau \]

\[ e_1 : B \]
\[ e_2 : \tau \]
\[ e_3 : \tau \]

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

\[ e_1 : \tau_1 \rightarrow \tau_2 \]
\[ e_2 : \tau_1 \]

\[ \frac{}{\text{[\(-E\)]} \quad e_1(e_2) : \tau_2} \]
AGT In A Nutshell

\[ \tau \in \text{type}^\# := \mathbb{B} \mid \tau \rightarrow \tau \mid ? \]

\[ e \in \text{exp}^\# := b \mid \text{if}(e)\{e\}\{e\} \]

\[ \quad \mid x \mid \lambda(x).e \mid e(e) \mid e^\circ \tau \]

\[
\begin{align*}
e_1 : \tau_1 & \quad \tau_1 \sim \mathbb{B} \\
e_2 : \tau_2 \\
e_3 : \tau_3 \\
\hline
\text{if}(e_1)\{e_2\}\{e_3\} : \tau_2 \lor \tau_3
\end{align*}
\]

\[
\begin{align*}
e_1 : \tau_1 \rightarrow \tau_2 & \\
e_2 : \tau_1 & \\
\hline
\hline
\text{if}(e_2): [\rightarrow \text{E}] \\
e_1(e_2) : \tau_2
\end{align*}
\]
AGT In A Nutshell

\[ \tau \in \text{type}^\# \equiv B | \tau \rightarrow \tau | ? \]

\[ e \in \text{exp}^\# \equiv b | \textbf{if}(e)\{e\}\{e\} | x | \lambda(x).e | e(e) | e^e \tau \]

\[ e_1 : \tau_1 \quad \tau_1 \sim B \]
\[ e_2 : \tau_2 \]
\[ e_3 : \tau_3 \]

\[ \text{if}(e_1)\{e_2\}\{e_3\} : \tau_2 \lor \tau_3 \]

\[ e_1 : \tau_1 \quad \tau_1 \sim \tau_1 \rightarrow \tau_{21} \]
\[ e_2 : \tau_2 \quad \tau_2 \sim \tau_{11} \]

\[ e_1(e_2) : \tau_{21} \]
Plausibility

\[ e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \]
\[ e_2 : \tau_2 \quad \tau_2 \sim \tau_{11} \]

\[ \text{[\rightarrow-E]} \]
\[ e_1(e_2) : \tau_{21} \]
Plausibility

\[ e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \]
\[ e_2 : \tau_2 \quad \tau_2 \sim \tau_{11} \]

\[ \text{[\xrightarrow{\rightarrow-E}]} \]

\[ e_1(e_2) : \tau_{21} \]

“It’s plausible that \( e_1 \) has some arrow type \( \tau_{11} \rightarrow \tau_{21} \)”
Plausibility

\[
\begin{align*}
e_1 : \tau_1 & \quad \tau_1 \sim \tau_{11} \to \tau_{21} \\
e_2 : \tau_2 & \quad \tau_2 \sim \tau_{11} \\
\end{align*}
\]

\[\emptyarg \to \mathbf{-E} \]

\[
e_1(e_2) : \tau_{21}
\]

“It’s plausible that \( e_1 \) has some arrow type \( \tau_{11} \to \tau_{21} \)”

\[
\begin{align*}
e : \tau_1 & \quad \tau_1 \sim \tau_2 \\
\end{align*}
\]

\[\emptyarg \to \mathbf{-I} \]

\[
(e : \tau_2) : \tau_2
\]
Plausibility

\[ e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \]
\[ e_2 : \tau_2 \quad \tau_2 \sim \tau_{11} \]

\[ \frac{\text{[\(\rightarrow\)-E]}}{e_1(e_2) : \tau_{21}} \]

"It’s plausible that \( e_1 \) has some arrow type \( \tau_{11} \rightarrow \tau_{21} \)"

\[ e : \tau_1 \quad \tau_1 \sim \tau_2 \]

\[ \frac{\text{[\(\bowtie\)-I]}}{(e : \tau_2) : \tau_2} \]

"I claim \( e \) might have type \( \tau_2 \)"
Plausibility

\[ e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \]
\[ e_2 : \tau_2 \quad \tau_2 \sim \tau_{11} \]

\[ \frac{}{e_1(e_2) : \tau_{21}} \rightarrow \text{-} \text{E} \]

“It’s plausible that \( e_1 \) has some arrow type \( \tau_{11} \rightarrow \tau_{21} \)”

\[ e : \tau_1 \quad \tau_1 \sim \tau_2 \]

\[ \frac{}{(e : \tau_2) : \tau_2} \phi \text{-} \text{I} \]

“I claim \( e \) might have type \( \tau_2 \)”

\[ ? \sim \tau \quad \tau \sim ? \]
Plausibility

\[ e_1 : \tau_1 \quad \tau_1 \sim \tau_{11} \rightarrow \tau_{21} \]
\[ e_2 : \tau_2 \quad \tau_2 \sim \tau_{11} \]

\[ \vdash -E \]

\[ e_1(e_2) : \tau_{21} \]

“It’s plausible that \( e_1 \) has some arrow type \( \tau_{11} \rightarrow \tau_{21} \)”

\[ \vdash \top \]

\[ e : \tau_1 \quad \tau_1 \sim \tau_2 \]

\[ \vdash \top -I \]

\[ (e \top \tau_2) : \tau_2 \]

“I claim \( e \) might have type \( \tau_2 \)”

\[ \vdash \top \top \top \]

<Gradual Rob> 😑

“If you say so…”
Consistent Equality

g\tau \sim g\tau
Consistent Equality

$g\tau \sim g\tau$

“meaning” of a gradual type

\[
\begin{align*}
\mathbb{B} & : \text{type}^\# \to \mathcal{P}(\text{type}) \\
\mathbb{B} & = \{ \mathbb{B} \} \\
[g\tau_1 \rightarrow g\tau_2] & = \{ \tau_1 \rightarrow \tau_2 \mid \tau_1 \in [g\tau_1] \land \tau_2 \in [g\tau_2] \} \\
[?] & = \{ \tau \mid \tau \in \text{type} \}
\end{align*}
\]
Consistent Equality

$$g\tau \sim g\tau$$

"meaning" of a gradual type

$$[\_] : \text{type}^\# \rightarrow \mathcal{P}(\text{type})$$
$$[B] = \{B\}$$
$$[g\tau_1 \rightarrow g\tau_2] = \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in [g\tau_1] \land \tau_2 \in [g\tau_2]\}$$
$$[?] = \{\tau \mid \tau \in \text{type}\}$$

consistent equalities are "plausibilities"

$$\tau_1 \in [g\tau_1]$$
$$\tau_2 \in [g\tau_2]$$
$$\tau_1 = \tau_2$$

==========

$$g\tau_1 \sim g\tau_2$$
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

\( \mathbb{Z} \cong \{ -, 0, + \} \)
abstract interpretation

\( \vdash \forall (x). P(x) \)
formal verification
Formally Verifying Derived Gradual Type Systems

\[
\mathbb{Z} \leftrightarrow \{-, 0, +\} \quad \vdash \quad \forall (x) \cdot P(x)
\]

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

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

\[ \llbracket \_ \rrbracket : \text{type}^\sharp \to \mathcal{P}(\text{type}) \]
\[ \llbracket B \rrbracket = \{ B \} \]
\[ \llbracket g\tau_1 \to g\tau_2 \rrbracket = \{ \tau_1 \to \tau_2 \mid \tau_1 \in \llbracket g\tau_1 \rrbracket \land \tau_2 \in \llbracket g\tau_2 \rrbracket \} \]
\[ \llbracket ? \rrbracket = \{ \tau \mid \tau \in \text{type} \} \]
Galois Connections

\( \gamma : \text{type} \rightarrow \mathcal{P}(\text{type}) \)

\( \gamma(\emptyset) = \{\emptyset\} \)

\( \gamma(g\tau_1 \rightarrow g\tau_2) = \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\} \)

\( \gamma(?) = \{\tau \mid \tau \in \text{type}\} \)
Galois Connections

\[ \gamma : \text{type}^\sharp \rightarrow \mathcal{P}(\text{type}) \]
\[ \gamma(\mathbb{B}) = \{\mathbb{B}\} \]
\[ \gamma(g\tau_1 \rightarrow g\tau_2) = \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\} \]
\[ \gamma(?) = \{\tau \mid \tau \in \text{type}\} \]

\[ \alpha : \mathcal{P}(\text{type}) \rightarrow \text{type}^\sharp \]
\[ \alpha(\{\tau_1 \ldots \tau_n\}) = \bigcup_i \eta(\tau_i) \]
Galois Connections

\( \gamma : \text{type}^\# \to \mathcal{P}(\text{type}) \)

\( \gamma(\Box) = \{\Box\} \)

\( \gamma(\tau_1 \to \tau_2) = \{\tau_1 \to \tau_2 \mid \tau_1 \in \gamma(\tau_1) \land \tau_2 \in \gamma(\tau_2)\} \)

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

\( \alpha : \mathcal{P}(\text{type}) \to \text{type}^\# \)

\( \alpha(\{\tau_1 \ldots \tau_n\}) = \bigcup_i \eta(\tau_i) \)

\( \eta : \text{type} \to \text{type}^\# \)

\( \eta(\Box) = \Box \)

\( \eta(\tau_1 \to \tau_2) = \eta(\tau_1) \to \eta(\tau_2) \)

\( \tau_1 \sqcup \tau_2 = ? \quad \text{when} \quad \tau_1 \neq \tau_2 \)

\( \tau_1 \quad \text{when} \quad \tau_1 = \tau_2 \)
Galois Connections

\[ \gamma : \text{type}^\# \to \mathcal{P}(\text{type}) \]
\[ \gamma(\mathbb{B}) = \{\mathbb{B}\} \]
\[ \gamma(\tau_1 \rightarrow \tau_2) = \{\tau_1 \rightarrow \tau_2 \mid \tau_1 \in \gamma(\tau_1) \land \tau_2 \in \gamma(\tau_2)\} \]
\[ \gamma(?) = \{\tau \mid \tau \in \text{type}\} \]

<table>
<thead>
<tr>
<th>Non-constructive</th>
<th>Constructive</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \alpha : \mathcal{P}(\text{type}) \to \text{type}^# ]</td>
<td>[ \eta : \text{type} \to \text{type}^# ]</td>
</tr>
<tr>
<td>[ \alpha({\tau_1 .. \tau_n}) = \bigsqcup_i \eta(\tau_i) ]</td>
<td>[ \eta(\mathbb{B}) = \mathbb{B} ]</td>
</tr>
<tr>
<td></td>
<td>[ \eta(\tau_1 \rightarrow \tau_2) = \eta(\tau_1) \rightarrow \eta(\tau_2) ]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Galois Connections

#### “specification effect”

\[ \gamma : \text{type}^\# \to \mathcal{P}(\text{type}) \]

\[ \gamma(\mathbb{B}) = \{\mathbb{B}\} \]

\[ \gamma(g\tau_1 \to g\tau_2) = \{\tau_1 \to \tau_2 | \tau_1 \in \gamma(g\tau_1) \land \tau_2 \in \gamma(g\tau_2)\} \]

\[ \gamma(?) = \{\tau | \tau \in \text{type}\} \]

<table>
<thead>
<tr>
<th>Non-constructive</th>
<th>Constructive</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \alpha : \mathcal{P}(\text{type}) \to \text{type}^# ]</td>
<td>[ \eta : \text{type} \to \text{type}^# ]</td>
</tr>
<tr>
<td>[ \alpha({\tau_1 .. \tau_n}) = \bigsqcup_i \eta(\tau_i) ]</td>
<td>[ \eta(\mathbb{B}) = \mathbb{B} ]</td>
</tr>
<tr>
<td>[ \eta(\tau_1 \to \tau_2) = \eta(\tau_1) \to \eta(\tau_2) ]</td>
<td>[ \tau_1 \lor \tau_2 = ? \text{ when } \tau_1 \neq \tau_2 ]</td>
</tr>
<tr>
<td>[ \tau_1 \text{ when } \tau_1 = \tau_2 ]</td>
<td></td>
</tr>
</tbody>
</table>
In Agda
In Agda

```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

```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

\[
\text{data } _\in \gamma[\_] : \text{ type } \to \text{ type}\# \to \text{ Set } \quad \text{where}
\]
\[
\langle B \rangle : \langle B \rangle \in \gamma[\langle B \rangle ]
\]
\[
\_ (\to)_ : \forall \{\tau_1\# \tau_2\# \tau_1 \tau_2\}
\]
\[
\quad \to \tau_1 \in \gamma[\tau_1\# ]
\]
\[
\quad \to \tau_2 \in \gamma[\tau_2\# ]
\]
\[
\quad \to (\tau_1 (\to) \tau_2) \in \gamma[\tau_1\# (\to) \tau_2\# ]
\]
\[
\langle ? \rangle : \forall \{\tau\} \to \tau \in \gamma[\langle ? \rangle ]
\]

\[
\eta : \text{ type } \to \text{ type}\#
\]
\[
\eta(\langle B \rangle) = \langle B \rangle
\]
\[
\eta(\tau_1 (\to) \tau_2) = \eta(\tau_1) (\to) \eta(\tau_2)
\]

- OCaml: Datatype
- Math: Inductive Judgment
- 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 (\(\eta\) is simpler than \(\alpha\))
Formally Verifying Derived Gradual Type Systems

\[
e : ?
\]

gradual types

\[
\mathbb{Z} \equiv \{-, 0, +\}
\]

abstract interpretation

\[
\vdash \forall (x) . P(x)
\]

formal verification
Formally Verifying Derived Gradual Type Systems

e : ?

\( \equiv \{ - , 0 , + \} \)

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

gradual types

abstract interpretation

formal verification
What I Did

• 1. Formally verified gradual type system in AGT

• 2. Simplified some proofs by using \( \eta \) instead of \( \alpha \)
“Simplified” How?

\[
\text{correct}[\text{cod}^t] / \eta \eta : \forall (\tau : \text{type}) \rightarrow \eta^t \cdot (\text{cod} \cdot \tau) \equiv \text{cod}^t \cdot (\eta^t \cdot \tau)
\]

\[
\text{correct}[\text{cod}^t] / \eta \eta \perp = \text{refl}
\]

\[
\text{correct}[\text{cod}^t] / \eta \eta \langle \mathbb{B} \rangle = \text{refl}
\]

\[
\text{correct}[\text{cod}^t] / \eta \eta \langle \tau_1 \rightarrow \tau_2 \rangle = \text{refl}
\]

VS

\[
\text{correct}[\text{cod}^t] / \eta \eta : \forall \tau^t \rightarrow (\text{pure} \cdot \eta^t)^* \cdot ((\text{pure} \cdot \text{cod})^* \cdot (\eta^{ct} \cdot \tau^t))
\]

\[
\leq \text{pure} \cdot \text{cod}^t \cdot \tau^t
\]

\[
\text{correct}[\text{cod}^t] / \eta \eta \tau^t = \text{extensionality}[\mathcal{P}] (Q \tau^t) \quad \text{where}
\]

\[
Q : \forall \tau_1^t \tau_2^t \rightarrow \tau_2^t \in (\text{pure} \cdot \eta^t)^* \cdot ((\text{pure} \cdot \text{cod})^* \cdot (\eta^{ct} \cdot \tau_1^t))
\]

\[
\rightarrow \tau_2^t \in \text{pure} \cdot \text{cod}^t \cdot \tau_1^t
\]

Q _ \perp (\exists \mathcal{P} \perp, (\exists \mathcal{P} \perp, \perp, \perp)) = \perp

Q \cdot \perp (\exists \mathcal{P} \perp, (\exists \mathcal{P} \perp, \perp, \perp)) = \perp

Q \cdot \langle \mathbb{B} \rangle \cdot \perp (\exists \mathcal{P} \perp, (\exists \mathcal{P} \perp, \mathbb{B}, \perp)) = \perp

Q \cdot (\rightarrow \tau_2^t) \tau_1^t (\exists \mathcal{P} \perp, (\exists \mathcal{P} \perp, \rightarrow \tau_2))
\]

\[
\rightarrow (\rightarrow \tau_2^t \in \eta[\tau_2^t]), \tau_1 \in \tau_2) \rightarrow \tau_1 \in \eta[\tau_1]
\]

\[
\text{complete}^{ct} \tau_2 \in \eta[\tau_2^t] \odot \text{respectful-arg} \tau_1 \in \tau_2 \odot \tau_1 \in \eta[\tau_1]
\]

\[\times 2\]
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].