Gradual Intersection Types

Pedro Ângelo, Mário Florido

February 5, 2018

1 Language Definition

Syntax

$$
\begin{aligned}
&Types\ \ T ::=\ Int \mid Bool \mid Dyn \mid T \rightarrow T' \mid T \cap \ldots \cap T \\
&\qquad T' ::=\ Int \mid Bool \mid Dyn \mid T' \rightarrow T' \\
&Ground\ Types\ G\ ::=\ Int \mid Bool \mid Dyn \rightarrow Dyn \\
&Casts\ \ c\ ::= c : T' \Rightarrow^{l} T'\ ^{cl} \mid blame\ T'\ T'\ l\ ^{cl} \mid \varnothing\ T'\ ^{cl} \\
&Expressions\ \ e\ ::= x \mid \lambda x : T\ .\ e \mid e\ e \mid n \mid true \mid false \\
&\qquad\qquad \mid e : T' \Rightarrow^{l} T' \mid e : c \cap \ldots \cap c \mid blame_{T}\ l \\
&Cast\ Values\ \ \ cv\ ::= cv1 \mid cv2 \\
&\qquad cv1\ ::= \varnothing\ T'\ ^{cl} : G \Rightarrow^{l} Dyn\ ^{cl} \\
&\qquad\qquad \mid \varnothing\ T'\ ^{cl} : T'_1 \rightarrow T'_2 \Rightarrow^{l} T'_3 \rightarrow T'_4\ ^{cl} \\
&\qquad\qquad \mid cv1 : G \Rightarrow^{l} Dyn\ ^{cl} \\
&\qquad\qquad \mid cv1 : T'_1 \rightarrow T'_2 \Rightarrow^{l} T'_3 \rightarrow T'_4\ ^{cl} \\
&\qquad cv2\ ::= blame\ T'\ T'\ l\ ^{cl} \\
&\qquad\qquad \mid \varnothing\ T'\ ^{cl} \\
&Values\ \ v\ ::= x \mid \lambda x : T\ .\ e \mid n \mid true \mid false \mid blame_{T}\ l \\
&\qquad \mid v : G \Rightarrow^{l} Dyn \\
&\qquad \mid v : T'_1 \rightarrow T'_2 \Rightarrow^{l} T'_3 \rightarrow T'_4 \\
&\qquad \mid v : cv_1 \cap \ldots \cap cv_n\ such\ that \\
&\qquad \neg(\forall_{i \in 1..n}\ .\ cv_i = blame\ T'\ T'\ l\ ^{cl}) \wedge \\
&\qquad \neg(\forall_{i \in 1..n}\ .\ cv_i = \varnothing\ T'\ ^{cl})
\end{aligned}
$$

Figure 1: Gradual Intersection System

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

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

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

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

$$\frac{\Gamma \vdash_{\cap G} e : T_1\ \ \ldots\ \ \Gamma \vdash_{\cap G} e : T_n}{\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n} \cap I \qquad \frac{\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n}{\Gamma \vdash_{\cap G} e : T_i} \cap E$$

$\boxed{T \sim T}$ Consistency

$$B \sim B \qquad T \sim Dyn \qquad Dyn \sim T \qquad \frac{T_1 \sim T_3 \qquad T_2 \sim T_4}{T_1 \to T_2 \sim T_3 \to T_4}$$

$$\frac{T_1 \sim T_1'\ \ \ldots\ \ T_n \sim T_n'}{T_1 \cap \ldots \cap T_n \sim T_1' \cap \ldots \cap T_n'} \qquad \frac{T \sim T_1\ \ \ldots\ \ T \sim T_n}{T \sim T_1 \cap \ldots \cap T_n} \qquad \frac{T_1 \sim T\ \ \ldots\ \ T_n \sim T}{T_1 \cap \ldots \cap T_n \sim T}$$

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

$$T_1 \to T_2 \rhd T_1 \to T_2 \qquad Dyn \rhd Dyn \to Dyn$$

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

$$Dyn \sqsubseteq T \qquad B \sqsubseteq B \qquad \frac{T_1 \sqsubseteq T_3 \qquad T_2 \sqsubseteq T_4}{T_1 \to T_2 \sqsubseteq T_3 \to T_4}$$

$$\frac{T_1 \sqsubseteq T_1'\ \ \ldots\ \ T_n \sqsubseteq T_n'}{T_1 \cap \ldots \cap T_n \sqsubseteq T_1' \cap \ldots \cap T_n'} \qquad \frac{T \sqsubseteq T_1\ \ \ldots\ \ T \sqsubseteq T_n}{T \sqsubseteq T_1 \cap \ldots \cap T_n} \qquad \frac{T_1 \sqsubseteq T\ \ \ldots\ \ T_n \sqsubseteq T}{T_1 \cap \ldots \cap T_n \sqsubseteq T}$$

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

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

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

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

$$\textit{rules in Figure 2 and}$$

$$\frac{\begin{array}{c}\Gamma \vdash_{\cap CC} e_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2} \\ \Gamma \vdash_{\cap CC} e_2 : T_1' \cap \ldots \cap T_n' \qquad T_{11} \sim T_1' \ldots T_{n1} \sim T_n'\end{array}}{\Gamma \vdash_{\cap CC} e_1\ e_2 : T_{12} \cap \ldots \cap T_{n2}} \text{ T-App}$$

$$\frac{\Gamma \vdash_{\cap CC} e : T_1 \qquad T_1 \sim T_2}{\Gamma \vdash_{\cap CC} e : T_1 \Rightarrow^l T_2 : T_2} \text{ T-Cast} \qquad \frac{}{\Gamma \vdash_{\cap CC} \mathit{blame}_T\ l : T} \text{ T-Blame}$$

$$\frac{\begin{array}{c}\Gamma \vdash_{\cap CC} e : T \qquad \vdash_{\cap IC} c_1 : T_1\ \ldots\ \vdash_{\cap IC} c_n : T_n \\ \mathit{initialType}(c_1) \cap \ldots \cap \mathit{initialType}(c_n) = T\end{array}}{\Gamma \vdash_{\cap CC} e : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n} \text{ T-IntersectionCast}$$

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

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

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

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

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

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

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

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

Figure 3: Intersection Cast Calculus ($\vdash_{\cap CC}$)

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

$$\frac{x : T \in \Gamma}{\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T} \text{ Var}$$

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

$$\frac{\begin{array}{c}\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM \\ PM \rhd T_1 \cap \ldots \cap T_n \to T \qquad \Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1' \cap \ldots \cap T_n' \\ T_1' \cap \ldots \cap T_n' \sim T_1 \cap \ldots \cap T_n \qquad instances(PM) = S_1 \\ instances(T_1 \cap \ldots \cap T_n \to T) = S_2 \qquad instances(T_1' \cap \ldots \cap T_n') = S_3 \\ instances(T_1 \cap \ldots \cap T_n) = S_4 \qquad S_1,\ S_2,\ e_1' \hookrightarrow e_1'' \qquad S_3,\ S_4,\ e_2' \hookrightarrow e_2''\end{array}}{\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow e_1''\ e_2'' : T} \text{ App}$$

$\boxed{\text{instances(T)} = \{\text{T}\}}$

$$instances(Int) = \{Int\}$$

$$instances(Bool) = \{Bool\}$$

$$instances(Dyn) = \{Dyn\}$$

$$\frac{instances(T_1) = \{T_{11}, \ldots, T_{1n}\}}{instances(T_1 \to T_2) = \{T_{11} \to T_2, \ldots, T_{1n} \to T_2\}}$$

$$\frac{instances(T_1) = \{T_{11}, \ldots, T_{1m}\} \ldots instances(T_n) = \{T_{n1}, \ldots, T_{nj}\}}{instances(T_1 \cap \ldots \cap T_n) = \{T_{11}, \ldots, T_{1m}, \ldots, T_{n1}, \ldots, T_{nj}\}}$$

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

$$\{T_1\},\ \{T_2\},\ e \hookrightarrow e : T_1 \Rightarrow^l T_2$$

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

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

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

Figure 4: Compilation to the Cast Calculus

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

Simulate casts on data types

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

Merge casts

$$\frac{\begin{array}{c} \textit{is value } v : cv_1 \cap \ldots \cap cv_n \\ v : c_1' \cap \ldots \cap c_m' = \textit{mergeIC}(v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2) \end{array}}{v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 \longrightarrow_{\cap CC} v : c_1' \cap \ldots \cap c_m'} \text{ M{\scriptsize ERGE}IC}\cap$$

$$\frac{\begin{array}{c} \textit{is value } v : T_1 \Rightarrow^l T_2 \\ v : c_1' \cap \ldots \cap c_m' = \textit{mergeCI}(v : T_1 \Rightarrow^l T_2 : c_1 \cap \ldots \cap c_n) \end{array}}{v : T_1 \Rightarrow^l T_2 : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : c_1' \cap \ldots \cap c_m'} \text{ M{\scriptsize ERGE}CI}\cap$$

$$\frac{\begin{array}{c} \textit{is value } v : cv_1 \cap \ldots \cap cv_n \\ v : c_1'' \cap \ldots \cap c_j'' = \textit{mergeII}(v : cv_1 \cap \ldots \cap cv_n : c_1' \cap \ldots \cap c_m') \end{array}}{v : cv_1 \cap \ldots \cap cv_n : c_1' \cap \ldots \cap c_m' \longrightarrow_{\cap CC} v : c_1'' \cap \ldots \cap c_j''} \text{ M{\scriptsize ERGE}II}\cap$$

Evaluate intersection casts

$$\frac{\begin{array}{c} \neg(\forall i \in 1..n \; . \; \textit{is cast value } c_i) \\ c_1 \longrightarrow_{\cap IC} cv_1 \;\; \ldots \;\; c_n \longrightarrow_{\cap IC} cv_n \end{array}}{v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n} \text{ E{\scriptsize VALUATE}C{\scriptsize ASTS}}\cap$$

Transition from cast values to values

$$\frac{}{\begin{array}{c} v : \textit{blame } T_1' \; T_1 \; l_1 \;^{cl_1} \cap \ldots \cap \textit{blame } T_n' \; T_n \; l_n \;^{cl_n} \\ \longrightarrow_{\cap CC} \textit{blame}_{(T_1 \cap \ldots \cap T_n)} \; l_1 \end{array}} \text{ P{\scriptsize ROPAGATE}B{\scriptsize LAME}}\cap$$

$$\frac{}{v : \varnothing \; T_1 \;^{cl_1} \cap \ldots \cap \varnothing \; T_n \;^{cl_n} \longrightarrow_{\cap CC} v} \text{ R{\scriptsize EMOVE}E{\scriptsize MPTY}}\cap$$

Figure 5: Cast Calculus Semantics ($\longrightarrow_{\cap CC}$)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 6: Definitions for auxiliary semantic functions

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}(c'_1, \ldots, c'_m) = filter\ (\lambda x\ .\ finalType\ x == T_1)\ (c_1, \ldots, c_n)\\ (cl_1, \ldots, cl_m) = map\ getCastLabel\ (c'_1, \ldots, c'_m)\end{array}}{mergeIC(e : c_1 \cap \ldots \cap c_n : T_1 \Rightarrow^l T_2) = e : (c'_1 : T_1 \Rightarrow^l T_2\ ^{cl_1}) \cap \ldots \cap (c'_m : T_1 \Rightarrow^l T_2\ ^{cl_m})}$$

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

$$\frac{\begin{array}{c}(c'_1, \ldots, c'_m) = filter\ (\lambda x\ .\ initialType\ x == T_2)\ (c_1, \ldots, c_n)\\ (c''_1, \ldots, c''_m) = map\ (\lambda x\ .\ joinCasts\ x\ (\varnothing\ T_1\ ^0 : T_1 \Rightarrow^l T_2\ ^0))\ (c'_1, \ldots, c'_m)\end{array}}{mergeCI(e : T_1 \Rightarrow^l T_2 : c_1 \cap \ldots \cap c_n) = e : c''_1 \cap \ldots \cap c''_m}$$

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

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

Figure 7: Definitions for auxiliary semantic functions

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

$$\frac{\vdash_{\cap IC} c : T_1 \qquad T_1 \sim T_2}{\vdash_{\cap IC} (c : T_1 \Rightarrow^{l} T_2\ ^{cl}) : T_1}\ \text{T-SingleC} \qquad \frac{}{\vdash_{\cap IC} \varnothing\ T\ ^{cl} : T}\ \text{T-EmptyC}$$

$$\frac{}{\vdash_{\cap IC}\ \mathit{blame}\ T_I\ T_F\ l\ ^{cl} : T_F}\ \text{T-BlameC}$$

Figure 8: Intersection Casts Type System ($\vdash_{\cap IC}$)

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

Push blame to top level

$$\frac{}{\mathit{blame}\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap IC} \mathit{blame}\ T_I\ T_2\ l_1\ ^{cl_1}}\ \textsc{PushBlameC}$$

Evaluate inside casts

$$\frac{\neg(\mathit{is\ cast\ value}\ c) \qquad c \longrightarrow_{\cap IC} c'}{c : T_1 \Rightarrow^{l} T_2\ ^{cl} \longrightarrow_{\cap IC} c' : T_1 \Rightarrow^{l} T_2\ ^{cl}}\ \textsc{EvaluateC}$$

Detect success or failure of casts

$$\frac{\mathit{is\ cast\ value}\ 1\ c \vee \mathit{is\ empty\ cast}\ c}{c : T \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap IC} c}\ \textsc{IdentityC}$$

$$\frac{\mathit{is\ cast\ value}\ 1\ c \vee \mathit{is\ empty\ cast}\ c}{c : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl_1} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap IC} c}\ \textsc{SucceedC}$$

$$\frac{\begin{array}{c}\mathit{is\ cast\ value}\ 1\ c \vee \mathit{is\ empty\ cast}\ c \\ \neg(\mathit{same\ ground}\ G_1\ G_2) \qquad \mathit{initialType}(c) = T_I\end{array}}{c : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{cl_1} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap IC} \mathit{blame}\ T_I\ G_2\ l_2\ ^{cl_1}}\ \textsc{FailC}$$

Mediate the transition between the two disciplines

$$\frac{\begin{array}{c}\mathit{is\ cast\ value}\ 1\ c \vee \mathit{is\ empty\ cast}\ c \\ G\ \mathit{is\ ground\ type\ of}\ T \qquad \neg(\mathit{ground}\ T)\end{array}}{c : T \Rightarrow^{l} \mathit{Dyn}\ ^{cl} \longrightarrow_{\cap IC} c : T \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} \mathit{Dyn}\ ^{cl}}\ \textsc{GroundC}$$

$$\frac{\begin{array}{c}\mathit{is\ cast\ value}\ 1\ c \vee \mathit{is\ empty\ cast}\ c \\ G\ \mathit{is\ ground\ type\ of}\ T \qquad \neg(\mathit{ground}\ T)\end{array}}{c : \mathit{Dyn} \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap IC} c : \mathit{Dyn} \Rightarrow^{l} G\ ^{cl} : G \Rightarrow^{l} T\ ^{cl}}\ \textsc{ExpandC}$$

Figure 9: Intersection Casts Semantics ($\longrightarrow_{\cap IC}$)

$\boxed{[e]_e = \text{e}}$ Erase identity casts

$$[x]_e = x$$

$$[\lambda x : T \ . \ e]_e = \lambda x : T \ . \ [e]_e$$

$$[e_1 \ e_2]_e = [e_1]_e \ [e_2]_e$$

$$[n]_e = n$$

$$[true]_e = true$$

$$[false]_e = false$$

$$[e : T \Rightarrow^l T]_e = [e]_e$$

$$[e : T_1 \Rightarrow^l T_2]_e = [e]_e : T_1 \Rightarrow^l T_2$$

$$\frac{[c_1]_c = \varnothing \ T_1 \ ^{cl_1} \ \ldots \ [c_n]_c = \varnothing \ T_n \ ^{cl_n}}{[e : c_1 \cap \ldots \cap c_n]_e = [e]_e}$$

$$\frac{[c_1]_c = c'_1 \ \ldots \ [c_n]_c = c'_n}{[e : c_1 \cap \ldots \cap c_n]_e = [e]_e : c'_1 \cap \ldots \cap c'_n}$$

$\boxed{[c]_c = \text{c}}$ Erase identity casts

$$[c : T \Rightarrow^l T \ ^{cl}]_c = [c]_c$$

$$[c : T_1 \Rightarrow^l T_2 \ ^{cl}]_c = [c]_c : T_1 \Rightarrow^l T_2 \ ^{cl}$$

$$[blame \ T_I \ T_F \ l \ ^{cl}]_c = blame \ T_I \ T_F \ l \ ^{cl}$$

$$[\varnothing \ T \ ^{cl}]_c = \varnothing \ T \ ^{cl}$$

Figure 10: Identity Cast Erasure

2 Proofs

Lemma 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.

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, $Bool = Bool$.
- $T_1 = Dyn$. This case is not considered due to the assumption that T_1 is a static type.

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 $T_{1n} = T_2$. Due to the idempotence property of intersection types, $T_2 \cap \ldots \cap T_2 = T_2$ Therefore, $T_{11} \cap \ldots \cap T_{1n} = T_2$.

□

Theorem 1 (Conservative Extension)**.** *Depends on Lemma 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 left and right direction of the implication, respectively.

Base case:

- Rule 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 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 $True$.
  - 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 $False$.
  - 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 $\to I$.
  - 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 $\to I'$.
  - 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 $\to E$.
  - 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 consistency ($T \sim T$), $T_1 \cap \ldots \cap T_n \sim T_1 \cap \ldots \cap T_n$. 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 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 $\cap I$.
  - 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 $\cap E$.
  - 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 $\Gamma \vdash_{\cap S} e : T_i$.

$\square$

Theorem 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* $T' \sqsubseteq T$ *for some* T'.

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

Base case:

- Rule Var. If $\Gamma \vdash_{\cap G} x : T$ and $x \sqsubseteq x$, then $\Gamma \vdash_{\cap G} x : T$ and $T \sqsubseteq T$.
- Rule Int. If $\Gamma \vdash_{\cap G} n : Int$ and $n \sqsubseteq n$, then $\Gamma \vdash_{\cap G} n : Int$ and $Int \sqsubseteq Int$.
- Rule $True$. If $\Gamma \vdash_{\cap G} true : Bool$ and $true \sqsubseteq true$, then $\Gamma \vdash_{\cap G} true : Bool$ and $Bool \sqsubseteq Bool$.
- Rule $False$. 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 $\cap I$. 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 $\Gamma \vdash_{\cap G} e : T$, $T_1' \cap \ldots \cap T_n' \sqsubseteq T_1 \cap \ldots \cap T_n$ and $e' \sqsubseteq e$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e' : T'$ and $T' \sqsubseteq T$. As $\Gamma \vdash_{\cap G} \lambda x : T_1' \cap \ldots \cap T_n'\ .\ e' : T_1' \cap \ldots \cap T_n' \to T'$, and by the definition of $\sqsubseteq$, $T_1' \cap \ldots \cap T_n' \to T' \sqsubseteq T_1 \cap \ldots \cap T_n \to T$, then it is proved.

- Rule $\cap I'$. If $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \,.\, e : T_i \to T$ and $\lambda x : T_1' \cap \ldots \cap T_n' \,.\, e' \sqsubseteq \lambda x : T_1 \cap \ldots \cap T_n \,.\, e$, then $\Gamma \vdash_{\cap G} e : T$, $T_1' \cap \ldots \cap T_n' \sqsubseteq T_1 \cap \ldots \cap T_n$ and $e' \sqsubseteq e$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e' : T'$ and $T' \sqsubseteq T$. As $\Gamma \vdash_{\cap G} \lambda x : T_1' \cap \ldots \cap T_n' \,.\, e' : T_i' \to T'$, and by the definition of $\sqsubseteq$, $T_i' \to T' \sqsubseteq T_i \to T$, then it is proved.

- Rule $\to E$. If $\Gamma \vdash_{\cap G} e_1\ e_2 : T$ and $e_1'\ e_2' \sqsubseteq e_1\ e_2$ then $\Gamma \vdash_{\cap G} e_1 : PM$, $PM \rhd T_{11} \cap \ldots \cap T_{1n} \to T$, $\Gamma \vdash_{\cap G} e_2 : T_{21} \cap \ldots \cap T_{2n}$, and $T_{21} \cap \ldots \cap T_{2n} \sim T_{11} \cap \ldots \cap T_{1n}$, $e_1' \sqsubseteq e_1$ and $e_2' \sqsubseteq e_2$. By the induction hypothesis, $\Gamma \vdash_{\cap G} e_1' : PM'$ *and* $PM' \sqsubseteq PM$ *and* $PM' \rhd T_{11}' \cap \ldots \cap T_{1n}' \to T'$ 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 $\rhd$, $T_{11}' \cap \ldots \cap T_{1n}' \to T' \sqsubseteq T_{11} \cap \ldots \cap T_{1n} \to T$, and therefore, $T' \sqsubseteq T$. As $\Gamma \vdash_{\cap G} e_1'\ e_2' : T'$, it is proved.

- Rule $\cap I$. If $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$ and $e' \sqsubseteq e$, then $\Gamma \vdash_{\cap G} e : T_1$ 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, $\Gamma \vdash_{\cap G} e' : T_1' \cap \ldots \cap T_n'$ and by the definition of $\sqsubseteq$, $T_1' \cap \ldots \cap T_n' \sqsubseteq T_1 \cap \ldots \cap T_n$, then it is proved.

- Rule $\cap E$. If $\Gamma \vdash_{\cap G} e : T_i$ and $e' \sqsubseteq e$, 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, $\Gamma \vdash_{\cap G} e' : T_1' \cap \ldots \cap T_n'$ and $T_1' \cap \ldots \cap T_n' \sqsubseteq T_1 \cap \ldots \cap T_n$. Therefore, $\Gamma \vdash_{\cap G} e' : T_i'$ and by the definition of $\sqsubseteq$, $T_i' \sqsubseteq T_i$, then it is proved.

□

Theorem 3 (Type preservation of cast insertion). *If* $\Gamma \vdash_{\cap G} e : T$ *then* $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$ *and* $\Gamma \vdash_{\cap CC} e' : T$.

Theorem 4 (Monotonicity of cast insertion). *If* $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : T$ *and* $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T$ *and* $e_1 \sqsubseteq e_2$ *then* $e_1' \sqsubseteq e_2'$.

Lemma 2 (Expressions annotated with only static types type with static types). *If e is annotated with only static types then:*

1. $\Gamma \vdash_{\cap G} e : T$, *for some static* T.
2. $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$, *for some static* T.

Proof. (1) We proceed by induction on the length of the derivation tree of $\vdash_{\cap G}$.

Base cases:

- Rule Var. If $\Gamma \vdash_{\cap G} x : T$, then $x : T \in \Gamma$. Therefore, there must have been at some point in the typing derivation the application of the rules ($\to I$) or ($\to I'$) to type the expression $\lambda x : T.e$, for some e. Both rules introduze the binding $x : T$ in Γ, such that T is a static type.

- Rule Int. As $\Gamma \vdash_{\cap G} n : Int$, it is proved.

- Rule $True$. As $\Gamma \vdash_{\cap G} true : Bool$, it is proved.

- Rule $False$. As $\Gamma \vdash_{\cap G} false : Bool$, it is proved.

Induction step:

- Rule $\to I$. If $\lambda x : T_1 \cap \ldots \cap T_n \;.\; e$ is annotated with only static types, then $T_1 \cap \ldots \cap T_n$ is a static type. By rule $(\to I)$, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap G} e : T$. By the induction hypothesis, T is a static type. Therefore $T_1 \cap \ldots \cap T_n \to T$ is a static type. As $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \;.\; e : T_1 \cap \ldots \cap T_n \to T$, then it is proved.

- Rule $\to I'$. If $\lambda x : T_1 \cap \ldots \cap T_n \;.\; e$ is annotated with only static types, then $T_1 \cap \ldots \cap T_n$ is a static type. By rule $(\to I')$, $\Gamma, x : T_i \vdash_{\cap G} e : T$. Since $T_1 \cap \ldots \cap T_n$ is a static type, then so is T_i. By the induction hypothesis, T is a static type, therefore so is $T_i \to T$. As $\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \ldots \cap T_n \;.\; e : T_i \to T$, then it is proved.

- Rule $\to E$. If $e_1\ e_2$ is annotated with only static types, then so are e_1 and e_2. By the induction hypothesis, PM is a static type. By the definition of $\rhd$, $T_1 \cap \ldots \cap T_n \to T$ is also a static type. Therefore, T is a static type. As $\Gamma \vdash_{\cap G} e_1\ e_2 : T$, then it is proved.

- Rule $\cap I$. If e is annotated with only static types, then by the induction hypothesis, $T_1 \ldots T_n$ are static types. Therefore, $T_1 \cap \ldots \cap T_n$ is a static type. As $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$, then it is proved.

- Rule $\cap E$. If e is annotated with only static types, then by the induction hypothesis, $T_1 \cap \ldots \cap T_n$ is a static type. Therefore, T_i is a static type. As $\Gamma \vdash_{\cap G} e : T_i$, then it is proved.

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

Base cases:

- Rule Var. If $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$, then there is a binding $x : T \in \Gamma$. Therefore, there must have been at some point in the typing derivation, the application of the rule for the term $\lambda x : T_1 \cap \ldots \cap T_n \;.\; e$ for some expressions e. If e is annotated with only static types, then the rule introduzes the binding $x : T$ in Γ, such that T is a static type.

Induction step:

- Rule *Abs*. If $\lambda x : T_1 \cap \ldots \cap T_n \;.\; e$ is annotated with only static types, then $T_1 \cap \ldots \cap T_n$ is a static type. By the induction hypothesis, T is a static type. Therefore $T_1 \cap \ldots \cap T_n \to T$ is a static type. As $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \;.\; e \rightsquigarrow \lambda x : T_1 \cap \ldots \cap T_n \;.\; e' : T_1 \cap \ldots \cap T_n \to T$, then it is proved.

- Rule *App*. If $e_1\ e_2$ is annotated with only static types, then so are e_1 and e_2. By the induction hypothesis, PM is a static type. By the definition of $\rhd$, $T_1 \cap \ldots \cap T_n \to T$ is also a static type. Therefore, T is a static type. As $\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow e_1'\ e_2' : T$, then it is proved.

$\square$

Lemma 3 (Static program compilation only adds identity casts)**.** *Depends on Lemmas 1 and 2. If e is annotated with only static types and* $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$*, then any casts* e' *contains are identity casts.*
By identity casts, we mean casts of the form $e : T \Rightarrow^l T$ *for some* T *and casts* $e : c_1 \cap \ldots \cap c_n$ *such that* $c_1 = \varnothing\ T_1\ ^0 : T_1 \Rightarrow T_1\ ^0$ *and ... and* $c_n = \varnothing\ T_n\ ^0 : T_n \Rightarrow T_n\ ^0$ *for some* $T_1,\ \ldots,\ T_n$*.*

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

Base cases:

- Rule Var. As $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T$, and x doesn't have any casts, then it is proved.

Induction step:

- Rule *Abs*. If $\Gamma \vdash_{\cap CC} (\lambda x : T_1 \cap \ldots \cap T_n\ .\ e) \rightsquigarrow (\lambda x : T_1 \cap \ldots \cap T_n\ .\ e') : T_1 \cap \ldots \cap T_n\ \rightarrow\ T$, then $\Gamma, x\ :\ T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e\ \rightsquigarrow\ e'\ :\ T$. By the induction hypothesis, e' either doesn't contain casts or contains only identity casts. As the rule doesn't introduze new casts, then it is proved.

- Rule *App*. If $\Gamma \vdash_{\cap CC} e_1\ e_2 \rightsquigarrow e_1'\ e_2' : T$, then $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM$ and $PM \rhd T_1 \cap \ldots \cap T_n \rightarrow T$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1' \cap \ldots \cap T_n'$ and $T_1' \cap \ldots \cap T_n' \sim T_1 \cap \ldots \cap T_n$ and $instances(PM) = S_1$ and $instances(T_1 \cap \ldots \cap T_n \rightarrow T) = S_2$ and $instances(T_1' \cap \ldots \cap T_n') = S_3$ and $instances(T_1 \cap \ldots \cap T_n) = S_4$ and $S_1,\ S_2,\ e_1' \hookrightarrow e_1''$ and $S_3,\ S_4,\ e_2' \hookrightarrow e_2''$. By the induction hypothesis, both e_1' as well as e_2' either only have identity casts or no casts at all. By Lemma 2, PM and $T_1' \cap \ldots \cap T_n'$ are static types. Therefore, by the definition of $\rhd$, $PM = T_1 \cap \ldots \cap T_n \rightarrow T$ and by Lemma 1, $T_1' \cap \ldots \cap T_n' = T_1 \cap \ldots \cap T_n$. Therefore by the definition of $instances(T) = \{T\}$ and $S,\ S,\ e \hookrightarrow e$, only identity casts are introduzed.

$\square$

Lemma 4 (Elimination of identity casts in c)**.** *For any cast c, such that* $\vdash_{\cap IC} c : T_F$*,* $initialType(c) = T_I$ *then:*

1. $\vdash_{\cap IC} [c]_c : T_F$ *and* $initialType([c]_c) = T_I$*.*
2. $c \longrightarrow_{\cap IC} cv \iff [c]_c \longrightarrow_{\cap IC} cv$*.*

Proof. (1) We proceed by structural induction on c.

Base cases:

- c $= \varnothing\ T\ ^{cl}$. As $\vdash_{\cap IC} \varnothing\ T\ ^{cl} : T$, $initialType(\varnothing\ T\ ^{cl}) = T$ and $[c]_c = \varnothing\ T\ ^{cl}$, then $\vdash_{\cap IC} [c]_c : T$ and $initialType([c]_c) = T$.

- c $= blame\ T_I\ T_F\ l\ ^{cl}$. As $\vdash_{\cap IC} blame\ T_I\ T_F\ l\ ^{cl} : T_F$, $initialType(blame\ T_I\ T_F\ l\ ^{cl}) = T_I$ and $[c]_c = blame\ T_I\ T_F\ l\ ^{cl}$, then $\vdash_{\cap IC} [c]_c : T_F$ and $initialType([c]_c) = T_I$.

Induction step:

- c $= c' : T_1 \Rightarrow^l T_2\ ^{cl}$. There are two cases:
  - $T_1 \neq T_2$. As $\vdash_{\cap IC} c' : T_1 \Rightarrow^l T_2\ ^{cl} : T_2$ and $initialType(c' : T_1 \Rightarrow^l T_2\ ^{cl}) = initialType(c')$, then $\vdash_{\cap IC} c' : T_1$. By the induction hypothesis, $\vdash_{\cap IC} [c']_c : T_1$ and $initialType([c']_c) = initialType(c')$. With $[c]_c = [c']_c : T_1 \Rightarrow^l T_2\ ^{cl}$, $\vdash_{\cap IC} [c]_c : T_2$ and $initialType([c]_c) = initialType([c']_c) = initialType(c') = initialType(c)$.
  - $T_1 = T_2$. As $\vdash_{\cap IC} c' : T_1 \Rightarrow^l T_1\ ^{cl} : T_1$ and $initialType(c' : T_1 \Rightarrow^l T_1\ ^{cl}) = initialType(c')$ then $\vdash_{\cap IC} c' : T_1$. By the induction hypothesis, $\vdash_{\cap IC} [c']_c : T_1$ and $initialType([c']_c) = initialType(c')$. With $[c]_c = [c']_c$, $\vdash_{\cap IC} [c]_c : T_1$ and $initialType([c]_c) = initialType([c']_c) = initialType(c') = initialType(c)$.

(2) We proceed by induction on the length of the derivation tree of $\longrightarrow_{\cap IC}$.

Base cases:

- Rule $PushBlameC$.
  - There are two cases:
    - $T_1 \neq T_2$. As $[c]_c = blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2}$ and by rule PushBlameC, $blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ T_2\ l_1\ ^{cl_1}$ it is proved.
    - $T_1 = T_2$. If $T_1 = T_2$, then by rules T-SingleC and T-BlameC, $T_F = T_1$. Therefore, $c = blame\ T_I\ T_1\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_1\ ^{cl_2}$. By rule PushBlameC, $blame\ T_I\ T_1\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_1\ ^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ T_1\ l_1\ ^{cl_1}$. Since $[c]_c = blame\ T_I\ T_1\ l_1\ ^{cl_1}$, and it is already a value, it is proved.
  - There are two cases:
    - $T_1 \neq T_2$. As c equals $blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2}$ or may contain adicional identity casts, then $[c]_c = blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2}$. By the rule PushBlameC, $blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ ^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ T_2\ l_1\ ^{cl_1}$. By the rules PushBlameC and IdentityC, $c \longrightarrow_{\cap IC} blame\ T_I\ T_2\ l_1\ ^{cl_1}$, then it is proved.
    - $T_1 = T_2$. As c equals $blame\ T_I\ T_F\ l_1\ ^{cl_1} : T_1 \Rightarrow^{l_2} T_1\ ^{cl_2}$ or may contain adicional identity casts, then $[c]_c = blame\ T_I\ T_F\ l_1\ ^{cl_1}$. As $blame\ T_I\ T_F\ l_1\ ^{cl_1}$ is already a value, it reduced to itself. By rules $T - SingleC$ and $T - BlameC$, $T_F = T_1$. By the rules PushBlameC and IdentityC, $c \longrightarrow_{\cap IC} blame\ T_I\ T_F\ l_1\ ^{cl_1}$, then it is proved.
- Rule $IdentityC$.
  - By rule IdentityC, $c : T \Rightarrow^l T\ ^{cl} \longrightarrow_{\cap IC} c$. As c is a value, it doesn't contain identity casts, therefore $[c]_c = c$. As $[c]_c$ is already a value, it reduces to itself, therefore it is proved.
  - As c equals $c' : T \Rightarrow^l T\ ^{cl}$ or may contain adicional identity casts, then $[c]_c = c'$. As c' is already a vaue, it reduced to itself. By rules IdentityC, $c \longrightarrow_{\cap IC} c'$, then it is proved.

- Rule *SucceedC*.

  - By rule SucceedC, $c : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap IC} c$. As c is already a value, then it doesn't contain identity casts, so $[c]_c = c : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2}$. Therefore, $[c]_c \longrightarrow_{\cap IC} c$.
  - As c equals $c' : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2}$ or may contain adicional identity casts, then $[c]_c = c' : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2}$. By rule SucceedC, $c' : G \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G\ ^{cl_2} \longrightarrow_{\cap IC} c'$. By rules SucceedC and IdentityC, $c \longrightarrow_{\cap IC} c'$, then it is proved.

- Rule *FailC*.

  - By rule FailC, $c : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{cl_1}$. As c is already a value, then it doesn't contain identity casts, so $[c]_c = c : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2}$. Therefore, $[c]_c \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{cl_1}$.
  - As c equals $c' : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2}$ or may contain adicional identity casts, then $[c]_c = c' : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2}$. By rule FailC, $c' : G_1 \Rightarrow^{l_1} Dyn\ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ ^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{cl_1}$. By rules FailC and IdentityC, $c \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{cl_1}$, then it is proved.

- Rule *GroundC*.

  - By rule GroundC, $c : T \Rightarrow^{l} Dyn\ ^{cl} \longrightarrow_{\cap IC} c : T \Rightarrow^{l} G : G \Rightarrow^{l} Dyn\ ^{cl}$. As c is a value, it doesn't contain identity casts, therefore $[c]_c = c : T \Rightarrow^{l} Dyn\ ^{cl}$. Therefore $[c]_c \longrightarrow_{\cap IC} c : T \Rightarrow^{l} G : G \Rightarrow^{l} Dyn\ ^{cl}$.
  - As c equals $c' : T \Rightarrow^{l} Dyn\ ^{cl}$ or may contain adicional identity casts, then $[c]_c = c' : T \Rightarrow^{l} Dyn\ ^{cl}$. By rule GroundC, $c' : T \Rightarrow^{l} Dyn\ ^{cl} \longrightarrow_{\cap IC} c' : T \Rightarrow^{l} G : G \Rightarrow^{l} Dyn\ ^{cl}$. By rules GroundC and IdentityC, $c \longrightarrow_{\cap IC} c' : T \Rightarrow^{l} G : G \Rightarrow^{l} Dyn\ ^{cl}$, then it is proved.

- Rule *ExpandC*.

  - By rule ExpandC, $c : Dyn \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap IC} c : Dyn \Rightarrow^{l} G : G \Rightarrow^{l} T\ ^{cl}$. As c is a value, it doesn't contain identity casts, therefore $[c]_c = c : Dyn \Rightarrow^{l} T\ ^{cl}$. Therefore $[c]_c \longrightarrow_{\cap IC} c : Dyn \Rightarrow^{l} G : G \Rightarrow^{l} T\ ^{cl}$.
  - As c equals $c' : Dyn \Rightarrow^{l} T\ ^{cl}$ or may contain adicional identity casts, then $[c]_c = c' : Dyn \Rightarrow^{l} T\ ^{cl}$. By rule ExpandC, $c' : Dyn \Rightarrow^{l} T\ ^{cl} \longrightarrow_{\cap IC} c' : Dyn \Rightarrow^{l} G : G \Rightarrow^{l} T\ ^{cl}$. By rules ExpandC and IdentityC, $c \longrightarrow_{\cap IC} c' : Dyn \Rightarrow^{l} G : G \Rightarrow^{l} T\ ^{cl}$, then it is proved.

Induction step:

- Rule *EvaluateC*.

  - There are two cases:
    * $T_1 \neq T_2$. By rule EvaluateC, $c \longrightarrow_{\cap IC} c'$. By the induction hypothesis, $[c]_c \longrightarrow_{\cap IC} c'$. As $[c]_c$ equals $[c]_c : T_1 \Rightarrow^{l} T_2\ ^{cl}$, then by rule EvaluateC, $[c]_c \longrightarrow_{\cap IC} c' : T_1 \Rightarrow T_2\ ^{cl}$.

* $T_1 = T_2$. By the induction hypothesis, as $c \longrightarrow_{\cap IC} cv'$, then $\lceil c \rceil_c \longrightarrow_{\cap IC} cv'$. By rule EvaluateC, $c : T_1 \Rightarrow^l T_1\ ^{cl} \longrightarrow_{\cap IC} cv' : T_1 \Rightarrow^l T_1\ ^{cl}$. However, as $cv' : T_1 \Rightarrow^l T_1\ ^{cl}$ is not a value, the rule IdentityC must be applied, therefore $c : T_1 \Rightarrow^l T_1\ ^{cl} \longrightarrow_{\cap IC} cv'$. As $\lceil c \rceil_c \longrightarrow_{\cap IC} cv'$, then it is proved.

- There are two cases:
  * $T_1 \neq T_2$. As c equals $c' : T_1 \Rightarrow^l T_2\ ^{cl}$ or may contain aditional identity casts, then $\lceil c \rceil_c = \lceil c' \rceil_c : T_1 \Rightarrow^l T_2\ ^{cl}$. By rule EvaluateC, $\lceil c' \rceil_c : T_1 \Rightarrow^l T_2\ ^{cl} \longrightarrow_{\cap IC} c'' : T_1 \Rightarrow^l T_2\ ^{cl}$. By rule EvaluateC, $\lceil c' \rceil_c \longrightarrow_{\cap IC} c''$, then by the induction hypothesis $c' \longrightarrow_{\cap IC} c''$. Therefore, by rules EvaluateC and IdentityC, $c \longrightarrow_{\cap IC} c'' : T_1 \Rightarrow^l T_2\ ^cl$, then it is proved.
  * $T_1 = T_2$. As c equals $c' : T_1 \Rightarrow^l T_1\ ^{cl}$ or may contain aditional identity casts, then $\lceil c \rceil_c = \lceil c' \rceil_c$. By rule EvaluateC, $\lceil c' \rceil_c \longrightarrow_{\cap IC} c''$, then by the induction hypothesis $c' \longrightarrow_{\cap IC} c''$. By rules EvaluateC and IdentityC, $c' : T_1 \Rightarrow^l T_1\ ^{cl} \longrightarrow_{\cap IC} c''$, then it is proved.

□

Lemma 5 (Elimination of identity casts in e)**.** *Depends on Lemma 4. For any expression e, such that* $\Gamma \vdash_{\cap CC} e : T$*:*

1. $\Gamma \vdash_{\cap CC} \lceil e \rceil_e : T$.
2. *If* $\Gamma \vdash_{\cap CC} e' \rightsquigarrow e$ *for some* e' *then* $e \longrightarrow_{\cap CC} v \iff \lceil e \rceil_e \longrightarrow_{\cap CC} v$.

Proof. (1) We proceed by induction on the length of the derivation tree of $\Gamma \vdash_{\cap CC} e : T$.

Base cases:

- Rule Var. As x doesn't contain casts, then $\lceil e \rceil_e = x$. Therefore it is proved.
- Rule Int. As n doesn't contain casts, then $\lceil e \rceil_e = n$. Therefore it is proved.
- Rule $True$. As $true$ doesn't contain casts, then $\lceil e \rceil_e = true$. Therefore it is proved.
- Rule $False$. As $false$ doesn't contain casts, then $\lceil e \rceil_e = false$. Therefore it is proved.
- Rule $T\!-\!Blame$. As $blame_T\ l$ doesn't contain casts, then $\lceil e \rceil_e = blame_T\ l$. Therefore it is proved.

Induction step:

- Rule $\rightarrow I$. If $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n\ .\ e : T_1 \cap \ldots \cap T_n \rightarrow T$, then $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} \lceil e \rceil_e : T$. As $\lceil e \rceil_e = \lambda x : T_1 \cap \ldots \cap T_n\ .\ \lceil e \rceil_e$, then $\Gamma \vdash_{\cap CC} \lceil e \rceil_e : T_1 \cap \ldots \cap T_n \rightarrow T$.

- Rule $\rightarrow I'$. If $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \, . \, e : T_i \rightarrow T$, then $\Gamma, x : T_i \vdash_{\cap CC} e : T$. By the induction hypothesis, $\Gamma, x : T_i \vdash_{\cap CC} [e]_e : T$. As $[e]_e = \lambda x : T_1 \cap \ldots \cap T_n \, . \, [e]_e$, then $\Gamma \vdash_{\cap CC} [e]_e : T_i \rightarrow T$.

- Rule $\rightarrow E$. If $\Gamma \vdash_{\cap CC} e_1 \; e_2 : T$, then $\Gamma \vdash_{\cap CC} e_1 : PM$, $PM \rhd T_1 \cap \ldots \cap T_n \rightarrow T$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T'_1 \cap \ldots \cap T'_n \sim T_1 \cap \ldots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e_1]_e : PM$ and $\Gamma \vdash_{\cap CC} [e_2]_e : T'_1 \cap \ldots \cap T'_n$. As $[e]_e = [e_1]_e \; [e_2]_e$, therefore $\Gamma \vdash_{\cap CC} [e]_e : T$.

- Rule $\cap I$. If $\Gamma \vdash_{\cap CC} e : T_1 \cap \ldots \cap T_n$, then $\Gamma \vdash_{\cap CC} e : T_1$ and ... and $\Gamma \vdash_{\cap CC} e : T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e]_e : T_1$ and ... and $\Gamma \vdash_{\cap CC} [e]_e : T_n$. Therefore $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \ldots \cap T_n$.

- Rule $\cap E$. If $\Gamma \vdash_{\cap CC} e : T_i$, then $\Gamma \vdash_{\cap CC} e : T_1 \cap \ldots \cap T_n$, such that $T_i \in \{T_1, \ldots, T_n\}$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \ldots \cap T_n$. Therefore $\Gamma \vdash_{\cap CC} [e]_e : T_i$.

- Rule $T{-}App$. If $\Gamma \vdash_{\cap CC} e_1 \; e_2 : T_{12} \cap \ldots \cap T_{n2}$, then $\Gamma \vdash_{\cap CC} e_1 : T_{11} \rightarrow T_{12} \cap \ldots \cap T_{n1} \rightarrow T_{n2}$ and $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T_{11} \sim T'_1 \ldots T_{n1} \sim T'_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e_1]_e : T_{11} \rightarrow T_{12} \cap \ldots \cap T_{n1} \rightarrow T_{n2}$ and $\Gamma \vdash_{\cap CC} [e_2]_e : T'_1 \cap \ldots \cap T'_n$. As $[e]_e = [e_1]_e \; [e_2]_e$, therefore $\Gamma \vdash_{\cap CC} [e]_e : T_{12} \cap \ldots \cap T_{n2}$.

- Rule $T{-}Cast$. There are two possibilities:

  - $T_1 \neq T_2$. If $\Gamma \vdash_{\cap CC} e' : T_1 \Rightarrow^l T_2 : T_2$, then $\Gamma \vdash_{\cap CC} e' : T_1$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e']_e : T_1$. As $[e]_e = [e']_e : T_1 \Rightarrow^l T_2$, then $\Gamma \vdash_{\cap CC} [e]_e : T_2$.
  - $T_1 = T_2$. If $\Gamma \vdash_{\cap CC} e' : T_1 \Rightarrow^l T_1 : T_1$, then $\Gamma \vdash_{\cap CC} e' : T_1$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e']_e : T_1$. As $[e]_e = [e']_e$, then $\Gamma \vdash_{\cap CC} [e]_e : T_1$.

- Rule $T{-}IntersectionCast$. If $\Gamma \vdash_{\cap CC} e' : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$, then $\Gamma \vdash_{\cap CC} e' : T$, $\vdash_{\cap IC} c_1 : T_1$ and ... and $\vdash_{\cap IC} c_n : T_n$ and $initialType(c_1) \cap \ldots \cap initialType(c_n) = T$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e']_e : T$. We now have 2 possibilities:

  - $\neg(\forall i \in 1..n \, . \, isEmptyCast \; [c_i]_c)$: For all casts c_i, with $i \in 1..n$, that don't contain identity casts, then $[c_i]_c = c_i$, therefore $\vdash_{\cap IC} [c_i]_c : T_i$ and $initialType([c_i]_c) = initialType(c_i)$. For the remaining casts, by Lemma 4, $\vdash_{\cap IC} [c_i]_c : T_i$ and $initialType([c_i]_c) = initialType(c_i)$. Therefore, with $[e]_e = [e']_e : [c_1]_c \cap \ldots \cap [c_n]_c$, $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \ldots \cap T_n$.
  - $\forall i \in 1..n \, . \, isEmptyCast \; [c_i]_c$: As all casts are empty casts, then for all casts $[c_i]_c$, by Lemma 4 and by rule T-EmptyC, $\vdash_{\cap IC} [c_i]_c : T_i$ and $initialType([c_i]_c) = T_i$. Therefore $[e]_e = [e']_e$. We now have two possibilities:

    * If T is not an intersection type, then $T_1 = \ldots = T_n = T$ and by idempotence of $\cap$, we have that $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \ldots \cap T_n$.
    * If T is an intersection type, then $T = T_1 \cap \ldots \cap T_n$. Therefore $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \ldots \cap T_n$.

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

Base cases:

- Rule $\beta{-}reduction$. With $[e]_e = (\lambda x : T\ .\ [e']_e)\ v$.
- Rule $SimulateArrow\cap$. As $v_1 : cv_1 \cap \ldots \cap cv_n$ and v_2 are values, then e doesn't contain identity casts. As $[e]_e = e$, then it is proved.
- Rule $MergeIC\cap$. This case is not considered due to the fact that cast insertion doesn't introduce such expressions.
- Rule $MergeCI\cap$. This case is not considered due to the fact that cast insertion doesn't introduce such expressions.
- Rule $MergeII\cap$. This case is not considered due to the fact that cast insertion doesn't introduce such expressions.
- Rule $EvaluateCasts\cap$.
  - By rule EvaluateCasts$\cap$, $v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$, with $c_1 \longrightarrow_{\cap IC} cv_1$ and ... and $c_n \longrightarrow_{\cap IC} cv_n$. With $[e]_e = v : [c_1]_c \cap \ldots \cap [c_n]_c$, by Lemma 4, $[c_1]_c \longrightarrow_{\cap IC} cv_1$ and ... and $[c_n]_c \longrightarrow_{\cap IC} cv_n$. Therefore, by rule EvaluateCasts$\cap$, $v : [c_1]_c \cap \ldots \cap [c_n]_c \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$.
  - By rule EvaluateCasts$\cap$, $v : [c_1]_c \cap \ldots \cap [c_n]_c \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$, with $[c_1]_c \longrightarrow_{\cap IC} cv_1$ and ... and $[c_n]_c \longrightarrow_{\cap IC} cv_n$. With $[e]_e = v : [c_1]_c \cap \ldots \cap [c_n]_c$, by Lemma 4, $c_1 \longrightarrow_{\cap IC} cv_1$ and ... and $c_n \longrightarrow_{\cap IC} cv_n$. Therefore, by rule EvaluateCasts$\cap$, $v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$.
- Rule $PropagateBlame\cap$. This case is not considered due to the fact that cast insertion doesn't introduce such expressions.
- Rule $RemoveEmpty\cap$. This case is not considered due to the fact that cast insertion doesn't introduce such expressions.

Induction step:

- e =

$\square$

Theorem 5 (Conservative Extension)**.** *Depends on Lemmas 3 and 5. If e is fully static, T is a static type and $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$, then $e \longrightarrow_{\cap S} v \iff e' \longrightarrow_{\cap CC} v$.*

Proof. By Lemma 3, we have that $[e']_e = e$. By Lemma 5, we have that $e' \longrightarrow_{\cap CC} v \iff [e']_e \longrightarrow_{\cap CC} v$. As $[e']_e = e$ and e doesn't contain casts, then we can evaluate e using just the reduction rules (of the gradual operational semantics) analogous to the static operational semantics' reduction rules. Therefore, we have that $e' \longrightarrow_{\cap CC} v \iff e \longrightarrow_{\cap S} v$. $\square$

Lemma 6 (Subject reduction of $\beta{-}reduction$)**.** *If e is a redex and e' is its contractum, then $\Gamma \vdash_{\cap CC} e : T \Rightarrow \Gamma \vdash_{\cap CC} e' : T$.*

Proof. Let $e = (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e_1) \ e_2$. There exists a type $T_1 \cap \ldots \cap T_n$ such that we can deduce $\Gamma \vdash_{\cap CC} e : T$ from $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e_1 : T_1 \cap \ldots \cap T_n \to T$ and $\Gamma \vdash_{\cap CC} e_2 : T_1 \cap \ldots \cap T_n$ (x does not occur in Γ). Moreover, $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \ldots \cap T_n \ . \ e_1 : T_1 \cap \ldots \cap T_n \to T$ only if $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e_1 : T$. By definition, $e' = [x \mapsto e_2] \ e_1$. To obtain $\Gamma \vdash_{\cap CC} [x \mapsto e_2] \ e_1 : T$, it is sufficient to replace, in the proof of $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} e_1 : T$, the statements $x : T_i$ (introduzed by the rules Var and $\cap E$) by the deductions of $\Gamma \vdash_{\cap CC} e_2 : T_i$ for $1 \leq i \leq n$. Proof adapted from [1]. □

Lemma 7 (Subject reduction of $\longrightarrow_{\cap IC}$). *If $\vdash_{\cap IC} c : T$ for some T and $c \longrightarrow_{\cap IC} c'$ then $\vdash_{\cap IC} c' : T$.*

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

Base cases:

- Rule *PushBlameC*. $\vdash_{\cap IC} blame \ T_I \ T_F \ l_1 \ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{cl_2} : T_2$ and by rule PushBlameC, $blame \ T_I \ T_F \ l_1 \ ^{cl_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{cl_2} \longrightarrow_{\cap IC} blame \ T_I \ T_2 \ l_1 \ ^{cl_1}$. As $\vdash_{\cap IC} blame \ T_I \ T_2 \ l_1 \ ^{cl_1} : T_2$, then it is proved.

- Rule *IdentityC*. If $\vdash_{\cap IC} c : T \Rightarrow^{l} T \ ^{cl} : T$, then $\vdash_{\cap IC} c : T$. By rule IdentityC, $c : T \Rightarrow^{l} T \ ^{cl} \longrightarrow_{\cap IC} c$. Therefore it is proved.

- Rule *SucceedC*. If $\vdash_{\cap IC} c : G \Rightarrow^{l_1} Dyn \ ^{cl_1} : Dyn \Rightarrow^{l_2} G \ ^{cl_2} : G$, then $\vdash_{\cap IC} c : G$. By rule SucceedC, $c : G \Rightarrow^{l_1} Dyn \ ^{cl_1} : Dyn \Rightarrow^{l_2} G \ ^{cl_2} \longrightarrow_{\cap IC} c$. Therefore it is proved.

- Rule *FailC*. If $\vdash_{\cap IC} c : G_1 \Rightarrow^{l_1} Dyn \ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2 \ ^{cl_2} : G_2$, and by rule FailC, $c : G_1 \Rightarrow^{l_1} Dyn \ ^{cl_1} : Dyn \Rightarrow^{l_2} G_2 \ ^{cl_2} \longrightarrow_{\cap IC} blame \ T_I \ G_2 \ l_2 \ ^{cl_1}$ and $\vdash_{\cap IC} blame \ T_I \ G_2 \ l_2 \ ^{cl_1} : G_2$, it is proved.

- Rule *GroundC*. If $\vdash_{\cap IC} c : T \Rightarrow^{l} Dyn \ ^{cl} : Dyn$ then $\vdash_{\cap IC} c : T$. By rule GroundC, $c : T \Rightarrow^{l} Dyn \ ^{cl} \longrightarrow_{\cap IC} c : T \Rightarrow^{l} G \ ^{cl} : G \Rightarrow^{l} Dyn \ ^{cl}$. As $\vdash_{\cap IC} c : T \Rightarrow^{l} G \ ^{cl} : G \Rightarrow^{l} Dyn \ ^{cl} : Dyn$, it is proved.

- Rule *ExpandC*. If $\vdash_{\cap IC} c : Dyn \Rightarrow^{l} T \ ^{cl} : T$ then $\vdash_{\cap IC} c : Dyn$. By rule ExpandC, $c : Dyn \Rightarrow^{l} T \ ^{cl} \longrightarrow_{\cap IC} c : Dyn \Rightarrow^{l} G \ ^{cl} : G \Rightarrow^{l} T \ ^{cl}$. As $\vdash_{\cap IC} c : Dyn \Rightarrow^{l} G \ ^{cl} : G \Rightarrow^{l} T \ ^{cl} : T$, it is proved.

Induction step:

- Rule *EvaluateC*. If $\vdash_{\cap IC} c : T_1 \Rightarrow^{l} T_2 \ ^{cl} : T_2$ then $\vdash_{\cap IC} c : T_1$. By rule EvaluateC, $c \longrightarrow_{\cap IC} c'$. By the induction hypothesis, $\vdash_{\cap IC} c' : T_1$. By rule EvaluateC, $c : T_1 \Rightarrow^{l} T_2 \ ^{cl} \longrightarrow_{\cap IC} c' : T_1 \Rightarrow^{l} T_2 \ ^{cl}$. As $\vdash_{\cap IC} c' : T_1 \Rightarrow^{l} T_2 \ ^{cl} : T_2$ it is proved.

□

Lemma 8 (Initial type preservation of $\longrightarrow_{\cap IC}$). *If $initialType(c) = T$ for some T and $c \longrightarrow_{\cap IC} c'$ then $initialType(c') = T$.*

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

Base cases:

- Rule *PushBlameC*. 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 PushBlameC, $blame\ T_I\ T_F\ l_1\ {}^{cl_1} : T_1 \Rightarrow^{l_2} T_2\ {}^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ T_2\ l_1\ {}^{cl_1}$. Since $initialType(blame\ T_I\ T_2\ l_1\ {}^{cl_1}) = T_I$, it is proved.

- Rule *IdentityC*. By the definitions of *initialType*, $initialType(c : T \Rightarrow^l T\ {}^{cl}) = initialType(c)$. By rule IdentityC, $c : T \Rightarrow^l T\ {}^{cl} \longrightarrow_{\cap IC} c$. Therefore it is proved.

- Rule *SucceedC*. By the definition of *initialType*, $initialType(c : G \Rightarrow^{l_1} Dyn\ {}^{cl_1} : Dyn \Rightarrow^{l_2} G\ {}^{cl_2}) = initialType(c)$. By rule SucceedC, $c : G \Rightarrow^{l_1} Dyn\ {}^{cl_1} : Dyn \Rightarrow^{l_2} G\ {}^{cl_2} \longrightarrow_{\cap IC} c$. Therefore it is proved.

- Rule *FailC*. By the definition of *initialType*, $initialType(c : G_1 \Rightarrow^{l_1} Dyn\ {}^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ {}^{cl_2}) = T_I$. By rule FailC, $c : G_1 \Rightarrow^{l_1} Dyn\ {}^{cl_1} : Dyn \Rightarrow^{l_2} G_2\ {}^{cl_2} \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ {}^{cl_1}$. Since $initialType(blame\ T_I\ G_2\ l_2\ {}^{cl_1}) = T_I$, it is proved.

- Rule *GroundC*. By the definition of *initialType*, $initialType(c : T \Rightarrow^l Dyn\ {}^{cl}) = initialType(c)$. By rule GroundC, $c : T \Rightarrow^l Dyn\ {}^{cl} \longrightarrow_{\cap IC} c : T \Rightarrow^l G\ {}^{cl} : G \Rightarrow^l Dyn\ {}^{cl}$. Since $initialType(c : T \Rightarrow^l G\ {}^{cl} : G \Rightarrow^l Dyn\ {}^{cl}) = initialType(c)$, it is proved.

- Rule *ExpandC*. By the definition of *initialType*, $initialType(c : Dyn \Rightarrow^l T\ {}^{cl}) = initialType(c)$. By rule ExpandC, $c : Dyn \Rightarrow^l T\ {}^{cl} \longrightarrow_{\cap IC} c : Dyn \Rightarrow^l G\ {}^{cl} : G \Rightarrow^l T\ {}^{cl}$. Since $initialType(c : Dyn \Rightarrow^l G\ {}^{cl} : G \Rightarrow^l T\ {}^{cl}) = initialType(c)$, it is proved.

Induction step:

- Rule *EvaluateC*. By the definition of *initialType*, $initialType(c : T_1 \Rightarrow^l T_2\ {}^{cl}) = initialType(c)$. By rule EvaluateC, $c \longrightarrow_{\cap IC} c'$. By the induction hypothesis, $initialType(c') = initialType(c)$. By rule EvaluateC, $c : T_1 \Rightarrow^l T_2\ {}^{cl} \longrightarrow_{\cap IC} c' : T_1 \Rightarrow^l T_2\ {}^{cl}$. Since $initialType(c' : T_1 \Rightarrow^l T_2\ {}^{cl}) = initialType(c')$, it is proved.

□

Theorem 6 (Subject reduction of $\longrightarrow_{\cap CC}$)**.** *Depends on Lemmas 6, 7 and 8. If* $\Gamma \vdash_{\cap CC} e : T$ *and* $e \longrightarrow_{\cap CC} e'$ *then* $\Gamma \vdash_{\cap CC} e' : T$.

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

Base case:

- Rule *β−reduction*. The proof of $\Gamma \vdash_{\cap CC} e' : T$ can be obtained from that of $\Gamma \vdash_{\cap CC} e : T$ by replacing any deduction of a type for e, by the corresponding deduction of the same type for e' (by Lemma 6).

- Rule *Simulate∩*. If $\Gamma \vdash_{\cap CC} (v_1 : cv_1 \cap \ldots \cap cv_n)\ v_2 : T_{12} \cap \ldots \cap T_{n2}$, then $\Gamma \vdash_{\cap CC} v_1 : cv_1 \cap \ldots \cap cv_n : T_1 \cap \ldots \cap T_n$ with $\vdash_{\cap IC} cv_1 : T_1$ and ... and $\vdash_{\cap IC} cv_n : T_n$, such that $\exists i \in 1..n\ .\ T_i = T_{i1} \rightarrow T_{i2}$ and $\Gamma \vdash_{\cap CC} v_1 : T'_1 \cap \ldots \cap T'_l$ and $I_1 = initialType(cv_1)$ and ... and $I_n = initialType(cv_n)$ such that either $T'_1 \cap \ldots \cap T'_l = I_1 \cap \ldots \cap I_n$ or $\{I_1, \ldots, I_n\} \subset \{T'_1, \ldots, T'_l\}$

and $\Gamma \vdash_{\cap CC} v_2 : T_{11} \cap \ldots \cap T_{n1}$. For the sake of simplicity lets elide cast labels and blame labels. As $\vdash_{\cap IC} cv'_1 : T_{11} \rightarrow T_{12}$ and ... and $\vdash_{\cap IC} cv'_m : T_{m1} \rightarrow T_{m2}$ then $cv'_1 = cv''_1 : T'_{11} \rightarrow T'_{12} \Rightarrow T_{11} \rightarrow T_{12}$ and ... and $cv'_m = cv''_m : T'_{m1} \rightarrow T'_{m2} \Rightarrow T_{m1} \rightarrow T_{m2}$. By the definition of simulateArrow, $c_{11} : \varnothing\ T_{11}\ : T_{11} \Rightarrow T'_{11}$ 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 $initialType(r_1) = I_1$ and ... and $initialType(r_m) = I_m$ and $\vdash_{\cap IC} r_1 : T'_{11} \rightarrow T'_{12}$ and ... and $\vdash_{\cap IC} r_m : T'_{m1} \rightarrow T'_{m2}$. Therefore $\Gamma \vdash_{\cap CC} v_1 : r_1 \cap \ldots \cap r_m : T'_{11} \rightarrow T'_{12} \cap \ldots \cap T'_{m1} \rightarrow T'_{m2}$ and $\Gamma \vdash_{\cap CC} v_2 : c_{11} \cap \ldots \cap c_{m1} : T'_{11} \cap \ldots \cap T'_{m1}$ and therefore $\Gamma \vdash_{\cap CC} (v_1 : r_1 \cap \ldots \cap r_m)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : T'_{12} \cap \ldots \cap T'_{m2}$. Therefore, $\Gamma \vdash_{\cap CC} (v_1 : r_1 \cap \ldots \cap r_m)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : c_{12} \cap \ldots \cap c_{m2} : T_{12} \cap \ldots \cap T_{m2}$, such that $\{T_{12}, \ldots, T_{m2}\} \subset \{T_{12}, \ldots, T_{n2}\}$. By rule Simulate$\cap$, $(v_1 : cv_1 \cap \ldots \cap cv_n)\ v_2 \longrightarrow_{\cap CC} (v_1 : r_1 \cap \ldots \cap r_m)\ (v_2 : c_{11} \cap \ldots \cap c_{m1}) : c_{12} \cap \ldots \cap c_{m2}$, therefore it is proved.

- Rule $MergeIC\cap$. If $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 : T_2$ then $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : T_{11} \cap \ldots \cap T_{1n}$ and such that $\exists T_{1i} . T_{1i} = T_1$ and $\vdash_{\cap IC} cv_1 : T_{11}$ and $I_1 = initialType(cv_1)$ and ... and $\vdash_{\cap IC} cv_n : T_{1n}$ and $I_n = initialType(cv_n)$ and $\Gamma \vdash_{\cap CC} v : I_1 \cap \ldots \cap I_n$ and $m \leq n$. By the definition of mergeIC, $\vdash_{\cap IC} c'_1 : T_2$ and $initialType(c'_1) : I_1$ and ... and $\vdash_{\cap IC} c'_m : T_2$ and $initialType(c'_m) : I_m$. As $\Gamma \vdash_{\cap CC} v : I_1 \cap \ldots \cap I_m$ and therefore $\Gamma \vdash_{\cap CC} v : c'_1 \cap \ldots \cap c'_m : T_2 \cap \ldots \cap T_2$ and $T_2 \cap \ldots \cap T_2 = T_2$ and by rule MergeIC$\cap$, $v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 \longrightarrow_{\cap CC} v : c'_1 \cap \ldots \cap c'_m$, then it is proved.

- Rule $MergeCI\cap$. If $\Gamma \vdash_{\cap CC} v : T_1 \Rightarrow^l T_2 : c_1 \cap \ldots \cap c_n : F_1 \cap \ldots \cap F_n$ then $\Gamma \vdash_{\cap CC} v : T_1 \Rightarrow T_2 : T_2$ and $\Gamma \vdash_{\cap CC} v : T_1$ and $\vdash_{\cap IC} c_1 : F_1$ and $initialType(c_1) = T_2$ and ... and $\vdash_{\cap IC} c_n : F_n$ and $initialType(c_n) = T_2$. By the definition of mergeCI, $mergeCI(v : T_1 \Rightarrow^l T_2 : c_1 \cap \ldots \cap c_n) = v : c'_1 \cap \ldots \cap c'_n$, such that $\vdash_{\cap IC} c'_1 : F_1$ and $initialType(c'_1) : T_1$ and ... and $\vdash_{\cap IC} c'_n : F_n$ and $initialType(c'_n) : T_1$. As $\Gamma \vdash_{\cap CC} v : c'_1 \cap \ldots \cap c'_n : F_1 \cap \ldots \cap F_n$ and by rule MergeCI$\cap$, $v : T_1 \Rightarrow^l T_2 : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : c'_1 \cap \ldots \cap c'_n$, then it is proved.

- Rule $MergeII\cap$. 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 $\vdash_{\cap IC} c'_1 : F'_1$ and $initialType(c'_1) = I'_1$ and ... and $\vdash_{\cap IC} c'_m : F'_m$ and $initialType(c'_m) = I'_m$ and $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : F_1 \cap \ldots \cap F_n$ and $\vdash_{\cap IC} cv_1 : F_1$ and $initialType(cv_1) = I_1$ and ... and $\vdash_{\cap IC} cv_n : F_n$ and $initialType(cv_n) = I_n$ and $\Gamma \vdash_{\cap CC} v : T_1 \cap \ldots \cap T_l$ such that either $T_1 \cap \ldots \cap T_l = I_1 \cap \ldots \cap I_n$ or $\{I_1, \ldots, I_n\} \subset \{T_1, \ldots, T_l\}$. There are two possibilities:
  - $F_1 \cap \ldots \cap F_n = I'_1 \cap \ldots \cap I'_m$. By the definition of mergeII, $\vdash_{\cap IC} c''_1 : F''_1$ and ... and $\vdash_{\cap IC} c''_j : F''_j$ such that $F''_1 \cap \ldots \cap F''_j = F'_1 \cap \ldots \cap F'_m$ and $initialType(c''_1) = I''_1$ and ... and $initialType(c''_j) = I''_j$ such that $I''_1 \cap \ldots \cap I''_j = I_1 \cap \ldots \cap I_n$. Therefore $\Gamma \vdash_{\cap CC} v : c''_1 \cap \ldots \cap c''_j : F''_1 \cap \ldots \cap F''_j$. By rule MergeII$\cap$, $v : cv_1 \cap \ldots \cap cv_n : c'_1 \cap \ldots \cap c'_m \longrightarrow_{\cap CC} v : c''_1 \cap \ldots \cap c''_j$. Therefore it is proved.
  - $\{I'_1, \ldots, I'_m\} \subset \{F_1, \ldots, F_n\}$. By the definition of mergeII, $\vdash_{\cap IC} c''_1 : F''_1$ and $initialType(c''_1) = I''_1$ and ... and $\vdash_{\cap IC} c''_j : F''_j$ and

$initialType(c''_j) = I''_j$ such that $\{I''_1, \ldots, I''_j\} \subset \{I_1, \ldots, I_n\}$ and $\{F''_1, \ldots, F''_j\} \subset \{F'_1, \ldots, F'_m\}$. Therefore, $\Gamma \vdash_{\cap CC} v : c''_1 \cap \ldots \cap c''_j : F''_1 \cap \ldots \cap F''_j$. By rule MergeII$\cap$, $v : cv_1 \cap \ldots \cap cv_n : c'_1 \cap \ldots \cap c'_m \longrightarrow_{\cap CC} v : c''_1 \cap \ldots \cap c''_j$. Therefore, it is proved.

- Rule *EvaluateCasts*$\cap$. If $\Gamma \vdash_{\cap CC} v : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$ then $\vdash_{\cap IC} c_1 : T_1$ and $I_1 = initialType(c_1)$ and ... and $\vdash_{\cap IC} c_n : T_n$ and $I_n = initialType(c_n)$ and $\Gamma \vdash_{\cap CC} v : I_1 \cap \ldots \cap I_n$. By rule EvaluateCasts$\cap$, $c_1 \longrightarrow_{\cap IC} cv_1$ and ... and $c_n \longrightarrow_{\cap IC} cv_n$. By Lemmas 7 and 8, $\vdash_{\cap IC} cv_1 : T_1$ and $initialType(cv_1) = I_1$ and ... and $\vdash_{\cap IC} cv_n : T_n$ and $initialType(cv_n) = I_n$. Therefore $\Gamma \vdash_{\cap CC} v : cv_1 \cap \ldots \cap cv_n : T_1 \cap \ldots \cap T_n$. By rule EvaluateCasts$\cap$, $v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$, then it is proved.

- Rule *PropagateBlame*$\cap$. 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 PropagateBlame$\cap$ $v : blame\ T'_1\ T_1\ l_1\ ^{m_1} \cap \ldots \cap blame\ T'_n\ T_n\ l_n\ ^{m_n} \longrightarrow_{\cap CC} blame_{(T_1 \cap \ldots \cap T_n)}\ l_1$, and $\Gamma \vdash_{\cap CC} blame_{(T_1 \cap \ldots \cap T_n)}\ l_1 : T_1 \cap \ldots \cap T_n$, then it is proved.

- Rule *RemoveEmpty*$\cap$.. 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 $\vdash_{\cap IC} \varnothing\ T_1\ ^{m_1} : T_1$ and $initialType(\varnothing\ T_1\ ^{m_1}) = T_1$ and ... and $\vdash_{\cap IC} \varnothing\ T_n\ ^{m_n} : T_n$ and $initialType(\varnothing\ T_n\ ^{m_n}) = T_n$ and $\Gamma \vdash_{\cap CC} v : T_1 \cap \ldots \cap T_n$. By rule RemoveEmpty$\cap$, $v : \varnothing\ T_1\ ^{m_1} \cap \ldots \cap \varnothing\ T_n\ ^{m_n} \longrightarrow_{\cap CC} v$, therefore it is proved.

Induction step:

- Rule *E−App*1. If $\Gamma \vdash_{\cap CC} e_1\ e_2 : T$, then either:
  - $\Gamma \vdash_{\cap CC} e_1 : PM$, $PM \rhd T_1 \cap \ldots \cap T_n \to T$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T'_1 \cap \ldots \cap T'_n \sim T_1 \cap \ldots \cap T_n$ or
  - $\Gamma \vdash_{\cap CC} e_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2}$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T_{11} \sim T'_1 \ldots T_{n1} \sim T'_n$.

  By rule E-App1, $e_1 \longrightarrow_{\cap IC} e'_1$, so by the induction hypothesis either:
  - $\Gamma \vdash_{\cap CC} e'_1 : PM$ or
  - $\Gamma \vdash_{\cap CC} e'_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2}$.

  Therefore, $\Gamma \vdash_{\cap CC} e'_1\ e_2 : T$. As by rule E-App1, $e_1\ e_2 \longrightarrow_{\cap IC} e'_1\ e_2$, it is proved.

- Rule *E−App*2. If $\Gamma \vdash_{\cap CC} v_1\ e_2 : T$, then either:
  - $\Gamma \vdash_{\cap CC} v_1 : PM$, $PM \rhd T_1 \cap \ldots \cap T_n \to T$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T'_1 \cap \ldots \cap T'_n \sim T_1 \cap \ldots \cap T_n$ or
  - $\Gamma \vdash_{\cap CC} v_1 : T_{11} \to T_{12} \cap \ldots \cap T_{n1} \to T_{n2}$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T_{11} \sim T'_1 \ldots T_{n1} \sim T'_n$.

  By rule E-App2, $e_2 \longrightarrow_{\cap IC} e'_2$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e'_2 : T'_1 \cap \ldots \cap T'_n$. Therefore, $\Gamma \vdash_{\cap CC} v_1\ e'_2 : T$. As by rule E-App2, $v_1\ e_2 \longrightarrow_{\cap IC} v_1\ e'_2$, it is proved.

- Rule $E{-}EvaluateCasts$. If $\Gamma \vdash_{\cap CC} e : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$, then $\Gamma \vdash_{\cap CC} e : T$, $\vdash_{\cap IC} c_1 : T_1$ and ... and $\vdash_{\cap IC} c_n : T_n$ and $initialType(c_1) \cap \ldots \cap initialType(c_n) = T$. By rule E-EvaluateCasts, $e \longrightarrow_{\cap IC} e'$, so by the induction hypothesis, $\Gamma \vdash_{\cap CC} e' : T$. Therefore, $\Gamma \vdash_{\cap CC} e' : c_1 \cap \ldots \cap c_n : T_1 \cap \ldots \cap T_n$. As by rule E-EvaluateCasts, $e : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap IC} e' : c_1 \cap \ldots \cap c_n$, it is proved.

□

Theorem 7 (Progress of $\longrightarrow_{\cap CC}$)**.** *If* $\Gamma \vdash_{\cap CC} e : T$ *then* $e \longrightarrow_{\cap CC} v$*.*

Theorem 8 (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 9 (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* $e_1 \longrightarrow_{\cap IC} e_1'$ *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* $e_2' \longrightarrow_{\cap CC} blame_T\ l$*.*

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.