Gradual Intersection Types

Pedro Ângelo, Mário Florido

March 27, 2018

1 Language Definition

Syntax

$$\begin{array}{l} \mathit{Types}\ T ::= \ \mathit{Int} \mid \mathit{Bool} \mid T \to T \mid T \cap \ldots \cap T \\ \mathit{Expressions}\ e\ ::= x \mid \lambda x : T\ .\ e \mid e\ e \mid n \mid \mathit{true} \mid \mathit{false} \end{array}$$

$\boxed{\Gamma \vdash_{\cap S} e : T}$ Typing

$$\frac{x : T \in \Gamma}{\Gamma \vdash_{\cap S} x : T}\ \textsc{T-Var} \qquad \frac{\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap S}\ e : T}{\Gamma \vdash_{\cap S} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_1 \cap \ldots \cap T_n \to T}\ \textsc{T-Abs}$$

$$\frac{\Gamma, x : T_i \vdash_{\cap S}\ e : T}{\Gamma \vdash_{\cap S} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_i \to T}\ \textsc{T-Abs'}$$

$$\frac{\Gamma \vdash_{\cap S} e_1 : T_1 \cap \ldots \cap T_n \to T \qquad \Gamma \vdash_{\cap S} e_2 : T_1 \cap \ldots \cap T_n}{\Gamma \vdash_{\cap S} e_1\ e_2 : T}\ \textsc{T-App}$$

$$\frac{\Gamma \vdash_{\cap S} e : T_1\ \ \ldots\ \ \Gamma \vdash_{\cap S} e : T_n}{\Gamma \vdash_{\cap S} e : T_1 \cap \ldots \cap T_n}\ \textsc{T-Gen} \qquad \frac{\Gamma \vdash_{\cap S} e : T_1 \cap \ldots \cap T_n}{\Gamma \vdash_{\cap S} e : T_i}\ \textsc{T-Inst} \qquad \frac{}{\Gamma \vdash_{\cap S} n : \mathit{Int}}\ \textsc{T-Int}$$

$$\frac{}{\Gamma \vdash_{\cap S} \mathit{true} : \mathit{Bool}}\ \textsc{T-True} \qquad \frac{}{\Gamma \vdash_{\cap S} \mathit{false} : \mathit{Bool}}\ \textsc{T-False}$$

Figure 1: Static Intersection Type System ($\vdash_{\cap S}$)

Syntax

$$\begin{aligned}
&\mathit{Types}\ \ T ::=\ \mathit{Int} \mid \mathit{Bool} \mid \mathit{Dyn} \mid T \to T \mid T \cap \ldots \cap T \\
&\mathit{Expressions}\ \ e\ \ ::= x \mid \lambda x : T\ .\ e \mid e\ e \mid n \mid \mathit{true} \mid \mathit{false}
\end{aligned}$$

$\boxed{\Gamma \vdash_{\cap G} e : T}$ Typing

$$\frac{x : T \in \Gamma}{\Gamma \vdash_{\cap G} x : T}\ \text{T-Var} \qquad \frac{\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap G}\ e : T}{\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_1 \cap \ldots \cap T_n \to T}\ \text{T-Abs}$$

$$\frac{\Gamma, x : T_i \vdash_{\cap G}\ e : T}{\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_i \to T}\ \text{T-Abs'}$$

$$\frac{\begin{array}{cc}\Gamma \vdash_{\cap G} e_1 : PM & PM \rhd T_1 \cap \ldots \cap T_n \to T \\ \Gamma \vdash_{\cap G} e_2 : T_1' \cap \ldots \cap T_n' & T_1' \cap \ldots \cap T_n' \sim T_1 \cap \ldots \cap T_n\end{array}}{\Gamma \vdash_{\cap G} e_1\ e_2 : T}\ \text{T-App}$$

$$\frac{\Gamma \vdash_{\cap G} e : T_1\ \ \ldots\ \ \Gamma \vdash_{\cap G} e : T_n}{\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n}\ \text{T-Gen} \qquad \frac{\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n}{\Gamma \vdash_{\cap G} e : T_i}\ \text{T-Inst}$$

$$\frac{}{\Gamma \vdash_{\cap G} n : \mathit{Int}}\ \text{T-Int} \qquad \frac{}{\Gamma \vdash_{\cap G} \mathit{true} : \mathit{Bool}}\ \text{T-True} \qquad \frac{}{\Gamma \vdash_{\cap G} \mathit{false} : \mathit{Bool}}\ \text{T-False}$$

$\boxed{T \sim T}$ Consistency

$$T \sim T \qquad T \sim \mathit{Dyn} \qquad \mathit{Dyn} \sim T \qquad \frac{T_1 \sim T_3 \qquad T_2 \sim T_4}{T_1 \to T_2 \sim T_3 \to T_4} \qquad \frac{T_1 \sim T_1'\ \ \ldots\ \ T_n \sim T_n'}{T_1 \cap \ldots \cap T_n \sim T_1' \cap \ldots \cap T_n'}$$

$\boxed{T \rhd T}$ Pattern Matching

$$T_1 \to T_2 \rhd T_1 \to T_2 \qquad\qquad \mathit{Dyn} \rhd \mathit{Dyn} \to \mathit{Dyn}$$

Figure 2: Gradual Intersection Type System ($\vdash_{\cap G}$)

$\boxed{T \sqsubseteq T}$ Type Precision

$$Dyn \sqsubseteq T \qquad T \sqsubseteq T \qquad \frac{T_1 \sqsubseteq T_3 \qquad T_2 \sqsubseteq T_4}{T_1 \to T_2 \sqsubseteq T_3 \to T_4} \qquad \frac{T_1 \sqsubseteq T_1' \ \ldots \ T_n \sqsubseteq T_n'}{T_1 \cap \ldots \cap T_n \sqsubseteq T_1' \cap \ldots \cap T_n'}$$

$$\frac{T \sqsubseteq T_1 \ \ldots \ T \sqsubseteq T_n}{T \sqsubseteq T_1 \cap \ldots \cap T_n} \qquad \frac{T_1 \sqsubseteq T \ \ldots \ T_n \sqsubseteq T}{T_1 \cap \ldots \cap T_n \sqsubseteq T}$$

$\boxed{c \sqsubseteq c}$ Cast Precision

$$\frac{c \sqsubseteq c' \qquad T_1 \sqsubseteq T_1' \qquad T_2 \sqsubseteq T_2'}{c : T_1 \Rightarrow^{l} T_2 \ ^{cl} \sqsubseteq c' : T_1' \Rightarrow^{l'} T_2' \ ^{cl'}} \qquad \frac{c \sqsubseteq c' \qquad \vdash_{\cap CI} c' : T \qquad T_1 \sqsubseteq T \qquad T_2 \sqsubseteq T}{c : T_1 \Rightarrow^{l} T_2 \ ^{cl} \sqsubseteq c'}$$

$$\frac{c \sqsubseteq c' \qquad \vdash_{\cap CI} c : T \qquad T \sqsubseteq T_1 \qquad T \sqsubseteq T_2}{c \sqsubseteq c' : T_1 \Rightarrow^{l} T_2 \ ^{cl}} \qquad \frac{T_I \sqsubseteq T_I' \qquad T_F \sqsubseteq T_F'}{blame\ T_I\ T_F\ l\ ^{cl} \sqsubseteq blame\ T_I'\ T_F'\ l'\ ^{cl'}}$$

$$\frac{T \sqsubseteq T'}{\varnothing\ T\ ^{cl} \sqsubseteq \varnothing\ T'\ ^{cl'}}$$

$\boxed{e \sqsubseteq e}$ Expression Precision

$$x \sqsubseteq x \qquad \frac{T \sqsubseteq T' \qquad e \sqsubseteq e'}{\lambda x : T\ .\ e \sqsubseteq \lambda x : T'\ .\ e'} \qquad \frac{e_1 \sqsubseteq e_1' \qquad e_2 \sqsubseteq e_2'}{e_1\ e_2 \sqsubseteq e_1'\ e_2'} \qquad n \sqsubseteq n \qquad true \sqsubseteq true$$

$$false \sqsubseteq false \qquad \frac{e \sqsubseteq e' \qquad c_1 \sqsubseteq c_1' \ldots c_n \sqsubseteq c_n'}{e : c_1 \cap \ldots \cap c_n \sqsubseteq e' : c_1' \cap \ldots \cap c_n'}$$

$$\frac{e \sqsubseteq e' \qquad \Gamma \vdash_{\cap CC} e' : T \qquad \vdash_{\cap CI} c_1 : T_1 \ldots \vdash_{\cap CI} c_n : T_n \qquad T_1 \cap \ldots \cap T_n \sqsubseteq T}{e : c_1 \cap \ldots \cap c_n \sqsubseteq e'}$$

$$\frac{e \sqsubseteq e' \qquad \Gamma \vdash_{\cap CC} e : T \qquad \vdash_{\cap CI} c_1 : T_1 \ldots \vdash_{\cap CI} c_n : T_n \qquad T \sqsubseteq T_1 \cap \ldots \cap T_n}{e \sqsubseteq e' : c_1 \cap \ldots \cap c_n}$$

$$\frac{\Gamma \vdash_{\cap CC} e : T \qquad T \sqsubseteq T'}{e \sqsubseteq blame_{T'}\ l}$$

Figure 3: Precision ($\sqsubseteq$)

Syntax

$$\begin{aligned} &\textit{Types}\ T ::=\ \textit{Int} \mid \textit{Bool} \mid \textit{Dyn} \mid T \to T \\ &\textit{Casts}\ \ c\ \ ::= c : T \Rightarrow^{l} T\ ^{cl} \mid \textit{blame}\ T\ T\ l\ ^{cl} \mid \varnothing\ T\ ^{cl} \end{aligned}$$

$\boxed{\vdash_{\cap CI} c : T}$ Typing

$$\frac{\vdash_{\cap CI} c : T_1 \qquad T_1 \sim T_2}{\vdash_{\cap CI} (c : T_1 \Rightarrow^{l} T_2\ ^{cl}) : T_2}\ \text{T-SINGLECI} \qquad \frac{}{\vdash_{\cap CI} \textit{blame}\ T_I\ T_F\ l\ ^{cl} : T_F}\ \text{T-BLAMECI}$$

$$\frac{}{\vdash_{\cap CI} \varnothing\ T\ ^{cl} : T}\ \text{T-EMPTYCI}$$

$\boxed{\text{initialType(c)} = \text{T}}$ $\boxed{\text{finalType(c)} = \text{T}}$

$$\textit{initialType}(c : T_1 \Rightarrow^{l} T_2\ ^{cl}) = \textit{initialType}(c) \qquad \textit{finalType}(c : T_1 \Rightarrow^{l} T_2\ ^{cl}) = T_2$$

$$\textit{initialType}(\varnothing\ T\ ^{cl}) = T \qquad \textit{finalType}(\varnothing\ T\ ^{cl}) = T$$

$$\textit{initialType}(\textit{blame}\ T_I\ T_F\ l\ ^{cl}) = T_I \qquad \textit{finalType}(\textit{blame}\ T_I\ T_F\ l\ ^{cl}) = T_F$$

Figure 4: Cast Intersection Type System ($\vdash_{\cap CI}$)

Syntax

$$\begin{aligned} &\textit{Types}\ T ::=\ \textit{Int} \mid \textit{Bool} \mid \textit{Dyn} \mid T \to T \mid T \cap \ldots \cap T \\ &\textit{Expressions}\ e\ ::= x \mid \lambda x : T\ .\ e \mid e\ e \mid n \mid \textit{true} \mid \textit{false} \mid e : c \cap \ldots \cap c \mid \textit{blame}_T\ l \end{aligned}$$

$\boxed{\Gamma \vdash_{\cap CC} e : T}$ Typing

Static Intersection Type System ($\vdash_{\cap S}$) *rules and*

$$\frac{\Gamma \vdash_{\cap CC} e_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2} \qquad \Gamma \vdash_{\cap CC} e_2 : T_{11} \cap \ldots \cap T_{n1}}{\Gamma \vdash_{\cap CC} e_1\ e_2 : T_{12} \cap \ldots \cap T_{n2}}\ \text{T-APP'}$$

$$\frac{\begin{array}{c}\Gamma \vdash_{\cap CC} e : T'_1 \cap \ldots \cap T'_n \qquad \vdash_{\cap CI} c_1 : T_1\ \ \ldots\ \ \vdash_{\cap CI} c_n : T_n \\ T'_1 \cap \ldots \cap T'_n = \textit{initialType}(c_1) \cap \ldots \cap \textit{initialType}(c_n)\end{array}}{\Gamma \vdash_{\cap CC} e : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n}\ \text{T-CASTINTERSECTION}$$

$$\frac{}{\Gamma \vdash_{\cap CC} \textit{blame}_T\ l : T}\ \text{T-BLAME}$$

Figure 5: Intersection Cast Calculus ($\vdash_{\cap CC}$)

$\boxed{\Gamma \vdash_{\cap CC} e \rightsquigarrow e : T}$ Compilation

$$\frac{x : T \in \Gamma}{\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T} \text{ C-VAR}$$

$$\frac{\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e \rightsquigarrow e' : T}{\Gamma \vdash_{\cap CC} (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e) \rightsquigarrow (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e') : T_1 \cap \ldots \cap T_n \to T} \text{ C-ABS}$$

$$\frac{\Gamma, x : T_i \vdash_{\cap CC} e \rightsquigarrow e' : T}{\Gamma \vdash_{\cap CC} (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e) \rightsquigarrow (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e') : T_i \to T} \text{ C-ABS'}$$

$$\frac{\begin{array}{c}\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM \qquad PM \rhd T_1 \cap \ldots \cap T_n \to T \qquad \Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1' \cap \ldots \cap T_n' \\ T_1' \cap \ldots \cap T_n' \sim T_1 \cap \ldots \cap T_n \qquad PM \trianglelefteq S_1 \qquad T_1 \cap \ldots \cap T_n \to T \trianglelefteq S_2 \\ T_1' \cap \ldots \cap T_n' \trianglelefteq S_3 \qquad T_1 \cap \ldots \cap T_n \trianglelefteq S_4 \qquad S_1,\ S_2,\ e_1' \hookrightarrow e_1'' \qquad S_3,\ S_4,\ e_2' \hookrightarrow e_2''\end{array}}{\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow e_1''\ e_2'' : T} \text{ C-APP}$$

$$\frac{\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_1 \ \ldots \ \Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_n}{\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_1 \cap \ldots \cap T_n} \text{ C-GEN} \qquad \frac{\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_1 \cap \ldots \cap T_n}{\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_i} \text{ C-INST}$$

$$\frac{}{\Gamma \vdash_{\cap CC} n \rightsquigarrow n : Int} \text{ C-INT} \qquad \frac{}{\Gamma \vdash_{\cap CC} true \rightsquigarrow true : Bool} \text{ C-TRUE}$$

$$\frac{}{\Gamma \vdash_{\cap CC} false \rightsquigarrow false : Bool} \text{ C-FALSE}$$

$\boxed{\text{T} \trianglelefteq \{\text{T}\}}$ Instances

$$Int \trianglelefteq \{Int\} \qquad Bool \trianglelefteq \{Bool\} \qquad Dyn \trianglelefteq \{Dyn\}$$

$$\frac{T_1 \trianglelefteq \{T_{11}, \ldots, T_{1n}\}}{T_1 \to T_2 \trianglelefteq \{T_{11} \to T_2, \ldots, T_{1n} \to T_2\}} \qquad \frac{T_1 \trianglelefteq \{T_{11}, \ldots, T_{1m}\} \ldots T_n \trianglelefteq \{T_{n1}, \ldots, T_{nj}\}}{T_1 \cap \ldots \cap T_n \trianglelefteq \{T_{11}, \ldots, T_{1m}, \ldots, T_{n1}, \ldots, T_{nj}\}}$$

$\boxed{S,\ S,\ e \hookrightarrow e}$ Cast Insertion

$$\{T_1\},\ \{T_2\},\ e \hookrightarrow e : (\varnothing\ T_1\ ^0 : T_1 \Rightarrow^l T_2\ ^0)$$

$$\{T_{11}, \ldots, T_{1n}\},\ \{T_{21}, \ldots, T_{2n}\},\ e \hookrightarrow e : (\varnothing\ T_{11}\ ^0 : T_{11} \Rightarrow^{l_1} T_{21}\ ^0) \cap \ldots \cap (\varnothing\ T_{1n}\ ^0 : T_{1n} \Rightarrow^{l_n} T_{2n}\ ^0)$$

$$\{T_{11}, \ldots, T_{1n}\},\ \{T_2\},\ e \hookrightarrow e : (\varnothing\ T_{11}\ ^0 : T_{11} \Rightarrow^{l_1} T_2\ ^0) \cap \ldots \cap (\varnothing\ T_{1n}\ ^0 : T_{1n} \Rightarrow^{l_n} T_2\ ^0)$$

$$\{T_1\},\ \{T_{21}, \ldots, T_{2n}\},\ e \hookrightarrow e : (\varnothing\ T_1\ ^0 : T_1 \Rightarrow^{l_1} T_{21}\ ^0) \cap \ldots \cap (\varnothing\ T_1\ ^0 : T_1 \Rightarrow^{l_n} T_{2n}\ ^0)$$

Figure 6: Compilation to the Intersection Cast Calculus

Syntax

$$
\begin{aligned}
&\mathit{Types}\ T ::= \ \mathit{Int} \mid \mathit{Bool} \mid \mathit{Dyn} \mid T \to T\\
&\mathit{Ground\ Types}\ G\ ::=\ \mathit{Int} \mid \mathit{Bool} \mid \mathit{Dyn} \to \mathit{Dyn}\\
&\mathit{Casts}\ \ c\ ::= c : T \Rightarrow^{l} T\ ^{cl} \mid \mathit{blame}\ T\ T\ l\ ^{cl} \mid \varnothing\ T\ ^{cl}\\
&\mathit{Cast\ Values}\ \ \ cv\ ::= cv1 \mid \mathit{blame}\ T\ T\ l\ ^{cl}\\
&\qquad\qquad\qquad cv1\ ::= \varnothing\ T\ ^{cl} \mid cv1 : G \Rightarrow^{l} \mathit{Dyn}\ ^{cl} \mid cv1 : T_1 \to T_2 \Rightarrow^{l} T_3 \to T_4\ ^{cl}
\end{aligned}
$$

$\boxed{c \longrightarrow_{\cap CI} c}$ Evaluation

Push blame to top level

$$\frac{}{\mathit{blame}\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap CI} \mathit{blame}\ T_I\ T_2\ l_1\ ^{cl_1}}\ \text{E-PushBlameCI}$$

Evaluate inside casts

$$\frac{\neg(\mathit{is\ cast\ value}\ c) \qquad c \longrightarrow_{\cap CI} c'}{c : T_1 \Rightarrow^{l} T_2\ ^{cl} \longrightarrow_{\cap CI} c' : T_1 \Rightarrow^{l} T_2\ ^{cl}}\ \text{E-EvaluateCI}$$

Detect success or failure of casts

$$\frac{}{cv1 : T \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1}\ \text{E-IdentityCI}$$

$$\frac{}{cv1 : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl_1} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap CI} cv1}\ \text{E-SucceedCI}$$

$$\frac{\neg(\mathit{same\ ground}\ G_1\ G_2) \qquad \mathit{initialType}(c) = T_I}{cv1 : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl_1} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap CI} \mathit{blame}\ T_I\ G_2\ l_2\ ^{cl_1}}\ \text{E-FailCI}$$

Mediate the transition between the two disciplines

$$\frac{G\ \mathit{is\ ground\ type\ of}\ T \qquad \neg(\mathit{ground}\ T)}{cv1 : T \Rightarrow^{l} \mathit{Dyn}\ ^{cl} \longrightarrow_{\cap CI} cv1 : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} \mathit{Dyn}\ ^{cl}}\ \text{E-GroundCI}$$

$$\frac{G\ \mathit{is\ ground\ type\ of}\ T \qquad \neg(\mathit{ground}\ T)}{cv1 : \mathit{Dyn} \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1 : \mathit{Dyn} \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl}}\ \text{E-ExpandCI}$$

Figure 7: Cast Intersection Operational Semantics ($\longrightarrow_{\cap CI}$)

Syntax

$$\begin{array}{l} \mathit{Types}\ T ::= \mathit{Int} \mid \mathit{Bool} \mid \mathit{Dyn} \mid T \to T \mid T \cap \ldots \cap T \\ \mathit{Expressions}\ e\ ::= x \mid \lambda x : T\ .\ e \mid e\ e \mid n \mid \mathit{true} \mid \mathit{false} \mid e : c \cap \ldots \cap c \mid \mathit{blame}_T\ l \\ \mathit{Values}\ v\ ::= x \mid \lambda x : T\ .\ e \mid n \mid \mathit{true} \mid \mathit{false} \mid \mathit{blame}_T\ l \mid v : cv_1 \cap \ldots \cap cv_n\ \mathit{such\ that} \\ \qquad \neg(\forall_{i \in 1..n}\ .\ cv_i = \mathit{blame}\ T\ T\ l\ ^{cl}) \ \wedge\ \neg(\forall_{i \in 1..n}\ .\ cv_i = \varnothing\ T\ ^{cl}) \end{array}$$

$\boxed{e \longrightarrow_{\cap CC} e}$ Evaluation

Push blame to top level

$$\frac{\Gamma \vdash_{\cap CC} (\mathit{blame}_{T_2}\ l)\ e_2 : T_1}{(\mathit{blame}_{T_2}\ l)\ e_2 \longrightarrow_{\cap CC} \mathit{blame}_{T_1}\ l}\ \text{E-PushBlame1}$$

$$\frac{\Gamma \vdash_{\cap CC} e_1\ (\mathit{blame}_{T_2}\ l) : T_1}{e_1\ (\mathit{blame}_{T_2}\ l) \longrightarrow_{\cap CC} \mathit{blame}_{T_1}\ l}\ \text{E-PushBlame2}$$

$$\frac{\vdash_{\cap CI} c_1 : T_1 \ldots \vdash_{\cap CI} c_n : T_n}{\mathit{blame}_T\ l : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} \mathit{blame}_{T_1 \cap \ldots \cap T_n}\ l}\ \text{E-PushBlameCast}$$

Evaluate expressions

$$\frac{}{(\lambda x : T_1 \cap \ldots \cap T_n\ .\ e)\ v \longrightarrow_{\cap CC}\ [x \mapsto v]e}\ \text{E-AppAbs} \qquad \frac{e_1 \longrightarrow_{\cap CC} e_1'}{e_1\ e_2 \longrightarrow_{\cap CC}\ e_1'\ e_2}\ \text{E-App1}$$

$$\frac{e_2 \longrightarrow_{\cap CC} e_2'}{v_1\ e_2 \longrightarrow_{\cap CC}\ v_1\ e_2'}\ \text{E-App2} \qquad \frac{e \longrightarrow_{\cap CC} e'}{e : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} e' : c_1 \cap \ldots \cap c_n}\ \text{E-Evaluate}$$

Simulate casts on data types

$$\frac{\begin{array}{c} \mathit{is\ value}\ (v_1 : cv_1 \cap \ldots \cap cv_n) \qquad \exists i \in 1..n\ .\ \mathit{isArrowCompatible}(cv_i) \\ ((c_{11}, c_{12}, c_1^s), \ldots, (c_{m1}, c_{m2}, c_m^s)) = \mathit{simulateArrow}(cv_1, \ldots, cv_n) \end{array}}{\begin{array}{c} (v_1 : cv_1 \cap \ldots \cap cv_n)\ v_2 \longrightarrow_{\cap CC} \\ (v_1 : c_1^s \cap \ldots \cap c_m^s)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : c_{12} \cap \ldots \cap c_{m2} \end{array}}\ \text{E-SimulateArrow}$$

Merge casts

$$\frac{\begin{array}{c} \mathit{is\ value}\ (v : cv_1 \cap \ldots \cap cv_n) \\ v : c_1'' \cap \ldots \cap c_j'' = \mathit{mergeCasts}(v : cv_1 \cap \ldots \cap cv_n : c_1' \cap \ldots \cap c_m') \end{array}}{v : cv_1 \cap \ldots \cap cv_n : c_1' \cap \ldots \cap c_m' \longrightarrow_{\cap CC} v : c_1'' \cap \ldots \cap c_j''}\ \text{E-MergeCasts}$$

Evaluate casts

$$\frac{\neg(\forall i \in 1..n\ .\ \mathit{is\ cast\ value}\ c_i) \qquad c_1 \longrightarrow_{\cap CI} cv_1\ \ \ldots\ \ c_n \longrightarrow_{\cap CI} cv_n}{v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n}\ \text{E-EvaluateCasts}$$

Transition from cast values to values

$$\frac{}{v : \mathit{blame}\ I_1\ F_1\ l_1\ ^{cl_1} \cap \ldots \cap \mathit{blame}\ I_n\ F_n\ l_n\ ^{cl_n} \longrightarrow_{\cap CC} \mathit{blame}_{(F_1 \cap \ldots \cap F_n)}\ l_1}\ \text{E-PropagateBlame}$$

$$\frac{}{v : \varnothing\ T_1\ ^{cl_1} \cap \ldots \cap \varnothing\ T_n\ ^{cl_n} \longrightarrow_{\cap CC} v}\ \text{E-RemoveEmpty}$$

Figure 8: Intersection Cast Calculus Operational Semantics ($\longrightarrow_{\cap CC}$)

$\boxed{\langle c\rangle^{cl} = \mathrm{c}}$

$$\langle c : T_1 \Rightarrow^l T_2\ ^{cl}\rangle^{cl'} = \langle c\rangle^{cl'} : T_1 \Rightarrow^l T_2\ ^{cl'}$$

$$\langle \mathit{blame}\ T_I\ T_F\ l\ ^{cl'}\rangle^{cl} = \mathit{blame}\ T_I\ T_F\ l\ ^{cl}$$

$$\langle \varnothing\ T\ ^{cl'}\rangle^{cl} = \varnothing\ T\ ^{cl}$$

$\boxed{\text{isArrowCompatible(c)} = \text{Bool}}$

$$\mathit{isArrowCompatible}(c : T_{11} \to T_{12} \Rightarrow^l T_{21} \to T_{22}\ ^{cl}) = \mathit{isArrowCompatible}(c)$$

$$\mathit{isArrowCompatible}(\varnothing\ (T_1 \to T_2)\ ^{cl}) = \mathit{True}$$

$\boxed{\mathit{separateIntersectionCast}(c) = \text{(c, c)}}$

$$\mathit{separateIntersectionCast}(c : T_1 \Rightarrow^l T_2\ ^{cl}) = (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}, c)$$

$$\mathit{separateIntersectionCast}(\varnothing\ T\ ^{cl}) = (\varnothing\ T\ ^{cl}, \varnothing\ T\ ^{cl})$$

$\boxed{\mathit{breakdownArrowType}(c) = \text{(c, c)}}$

$$\begin{gathered}\mathit{breakdownArrowType}(\varnothing\ T_{11} \to T_{12}\ ^{cl} : T_{11} \to T_{12} \Rightarrow^l T_{21} \to T_{22}\ ^{cl}) = \\ (\varnothing\ T_{21}\ ^{cl} : T_{21} \Rightarrow^l T_{11}\ ^{cl}, \varnothing\ T_{12}\ ^{cl} : T_{12} \Rightarrow^l T_{22}\ ^{cl})\end{gathered}$$

$$\mathit{breakdownArrowType}(\varnothing\ T_1 \to T_2\ ^{cl}) = (\varnothing\ T_1\ ^{cl}, \varnothing\ T_2\ ^{cl}\ )$$

$\boxed{\text{simulateArrow}(c_1, \ldots, c_n) = ((c_{11}, c_{12}, c_1^s), \ldots, (c_{m1}, c_{m2}, c_m^s))}$

$$\frac{\begin{gathered}(c'_1, \ldots, c'_m) = \mathit{filter}\ \mathit{isArrowCompatible}\ (c_1, \ldots, c_n) \\ ((c_1^f, c_1^s), \ldots, (c_m^f, c_m^s)) = \mathit{map}\ \mathit{separateIntersectionCast}\ (\langle c'_1\rangle^0, \ldots, \langle c'_m\rangle^0) \\ ((c_{11}, c_{12}), \ldots, (c_{m1}, c_{m2})) = \mathit{map}\ \mathit{breakdownArrowType}\ (\langle c_1^f\rangle^1, \ldots, \langle c_m^f\rangle^m)\end{gathered}}{\mathit{simulateArrow}(c_1, \ldots, c_n) = ((c_{11}, c_{12}, c_1^s), \ldots, (c_{m1}, c_{m2}, c_m^s))}$$

Figure 9: Definitions for auxiliary semantic functions

$\boxed{\text{getCastLabel(c) = cl}}$

$$getCastLabel(c : T_1 \Rightarrow^l T_2\ ^{cl}) = cl$$

$$getCastLabel(blame\ T_I\ T_F\ l\ ^{cl}) = cl$$

$$getCastLabel(\varnothing\ T\ ^{cl}) = cl$$

$\boxed{\text{sameCastLabel(c, c) = Bool}}$

$$sameCastLabel(c_1, c_2) = getCastLabel(c_1) == 0$$

$$sameCastLabel(c_1, c_2) = getCastLabel(c_2) == 0$$

$$sameCastLabel(c_1, c_2) = getCastLabel(c_1) == getCastLabel(c_2)$$

$\boxed{\text{joinCasts(c, c) = c}}$

$$joinCasts(c : T_1 \Rightarrow^l T_2\ ^{cl}, c') = joinCasts(c, c') : T_1 \Rightarrow^l T_2\ ^{cl}$$

$$joinCasts(blame\ T_I\ T_F\ l\ ^{cl}, c) = blame\ T_I\ T_F\ l\ ^{cl}$$

$$joinCasts(\varnothing\ T\ ^{cl}, c) = \langle c \rangle^{cl}$$

$\boxed{mergeCasts(e) = e}$

$$\frac{\begin{array}{c}(c'_1, \ldots, c'_o) = [joinCast\ y\ x \mid x \leftarrow (c_{11}, \ldots, c_{1m}),\ y \leftarrow (c_{21}, \ldots, c_{2n}),\\ sameCastLabel\ y\ x\ \&\&\ initialType(y) == finalType(x)]\end{array}}{mergeCasts(e : c_{11} \cap \ldots \cap c_{1m} : c_{21} \cap \ldots \cap c_{2n}) = e : c'_1 \cap \ldots \cap c'_o}$$

Figure 10: Definitions for auxiliary semantic functions

$\boxed{e =_c e}$ Equality of Casts

$$x =_c x \qquad n =_c n \qquad true =_c true \qquad false =_c false \qquad blame_T\ l =_c blame_T\ l$$

$$\frac{e =_c e'}{\lambda x : T\ .\ e =_c \lambda x : T\ .\ e'} \qquad \frac{e_1 =_c e_1' \qquad e_2 =_c e_2'}{e_1\ e_2 =_c e_1'\ e_2'} \qquad \frac{e =_c e'}{e =_c e' : (\varnothing\ T\ ^{cl})}$$

$$blame_T\ l =_c e : (blame\ T'\ T\ l\ ^{cl}) \qquad \frac{e =_c e' : c}{e : T_1 \Rightarrow^l T_2 =_c e' : (c : T_1 \Rightarrow^l T_2\ ^{cl})}$$

Figure 11: Equality of Casts

2 Gradual Intersection Lambda Calculus as an extension of the GTLC

Theorem 2.1 (Instances of Intersection Types)**.** *If $T \trianglelefteq \{T_1, \ldots, T_n\}$ then $\{T_1, \ldots, T_n\}$ is the set of all the instances of T and for each $i \in 1..n$, T_i is a simple type.*

Proof. We proceed by structural induction on T. Base cases:

- $T = Int$. If $Int \trianglelefteq \{Int\}$ then Int is the only instance of Int and Int is a simple type.
- $T = Bool$. If $Bool \trianglelefteq \{Bool\}$ then $Bool$ is the only instance of $Bool$ and $Bool$ is a simple type.
- $T = Dyn$. If $Dyn \trianglelefteq \{Dyn\}$ then Dyn is the only instance of Dyn and Dyn is a simple type.

Induction step:

- $T = T_1 \to T_2$. If $T_1 \to T_2 \trianglelefteq \{T_{11} \to T_2, \ldots, T_{1n} \to T_2\}$ then, by the definition of $\trianglelefteq$, $T_1 \trianglelefteq \{T_{11}, \ldots, T_{1n}\}$. By the induction hypothesis, $\{T_{11}, \ldots, T_{1n}\}$ is the set of all the instances of T_1 and T_{11} and ... and T_{1n} are all simple types. As T_2 is a simple type, then T_2 is the only instance of T_2. Therefore, $\{T_{11} \to T_2, \ldots, T_{1n} \to T_2\}$ is the set of all the instances of $T_1 \to T_2$ and $T_{11} \to T_2$ and ... and $T_{1n} \to T_2$ are all simple types.
- $T = T_1 \cap \ldots \cap T_n$. If $T_1 \cap \ldots \cap T_n \trianglelefteq \{T_{11}, \ldots, T_{1m}, \ldots, T_{n1}, \ldots, T_{nj}\}$ then, by the definition of $\trianglelefteq$, $T_1 \trianglelefteq \{T_{11}, \ldots, T_{1m}\}$ and ... and $T_n \trianglelefteq \{T_{n1}, \ldots, T_{nj}\}$. By the induction hypothesis, $\{T_{11}, \ldots, T_{1m}\}$ is the set of all the instances of T_1 and T_{11} and ... and T_{1m} are all simple types and ... and $\{T_{n1}, \ldots, T_{nj}\}$ is the set of all the instances of T_n and T_{n1} and ... and T_{nj} are all simple types. Then, $\{T_{11}, \ldots, T_{1m}, \ldots, T_{n1}, \ldots, T_{nj}\}$ is the set of all the instance of $T_1 \cap \ldots \cap T_n$ and T_{11} and ... and T_{1m} and ... and T_{n1} and ... and T_{nj} are all simple types.

□

Theorem 2.2 (Conservative Extension)**.** *If e is annotated with only simple types and T is a simple type, then $\Gamma \vdash_G e : T \iff \Gamma \vdash_{\cap G} e : T$.*

Proof. We will first prove the right direction of the implication, that if $\Gamma \vdash_G e : T$ then $\Gamma \vdash_{\cap G} e : T$. We proceed by induction on the length of the derivation tree of $\vdash_G$. Base cases:

- Rule T-Var. If $\Gamma \vdash_G x : T$, then by rule T-Var, $x : T \in \Gamma$. Therefore, $\Gamma \vdash_{\cap G} x : T$.
- Rule T-Int. If $\Gamma \vdash_G n : Int$, then by rule T-Int, $\Gamma \vdash_{\cap G} n : Int$.
- Rule T-True. If $\Gamma \vdash_G true : Bool$, then by rule T-True, $\Gamma \vdash_{\cap G} true : Bool$.
- Rule T-False. If $\Gamma \vdash_G false : Bool$, then by rule T-False, $\Gamma \vdash_{\cap G} false : Bool$.

Induction step:

- Rule T-Abs. If $\Gamma \vdash_G \lambda x : T_1 \,.\, e : T_1 \to T_2$, then by rule T-Abs, $\Gamma, x : T_1 \vdash_G e : T_2$. By the induction hypothesis, $\Gamma, x : T_1 \vdash_{\cap G} e : T_2$. Therefore, by rule T-Abs, $\Gamma \vdash_{\cap G} \lambda x : T_1 \,.\, e : T_1 \to T_2$.
- Rule T-App. If $\Gamma \vdash_G e_1\ e_2 : T_2$ then by rule T-App, $\Gamma \vdash_G e_1 : PM$, $PM \triangleright T_1 \to T_2$, $\Gamma \vdash_G e_2 : T_1'$ and $T_1' \sim T_1$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e_1 : PM$ and $\Gamma \vdash_{\cap G} e_2 : T_1'$. Therefore, by rule T-App, $\Gamma \vdash_{\cap G} e_1\ e_2 : T_2$.

We will now prove the left direction of the implication, that if $\Gamma \vdash_{\cap G} e : T$ then $\Gamma \vdash_G e : T$. We proceed by induction on the length of the derivation tree of $\vdash_{\cap G}$. Base cases:

- Rule T-Var. If $\Gamma \vdash_{\cap G} x : T$, then by rule T-Var, $x : T \in \Gamma$. Therefore, $\Gamma \vdash_G e : T$.
- Rule T-Int. If $\Gamma \vdash_{\cap G} n : Int$, then by rule T-Int, $\Gamma \vdash_G n : Int$.
- Rule T-True. If $\Gamma \vdash_{\cap G} true : Bool$, then by rule T-True, $\Gamma \vdash_G true : Bool$.
- Rule T-False. If $\Gamma \vdash_{\cap G} false : Bool$, then by rule T-False, $\Gamma \vdash_G false : Bool$.

Induction step:

- Rule T-Abs. If $\Gamma \vdash_{\cap G} \lambda x : T_1 : e : T_1 \to T_2$, then by rule T-Abs, $\Gamma, x : T_1 \vdash_{\cap G} e : T_2$. By the induction hypothesis, $\Gamma, x : T_1 \vdash_G e : T_2$. Therefore, by rule T-Abs, $\Gamma \vdash_G \lambda x : T_1 \ . \ e : T_1 \to T_2$.
- Rule T-Abs'. If $\Gamma \vdash_{\cap G} \lambda x : T_1 : e : T_1 \to T_2$, then by rule T-Abs', $\Gamma, x : T_1 \vdash_{\cap G} e : T_2$. By the induction hypothesis, $\Gamma, x : T_1 \vdash_G e : T_2$. Therefore, by rule T-Abs, $\Gamma \vdash_G \lambda x : T_1 \ . \ e : T_1 \to T_2$.
- Rule T-App. If $\Gamma \vdash_{\cap G} e_1 \ e_2 : T_2$ then by rule T-App, $\Gamma \vdash_{\cap G} e_1 : PM$, $PM \rhd T_1 \to T_2$, $\Gamma \vdash_{\cap G} e_2 : T_1'$ and $T_1' \sim T_1$. By the induction hypothesis, $\Gamma \vdash_G e_1 : PM$ and $\Gamma \vdash_G e_2 : T_1'$. Therefore, by rule T-App, $\Gamma \vdash_G e_1 \ e_2 : T_2$.
- Rule T-Gen. If $\Gamma \vdash_{\cap G} e : T$, then by rule T-Gen, $\Gamma \vdash_{\cap G} e : T$. By the induction hypothesis, $\Gamma \vdash_G e : T$.
- Rule T-Inst. If $\Gamma \vdash_{\cap G} e : T$, then by rule T-Inst, $\Gamma \vdash_{\cap G} e : T$. By the induction hypothesis, $\Gamma \vdash_G e : T$.

□

Theorem 2.3 (Conservative Extension)**.** *If e is annotated with only simple types and T is a simple type then $\Gamma \vdash_{CC} e \rightsquigarrow e_1 : T \iff \Gamma \vdash_{\cap CC} e \rightsquigarrow e_2 : T$ and $e_1 =_c e_2$.*

Proof. We will first prove the right direction of the implication, that if $\Gamma \vdash_{CC} e \rightsquigarrow e_1 : T$ then $\Gamma \vdash_{\cap CC} e \rightsquigarrow e_2 : T$ and $e_1 =_c e_2$. We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{CC} e \rightsquigarrow e_1 : T$. Base cases:

- Rule C-Var. If $\Gamma \vdash_{CC} x \rightsquigarrow x : T$, then by rule C-Var, $x : T \in \Gamma$. Therefore, by rule C-Var, $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$.
- Rule C-Int. If $\Gamma \vdash_{CC} n \rightsquigarrow n : Int$, then by rule C-Int, $\Gamma \vdash_{\cap CC} n \rightsquigarrow n : Int$.
- Rule C-True. If $\Gamma \vdash_{CC} true \rightsquigarrow true : Bool$, then by rule C-True, $\Gamma \vdash_{\cap CC} true \rightsquigarrow true : Bool$.
- Rule C-False. If $\Gamma \vdash_{CC} false \rightsquigarrow false : Bool$, then by rule C-False, $\Gamma \vdash_{\cap CC} false \rightsquigarrow false : Bool$.

Induction step:

- Rule C-Abs. If $\Gamma \vdash_{CC} \lambda x : T_1 \ . \ e \rightsquigarrow \lambda x : T_1 \ . \ e' : T_1 \to T_2$, then by rule C-Abs, $\Gamma, x : T_1 \vdash_{CC} e \rightsquigarrow e' : T_2$. By the induction hypothesis, $\Gamma, x : T_1 \vdash_{\cap CC} e \rightsquigarrow e' : T_2$. Therefore, by rule C-Abs, $\Gamma \vdash_{\cap CC} \lambda x : T_1 \ . \ e \rightsquigarrow \lambda x : T_1 \ . \ e' : T_1 \to T_2$.

- Rule C-App. If $\Gamma \vdash_{CC} e_1\ e_2 \rightsquigarrow (e_1' : PM \Rightarrow^l T_1 \to T_2)\ (e_2' : T_1' \Rightarrow^l T_1) : T_2$, then by rule C-App, $\Gamma \vdash_{CC} e_1 \rightsquigarrow e_1' : PM$, $PM \rhd T_1 \to T_2$, $\Gamma \vdash_{CC} e_2 \rightsquigarrow e_2' : T_1'$ and $T_1' \sim T_1$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1'$. By definition of $\unlhd$, $PM \unlhd \{PM\}$, $T_1 \to T_2 \unlhd \{T_1 \to T_2\}$, $T_1' \unlhd \{T_1'\}$ and $T_1 \unlhd \{T_1\}$. By the definition of $\hookrightarrow$, $\{PM\}$, $\{T_1 \to T_2\}$, $e_1' \hookrightarrow e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0$ and $\{T_1'\}$, $\{T_1\}$, $e_2' \hookrightarrow e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0$. Therefore, $\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow (e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0)\ (e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0) : T_2$. By the definition of $=_c$, $(e_1' : PM \Rightarrow^l T_1 \to T_2) =_c (e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0)$ and $(e_2' : T_1' \Rightarrow^l T_1) =_c (e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0)$. Therefore, $(e_1' : PM \Rightarrow^l T_1 \to T_2)\ (e_2' : T_1' \Rightarrow^l T_1) =_c (e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0)\ (e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0)$.

We will now prove the left direction of the implication, that if $\Gamma \vdash_{\cap CC} e \rightsquigarrow e_2 : T$ then $\Gamma \vdash_{CC} e \rightsquigarrow e_1 : T$ and $e_1 =_c e_2$. We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{\cap CC} e \rightsquigarrow e_2 : T$. Base cases:

- Rule C-Var. If $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$, then by rule C-Var, $x : T \in \Gamma$. Therefore, by rule C-Var, $\Gamma \vdash_{CC} x \rightsquigarrow x : T$.
- Rule C-Int. If $\Gamma \vdash_{\cap CC} n \rightsquigarrow n : Int$, then by rule C-Int, $\Gamma \vdash_{CC} n \rightsquigarrow n : Int$.
- Rule C-True. If $\Gamma \vdash_{\cap CC} true \rightsquigarrow true : Bool$, then by rule C-True, $\Gamma \vdash_{CC} true \rightsquigarrow true : Bool$.
- Rule C-False. If $\Gamma \vdash_{\cap CC} false \rightsquigarrow false : Bool$, then by rule C-False, $\Gamma \vdash_{CC} false \rightsquigarrow false : Bool$.

Induction step:

- Rule C-Abs. If $\Gamma \vdash_{\cap CC} \lambda x : T_1\ .\ e \rightsquigarrow \lambda x : T_1\ .\ e' : T_1 \to T_2$, then by rule C-Abs, $\Gamma, x : T_1 \vdash_{\cap CC} e \rightsquigarrow e' : T_2$. By the induction hypothesis, $\Gamma, x : T_1 \vdash_{CC} e \rightsquigarrow e' : T_2$. Therefore, by rule C-Abs, $\Gamma \vdash_{CC} \lambda x : T_1\ .\ e \rightsquigarrow \lambda x : T_1\ .\ e' : T_1 \to T_2$.
- Rule C-Abs' If $\Gamma \vdash_{\cap CC} \lambda x : T_1\ .\ e \rightsquigarrow \lambda x : T_1\ .\ e' : T_1 \to T_2$, then by rule C-Abs', $\Gamma, x : T_1 \vdash_{\cap CC} e \rightsquigarrow e' : T_2$. By the induction hypothesis, $\Gamma, x : T_1 \vdash_{CC} e \rightsquigarrow e' : T_2$. Therefore, by rule C-Abs, $\Gamma \vdash_{CC} \lambda x : T_1\ .\ e \rightsquigarrow \lambda x : T_1\ .\ e' : T_1 \to T_2$.
- Rule C-App. If $\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow e_1''\ e_2'' : T_2$ then by rule C-App, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM$, $PM \rhd T_1 \to T_2$, $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1'$, $T_1' \sim T_1$, $PM \unlhd S_1$, $T_1 \to T_2 \unlhd S_2$, $T_1' \unlhd S_3$, $T_1 \unlhd S_4$, S_1, S_2, $e_1' \hookrightarrow e_1''$ and S_3, S_4, $e_2' \hookrightarrow e_2''$. Since $e_1\ e_2$ is annotated with only simple types, then by the definition of $\unlhd$, $e_1'' = (e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0)$ and $e_2'' = (e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0)$. By the induction hypothesis, $\Gamma \vdash_{CC} e_1 \rightsquigarrow e_1' : PM$ and $\Gamma \vdash_{CC} e_2 \rightsquigarrow e_2' : T_1'$. Therefore, by rule C-App, $\Gamma \vdash_{CC} e_1\ e_2 \rightsquigarrow (e_1' : PM \Rightarrow^l T_1 \to T_2)\ (e_2' : T_1' \Rightarrow^l T_1) : T_2$. By the definition of $=_c$, $(e_1' : PM \Rightarrow^l T_1 \to T_2) =_c (e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0)$ and $(e_2' : T_1' \Rightarrow^l T_1) =_c (e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0)$. Therefore, $(e_1' : PM \Rightarrow^l T_1 \to T_2)\ (e_2' : T_1' \Rightarrow^l T_1) =_c (e_1' : \varnothing\ PM\ ^0 : PM \Rightarrow^l T_1 \to T_2\ ^0)\ (e_2' : \varnothing\ T_1'\ ^0 : T_1' \Rightarrow^l T_1\ ^0)$.
- Rule C-Gen. If $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$ then by rule C-Gen, $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$. By the induction hypothesis, $\Gamma \vdash_{CC} e \rightsquigarrow e' : T$.
- Rule C-Inst. If $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$ then by rule C-Inst, $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$. By the induction hypothesis, $\Gamma \vdash_{CC} e \rightsquigarrow e' : T$.

□

Theorem 2.4 (Conservative Extension)**.** *Depends on Lemma 3.5. If e_2 are annotated with only simple types, T is a simple type, $\Gamma \vdash_{CC} e_1 : T$, $\Gamma \vdash_{\cap CC} e_2 : T$ and $e_1 =_c e_2$ then $e_1 \longrightarrow_{CC} e_1' \iff e_2 \longrightarrow_{\cap CC} e_2'$, and $e_1' =_c e_2'$.*

Proof. We will first prove the right direction of the implication, that if $e_1 \longrightarrow_{CC} e_1'$ then $e_2 \longrightarrow^*_{\cap CC} e_2'$ and $e_1 =_c e_2$. We proceed by induction on the length of the derivation tree of $e_1 =_c e_2$. Base cases:

- $x =_c x$. As x doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.
- $n =_c n$. As n doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.
- $true =_c true$. As $true$ doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.
- $false =_c false$. As $false$ doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.
- $blame_T\ l =_c blame_T\ l$. As $blame_T\ l$ doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.
- $blame_T\ l =_c e : (blame\ T'\ T\ l\ ^{cl})$. As $blame_T\ l$ doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.

Induction step:

- $\lambda x : T\ .\ e =_c \lambda x : T\ .\ e'$. As $\lambda x : T\ .\ e$ doesn't reduce by $\longrightarrow_{CC}$, this case is not considered.
- $e_1\ e_2 =_c e_3\ e_4$. There are six possibilities:
  - Rule E-PushBlame1. If $blame_{T' \to T}\ l\ e_2 = e_3\ e_4$ and $blame_{T' \to T}\ l\ e_2 \longrightarrow_{CC} blame_T\ l$ then by the definition of $=_c$, $blame_{T' \to T}\ l =_c e_3$. There are two possibilities. By the definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, either
    * $e_3 \longrightarrow^*_{\cap CC} blame_{T' \to T}\ l$. By rule E-App1, $e_3\ e_4 \longrightarrow^*_{\cap CC} blame_{T' \to T}\ l\ e_4$. By rule E-PushBlame1, $blame_{T' \to T}\ l\ e_4 \longrightarrow^*_{\cap CC} blame_T\ l$ and $blame_T\ l =_c blame_T\ l$.
    * $e_3 \longrightarrow^*_{\cap CC} e : (blame\ T''\ (T' \to T)\ l\ ^{cl})$. By repeated application of rule E-Evaluate and by Lemma 3.5, $e : blame\ T''\ (T' \to T)\ l\ ^{cl}) \longrightarrow^*_{\cap CC} v : blame\ T''\ (T' \to T)\ l\ ^{cl})$. By rule E-PropagateBlame, $v : blame\ T''\ (T' \to T)\ l\ ^{cl}) \longrightarrow^*_{\cap CC} blame_{T' \to T}\ l$. By rule E-App1, $e_3\ e_4 \longrightarrow^*_{\cap CC} blame_{T' \to T}\ l\ e_4$. By rule E-PushBlame1, $blame_{T' \to T}\ l\ e_4 \longrightarrow^*_{\cap CC} blame_T\ l$ and $blame_T\ l =_c blame_T\ l$.
  - Rule E-PushBlame2. If $e_1\ blame_{T'}\ l = e_3\ e_4$ and $e_1\ blame_{T'}\ l \longrightarrow_{CC} blame_T\ l$ then by the definition of $=_c$, $blame_{T'}\ l =_c e_4$. There are two possibilities. By the definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, either
    * $e_4 \longrightarrow^*_{\cap CC} blame_{T'}\ l$. By rule E-App2, $e_3\ e_4 \longrightarrow^*_{\cap CC} e_3\ blame_{T'}\ l$. By rule E-PushBlame2, $e_3\ blame_{T'}\ l \longrightarrow^*_{\cap CC} blame_T\ l$ and $blame_T\ l =_c blame_T\ l$.
    * $e_4 \longrightarrow^*_{\cap CC} e : blame\ T''\ T'\ l\ ^{cl}$. By repeated application of rule E-Evaluate and by Lemma 3.5, $e : blame\ T''\ T'\ l\ ^{cl} \longrightarrow^*_{\cap CC} v : blame\ T''\ T'\ l\ ^{cl}$. By rule E-PropagateBlame, $v : blame\ T''\ T'\ l\ ^{cl} \longrightarrow^*_{\cap CC} blame_{T'}\ l$. By rule E-App2, $e_3\ e_4 \longrightarrow^*_{\cap CC} e_3\ blame_{T'}\ l$. By rule E-PushBlame2, $e_3\ blame_{T'}\ l \longrightarrow^*_{\cap CC} blame_T\ l$ and $blame_T\ l =_c blame_T\ l$.
  - Rule E-App1. If $e_1\ e_2 =_c e_3\ e_4$ and $e_1\ e_2 \longrightarrow_{CC} e_1'\ e_2$ then by the definition of $=_c$, $e_1 =_c e_3$ and $e_2 =_c e_4$, and by rule E-App1, $e_1 \longrightarrow_{CC} e_1'$. By the induction hypothesis, $e_3 \longrightarrow_{\cap CC} e_3'$ and $e_1' =_c e_3'$. Then, by rule E-App1, $e_3\ e_4 \longrightarrow_{\cap CC} e_3'\ e_4$. By definition of $=_c$, $e_1'\ e_2 =_c e_3'\ e_4$.

– Rule E-App2. If $v_1\ e_2 =_c e_3\ e_4$ and $v_1\ e_2 \longrightarrow_{CC} v_1\ e_2'$ then by the definition of $=_c$, $v_1 =_c e_3$ and $e_2 =_c e_4$, and by rule E-App2, $e_2 \longrightarrow_{CC} e_2'$. By the induction hypothesis, $e_4 \longrightarrow_{\cap CC} e_4'$ and $e_2' =_c e_4'$. By definition of $=_c$, and by applying rule E-RemoveEmpty zero or more times, $e_3 \longrightarrow^*_{\cap CC} v_1$. If $e_3 \longrightarrow^*_{\cap CC} v_1'$ such that $v_1 =_c v_1'$, by rule E-App1, $e_3\ e_4 \longrightarrow_{\cap CC} v_1'\ e_4$, and by rule E-App2, $v_1'\ e_4 \longrightarrow_{\cap CC} v_1'\ e_4'$. By definition of $=_c$, $v_1\ e_2' =_c v_1'\ e_4'$.

– Rule E-AppAbs. If $(\lambda x : T' \,.\, e)\ v =_c e_3\ e_4$ and $(\lambda x : T' \,.\, e)\ v \longrightarrow_{CC} [x \mapsto v]e$ then by the definition of $=_c$, $(\lambda x : T' \,.\, e) =_c e_3$ and $v =_c e_4$. By the definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, $e_3 \longrightarrow^*_{\cap CC} \lambda x : T' \,.\, e'$ and $e_4 \longrightarrow^*_{\cap CC} v'$, such that, by definition of $=_c$, $(\lambda x : T' \,.\, e) =_c (\lambda x : T' \,.\, e')$ and $v =_c v'$ and $e =_c e'$. By rule E-AppAbs, $(\lambda x : T' \,.\, e')\ v' \longrightarrow_{\cap CC} [x \mapsto v']e'$ and by definition of $=_c$, $[x \mapsto v]e =_c [x \mapsto v']e'$.

– Rule C-BETA. If $(v_1 : T_1 \to T_2 \Rightarrow^l T_3 \to T_4)\ v_2 =_c e_3\ e_4$ and $(v_1 : T_1 \to T_2 \Rightarrow^l T_3 \to T_4)\ v_2 \longrightarrow_{CC} (v_1\ (v_2 : T_3 \Rightarrow^l T_1)) : T_2 \Rightarrow^l T_4$ then by the definition of $=_c$, $v_1 : T_1 \to T_2 \Rightarrow^l T_3 \to T_4 =_c e_3$ and $v_2 =_c e_4$. By definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, $e_3 \longrightarrow^*_{\cap CC} v_1' : (\varnothing\ T_1 \to T_2\ ^{cl} : T_1 \to T_2 \Rightarrow^l T_3 \to T_4)$ such that $v_1 =_c v_1'$, and $e_4 \longrightarrow^*_{\cap CC} v_2'$ such that $v_2 =_c v_2'$. By rule E-SimulateArrow, $(v_1' : (\varnothing\ T_1 \to T_2\ ^{cl} : T_1 \to T_2 \Rightarrow^l T_3 \to T_4))\ v_2' \longrightarrow_{\cap CC} ((v_1' : \varnothing\ T_1 \to T_2\ ^{cl})\ (v_2' : (\varnothing\ T_3\ ^0 : T_3 \Rightarrow^l T_1\ ^0))) : (\varnothing\ T_2\ ^0 : T_2 \Rightarrow^l T_4\ ^0)$. By the definition of $=_c$, $(v_1\ (v_2 : T_3 \Rightarrow^l T_1)) : T_2 \Rightarrow^l T_4 =_c ((v_1' : \varnothing\ T_1 \to T_2\ ^{cl})\ (v_2' : (\varnothing\ T_3\ ^0 : T_3 \Rightarrow^l T_1\ ^0))) : (\varnothing\ T_2\ ^0 : T_2 \Rightarrow^l T_4\ ^0)$.

- $e_1 =_c e_2 : (\varnothing\ T\ ^{cl})$. If $e_1 =_c e_2 : \varnothing\ T\ ^{cl}$ and $e_1 \longrightarrow_{CC} e_1'$ then by the definition of $=_c$, $e_1 =_c e_2$. By the induction hypothesis, $e_2 \longrightarrow_{\cap CC} e_2'$ and $e_1' =_c e_2'$. By rule E-Evaluate, $e_2 : \varnothing\ T\ ^{cl} \longrightarrow_{\cap CC} e_2' : \varnothing\ T\ ^{cl}$. As $e_1' =_c e_2'$ then by definition of $=_c$, $e_1' =_c e_2' : \varnothing\ T\ ^{cl}$.

- $e : T_1 \Rightarrow^l T_2 =_c e' : (c : T_1 \Rightarrow^l T_2\ ^{cl})$. There are seven possibilities:

  – Rule E-Evaluate. If $e_1 : T_1 \Rightarrow^l T_2 =_c e$ and $e_1 : T_1 \Rightarrow^l T_2 \longrightarrow_{CC} e_1' : T_1 \Rightarrow^l T_2$, then by the definition of $=_c$ and by applying rule E-Evaluate zero or more times, $e \longrightarrow^*_{\cap CC} e_2 : (c : T_1 \Rightarrow^l T_2\ ^{cl})$ such that $e_1 =_c e_2 : c$, and by rule E-Evaluate, $e_1 \longrightarrow_{CC} e_1'$. By the induction hypothesis, $e_2 : c \longrightarrow^*_{\cap CC} e_2' : c$ and $e_1' =_c e_2' : c$. If $e_2 : c \longrightarrow^*_{\cap CC} e_2' : c$ then by rule E-Evaluate, $e_2 \longrightarrow^*_{\cap CC} e_2'$. By rule E-Evaluate, $e_2 : (c : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow_{\cap CC} e_2' : (c : T_1 \Rightarrow^l T_2\ ^{cl})$. As $e_1' =_c e_2' : c$ then by the definition of $=_c$, $e_1' : T_1 \Rightarrow^l T_2 =_c e_2' : (c : T_1 \Rightarrow^l T_2\ ^{cl})$.

  – Rule CTX-BLAME. If $blame_{T_1}\ l : T_1 \Rightarrow^l T_2 =_c e$ and $blame_{T_1}\ l : T_1 \Rightarrow^l T_2 \longrightarrow_{CC} blame_{T_2}\ l$ then there are three possibilities. By the definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, either

    * $e \longrightarrow^*_{\cap CC} blame_{T_1}\ l : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl})$. By rule E-PushBlameCast, $blame_{T_1}\ l : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow_{\cap CC} blame_{T_2}\ l$ and $blame_{T_2}\ l =_c blame_{T_2}\ l$.
    * $e \longrightarrow^*_{\cap CC} e' : (blame\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl})$. By repeated application of rule E-Evaluate and by Lemma 3.5, $e' : (blame\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow^*_{\cap CC} v : (blame\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl})$. By rule E-EvaluateCasts and by rule E-PushBlameCI, $v : (blame\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow^*_{\cap CC} v : (blame\ T'\ T_2\ l\ ^{cl})$. By rule E-PropagateBlame, $v : (blame\ T'\ T_2\ l\ ^{cl}) \longrightarrow^*_{\cap CC} blame_{T_2}\ l)$ and $blame_{T_2}\ l =_c blame_{T_2}\ l$.
    * $e \longrightarrow^*_{\cap CC} e' : (blame\ T'\ T_1\ l\ ^{cl}) : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl})$. By repeated application of rule E-Evaluate and by Lemma 3.5, $e' : (blame\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow^*_{\cap CC} v : (blame\ T'\ T_1\ l\ ^{cl}) : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl})$. By rule E-MergeCasts, $v : (blame\ T'\ T_1\ l\ ^{cl}) : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow_{\cap CC} v : (blame\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^l$

$T_2\ ^{cl}$). By rule E-EvaluateCasts and by rule E-PushBlameCI, $v : (\mathit{blame}\ T'\ T_1\ l\ ^{cl} : T_1 \Rightarrow^{l} T_2\ ^{cl}) \longrightarrow^{*}_{\cap CC} v : (\mathit{blame}\ T'\ T_2\ l\ ^{cl})$. By rule E-PropagateBlame, $v : (\mathit{blame}\ T'\ T_2\ l\ ^{cl}) \longrightarrow^{*}_{\cap CC} \mathit{blame}_{T_2}\ l)$ and $\mathit{blame}_{T_2}\ l =_c \mathit{blame}_{T_2}\ l$.

- Rule ID-BASE and Rule ID-STAR. If $v : T \Rightarrow^{l} T =_c e$ and $v : T \Rightarrow^{l} T \longrightarrow_{CC} v$, then by the definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, $e \longrightarrow^{*}_{\cap CC} v' : (cv : T \Rightarrow^{l} T\ ^{cl})$, such that $v =_c v' : cv$. By rule E-EvaluateCasts and by rule E-IdentityCI, $v' : (cv : T \Rightarrow^{l} T\ ^{cl}) \longrightarrow_{\cap CC} v' : cv$ and $v =_c v' : cv$.
- Rule SUCCEED. If $v : G \Rightarrow^{l_1} \mathit{Dyn} : \mathit{Dyn} \Rightarrow^{l_2} G =_c e$ and $v : G \Rightarrow^{l_1} \mathit{Dyn} : \mathit{Dyn} \Rightarrow^{l_2} G \longrightarrow_{CC} v$ then there are two possibilities. By definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, either
  * $e \longrightarrow^{*}_{\cap CC} v' : (cv : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl})$ or
  * $e \longrightarrow^{*}_{\cap CC} v' : (cv : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl}) : (\varnothing\ \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl})$

  such that $v =_c v' : cv$. As, by rule E-MergeCasts, $v' : (cv : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl}) : (\varnothing\ \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl}) \longrightarrow_{\cap CC} v' : (cv : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl})$, we only need to address the first case. By rule E-EvaluateCasts and by rule E-SucceedCI, $v' : (cv : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl}) \longrightarrow_{\cap CC} v' : cv$ and $v =_c v' : cv$.
- Rule FAIL. If $v : G_1 \Rightarrow^{l_1} \mathit{Dyn} : \mathit{Dyn} \Rightarrow^{l_2} G_2 =_c e$ and $v : G_1 \Rightarrow^{l_1} \mathit{Dyn} : \mathit{Dyn} \Rightarrow^{l_2} G_2 \longrightarrow_{CC} \mathit{blame}_{G_2}\ l_2$ then there are two possibilities. By definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, either
  * $e \longrightarrow^{*}_{\cap CC} v' : (cv : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl})$ or
  * $e \longrightarrow^{*}_{\cap CC} v' : (cv : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl}) : (\varnothing\ \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl})$

  such that $v =_c v' : cv$. As, by rule E-MergeCasts, $v' : (cv : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl}) : (\varnothing\ \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl}) \longrightarrow_{\cap CC} v' : (cv : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl})$, we only need to address the first case. By rule E-EvaluateCasts and by rule E-FailCI, $v' : (cv : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl}) \longrightarrow_{\cap CC} v' : \mathit{blame}\ T_I\ G_2\ l_2\ ^{cl}$. By rule E-PropagateBlame, $v' : \mathit{blame}\ T_I\ G_2\ l_2\ ^{cl} \longrightarrow_{\cap CC} \mathit{blame}_{G_2}\ l_2$ and $\mathit{blame}_{G_2}\ l_2 =_c \mathit{blame}_{G_2}\ l_2$.
- Rule GROUND. If $v : T \Rightarrow^{l} \mathit{Dyn} =_c e$ and $v : T \Rightarrow^{l} \mathit{Dyn} \longrightarrow_{CC} v : T \Rightarrow^{l} G : G \Rightarrow^{l} \mathit{Dyn}$ then by definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, $e \longrightarrow^{*}_{\cap CC} v' : (cv : T \Rightarrow^{l} \mathit{Dyn}\ ^{cl})$ such that $v =_c v' : cv$. By rule E-EvaluateCasts and by rule E-GroundCI, $v' : (cv : T \Rightarrow^{l} \mathit{Dyn}\ ^{cl}) \longrightarrow_{\cap CC} v' : (cv : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} \mathit{Dyn}\ ^{cl})$. As $v =_c v' : cv$, then by definition of $=_c$, $v : T \Rightarrow^{l} G : G \Rightarrow^{l} \mathit{Dyn} =_c v' : (cv : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} \mathit{Dyn}\ ^{cl})$.
- Rule EXPAND. If $v : \mathit{Dyn} \Rightarrow^{l} T =_c e$ and $v : \mathit{Dyn} \Rightarrow^{l} T \longrightarrow_{CC} v : \mathit{Dyn} \Rightarrow^{l} G : G \Rightarrow^{l} T$ then by definition of $=_c$ and by applying rule E-RemoveEmpty zero or more times, $e \longrightarrow^{*}_{\cap CC} v' : (cv : \mathit{Dyn} \Rightarrow^{l} T\ ^{cl})$ such that $v =_c v' : cv$. By rule E-EvaluateCasts and by rule E-ExpandCI, $v' : (cv : \mathit{Dyn} \Rightarrow^{l} T\ ^{cl}) \longrightarrow_{\cap CC} v' : (cv : \mathit{Dyn} \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl})$. As $v =_c v' : cv$, then by definition of $=_c$, $v : \mathit{Dyn} \Rightarrow^{l} G : G \Rightarrow^{l} T =_c v' : (cv : \mathit{Dyn} \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl})$.

We will now prove the left direction of the implication, that if $e_2 \longrightarrow_{\cap CC} e_2'$ then $e_1 \longrightarrow_{CC} e_1'$ and $e_1 =_c e_2$. We proceed by induction on the length of the derivation tree of $e_1 =_c e_2$. Base cases:

- $x =_c x$. As x doesn't reduce by $\longrightarrow_{\cap CC}$, this case is not considered.
- $n =_c n$. As n doesn't reduce by $\longrightarrow_{\cap CC}$, this case is not considered.
- $\mathit{true} =_c \mathit{true}$. As true doesn't reduce by $\longrightarrow_{\cap CC}$, this case is not considered.

- $false =_c false$. As $false$ doesn't reduce by $\longrightarrow_{\cap CC}$, this case is not considered.
- $blame_T\ l =_c blame_T\ l$. As $blame_T\ l$ doesn't reduce by $\longrightarrow_{\cap CC}$, this case is not considered.
- $blame_T\ l =_c e : (blame\ T'\ T\ l\ ^{cl})$. There are two possibilities:
  - Rule E-Evaluate. If $e : (blame\ T'\ T\ l\ ^{cl}) \longrightarrow_{\cap CC} e' : (blame\ T'\ T\ l\ ^{cl})$ and as $blame_T\ l$ is already a value, then $blame_T\ l =_c' e : (blame\ T'\ T\ l\ ^{cl})$.
  - Rule E-PropagateBlame. If $v : (blame\ T'\ T\ l\ ^{cl}) \longrightarrow_{\cap CC} blame_T\ l$ and as $blame_T\ l$ is already a value, then $blame_T\ l =_c blame_T\ l$.

Induction step:

- $\lambda x : T\ .\ e =_c \lambda x : T\ .\ e'$. As $\lambda x : T\ .\ e'$ doesn't reduce by $\longrightarrow_{\cap CC}$, this case is not considered.
- $e_1\ e_2 =_c e_3\ e_4$. There are 6 possibilities:
  - Rule E-PushBlame1. If $blame_{T' \to T}\ l\ e_2 = blame_{T' \to T}\ l\ e_4$ and $blame_{T' \to T}\ l\ e_4 \longrightarrow_{\cap CC} blame_T\ l$ then by rule E-PushBlame1, $blame_{T' \to T}\ l\ e_2 \longrightarrow_{CC} blame_T\ l$ and $blame_T\ l =_c blame_T\ l$.
  - Rule E-PushBlame2. If $e_1\ blame_{T'}\ l = e_3\ blame_{T'}\ l$ and $e_3\ blame_{T'}\ l \longrightarrow_{\cap CC} blame_T\ l$ then by rule E-PushBlame2, $e_1\ blame_{T'}\ l \longrightarrow_{CC} blame_T\ l$ and $blame_T\ l =_c blame_T\ l$.
  - Rule E-App1. If $e_1\ e_2 =_c e_3\ e_4$ and $e_3\ e_4 \longrightarrow_{\cap CC} e_3'\ e_4$ then by the definition of $=_c$, $e_1 =_c e_3$ and $e_2 =_c e_4$, and by rule E-App1, $e_3 \longrightarrow_{\cap CC} e_3'$. By the induction hypothesis, $e_1 \longrightarrow_{CC} e_1'$ and $e_1' =_c e_3'$. Then, by rule E-App1, $e_1\ e_2 \longrightarrow_{CC} e_1'\ e_2$. By definition of $=_c$, $e_1'\ e_2 =_c e_3'\ e_4$.
  - Rule E-App2. If $v_1\ e_2 =_c v_3\ e_4$ and $v_3\ e_4 \longrightarrow_{\cap CC} v_3\ e_4'$ then by the definition of $=_c$, $v_1 =_c v_3$ and $e_2 =_c e_4$, and by rule E-App2, $e_4 \longrightarrow_{\cap CC} e_4'$. By the induction hypothesis, $e_2 \longrightarrow_{CC} e_2'$ and $e_2' =_c e_4'$. Then, by rule E-App2, $v_1\ e_2 \longrightarrow_{CC} v_1\ e_2'$. By definition of $=_c$, $v_1\ e_2' =_c v_3\ e_4'$.
  - Rule E-AppAbs. If $(\lambda x : T'\ .\ e)\ v_2 =_c (\lambda x : T'\ .\ e')\ v_4$ and $(\lambda x : T'\ .\ e')\ v_4 \longrightarrow_{\cap CC} [x \mapsto v_4]e'$ then by the definition of $=_c$, $(\lambda x : T'\ .\ e) =_c (\lambda x : T'\ .\ e')$ and $v_2 =_c v_4$ and $e =_c e'$. By rule E-AppAbs, $(\lambda x : T'\ .\ e)\ v_2 \longrightarrow_{CC} [x \mapsto v_2]e$. As $v_2 =_c v_4$ and $e =_c e'$, then by definition of $=_c$, $[x \mapsto v_2]e =_c [x \mapsto v_4]e'$.
  - Rule E-SimulateArrow. There are two possibilities:
    * If $v_1\ v_2 =_c (v_3 : \varnothing\ T' \to T\ ^{cl})\ v_4$ and $(v_3 : \varnothing\ T' \to T\ ^{cl})\ v_4 \longrightarrow_{\cap CC} ((v_3 : \varnothing\ T' \to T\ ^{cl})\ (v_4 : \varnothing\ T'\ ^{cl})) : \varnothing\ T\ ^{cl}$ then by definition of $=_c$, $v_1 =_c (v_3 : \varnothing\ T' \to T\ ^{cl})$ and $v_2 =_c v_4$ and $v_1 =_c v_3$. By the definition of $=_c$, $v_2 =_c v_4 : \varnothing\ T'\ ^{cl}$. By the definition of $=_c$, $v_1\ v_2 =_c ((v_3 : \varnothing\ T' \to T\ ^{cl})\ (v_4 : \varnothing\ T'\ ^{cl}))$. By the definition of $=_c$, $v_1\ v_2 =_c ((v_3 : \varnothing\ T' \to T\ ^{cl})\ (v_4 : \varnothing\ T'\ ^{cl})) : \varnothing\ T\ ^{cl}$.
    * If $(v_1 : T_1 \to T_2 \Rightarrow^l T_3 \to T_4)\ v_2 =_c (v_3 : (cv : T_1 \to T_2 \Rightarrow^l T_3 \to T_4\ ^{cl}))\ v_4$ and $(v_3 : (cv : T_1 \to T_2 \Rightarrow^l T_3 \to T_4\ ^{cl}))\ v_4 \longrightarrow_{\cap CC} ((v_3 : cv)\ (v_4 : (\varnothing\ T_3\ ^{cl} : T_3 \Rightarrow^l T_1\ ^{cl}))) : (\varnothing\ T\ ^{cl} : T_2 \Rightarrow^l T_4\ ^{cl})$ then by definition of $=_c$, $v_1 =_c v_3 : cv$ and $v_2 =_c v_4$. By rule C-BETA, $(v_1 : T_1 \to T_2 \Rightarrow^l T_3 \to T_4)\ v_2 \longrightarrow_{CC} (v_1\ (v_2 : T_3 \Rightarrow^l T_1)) : T_2 \Rightarrow^l T_4$. As $v_2 =_c v_4$, then by definition of $=_c$, $v_2 : T_3 \Rightarrow^l T_1 =_c v_4 : (\varnothing\ T_3\ ^{cl} : T_3 \Rightarrow^l T_1\ ^{cl})$. As $v_1 =_c v_3 : cv$ and $v_2 : T_3 \Rightarrow^l T_1 =_c v_4 : (\varnothing\ T_3\ ^{cl} : T_3 \Rightarrow^l T_1\ ^{cl})$, then by the definition of $=_c$, $(v_1\ (v_2 : T_3 \Rightarrow^l T_1)) =_c ((v_3 : cv)\ (v_4 : (\varnothing\ T_3\ ^{cl} : T_3 \Rightarrow^l T_1\ ^{cl})))$. As $(v_1\ (v_2 : T_3 \Rightarrow^l T_1)) =_c ((v_3 : cv)\ (v_4 : (\varnothing\ T_3\ ^{cl} : T_3 \Rightarrow^l T_1\ ^{cl})))$, then by the definition of $=_c$, $(v_1\ (v_2 : T_3 \Rightarrow^l T_1)) : T_2 \Rightarrow^l T_4 =_c ((v_3 : cv)\ (v_4 : (\varnothing\ T_3\ ^{cl} : T_3 \Rightarrow^l T_1\ ^{cl}))) : (\varnothing\ T\ ^{cl} : T_2 \Rightarrow^l T_4\ ^{cl})$.

- $e_1 =_c e_2 : (\varnothing\ T\ ^{cl})$. There are two possibilities:
  - Rule E-Evaluate. If $e_1 =_c e_2 : (\varnothing\ T\ ^{cl})$ and $e_2 : (\varnothing\ T\ ^{cl}) \longrightarrow_{\cap CC} e'_2 : (\varnothing\ T\ ^{cl})$ then by the definition of $=_c$, $e_1 =_c e_2$, and by rule E-Evaluate, $e_2 \longrightarrow_{\cap CC} e'_2$. By the induction hypothesis, $e_1 \longrightarrow_{CC} e'_1$ and $e'_1 =_c e'_2$. As $e'_1 =_c e'_2$ then by definition of $=_c$, $e'_1 =_c e'_2 : (\varnothing\ T\ ^{cl})$.
  - Rule E-RemoveEmpty. If $v_1 =_c v_2 : (\varnothing\ T\ ^{cl})$ and $v_2 : (\varnothing\ T\ ^{cl}) \longrightarrow_{\cap CC} v_2$ then by the definition of $=_c$, $v_1 =_c v_2$.
- $e : T_1 \Rightarrow^l T_2 =_c e' : (c : T_1 \Rightarrow^l T_2\ ^{cl})$. There are four possibilities:
  - Rule E-PushBlameCast. If $blame_{T_1}\ l : T_1 \Rightarrow^l T_2 =_c blame_{T_1}\ l : (c : T_1 \Rightarrow^l T_2\ ^{cl})$ and $blame_{T_1}\ l : (c : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow_{\cap CC} blame_{T_2}\ l$ then by rule CTX-BLAME, $blame_{T_1}\ l : T_1 \Rightarrow^l T_2 \longrightarrow_{CC} blame_{T_2}\ l$ and $blame_{T_2}\ l =_c blame_{T_2}\ l$.
  - Rule E-Evaluate. If $e_1 : T_1 \Rightarrow^l T_2 =_c e_2 : (c : T_1 \Rightarrow^l T_2\ ^{cl})$ and $e_2 : (c : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow_{\cap CC} e'_2 : (c : T_1 \Rightarrow^l T_2\ ^{cl})$ then by definition of $=_c$, $e_1 =_c e_2 : c$, and by rule E-Evaluate, $e_2 \longrightarrow_{\cap CC} e'_2$. By rule E-Evaluate, $e_2 : c \longrightarrow_{\cap CC} e'_2 : c$. By the induction hypothesis, $e_1 \longrightarrow_{CC} e'_1$ and $e'_1 =_c e'_2 : c$. By rule E-Evaluate, $e_1 : T_1 \Rightarrow^l T_2 \longrightarrow_{CC} e'_1 : T_1 \Rightarrow^l T_2$. As $e'_1 =_c e'_2 : c$, then by the definition of $=_c$, $e'_1 : T_1 \Rightarrow^l T_2 =_c e'_2 : (c : T_1 \Rightarrow^l T_2\ ^{cl})$.
  - Rule E-MergeCasts. If $v : T_1 \Rightarrow^l T_2 =_c (v' : cv) : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl})$ and $(v' : cv) : (\varnothing\ T_1\ ^{cl} : T_1 \Rightarrow^l T_2\ ^{cl}) \longrightarrow_{\cap CC} v' : (cv : T_1 \Rightarrow^l T_2\ ^{cl})$ then by the definition of $=_c$, $v =_c v' : cv$. As $v =_c v' : cv$, then by the definition of $=_c$, $v : T_1 \Rightarrow^l T_2 =_c v' : (cv : T_1 \Rightarrow^l T_2\ ^{cl})$.
  - Rule E-EvaluateCasts. There are seven possibilities:
    * Rule E-PushBlameCI. If $blame_{T_1}\ l_1 : T_1 \Rightarrow^{l_2} T_2 =_c v : (blame\ T'\ T_1\ l_1\ ^{cl} : T_1 \Rightarrow^{l_2} T_2\ ^{cl})$ and $v : (blame\ T'\ T_1\ l_1\ ^{cl} : T_1 \Rightarrow^{l_2} T_2\ ^{cl}) \longrightarrow_{\cap CC} v : blame\ T'\ T_2\ l_1\ ^{cl}$ then by rule CTX-BLAME $blame_{T_1}\ l_1 : T_1 \Rightarrow^{l_2} T_2 \longrightarrow_{CC} blame_{T_2}\ l_1$ and $blame_{T_2}\ l_1 =_c v : blame\ T'\ T_2\ l_1\ ^{cl}$.
    * Rule E-EvaluateCI. If $v_1 : T_1 \Rightarrow^l T_2 =_c v_2 : (c : T_1 \Rightarrow^l T_2)$ and $v_2 : (c : T_1 \Rightarrow^l T_2) \longrightarrow_{\cap CC} v_2 : (c' : T_1 \Rightarrow^l T_2)$ then $v_1 =_c v_2 : c$ and by rule E-EvaluateCasts, $v_2 : c \longrightarrow_{\cap CC} v_2 : c'$. By the induction hypothesis, $v_1 \longrightarrow_{CC} v'_1$ and $v'_1 =_c v_2 : c'$. By rule E-Evaluate, $v_1 : T_1 \Rightarrow^l T_2 \longrightarrow_{CC} v'_1 : T_1 \Rightarrow^l T_2$. As $v'_1 =_c v_2 : c'$, then by definition of $=_c$, $v'_1 : T_1 \Rightarrow^l T_2 =_c v_2 : (c' : T_1 \Rightarrow^l T_2)$.
    * E-IdentityCI. If $v_1 : T \Rightarrow^l T =_c v_2 : (cv1 : T \Rightarrow^l T)$ and $v_2 : (cv1 : T \Rightarrow^l T) \longrightarrow_{\cap CC} v_2 : cv1$ then by the definition of $=_c$, $v_1 =_c v_2 : cv1$. By rule ID-BASE or ID-STAR, $v_1 : T \rightarrow^l T \longrightarrow_{CC} v_1$ and $v_1 =_c v_2 : cv1$.
    * E-SucceedCI. If $v_1 : G \Rightarrow^{l_1} Dyn : Dyn \Rightarrow^{l_2} G =_c v_2 : (cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2})$ and $v_2 : (cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2}) \longrightarrow_{\cap CC} v_2 : cv1$ then by the definition of $=_c$, $v_1 =_c v_2 : cv1$. By rule SUCCEED, $v_1 : G \Rightarrow^{l_1} Dyn : Dyn \Rightarrow^{l_2} G \longrightarrow_{CC} v_1$ and $v_1 =_c v_2 : cv1$.
    * E-FailCI. If $v_1 : G_1 \Rightarrow^{l_1} Dyn : Dyn \Rightarrow^{l_2} G_2 =_c v_2 : (cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2})$ and $v_2 : (cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2}) \longrightarrow_{\cap CC} v_2 : blame\ T'\ G_2\ l_2\ ^{cl_1}$ then by the definition of $=_c$, $v_1 =_c v_2 : cv1$. By rule FAIL, $v_1 : G_1 \Rightarrow^{l_1} Dyn : Dyn \Rightarrow^{l_2} G_2 \longrightarrow_{CC} blame_{G_2}\ l_2$ and by the definition of $=_c$, $blame_{G_2}\ l_2 =_c v_2 : blame\ T'\ G_2\ l_2\ ^{cl_1}$.

* E-GroundCI. If $v_1 : T \Rightarrow^l Dyn =_c v_2 : (cv1 : T \Rightarrow^l Dyn\ ^{cl})$ and $v_2 : (cv1 : T \Rightarrow^l Dyn\ ^{cl}) \longrightarrow_{\cap CC} v_2 : (cv1 : T \Rightarrow^l G\ ^{cl} : G \Rightarrow^l Dyn\ ^{cl})$ then by the definition of $=_c$, $v_1 =_c v_2 : cv1$. By rule GROUND, $v_1 : T \Rightarrow^l Dyn \longrightarrow_{CC} v_1 : T \Rightarrow^l G : G \Rightarrow^l Dyn$. As $v_1 =_c v_2 : cv1$, then by the definition of $=_c$, $v_1 : T \Rightarrow^l G =_c v_2 : (cv1 : T \Rightarrow^l G\ ^{cl})$. As $v_1 : T \Rightarrow^l G =_c v_2 : (cv1 : T \Rightarrow^l G\ ^{cl})$, then by the definition of $=_c$, $v_1 : T \Rightarrow^l G : G \Rightarrow^l Dyn =_c v_2 : (cv1 : T \Rightarrow^l G\ ^{cl} : G \Rightarrow^l Dyn\ ^{cl})$.
* E-ExpandCI. If $v_1 : Dyn \Rightarrow^l T =_c v_2 : (cv1 : Dyn \Rightarrow^l T\ ^{cl})$ and $v_2 : (cv1 : Dyn \Rightarrow^l T\ ^{cl}) \longrightarrow_{\cap CC} v_2 : (cv1 : Dyn \Rightarrow^l G\ ^{cl} : G \Rightarrow^l T\ ^{cl})$ then by the definition of $=_c$, $v_1 =_c v_2 : cv1$. By rule EXPAND, $v_1 : Dyn \Rightarrow^l T \longrightarrow_{CC} v_1 : Dyn \Rightarrow^l G : G \Rightarrow^l T$. As $v_1 =_c v_2 : cv1$, then by the definition of $=_c$, $v_1 : Dyn \Rightarrow^l G =_c v_2 : (cv1 : Dyn \Rightarrow^l G\ ^{cl})$. As $v_1 : Dyn \Rightarrow^l G =_c v_2 : (cv1 : Dyn \Rightarrow^l G\ ^{cl})$, then by the definition of $=_c$, $v_1 : Dyn \Rightarrow^l G : G \Rightarrow^l T =_c v_2 : (cv1 : Dyn \Rightarrow^l G\ ^{cl} : G \Rightarrow^l T\ ^{cl})$.

□

3 Correctness Criteria

Lemma 3.1 (Consistency reduces to equality when comparing static types)**.** *If T_1 and T_2 are static types then $T_1 = T_2 \iff T_1 \sim T_2$.*

Proof. We proceed by structural induction on T_1.

Base cases:

- $T_1 = Int$.
  - If $Int = Int$ then, by the definition of $\sim$, $Int \sim Int$.
  - If $Int \sim Int$, then $Int = Int$.
- $T_1 = Bool$.
  - If $Bool = Bool$ then, by the definition of $\sim$, $Bool \sim Bool$.
  - If $Bool \sim Bool$, then $Bool = Bool$.

Induction step:

- $T_1 = T_{11} \to T_{12}$.
  - If $T_{11} \to T_{12} = T_{21} \to T_{22}$, for some T_{21} and T_{22}, then $T_{11} = T_{21}$ and $T_{12} = T_{22}$. By the induction hypothesis, $T_{11} \sim T_{21}$ and $T_{12} \sim T_{22}$. Therefore, by the definition of $\sim$, $T_{11} \to T_{12} \sim T_{21} \to T_{22}$.
  - If $T_{11} \to T_{12} \sim T_2$, then by the definition of $\sim$, $T_2 = T_{21} \to T_{22}$ and $T_{11} \sim T_{21}$ and $T_{12} \sim T_{22}$. By the induction hypothesis, $T_{11} = T_{21}$ and $T_{12} = T_{22}$. Therefore, $T_{11} \to T_{12} = T_{21} \to T_{22}$.
- $T_1 = T_{11} \cap \ldots \cap T_{1n}$.
  - If $T_{11} \cap \ldots \cap T_{1n} = T_2$, then $\exists T_{21} \ldots T_{2n} \,.\, T_2 = T_{21} \cap \ldots \cap T_{2n}$ and $T_{11} = T_{21}$ and ... and $T_{1n} = T_{2n}$. By the induction hypothesis, $T_{11} \sim T_{21}$ and ... and $T_{1n} \sim T_{2n}$. Therefore, by the definition of $\sim$, $T_{11} \cap \ldots \cap T_{1n} \sim T_{21} \cap \ldots \cap T_{2n}$.

- If $T_{11} \cap \ldots \cap T_{1n} \sim T_2$, then either:
  * $\exists T_{21} \ldots T_{2n}$. $T_2 = T_{21} \cap \ldots \cap T_{2n}$ and $T_{11} \sim T_{21}$ and ... and $T_{1n} \sim T_{2n}$. By the induction hypothesis, $T_{11} = T_{21}$ and ... and $T_{1n} = T_{2n}$. Therefore, $T_{11} \cap \ldots \cap T_{1n} = T_{21} \cap \ldots \cap T_{2n}$.
  * $T_{11} \sim T_2$ and ... and $T_{1n} \sim T_2$. By the induction hypothesis, $T_{11} = T_2$ and ... and $T_{1n} = T_2$. As $T_2 \cap \ldots \cap T_2 = T_2$, then $T_{11} \cap \ldots \cap T_{1n} = T_2$.

□

Theorem 3.1 (Conservative Extension)**.** *Depends on Lemma 3.1. If e is fully static and T is a static type, then $\Gamma \vdash_{\cap S} e : T \iff \Gamma \vdash_{\cap G} e : T$.*

Proof. We proceed by induction on the length of the derivation tree of $\vdash_{\cap S}$ and $\vdash_{\cap G}$ for the right and left direction of the implication, respectively.

Base cases:

- Rule T-Var.
  - If $\Gamma \vdash_{\cap S} x : T$, then $x : T \in \Gamma$. Therefore, $\Gamma \vdash_{\cap G} x : T$.
  - If $\Gamma \vdash_{\cap G} x : T$, then $x : T \in \Gamma$. Therefore, $\Gamma \vdash_{\cap S} e : T$.
- Rule T-Int.
  - If $\Gamma \vdash_{\cap S} n : Int$, then $\Gamma \vdash_{\cap G} n : Int$.
  - If $\Gamma \vdash_{\cap G} n : Int$, then $\Gamma \vdash_{\cap S} n : Int$.
- Rule T-True.
  - If $\Gamma \vdash_{\cap S} true : Bool$, then $\Gamma \vdash_{\cap G} true : Bool$.
  - If $\Gamma \vdash_{\cap G} true : Bool$, then $\Gamma \vdash_{\cap S} true : Bool$.
- Rule T-False.
  - If $\Gamma \vdash_{\cap S} false : Bool$, then $\Gamma \vdash_{\cap G} false : Bool$.
  - If $\Gamma \vdash_{\cap G} false : Bool$, then $\Gamma \vdash_{\cap S} false : Bool$.

Induction step:

- Rule T-Abs.
  - If $\Gamma \vdash_{\cap S} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_1 \cap \ldots \cap T_n \to T$, then $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap S} e : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap G} e : T$. Therefore, $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_1 \cap \ldots \cap T_n \to T$.
  - If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_1 \cap \ldots \cap T_n \to T$, then $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap G} e : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap S} e : T$. Therefore, $\Gamma \vdash_{\cap S} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_1 \cap \ldots \cap T_n \to T$.
- Rule T-Abs'.
  - If $\Gamma \vdash_{\cap S} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_i \to T$, then $\Gamma, x : T_i \vdash_{\cap S} e : T$. By the induction hypothesis, $\Gamma, x : T_i \vdash_{\cap G} e : T$. Therefore, $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_i \to T$.

- If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e : T_i \to T$, then $\Gamma, x : T_i \vdash_{\cap G} e : T$. By the induction hypothesis, $\Gamma, x : T_i \vdash_{\cap S} e : T$. Therefore, $\Gamma \vdash_{\cap S} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e : T_i \to T$.

- Rule T-App.
  - If $\Gamma \vdash_{\cap S} e_1 \ e_2 : T$ then $\Gamma \vdash_{\cap S} e_1 : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap S} e_2 : T_1 \cap \ldots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e_1 \ : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap G} e_2 : T_1 \cap \ldots \cap T_n$. By the definition of $\rhd$, $T_1 \cap \ldots \cap T_n \to T \rhd T_1 \cap \ldots \cap T_n \to T$. By the definition of $\sim$, $T_1 \cap \ldots \cap T_n \sim T_1 \cap \ldots \cap T_n$. Therefore, $\Gamma \vdash_{\cap G} e_1 \ e_2 : T$.
  - If $\Gamma \vdash_{\cap G} e_1 \ e_2 : T$ then $\Gamma \vdash_{\cap G} e_1 : PM$, $PM \rhd T_1 \cap \ldots \cap T_n \to T$, $\Gamma \vdash_{\cap G} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T'_1 \cap \ldots \cap T'_n \sim T_1 \cap \ldots \cap T_n$. By the definition of $\rhd$, $PM = T_1 \cap \ldots \cap T_n \to T$, therefore $\Gamma \vdash_{\cap G} e_1 : T_1 \cap \ldots \cap T_n \to T$. By Lemma 3.1, $T'_1 \cap \ldots \cap T'_n = T_1 \cap \ldots \cap T_n$, and therefore $\Gamma \vdash_{\cap G} e_2 : T_1 \cap \ldots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap S} e_1 \ : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap S} e_2 : T_1 \cap \ldots \cap T_n$. Therefore, $\Gamma \vdash_{\cap S} e_1 \ e_2 : T$.

- Rule T-Gen.
  - If $\Gamma \vdash_{\cap S} e : T_1 \cap \ldots \cap T_n$ then $\Gamma \vdash_{\cap S} e : T_1$ and ... and $\Gamma \vdash_{\cap S} e : T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e : T_1$ and ... and $\Gamma \vdash_{\cap G} e : T_n$. Therefore, $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$.
  - If $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$ then $\Gamma \vdash_{\cap G} e : T_1$ and ... and $\Gamma \vdash_{\cap G} e : T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap S} e : T_1$ and ... and $\Gamma \vdash_{\cap S} e : T_n$. Therefore $\Gamma \vdash_{\cap S} e : T_1 \cap \ldots \cap T_n$.

- Rule T-Inst.
  - If $\Gamma \vdash_{\cap S} e : T_i$ then $\Gamma \vdash_{\cap S} e : T_1 \cap \ldots \cap T_n$, such that $T_i \in \{T_1, ..., T_n\}$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$. As $T_i \in \{T_1, ..., T_n\}$, then $\Gamma \vdash_{\cap G} e : T_i$.
  - If $\Gamma \vdash_{\cap G} e : T_i$ then $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$, such that $T_i \in \{T_1, ..., T_n\}$. By the induction hypothesis, $\Gamma \vdash_{\cap S} e : T_1 \cap \ldots \cap T_n$. As $T_i \in \{T_1, ..., T_n\}$, then $\Gamma \vdash_{\cap S} e : T_i$.

□

Theorem 3.2 (Monotonicity w.r.t. precision). *If* $\Gamma \vdash_{\cap G} e : T$ *and* $e' \sqsubseteq e$ *then* $\Gamma \vdash_{\cap G} e' : T'$ *and* $T' \sqsubseteq T$.

Proof. We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{\cap G} e : T$.

Base cases:

- Rule T-Var. If $\Gamma \vdash_{\cap G} x : T$ and $x \sqsubseteq x$, then $\Gamma \vdash_{\cap G} x : T$ and $T \sqsubseteq T$.
- Rule T-Int. If $\Gamma \vdash_{\cap G} n : Int$ and $n \sqsubseteq n$, then $\Gamma \vdash_{\cap G} n : Int$ and $Int \sqsubseteq Int$.
- Rule T-True. If $\Gamma \vdash_{\cap G} true : Bool$ and $true \sqsubseteq true$, then $\Gamma \vdash_{\cap G} true : Bool$ and $Bool \sqsubseteq Bool$.
- Rule T-False. If $\Gamma \vdash_{\cap G} false : Bool$ and $false \sqsubseteq false$, then $\Gamma \vdash_{\cap G} false : Bool$ and $Bool \sqsubseteq Bool$.

Induction step:

- Rule T-Abs. If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e : T_1 \cap \ldots \cap T_n \to T$ and $\lambda x : T'_1 \cap \ldots \cap T'_n \ . \ e' \sqsubseteq \lambda x : T_1 \cap \ldots \cap T_n \ . \ e$, then by rule T-Abs, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap G} e : T$, and by the definition of $\sqsubseteq$, $T'_1 \cap \ldots \cap T'_n \sqsubseteq T_1 \cap \ldots \cap T_n$ and $e' \sqsubseteq e$. By the induction hypothesis, $\Gamma, x : T'_1 \cap \ldots \cap T'_n \vdash_{\cap G} e' : T'$ and $T' \sqsubseteq T$. By rule T-Abs, $\Gamma \vdash_{\cap G} \lambda x : T'_1 \cap \ldots \cap T'_n \ . \ e' : T'_1 \cap \ldots \cap T'_n \to T'$, and by the definition of $\sqsubseteq$, $T'_1 \cap \ldots \cap T'_n \to T' \sqsubseteq T_1 \cap \ldots \cap T_n \to T$.

- Rule T-Abs'. If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e : T_i \rightarrow T$ and $\lambda x : T'_1 \cap \ldots \cap T'_n \ . \ e' \sqsubseteq \lambda x : T_1 \cap \ldots \cap T_n \ . \ e$, then by rule T-Abs', $\Gamma, x : T_i \vdash_{\cap G} e : T$, and by the definition of $\sqsubseteq$, $T'_1 \cap \ldots \cap T'_n \sqsubseteq T_1 \cap \ldots \cap T_n$ and $e' \sqsubseteq e$. By the induction hypothesis, $\Gamma, x : T'_i \vdash_{\cap G} e' : T'$ and $T' \sqsubseteq T$. By rule T-Abs', $\Gamma \vdash_{\cap G} \lambda x : T'_1 \cap \ldots \cap T'_n \ . \ e' : T'_i \rightarrow T'$, and by the definition of $\sqsubseteq$, $T'_i \rightarrow T' \sqsubseteq T_i \rightarrow T$.

- Rule T-App. If $\Gamma \vdash_{\cap G} e_1 \ e_2 : T$ and $e'_1 \ e'_2 \sqsubseteq e_1 \ e_2$ then by rule T-App, $\Gamma \vdash_{\cap G} e_1 : PM$, $PM \rhd T_{11} \cap \ldots \cap T_{1n} \rightarrow T$, $\Gamma \vdash_{\cap G} e_2 : T_{21} \cap \ldots \cap T_{2n}$, and $T_{21} \cap \ldots \cap T_{2n} \sim T_{11} \cap \ldots \cap T_{1n}$, and by the definition of $\sqsubseteq$, $e'_1 \sqsubseteq e_1$ and $e'_2 \sqsubseteq e_2$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e'_1 : PM' \ and \ PM' \sqsubseteq PM \ and \ PM' \rhd T'_{11} \cap \ldots \cap T'_{1n} \rightarrow T'$ and $\Gamma \vdash_{\cap G} e'_2 : T'_{21} \cap \ldots \cap T'_{2n} \ and \ T'_{21} \cap \ldots \cap T'_{2n} \sqsubseteq T_{21} \cap \ldots \cap T_{2n} \ and \ T'_{21} \cap \ldots \cap T'_{2n} \sim T'_{11} \cap \ldots \cap T'_{1n}$. By the definition of $\sqsubseteq$ and $\rhd$, $T'_{11} \cap \ldots \cap T'_{1n} \rightarrow T' \sqsubseteq T_{11} \cap \ldots \cap T_{1n} \rightarrow T$, and therefore, $T' \sqsubseteq T$. As $\Gamma \vdash_{\cap G} e'_1 \ e'_2 : T'$, it is proved.

- Rule T-Gen. If $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$ and $e' \sqsubseteq e$, then by rule T-Gen, $\Gamma \vdash_{\cap G} e : T_1$ and ... and $\Gamma \vdash_{\cap G} e : T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e' : T'_1 \ and \ T'_1 \sqsubseteq T_1$ and ... and $\Gamma \vdash_{\cap G} e' : T'_n \ and \ T'_n \sqsubseteq T_n$. Then by rule T-Gen, $\Gamma \vdash_{\cap G} e' : T'_1 \cap \ldots \cap T'_n$ and by the definition of $\sqsubseteq$, $T'_1 \cap \ldots \cap T'_n \sqsubseteq T_1 \cap \ldots \cap T_n$.

- Rule T-Inst. If $\Gamma \vdash_{\cap G} e : T_i$ and $e' \sqsubseteq e$, then by rule T-Inst, $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$ such that $T_i \in \{T_1, ..., T_n\}$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e' : T'_1 \cap \ldots \cap T'_n$ and $T'_1 \cap \ldots \cap T'_n \sqsubseteq T_1 \cap \ldots \cap T_n$. Therefore, by rule T-Inst, $\Gamma \vdash_{\cap G} e' : T'_i$ and by the definition of $\sqsubseteq$, $T'_i \sqsubseteq T_i$.

$\square$

Theorem 3.3 (Type preservation of cast insertion)**.** *If* $\Gamma \vdash_{\cap G} e : T$ *then* $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$ *and* $\Gamma \vdash_{\cap CC} e' : T$.

Proof. We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{\cap G} e : T$.

Base cases:

- Rule T-Var. If $\Gamma \vdash_{\cap G} x : T$, then by rule T-Var, $x : T \in \Gamma$. By rule C-Var, $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$ and by rule T-Var, $\Gamma \vdash_{\cap CC} x : T$.

- Rule T-Int. As $\Gamma \vdash_{\cap G} n : Int$, then by rule C-Int, $\Gamma \vdash_{\cap CC} n \rightsquigarrow n : Int$ and by rule T-Int, $\Gamma \vdash_{\cap CC} n : Int$.

- Rule T-True. As $\Gamma \vdash_{\cap G} true : Bool$, then by rule C-True, $\Gamma \vdash_{\cap CC} true \rightsquigarrow true : Bool$ and by rule T-True, $\Gamma \vdash_{\cap CC} true : Bool$.

- Rule T-False. As $\Gamma \vdash_{\cap G} false : Bool$, then by rule C-False, $\Gamma \vdash_{\cap CC} false \rightsquigarrow false : Bool$ and by rule T-False, $\Gamma \vdash_{\cap CC} false : Bool$, it is proved.

Induction step:

- Rule T-Abs. If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e : T_1 \cap \ldots \cap T_n \rightarrow T$ then by rule T-Abs, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap G} e : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e \rightsquigarrow e' : T$ and $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e' : T$. By rule C-Abs, $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e \rightsquigarrow \lambda x : T_1 \cap \ldots \cap T_n \ . \ e' : T_1 \cap \ldots \cap T_n \rightarrow T$ and by rule T-Abs, $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e' : T_1 \cap \ldots \cap T_n \rightarrow T$.

- Rule T-Abs'. If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \;.\; e : T_i \rightarrow T$ then by rule T-Abs', $\Gamma, x : T_i \vdash_{\cap G} e : T$. By the induction hypothesis, $\Gamma, x : T_i \vdash_{\cap CC} e \rightsquigarrow e' : T$ and $\Gamma, x : T_i \vdash_{\cap CC} e' : T$. By rule C-Abs', $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \;.\; e \rightsquigarrow \lambda x : T_1 \cap \ldots \cap T_n \;.\; e' : T_i \rightarrow T$ and by rule T-Abs', $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \;.\; e' : T_i \rightarrow T$.

- Rule T-App. If $\Gamma \vdash_{\cap G} e_1\ e_2 : T$ then by rule T-App, $\Gamma \vdash_{\cap G} e_1 : PM$, $PM \rhd T_1 \cap \ldots \cap T_n \rightarrow T$, $\Gamma \vdash_{\cap G} e_2 : T_1' \cap \ldots \cap T_n'$ and $T_1' \cap \ldots \cap T_n' \sim T_1 \cap \ldots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM$ and $\Gamma \vdash_{\cap CC} e_1' : PM$, and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1' \cap \ldots \cap T_n'$ and $\Gamma \vdash_{\cap CC} e_2' : T_1' \cap \ldots \cap T_n'$. Therefore, by rule C-App, $\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow e_1''\ e_2'' : T$. By the definition of $\trianglelefteq$ and S, $S,\ e \hookrightarrow e$, by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} e_1'' : T_1 \rightarrow T \cap \ldots \cap T_n \rightarrow T$ and $\Gamma \vdash_{\cap CC} e_2'' : T_1 \cap \ldots \cap T_n$. By rule T-App', $\Gamma \vdash_{\cap CC} e_1''\ e_2'' : T \cap \ldots \cap T$ and then by the properties of intersection types (modulo repetitions), $\Gamma \vdash_{\cap CC} e_1''\ e_2'' : T$.

- Rule T-Gen. If $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$ then by rule T-Gen, $\Gamma \vdash_{\cap G} e : T_1$ and ... and $\Gamma \vdash_{\cap G} e : T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_1$ and ... and $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_n$, and $\Gamma \vdash_{\cap CC} e' : T_1$ and ... and $\Gamma \vdash_{\cap CC} e' : T_n$. By rule C-Gen, $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_1 \cap \ldots \cap T_n$ and by rule T-Gen, $\Gamma \vdash_{\cap CC} e' : T_1 \cap \ldots \cap T_n$.

- Rule T-Inst. If $\Gamma \vdash_{\cap G} e : T_i$ then by rule T-Inst, $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$, such that $T_i \in \{T_1, \ldots, T_n\}$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_1 \cap \ldots \cap T_n$ and $\Gamma \vdash_{\cap CC} e' : T_1 \cap \ldots \cap T_n$. By rule C-Inst, $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T_i$ and by rule T-Inst, $\Gamma \vdash_{\cap CC} e' : T_i$.

$\square$

Theorem 3.4 (Monotonicity w.r.t precision of cast insertion). *If $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : T_1$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_2$ and $e_1 \sqsubseteq e_2$ then $e_1' \sqsubseteq e_2'$ and $T_1 \sqsubseteq T_2$.*

Proof. We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : T$. Base cases:

- Rule C-Var. If $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$ and $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$, and $x \sqsubseteq x$, then $x \sqsubseteq x$ and $T \sqsubseteq T$.

- Rule C-Int. If $\Gamma \vdash_{\cap CC} n \rightsquigarrow n : Int$, $\Gamma \vdash_{\cap CC} n \rightsquigarrow n : Int$ and $n \sqsubseteq n$, then $n \sqsubseteq n$ and $Int \sqsubseteq Int$.

- Rule C-True. If $\Gamma \vdash_{\cap CC} true \rightsquigarrow true : Bool$, $\Gamma \vdash_{\cap CC} true \rightsquigarrow true : Bool$ and $true \sqsubseteq true$, then $true \sqsubseteq true$ and $Bool \sqsubseteq Bool$.

- Rule C-False. If $\Gamma \vdash_{\cap CC} false \rightsquigarrow false : Bool$, $\Gamma \vdash_{\cap CC} false \rightsquigarrow false : Bool$ and $false \sqsubseteq false$, then $false \sqsubseteq false$ and $Bool \sqsubseteq Bool$.

Induction step:

- Rule C-Abs. If $\Gamma \vdash_{\cap CC} \lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1 \rightsquigarrow \lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1' : T_{11} \cap \ldots \cap T_{1n} \rightarrow T_1$ and $\Gamma \vdash_{\cap CC} \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2 \rightsquigarrow \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2' : T_{21} \cap \ldots \cap T_{2n} \rightarrow T_2$ and $\lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1 \sqsubseteq \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2$ then by rule C-Abs, $\Gamma, x : T_{11} \cap \ldots \cap T_{1n} \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : T_1$ and $\Gamma, x : T_{21} \cap \ldots \cap T_{2n} \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_2$ and by the definition of $\sqsubseteq$, $T_{11} \cap \ldots \cap T_{1n} \sqsubseteq T_{21} \cap \ldots \cap T_{2n}$ and $e_1 \sqsubseteq e_2$. By the induction hypothesis, $e_1' \sqsubseteq e_2'$ and $T_1 \sqsubseteq T_2$. Therefore, by the definition of $\sqsubseteq$, $\lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1' \sqsubseteq \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2'$ and $T_{11} \cap \ldots \cap T_{1n} \rightarrow T_1 \sqsubseteq T_{21} \cap \ldots \cap T_{2n} \rightarrow T_2$.

- Rule C-Abs'. If $\Gamma \vdash_{\cap CC} \lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1 \rightsquigarrow \lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1' : T_{1i} \rightarrow T_1$, such that $T_{1i} \in \{T_{11}, \ldots, T_{1n}\}$, and $\Gamma \vdash_{\cap CC} \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2 \rightsquigarrow \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2' : T_{2i} \rightarrow T_2$, such that $T_{2i} \in \{T_{21}, \ldots, T_{2n}\}$, and $\lambda x : T_{11} \cap \ldots \cap T_{1n} \;.\; e_1 \sqsubseteq \lambda x : T_{21} \cap \ldots \cap T_{2n} \;.\; e_2$ then by the definition of C-Abs', $\Gamma, x : T_{1i} \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_1$ and $\Gamma, x : T_{2i} \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_2$ and by the definition of $\sqsubseteq$, $T_{11} \cap \ldots \cap T_{1n} \sqsubseteq T_{21} \cap \ldots \cap T_{2n}$ and $e_1 \sqsubseteq e_2$ and therefore $T_{1i} \sqsubseteq T_{2i}$. By the induction hypothesis, $e'_1 \sqsubseteq e'_2$ and $T_1 \sqsubseteq T_2$. Therefore, by the definition of $\sqsubseteq$, $\lambda x : T_{11} \cap \ldots \cap T_{1n} \,.\, e'_1 \sqsubseteq \lambda x : T_{21} \cap \ldots \cap T_{2n} \,.\, e'_2$ and $T_{1i} \to T_1 \sqsubseteq T_{2i} \to T_2$.

- Rule C-App. If $\Gamma \vdash_{\cap CC} e_{11}\ e_{12} \rightsquigarrow e''_{11}\ e''_{12} : T_1$ and $\Gamma \vdash_{\cap CC} e_{21}\ e_{22} \rightsquigarrow e''_{21}\ e''_{22} : T_2$ and $e_{11}\ e_{12} \sqsubseteq e_{21}\ e_{22}$ then by rule C-App, $\Gamma \vdash_{\cap CC} e_{11} \rightsquigarrow e'_{11} : PM_1$ and $PM_1 \rhd T_{11} \cap \ldots \cap T_{1n} \to T_1$ and $\Gamma \vdash_{\cap CC} e_{12} \rightsquigarrow e'_{12} : T'_{11} \cap \ldots \cap T'_{1n}$ and $T'_{11} \cap \ldots \cap T'_{1n} \sim T_{11} \cap \ldots \cap T_{1n}$ and $PM_1 \trianglelefteq S_{11}$ and $T_{11} \cap \ldots \cap T_{1n} \to T_1 \trianglelefteq S_{12}$ and $T'_{11} \cap \ldots \cap T'_{1n} \trianglelefteq S_{13}$ and $T_{11} \cap \ldots \cap T_{1n} \trianglelefteq S_{14}$ and $S_{11},\ S_{12},\ e'_{11} \hookrightarrow e''_{11}$ and $S_{13},\ S_{14},\ e'_{12} \hookrightarrow e''_{12}$ and $\Gamma \vdash_{\cap CC} e_{21} \rightsquigarrow e'_{21} : PM_2$ and $PM_2 \rhd T_{21} \cap \ldots \cap T_{2n} \to T_2$ and $\Gamma \vdash_{\cap CC} e_{22} \rightsquigarrow e'_{22} : T'_{21} \cap \ldots \cap T'_{2n}$ and $T'_{21} \cap \ldots \cap T'_{2n} \sim T_{21} \cap \ldots \cap T_{2n}$ and $PM_2 \trianglelefteq S_{21}$ and $T_{21} \cap \ldots \cap T_{2n} \to T_2 \trianglelefteq S_{22}$ and $T'_{21} \cap \ldots \cap T'_{2n} \trianglelefteq S_{23}$ and $T_{21} \cap \ldots \cap T_{2n} \trianglelefteq S_{24}$ and $S_{21},\ S_{22},\ e'_{21} \hookrightarrow e''_{21}$ and $S_{23},\ S_{24},\ e'_{22} \hookrightarrow e''_{22}$. As, by the definition of $\sqsubseteq$, $e_{11} \sqsubseteq e_{21}$ and $e_{12} \sqsubseteq e_{22}$ then by the induction hypothesis, $e'_{11} \sqsubseteq e'_{21}$ and $PM_1 \sqsubseteq PM_2$ and $e'_{12} \sqsubseteq e'_{22}$ and $T'_{11} \cap \ldots \cap T'_{1n} \sqsubseteq T'_{21} \cap \ldots \cap T'_{2n}$. By the definition of $\rhd$, we have that $PM_1 = T_{11} \cap \ldots \cap T_{1n} \to T_1$ and $PM_2 = T_{21} \cap \ldots \cap T_{2n} \to T_2$ and so $T_{11} \cap \ldots \cap T_{1n} \to T_1 \sqsubseteq T_{21} \cap \ldots \cap T_{2n} \to T_2$ and therefore by the definition of $\sqsubseteq$, $T_1 \sqsubseteq T_2$. As by the definition of $\trianglelefteq$, $S,\ S,\ e \hookrightarrow e$ and $\sqsubseteq$, $e''_{11} \sqsubseteq e''_{21}$ and $e''_{12} \sqsubseteq e''_{22}$, then by the definition of $\sqsubseteq$, $e''_{11}\ e''_{12} \sqsubseteq e''_{21}\ e''_{22}$ and $T_1 \sqsubseteq T_2$.
- Rule C-Gen. If $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_{11} \cap \ldots \cap T_{1n}$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_{21} \cap \ldots \cap T_{2n}$ and $e_1 \sqsubseteq e_2$ then by rule C-Gen, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_{11}$ and ... and $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_{1n}$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_{21}$ and ... and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_{2n}$. By the induction hypothesis, $e'_1 \sqsubseteq e'_2$ and $T_{11} \sqsubseteq T_{21}$ and ... and $T_{1n} \sqsubseteq T_{2n}$, and therefore by the definition of $\sqsubseteq$, $T_{11} \cap \ldots \cap T_{1n} \sqsubseteq T_{21} \cap \ldots \cap T_{2n}$.
- Rule C-Inst. If $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_{1i}$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_{2i}$ and $e_1 \sqsubseteq e_2$ then by rule C-Inst, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_{11} \cap \ldots \cap T_{1n}$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_{21} \cap \ldots \cap T_{2n}$. By the induction hypothesis, $e'_1 \sqsubseteq e'_2$ and $T_{11} \cap \ldots \cap T_{1n} \sqsubseteq T_{21} \cap \ldots \cap T_{2n}$, and therefore, by the definition of $\sqsubseteq$, $T_{1i} \sqsubseteq T_{2i}$.

□

Corollary 3.4.1 (Monotonicity of cast insertion)**.** *Corollary of Theorem 3.4. If* $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : T_1$ *and* $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T_2$ *and* $e_1 \sqsubseteq e_2$ *then* $e'_1 \sqsubseteq e'_2$.

Theorem 3.5 (Conservative Extension)**.** *If e is fully static, then* $e \longrightarrow_{\cap S} e' \iff e \longrightarrow_{\cap CC} e'$.

Proof. We proceed by induction on the length of the derivation tree of $\longrightarrow_{\cap S}$ and $\longrightarrow_{\cap CC}$ for the right and left direction of the implication, respectively. Base cases:

- Rule E-AppAbs. If $(\lambda x : T_1 \cap \ldots \cap T_n \,.\, e)\ v \longrightarrow_{\cap S} [x \mapsto v]e$ and $(\lambda x : T_1 \cap \ldots \cap T_n \,.\, e)\ v \longrightarrow_{\cap CC} [x \mapsto v]e$, then it is proved.

Induction step:

- Rule E-App1.
  - If $e_1\ e_2 \longrightarrow_{\cap S} e'_1\ e_2$ then by rule E-App1, $e_1 \longrightarrow_{\cap S} e'_1$. By the induction hypothesis, $e_1 \longrightarrow_{\cap CC} e'_1$. Therefore, by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap CC} e'_1\ e_2$
  - If $e_1\ e_2 \longrightarrow_{\cap CC} e'_1\ e_2$ then by rule E-App1, $e_1 \longrightarrow_{\cap CC} e'_1$. By the induction hypothesis, $e_1 \longrightarrow_{\cap S} e'_1$. Therefore, by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap S} e'_1\ e_2$

- Rule E-App2.
  - If $v_1\ e_2 \longrightarrow_{\cap S} v_1\ e_2'$ then by rule E-App2, $e_2 \longrightarrow_{\cap S} e_2'$. By the induction hypothesis, $e_2 \longrightarrow_{\cap CC} e_2'$. Therefore, by rule E-App2, $v_1\ e_2 \longrightarrow_{\cap CC} v_1\ e_2'$
  - If $v_1\ e_2 \longrightarrow_{\cap CC} v_1\ e_2'$ then by rule E-App2, $e_2 \longrightarrow_{\cap CC} e_2'$. By the induction hypothesis, $e_2 \longrightarrow_{\cap S} e_2'$. Therefore, by rule E-App2, $v_1\ e_2 \longrightarrow_{\cap S} v_1\ e_2'$

□

Lemma 3.2 (Type preservation of $\longrightarrow_{\cap CI}$). *If $c \longrightarrow_{\cap CI} c$ and*

- $\vdash_{\cap CI} c : T$ *then* $\vdash_{\cap CI} c' : T$.
- *initialType*$(c) = T$ *then initialType*$(c') = T$.

Proof. We proceed by induction on the length of the derivation tree of $\longrightarrow_{\cap CI}$.

Base cases:

- Rule E-PushBlameCI.
  - If $\vdash_{\cap CI}$ *blame* $T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} : T_2$ and by rule E-PushBlameCI, *blame* $T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap CI}$ *blame* $T_I\ T_2\ l_1\ ^{cl_1}$, then by rule T-BlameCI, $\vdash_{\cap CI}$ *blame* $T_I\ T_2\ l_1\ ^{cl_1} : T_2$, then it is proved.
  - By the definition of *initialType*, *initialType*(*blame* $T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2}) = T_I$. By rule E-PushBlameCI, *blame* $T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap CI}$ *blame* $T_I\ T_2\ l_1\ ^{cl_1}$. Since *initialType*(*blame* $T_I\ T_2\ l_1\ ^{cl_1}) = T_I$, it is proved.
- Rule E-IdentityCI.
  - If $\vdash_{\cap CI} cv1 : T \Rightarrow^{l} T\ ^{cl} : T$, then by rule T-SingleCI, $\vdash_{\cap CI} cv1 : T$. By rule E-IdentityCI, $cv1 : T \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1$.
  - By the definitions of *initialType*, *initialType*$(cv1 : T \Rightarrow^{l} T\ ^{cl}) =$ *initialType*$(cv1)$. By rule E-IdentityCI, $cv1 : T \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1$.
- Rule E-SucceedCI.
  - If $\vdash_{\cap CI} cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2} : G$, then by rule T-SingleCI, $\vdash_{\cap CI} cv1 : G$. By rule E-SucceedCI, $cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap CI} cv1$.
  - By the definition of *initialType*, *initialType*$(cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2}) =$ *initialType*$(cv1)$. By rule E-SucceedCI, $cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap CI} cv1$. Therefore it is proved.
- Rule E-FailCI.
  - If $\vdash_{\cap CI} cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2} : G_2$, and by rule E-FailCI, $cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap CI}$ *blame* $T_I\ G_2\ l_2\ ^{cl_1}$ then by rule T-BlameCI, $\vdash_{\cap CI}$ *blame* $T_I\ G_2\ l_2\ ^{cl_1} : G_2$.
  - By the definition of *initialType*, *initialType*$(cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2}) = T_I$. By rule E-FailCI, $cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap CI}$ *blame* $T_I\ G_2\ l_2\ ^{cl_1}$, then *initialType*(*blame* $T_I\ G_2\ l_2\ ^{cl_1}) = T_I$.

- Rule E-GroundCI.
  - If $\vdash_{\cap CI} cv1 : T \Rightarrow^{l} Dyn\ ^{cl} : Dyn$ then by rule T-SingleCI, $\vdash_{\cap CI} cv1 : T$. By rule E-GroundCI, $cv1 : T \Rightarrow^{l} Dyn\ ^{cl} \longrightarrow_{\cap CI} cv1 : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} Dyn\ ^{cl}$, then by rule T-SingleCI, $\vdash_{\cap CI} cv1 : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} Dyn\ ^{cl} : Dyn$.
  - By the definition of *initialType*, $initialType(cv1 : T \Rightarrow^{l} Dyn\ ^{cl}) = initialType(cv1)$. By rule E-GroundCI, $cv1 : T \Rightarrow^{l} Dyn\ ^{cl} \longrightarrow_{\cap CI} cv1 : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} Dyn\ ^{cl}$, then $initialType(cv1 : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} Dyn\ ^{cl}) = initialType(cv1)$.
- Rule E-ExpandCI.
  - If $\vdash_{\cap CI} cv1 : Dyn \Rightarrow^{l} T\ ^{cl} : T$ then by rule T-SingleCI, $\vdash_{\cap CI} cv1 : Dyn$. By rule E-ExpandCI, $cv1 : Dyn \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1 : Dyn \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl}$, then by rule T-SingleCI, $\vdash_{\cap CI} cv1 : Dyn \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl} : T$.
  - By the definition of *initialType*, $initialType(cv1 : Dyn \Rightarrow^{l} T\ ^{cl}) = initialType(cv1)$. By rule E-ExpandCI, $cv1 : Dyn \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1 : Dyn \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl}$. Since $initialType(cv1 : Dyn \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl}) = initialType(cv1)$, it is proved.

Induction step:

- Rule E-EvaluateCI.
  - If $\vdash_{\cap CI} c : T_1 \Rightarrow^{l} T_2\ ^{cl} : T_2$ then by rule T-SingleCI, $\vdash_{\cap CI} c : T_1$. By rule E-EvaluateCI, $c \longrightarrow_{\cap CI} c'$. By the induction hypothesis, $\vdash_{\cap CI} c' : T_1$. By rule E-EvaluateCI, $c : T_1 \Rightarrow^{l} T_2\ ^{cl} \longrightarrow_{\cap CI} c' : T_1 \Rightarrow^{l} T_2\ ^{cl}$, then by rule T-SingleCI, $\vdash_{\cap CI} c' : T_1 \Rightarrow^{l} T_2\ ^{cl} : T_2$.
  - By the definition of *initialType*, $initialType(c : T_1 \Rightarrow^{l} T_2\ ^{cl}) = initialType(c)$. By rule E-EvaluateCI, $c \longrightarrow_{\cap CI} c'$. By the induction hypothesis, $initialType(c') = initialType(c)$. By rule E-EvaluateCI, $c : T_1 \Rightarrow^{l} T_2\ ^{cl} \longrightarrow_{\cap CI} c' : T_1 \Rightarrow^{l} T_2\ ^{cl}$. Since $initialType(c' : T_1 \Rightarrow^{l} T_2\ ^{cl}) = initialType(c')$, it is proved.

□

Lemma 3.3 (Progress of $\longrightarrow_{\cap CI}$). *If $\Gamma \vdash_{\cap CI} c : T$ and $initialType(c) = T_I$ then either c is a cast value or there exists a c' such that $c \longrightarrow_{\cap CI} c'$.*

Proof. We proceed by induction on the length of the derivation tree of $\vdash_{\cap CI} c : T$.

Base cases:

- Rule T-BlameCI. As $\vdash_{\cap CI} blame\ T_I\ T_F\ l\ ^{cl} : T_F$, $initialType(blame\ T_I\ T_F\ l\ ^{cl}) = T_I$ and $blame\ T_I\ T_F\ l\ ^{cl}$ is a cast value, it is proved.
- Rule T-EmptyCI. As $\vdash_{\cap CI} \varnothing\ T\ ^{cl} : T$, $initialType(\varnothing\ T\ ^{cl}) = T$ and $\varnothing\ T\ ^{cl}$ is a cast value, it is proved.

Induction step:

- Rule T-SingleCI. If $\vdash_{\cap CI} c : T_1 \Rightarrow^{l} T_2\ ^{c}l : T_2$ and $initialType(c : T_1 \Rightarrow^{l} T_2\ ^{c}l) = T_I$ then by rule T-SingleCI, $\vdash_{\cap CI} c : T_1$ and $initialType(c) = T_I$. By the induction hypothesis, either c is a cast value or there is a c' such that $c \longrightarrow_{\cap CI} c'$. If c is a cast value, then c can either be of the form $blame\ T_I\ T_F\ l\ ^{cl}$, in which case by rule E-PushBlameCI, $blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap CI} blame\ T_I\ T_2\ l_1\ ^{cl_1}$ or c is a cast value 1. If c is a cast value 1 then $c : T_1 \Rightarrow^{l} T_2\ ^{c}l$ can be of one of the folowing forms:

- $cv1 : T \Rightarrow^{l} T\ ^{cl}$. Then by rule E-IdentityCI, $cv1 : T \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1$.
- $cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2}$. Then by rule E-SucceedCI, $cv1 : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap CI} cv1$.
- $cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2}$. Then by rule E-FailCI, $cv1 : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap CI} blame\ T_I\ G_2\ l_2\ ^{cl_1}$.
- $cv1 : T \Rightarrow^{l} Dyn\ ^{cl}$. Then by rule E-GroundCI, $cv1 : T \Rightarrow^{l} Dyn\ ^{cl} \longrightarrow_{\cap CI} cv1 : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} Dyn\ ^{cl}$.
- $cv1 : Dyn \Rightarrow^{l} T\ ^{cl}$. Then by rule E-ExpandCI, $cv1 : Dyn \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap CI} cv1 : Dyn \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl}$.

If there is a c' such that $c \longrightarrow_{\cap CI} c'$, then by rule E-EvaluateCI, $c : T_1 \Rightarrow^{l} T_2\ ^{c}l \longrightarrow_{\cap CI} c' : T_1 \Rightarrow^{l} T_2\ ^{c}l$.

□

Lemma 3.4 (Type preservation of $\longrightarrow_{\cap CC}$)**.** *Depends on Lemmas 3.2 and 3.3. If* $\Gamma \vdash_{\cap CC} e : T_1 \cap \ldots \cap T_n$ *and* $e \longrightarrow_{\cap CC} e'$ *then* $\Gamma \vdash_{\cap CC} e' : T_1 \cap \ldots \cap T_m$ *such that* $m \leq n$.

Proof. We proceed by induction on the length of the derivation tree of $\longrightarrow_{\cap CC}$.

Base cases:

- Rule E-PushBlame1. If $\Gamma \vdash_{\cap CC} blame_{T_2}\ l\ e_2 : T_1$ and $blame_{T_2}\ l\ e_2 \longrightarrow_{\cap CC} blame_{T_1}\ l$ then by rule T-Blame, $\Gamma \vdash_{\cap CC} blame_{T_1}\ l : T_1$.

- Rule E-PushBlame2. If $\Gamma \vdash_{\cap CC} e_1\ blame_{T_2}\ l : T_1$ and $e_1\ blame_{T_2}\ l \longrightarrow_{\cap CC} blame_{T_1}\ l$ then by rule T-Blame, $\Gamma \vdash_{\cap CC} blame_{T_1}\ l : T_1$.

- Rule E-PushBlameCast. If $\Gamma \vdash_{\cap CC} blame_T\ l : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$ and $blame_T\ l : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} blame_{T_1 \cap \ldots \cap T_n}\ l$ then by rule T-Blame, $\Gamma \vdash_{\cap CC} blame_{T_1 \cap \ldots \cap T_n}\ l : T_1 \cap \ldots \cap T_n$.

- Rule E-AppAbs. There exists a type $T_1 \cap \ldots \cap T_n$ such that we can deduce $\Gamma \vdash_{\cap CC} (\lambda x : T_1 \cap \ldots \cap T_n\ .\ e)\ v : T$ from $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_1 \cap \ldots \cap T_n \rightarrow T$ and $\Gamma \vdash_{\cap CC} v : T_1 \cap \ldots \cap T_n$ (x does not occur in Γ). Moreover, $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_1 \cap \ldots \cap T_n \rightarrow T$ only if $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e : T$. By rule E-AppAbs, $(\lambda x : T_1 \cap \ldots \cap T_n\ .\ e)\ v \longrightarrow_{\cap CC} [x \mapsto v]e$. To obtain $\Gamma \vdash_{\cap CC} [x \mapsto v]e : T$, it is sufficient to replace, in the proof of $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e : T$, the statements $x : T_i$ (introduzed by the rules T-Var and T-Inst) by the deductions of $\Gamma \vdash_{\cap CC} v : T_i$ for $1 \leq i \leq n$. (Proof adapted from [1])

- Rule E-SimulateArrow. If $\Gamma \vdash_{\cap CC} (v_1 : cv_1 \cap \ldots \cap cv_n)\ v_2 : T_{12} \cap \ldots \cap T_{n2}$, then by rule T-App', $\Gamma \vdash_{\cap CC} v_1 : cv_1 \cap \ldots \cap cv_n : T_1 \cap \ldots \cap T_n$ such that $\exists i \in 1..n\ .\ T_i = T_{i1} \rightarrow T_{i2}$ and $\Gamma \vdash_{\cap CC} v_2 : T_{11} \cap \ldots \cap T_{n1}$. As $\Gamma \vdash_{\cap CC} v_1 : cv_1 \cap \ldots \cap cv_n : T_1 \cap \ldots \cap T_n$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v_1 : T''_1 \cap \ldots \cap T''_l$ and $\vdash_{\cap CI} cv_1 : T_1$ and ... and $\vdash_{\cap CI} cv_n : T_n$ and $I_1 = initialType(cv_1)$ and ... and $I_n = initialType(cv_n)$ such that $\{I_1, \ldots, I_n\} \subseteq \{T''_1, \ldots, T''_l\}$ and $I_1 \cap \ldots \cap I_n = T''_1 \cap \ldots \cap T''_n$ and $n \leq l$. For the sake of simplicity lets elide cast labels and blame labels. By the definition of SimulateArrow, we have that $c'_1 = c''_1 : T'_{11} \rightarrow T'_{12} \Rightarrow T_{11} \rightarrow T_{12}$ and ... and $c'_m = c''_m : T'_{m1} \rightarrow T'_{m2} \Rightarrow T_{m1} \rightarrow T_{m2}$, for some $m \leq n$. Also, $c_{11} = \varnothing\ T_{11}\ : T_{11} \Rightarrow T'_{11}$ and ... and $c_{m1} = \varnothing\ T_{m1}\ : T_{m1} \Rightarrow T'_{m1}$ and $c_{12} : \varnothing\ T'_{12}\ : T'_{12} \Rightarrow T_{12}$ and ... and $c_{m2} = \varnothing\ T'_{m2}\ : T'_{m2} \Rightarrow T_{m2}$ and $initialType(c^s_1) = I_1$ and ... and $initialType(c^s_m) = I_m$ and $\vdash_{\cap CI} c^s_1 : T'_{11} \rightarrow T'_{12}$ and ... and $\vdash_{\cap CI} c^s_m : T'_{m1} \rightarrow T'_{m2}$. As by rule T-Gen and T-Inst $\Gamma \vdash_{\cap CC} v_1 : T''_1 \cap \ldots \cap T''_m$ and $I_1 \cap \ldots \cap I_m = T''_1 \cap \ldots \cap T''_m$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v_1 : c^s_1 \cap \ldots \cap c^s_m : T'_{11} \to T'_{12} \cap \ldots \cap T'_{m1} \to T'_{m2}$. As by rule T-Gen and T-Inst $\Gamma \vdash_{\cap CC} v_2 : T_{11} \cap \ldots \cap T_{m1}$ and $\vdash_{\cap CI} c_{11} : T'_{11}$ and ... and $\vdash_{\cap CI} c_{m1} : T'_{m1}$ and $initialType(c_{11}) = T_{11}$ and ... and $initialType(c_{m1}) = T_{m1}$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v_2 : c_{11} \cap \ldots \cap c_{m1} : T'_{11} \cap \ldots \cap T'_{m1}$. Therefore, by rule T-App', $\Gamma \vdash_{\cap CC} (v_1 : c^s_1 \cap \ldots \cap c^s_m)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : T'_{12} \cap \ldots \cap T'_{m2}$. As $\vdash_{\cap CI} c_{12} : T_{12}$ and ... and $\vdash_{\cap CI} c_{m2} : T_{m2}$ and $initialType(c_{12}) = T'_{12}$ and ... and $initialType(c_{m2}) = T'_{m2}$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} (v_1 : c^s_1 \cap \ldots \cap c^s_m)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : c_{12} \cap \ldots \cap c_{m2} : T_{12} \cap \ldots \cap T_{m2}$. By rule E-SimulateArrow, $(v_1 : cv_1 \cap \ldots \cap cv_n)\ v_2 \longrightarrow_{\cap CC}$ $(v_1 : c^s_1 \cap \ldots \cap c^s_m)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : c_{12} \cap \ldots \cap c_{m2}$, therefore it is proved.

- Rule E-MergeCasts. If $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : c'_1 \cap \ldots \cap c'_m : F'_1 \cap \ldots \cap F'_m$ then by rule T-CastIntersections, $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : F_1 \cap \ldots \cap F_n$ and $\vdash_{\cap CI} c'_1 : F'_1$ and ... and $\vdash_{\cap CI} c'_m : F'_m$ and $initialType(c'_1) = I'_1$ and $initialType(c'_m) = I'_m$ such that $\{I'_1, \ldots, I'_m\} \subseteq \{F_1, \ldots, F_n\}$ and $I'_1 \cap \ldots \cap I'_m = F_1 \cap \ldots \cap F_m$ and $m \leq n$. As $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : F_1 \cap \ldots \cap F_n$ then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v : T_1 \cap \ldots \cap T_l$ and $\vdash_{\cap CI} cv_1 : F_1$ and ... and $\vdash_{\cap CI} cv_n : F_n$ and $initialType(cv_1) : I_1$ and ... and $initialType(cv_n) : I_n$ such that $\{I_1, \ldots, I_n\} \subseteq \{T_1, \ldots, T_l\}$ and $I_1 \cap \ldots \cap I_n = T_1 \cap \ldots \cap T_n$ and $n \leq l$. By the definition of mergeCasts, $\vdash_{\cap CI} c''_1 : F''_1$ and ... and $\vdash_{\cap CI} c''_j : F''_j$ and $initialType(c''_1) = I''_1$ and ... and $initialType(c''_j) = I''_j$ such that $\{I''_1, \ldots, I''_j\} \subseteq \{T_1, \ldots, T_l\}$ and $I''_1 \cap \ldots \cap I''_j = T_1 \cap \ldots \cap T_j$ and $\{F''_1, \ldots, F''_j\} \subseteq \{F'_1, \ldots, F'_m\}$ and $F''_1 \cap \ldots \cap F''_j = F'_1 \cap \ldots \cap F'_j$ and $j \leq l$ and $j \leq m$. By rule T-Gen and T-Inst, $\Gamma \vdash_{\cap CC} v : T_1 \cap \ldots \cap T_j$ and therefore by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v : c''_1 \cap \ldots \cap c''_j : F''_1 \cap \ldots \cap F''_j$. By rule E-MergeCasts, $v : cv_1 \cap \ldots \cap cv_n : c'_1 \cap \ldots \cap c'_m \longrightarrow_{\cap CC} v : c''_1 \cap \ldots \cap c''_j$.

- Rule E-EvaluateCasts. If $\Gamma \vdash_{\cap CC} v : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$ then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v : T'_1 \cap \ldots \cap T'_n$ and $\vdash_{\cap CI} c_1 : T_1$ and ... and $\vdash_{\cap CI} c_n : T_n$ and $I_1 = initialType(c_1)$ and ... and $I_n = initialType(c_n)$ and $I_1 \cap \ldots \cap I_n = T'_1 \cap \ldots \cap T'_n$. By rule E-EvaluateCasts, $c_1 \longrightarrow_{\cap CI} cv_1$ and ... and $c_n \longrightarrow_{\cap CI} cv_n$. By Lemmas 3.2 and 3.3, $\vdash_{\cap CI} cv_1 : T_1$ and $initialType(cv_1) = I_1$ and ... and $\vdash_{\cap CI} cv_n : T_n$ and $initialType(cv_n) = I_n$. Therefore by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : T_1 \cap \ldots \cap T_n$. By rule E-EvaluateCasts, $v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$.

- Rule E-PropagateBlame. If $\Gamma \vdash_{\cap CC} v : blame\ T'_1\ T_1\ l_1\ {}^{m_1} \cap \ldots \cap blame\ T'_n\ T_n\ l_n\ {}^{m_n} : T_1 \cap \ldots \cap T_n$ and by rule E-PropagateBlame $v : blame\ T'_1\ T_1\ l_1\ {}^{m_1} \cap \ldots \cap blame\ T'_n\ T_n\ l_n\ {}^{m_n} \longrightarrow_{\cap CC} blame_{(T_1 \cap \ldots \cap T_n)}\ l_1$, then by rule T-Blame, $\Gamma \vdash_{\cap CC} blame_{(T_1 \cap \ldots \cap T_n)}\ l_1 : T_1 \cap \ldots \cap T_n$.

- Rule E-RemoveEmpty. If $\Gamma \vdash_{\cap CC} v : \varnothing\ T_1\ {}^{m_1} \cap \ldots \cap \varnothing\ T_n\ {}^{m_n} : T_1 \cap \ldots \cap T_n$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} v : T_1 \cap \ldots \cap T_n$ and $\vdash_{\cap CI} \varnothing\ T_1\ {}^{m_1} : T_1$ and ... and $\vdash_{\cap CI} \varnothing\ T_n\ {}^{m_n} : T_n$ and $initialType(\varnothing\ T_1\ {}^{m_1}) = T_1$ and ... and $initialType(\varnothing\ T_n\ {}^{m_n}) = T_n$. Therefore, by rule E-RemoveEmpty, $v : \varnothing\ T_1\ {}^{m_1} \cap \ldots \cap \varnothing\ T_n\ {}^{m_n} \longrightarrow_{\cap CC} v$.

Induction step:

- Rule E-App1. There are two possibilities:
  - If $\Gamma \vdash_{\cap CC} e_1\ e_2 : T$, then by rule T-App, $\Gamma \vdash_{\cap CC} e_1 : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap CC} e_2 : T_1 \cap \ldots \cap T_n$. By rule E-App1, $e_1 \longrightarrow_{\cap CI} e'_1$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e'_1 : T_1 \cap \ldots \cap T_n \to T$. As by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap CI} e'_1\ e_2$, then by rule T-App, $\Gamma \vdash_{\cap CC} e'_1\ e_2 : T$.

- If $\Gamma \vdash_{\cap CC} e_1\ e_2 : T_{12} \cap \ldots \cap T_{n2}$, then by rule T-App', $\Gamma \vdash_{\cap CC} e_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2}$ and $\Gamma \vdash_{\cap CC} e_2 : T_{11} \cap \ldots \cap T_{n1}$. By rule E-App1, $e_1 \longrightarrow_{\cap CI} e_1'$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e_1' : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2}$. As by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap CI} e_1'\ e_2$, then by rule T-App', $\Gamma \vdash_{\cap CC} e_1'\ e_2 : T_{12} \cap \cdots \cap T_{n2}$.

- Rule E-App2. There are two possibilities:
  - If $\Gamma \vdash_{\cap CC} v_1\ e_2 : T$, then by rule T-App, $\Gamma \vdash_{\cap CC} v_1 : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap CC} e_2 : T_1 \cap \ldots \cap T_n$. By rule E-App2, $e_2 \longrightarrow_{\cap CI} e_2'$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e_2' : T_1 \cap \ldots \cap T_n$. As by rule E-App2, $v_1\ e_2 \longrightarrow_{\cap CI} v_1\ e_2'$, then by rule T-App, $\Gamma \vdash_{\cap CC} v_1\ e_2' : T$.
  - If $\Gamma \vdash_{\cap CC} v_1\ e_2 : T_{12} \cap \ldots \cap T_{n2}$, then by rule T-App', $\Gamma \vdash_{\cap CC} v_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2}$ and $\Gamma \vdash_{\cap CC} e_2 : T_{11} \cap \ldots \cap T_{n1}$. By rule E-App2, $e_2 \longrightarrow_{\cap CI} e_2'$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e_2' : T_{11} \cap \ldots \cap T_{n1}$. As by rule E-App1, $v_1\ e_2 \longrightarrow_{\cap CI} v_1\ e_2'$, then by rule T-App', $\Gamma \vdash_{\cap CC} v_1\ e_2' : T_{12} \cap \cdots \cap T_{n2}$..

- Rule E-Evaluate. If $\Gamma \vdash_{\cap CC} e : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} e : T_1' \cap \ldots \cap T_n'$, $\vdash_{\cap CI} c_1 : T_1$ and ... and $\vdash_{\cap CI} c_n : T_n$ and $\mathit{initialType}(c_1) \cap \ldots \cap \mathit{initialType}(c_n) = T_1' \cap \ldots \cap T_n'$. By rule E-Evaluate, $e \longrightarrow_{\cap CI} e'$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e' : T$. As by rule E-Evaluate, $e : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CI} e' : c_1 \cap \ldots \cap c_n$, then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} e' : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$.

□

Lemma 3.5 (Progress of $\longrightarrow_{\cap CC}$)**.** *If $\Gamma \vdash_{\cap CC} e : T$ then either e is a value or there exists an e' such that $e \longrightarrow_{\cap CC} e'$.*

Proof. We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{\cap CC} e : T$.

Base cases:

- Rule T-Var. If $\Gamma \vdash_{\cap CC} x : T$, then x is a value.
- Rule T-Int. If $\Gamma \vdash_{\cap CC} n : Int$ then n is a value.
- Rule T-True. If $\Gamma \vdash_{\cap CC} true : Bool$ then $true$ is a value.
- Rule T-False. If $\Gamma \vdash_{\cap CC} false : Bool$ then $false$ is a value.

Induction step:

- Rule T-Abs. If $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_1 \cap \ldots \cap T_n \to T$ then $\lambda x : T_1 \cap \ldots \cap T_n\ .\ e$ is a value.
- Rule T-Abs'. If $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_i \to T$ then $\lambda x : T_1 \cap \ldots \cap T_n\ .\ e$ is a value.
- Rule T-App. If $\Gamma \vdash_{\cap CC} e_1\ e_2 : T$ then by rule T-App, $\Gamma \vdash_{\cap CC} e_1 : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap CC} e_2 : T_1 \cap \ldots \cap T_n$. By the induction hypothesis, e_1 is either a value or there is a e_1' such that $e_1 \longrightarrow_{\cap CC} e_1'$ and e_2 is either a value or there is a e_2' such that $e_2 \longrightarrow_{\cap CC} e_2'$. If e_1 is a value, then by rule E-PushBlame1, $(\mathit{blame}_{T_2}\ l)\ e_2 \longrightarrow_{\cap CC} \mathit{blame}_{T_1}\ l$. If e_2 is a value, then by rule E-PushBlame2, $e_1\ (\mathit{blame}_{T_2}\ l) \longrightarrow_{\cap CC} \mathit{blame}_{T_1}\ l$. If e_1 is not a value, then by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap CC} e_1'\ e_2$. If e_1 is a value and e_2 is not a value, then by rule E-App2, $v_1\ e_2 \longrightarrow_{\cap CC} v_1\ e_2'$. If both e_1 and e_2 are values then e_1 must be an abstraction $(\lambda x : T_1 \cap \ldots \cap T_n\ .\ e)$, and by rule E-AppAbs $(\lambda x : T_1 \cap \ldots \cap T_n\ .\ e)\ v_2 \longrightarrow_{\cap CC} [x \mapsto v_2]e$.

- Rule T-Gen. If $\Gamma \vdash_{\cap CC} e : T_1 \cap \ldots \cap T_n$ then by rule T-Gen, $\Gamma \vdash_{\cap CC} e : T_1$ and ... and $\Gamma \vdash_{\cap CC} e : T_n$. By the induction hypothesis, either e is a value or there exists an e' such that $e \longrightarrow_{\cap CC} e'$.

- Rule T-Inst. If $\Gamma \vdash_{\cap CC} e : T_i$ then by rule T-Inst, $\Gamma \vdash_{\cap CC} e : T_1 \cap \ldots \cap T_n$, such that $T_i \in \{T_1, \ldots, T_n\}$. By the induction hypothesis, either e is a value or there exists an e' such that $e \longrightarrow_{\cap CC} e'$.

- Rule T-App'. If $\Gamma \vdash_{\cap CC} e_1\ e_2 : T_{12} \cap \ldots \cap T_{n2}$ then by rule T-App', $\Gamma \vdash_{\cap CC} e_1 : T_{11} \rightarrow T_{12} \cap \ldots \cap T_{n1} \rightarrow T_{n2}$ and $\Gamma \vdash_{\cap CC} e_2 : T_{11} \cap \ldots \cap T_{n1}$. By the induction hypothesis, e_1 is either a value or there is a e_1' such that $e_1 \longrightarrow_{\cap CC} e_1'$ and e_2 is either a value or there is a e_2' such that $e_2 \longrightarrow_{\cap CC} e_2'$. If e_1 is a value, then by rule E-PushBlame1, $(blame_{T_2}\ l)\ e_2 \longrightarrow_{\cap CC} blame_{T_1}\ l$. If e_2 is a value, then by rule E-PushBlame2, $e_1\ (blame_{T_2}\ l) \longrightarrow_{\cap CC} blame_{T_1}\ l$. If e_1 is not a value, then by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap CC} e_1'\ e_2$. If e_1 is a value and e_2 is not a value, then by rule E-App2, $v_1\ e_2 \longrightarrow_{\cap CC} v_1\ e_2'$. If both e_1 and e_2 are values then e_1 must be an abstraction $(\lambda x : T_{11} \rightarrow T_{12} \cap \ldots \cap T_{n1} \rightarrow T_{n2}.\ e)$, and by rule E-AppAbs $(\lambda x : T_{11} \rightarrow T_{12} \cap \ldots \cap T_{n1} \rightarrow T_{n2}\ .\ e)\ v_2 \longrightarrow_{\cap CC} [x \mapsto v_2]e$.

- Rule T-CastIntersection. If $\Gamma \vdash_{\cap CC} e : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$ then by rule T-CastIntersection, $\Gamma \vdash_{\cap CC} e : T_1' \cap \ldots \cap T_n'$. By the induction hypothesis, e is either a value, or there is an e' such that $e \longrightarrow_{\cap CC} e'$. If e is a value, then either by rule E-EvaluateCasts, $v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$, or by rule E-PushBlameCast, $blame_{T_1' \cap \ldots \cap T_n'}\ l : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} blame_{T_1 \cap \ldots \cap T_n}\ l$. If there is an e' such that $e \longrightarrow_{\cap CC} e'$, then by rule E-Evaluate, $e : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} e' : c_1 \cap \ldots \cap c_n$.

- Rule T-Blame. If $\Gamma \vdash_{\cap CC} blame_T\ l : T$ then $blame_T\ l$ is a value.

□

Theorem 3.6 (Type Safety of $\longrightarrow_{\cap CC}$)**.** *Depends on Lemmas 3.4 and 3.5. Both Type Preservation and Progress hold for* $\longrightarrow_{\cap CC}$*.*

Proof. We have Type Preservation (by Lemma 3.4) and Progress (by Lemma 3.5) for $\longrightarrow_{\cap CC}$. □

Theorem 3.7 (Blame Theorem)**.** *If* $\Gamma \vdash_{\cap CC} e : T$ *and* $e \longrightarrow^*_{\cap CC} blame_T\ l$ *then* l *is not a safe cast of* e*.*

Theorem 3.8 (Gradual Guarantee)**.** *If* $\Gamma \vdash_{\cap CC} e_1 : T_1$ *and* $\Gamma \vdash_{\cap CC} e_2 : T_2$ *and* $e_1 \sqsubseteq e_2$ *then:*

1. *if* $e_2 \longrightarrow_{\cap CC} e_2'$ *then* $e_1 \longrightarrow^*_{\cap CC} e_1'$ *and* $e_1' \sqsubseteq e_2'$*.*
2. *if* $e_1 \longrightarrow_{\cap CC} e_1'$ *then either* $e_2 \longrightarrow^*_{\cap CC} e_2'$ *and* $e_1' \sqsubseteq e_2'$ *or* $e_2 \longrightarrow^*_{\cap CC} blame_{T_2}\ l$*.*

References

[1] Mario Coppo, Mariangiola Dezani-Ciancaglini, et al. An extension of the basic functionality theory for the λ-calculus. *Notre Dame journal of formal logic*, 21(4):685–693, 1980.