

Gradual Intersection Types

Pedro Ângelo

January 10, 2018

1 Language Definition

Syntax

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

Figure 1: Gradual Intersection System

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

$$\begin{array}{c}
\frac{\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\cap G} e : T}{\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \dots \cap T_n . e : T_1 \cap \dots \cap T_n \rightarrow T} \rightarrow I \\
\\
\frac{\Gamma, x : T_i \vdash_{\cap G} e : T}{\Gamma \vdash_{\cap G} \lambda x : T_1 \cap \dots \cap T_n . e : T_i \rightarrow T} \rightarrow I' \\
\\
\frac{\Gamma \vdash_{\cap G} e_1 : PM \quad PM \triangleright T_1 \cap \dots \cap T_n \rightarrow T \quad \Gamma \vdash_{\cap G} e_2 : T'_1 \cap \dots \cap T'_n \quad T'_1 \cap \dots \cap T'_n \sim T_1 \cap \dots \cap T_n}{\Gamma \vdash_{\cap G} e_1 e_2 : T} \rightarrow E \\
\\
\frac{\Gamma \vdash_{\cap G} e : T_1 \dots \Gamma \vdash_{\cap G} e : T_n}{\Gamma \vdash_{\cap G} e : T_1 \cap \dots \cap T_n} \cap I \quad \frac{\Gamma \vdash_{\cap G} e : T_1 \cap \dots \cap T_n}{\Gamma \vdash_{\cap G} e : T_i} \cap E \\
\\
\frac{x : T \in \Gamma}{\Gamma \vdash_{\cap G} x : T} Var
\end{array}$$

$\boxed{T \sim T}$ Consistency

$$\begin{array}{c}
\overline{B \sim B} \quad \overline{T \sim Dyn} \quad \overline{Dyn \sim T} \quad \frac{T_1 \sim T_3 \quad T_2 \sim T_4}{T_1 \rightarrow T_2 \sim T_3 \rightarrow T_4} \\
\\
\frac{T_1 \sim T'_1 \dots T_n \sim T'_n}{T_1 \cap \dots \cap T_n \sim T'_1 \cap \dots \cap T'_n} \quad \frac{T \sim T_1 \dots T \sim T_n}{T \sim T_1 \cap \dots \cap T_n} \quad \frac{T_1 \sim T \dots T_n \sim T}{T_1 \cap \dots \cap T_n \sim T}
\end{array}$$

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

$$\overline{(T_1 \rightarrow T_2) \triangleright T_1 \rightarrow T_2} \quad \overline{Dyn \triangleright Dyn \rightarrow Dyn}$$

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

$$\begin{array}{c}
\overline{Dyn \sqsubseteq T} \quad \overline{B \sqsubseteq B} \quad \frac{T_1 \sqsubseteq T_3 \quad T_2 \sqsubseteq T_4}{T_1 \rightarrow T_2 \sqsubseteq T_3 \rightarrow T_4} \\
\\
\frac{T_1 \sqsubseteq T'_1 \dots T_n \sqsubseteq T'_n}{T_1 \cap \dots \cap T_n \sqsubseteq T'_1 \cap \dots \cap T'_n} \quad \frac{T \sqsubseteq T_1 \dots T \sqsubseteq T_n}{T \sqsubseteq T_1 \cap \dots \cap T_n} \quad \frac{T_1 \sqsubseteq T \dots T_n \sqsubseteq T}{T_1 \cap \dots \cap T_n \sqsubseteq T}
\end{array}$$

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

$$\frac{}{x \sqsubseteq e} \quad \frac{T \sqsubseteq T' \quad e \sqsubseteq e'}{\lambda x : T . e \sqsubseteq \lambda x : T' . e'} \quad \frac{e_1 \sqsubseteq e'_1 \quad e_2 \sqsubseteq e'_2}{e_1 e_2 \sqsubseteq e'_1 e'_2}$$

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

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

rules in Figure 2 and

$$\frac{\Gamma \vdash_{\cap CC} e_1 : T_{11} \rightarrow T_{12} \cap \dots \cap T_{n1} \rightarrow T_{n2} \quad \Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \dots \cap T'_n \quad T_{11} \sim T'_1 \dots T_{n1} \sim T'_n}{\Gamma \vdash_{\cap CC} e_1 e_2 : T_{12} \cap \dots \cap T_{n2}} \text{T-APP}$$

$$\frac{\Gamma \vdash_{\cap CC} e : T_1 \quad T_1 \sim T_2}{\Gamma \vdash_{\cap CC} (e : T_1 \Rightarrow^l T_2) : T_2} \text{T-CAST} \quad \frac{}{\Gamma \vdash_{\cap CC} \text{blame}_T l : T} \text{T-BLAME}$$

$$\frac{\Gamma \vdash_{\cap CC} e : T \quad \vdash_{\cap IC} c_1 : T_1 \dots \vdash_{\cap IC} c_n : T_n \quad \text{initialType}(c_1) \cap \dots \cap \text{initialType}(c_n) =_{\cap} T}{\Gamma \vdash_{\cap CC} (e : c_1 \cap \dots \cap c_n) : T_1 \cap \dots \cap T_n} \text{T-INTERSECTIONCAST}$$

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

$$\text{initialType}(c : T_1 \Rightarrow^l T_2 \cap \dots) = \text{initialType}(c)$$

$$\text{initialType}(\emptyset T \cap \dots) = T$$

$$\text{initialType}(\text{blame } T_I T_F l \cap \dots) = T_I$$

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

$$\text{finalType}(c : T_1 \Rightarrow^l T_2 \cap \dots) = T_2$$

$$\text{finalType}(\emptyset T \cap \dots) = T$$

$$\text{finalType}(\text{blame } T_I T_F l \cap \dots) = T_F$$

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

$\boxed{\Gamma \vdash_{\text{NCC}} e \rightsquigarrow e : T}$ Compilation

$$\frac{x : T_1 \cap \dots \cap T_n \in \Gamma}{\Gamma \vdash_{\text{NCC}} x \rightsquigarrow x : T_i}$$

$$\frac{\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\text{NCC}} e \rightsquigarrow e' : T}{\Gamma \vdash_{\text{NCC}} (\lambda x : T_1 \cap \dots \cap T_n . e) \rightsquigarrow (\lambda x : T_1 \cap \dots \cap T_n . e') : T_1 \cap \dots \cap T_n \rightarrow T}$$

$$\frac{\begin{array}{l} \Gamma \vdash_{\text{CC}} e_1 \rightsquigarrow e'_1 : PM \quad PM \triangleright T_1 \cap \dots \cap T_n \rightarrow T \\ \Gamma \vdash_{\text{CC}} e_2 \rightsquigarrow e'_2 : T'_1 \cap \dots \cap T'_n \quad T'_1 \cap \dots \cap T'_n \sim T_1 \cap \dots \cap T_n \\ e''_1 = \text{addCasts}(\text{getInstances}(PM), \text{getInstances}(T_1 \cap \dots \cap T_n \rightarrow T), e'_1) \\ e''_2 = \text{addCasts}(\text{getInstances}(T'_1 \cap \dots \cap T'_n), \text{getInstances}(T_1 \cap \dots \cap T_n), e'_2) \end{array}}{\Gamma \vdash_{\text{CC}} e_1 e_2 \rightsquigarrow e''_1 e''_2 : T}$$

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

$$\text{getInstances}(Int) = \{Int\}$$

$$\text{getInstances}(Bool) = \{Bool\}$$

$$\text{getInstances}(Dyn) = \{Dyn\}$$

$$\begin{array}{l} \text{getInstances}(T_1 \rightarrow T_2) = \\ \text{let } \{T_{11}, \dots, T_{1n}\} = \text{getInstances}(T_1) \text{ in } \{T_{11} \rightarrow T_2, \dots, T_{1n} \rightarrow T_2\} \end{array}$$

$$\begin{array}{l} \text{getInstances}(T_1 \cap \dots \cap T_n) = \\ \text{let } \{T_{11}, \dots, T_{1m}\} = \text{getInstances}(T_1) \end{array}$$

$$\begin{array}{l} \dots \\ \text{let } \{T_{n1}, \dots, T_{nj}\} = \text{getInstances}(T_n) \\ \text{in } \{T_{11}, \dots, T_{1m}, \dots, T_{n1}, \dots, T_{nj}\} \end{array}$$

$\boxed{\text{addCasts}(\{T\}, \{T\}, e) = e}$

$$\text{addCasts}(\{T_1\}, \{T_2\}, e) = e : T_1 \Rightarrow^l T_2$$

$$\begin{array}{l} \text{addCasts}(\{T_{11}, \dots, T_{1n}\}, \{T_{21}, \dots, T_{2n}\}, e) = \\ e : (\emptyset T_{11}^0 : T_{11} \Rightarrow^l T_{21}^0) \cap \dots \cap (\emptyset T_{1n}^0 : T_{1n} \Rightarrow^l T_{2n}^0) \end{array}$$

$$\begin{array}{l} \text{addCasts}(\{T_{11}, \dots, T_{1n}\}, \{T_2\}, e) = \\ e : (\emptyset T_{11}^0 : T_{11} \Rightarrow^l T_2^0) \cap \dots \cap (\emptyset T_{1n}^0 : T_{1n} \Rightarrow^l T_2^0) \end{array}$$

$$\begin{array}{l} \text{addCasts}(\{T_1\}, \{T_{21}, \dots, T_{2n}\}, e) = \\ e : (\emptyset T_1^0 : T_1 \Rightarrow^l T_{21}^0) \cap \dots \cap (\emptyset T_1^0 : T_1 \Rightarrow^l T_{2n}^0) \end{array}$$

Figure 4: Compilation to the Cast Calculus

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

Simulate casts on data types

$$\frac{\begin{array}{l} isValue\ v_1 : cv_1 \cap \dots \cap cv_n \quad \exists i \in 1..n . isArrowCompatible\ cv_i \\ (cv'_1, \dots, cv'_m) = filter\ isArrowCompatible\ (cv_1, \dots, cv_n) \\ ((c_{11}, c_{12}, r_1), \dots, (c_{m1}, c_{m2}, r_m)) = map\ simulateArrow\ (cv'_1, \dots, cv'_m) \end{array}}{(v_1 : cv_1 \cap \dots \cap cv_n)\ v_2 \longrightarrow_{\cap CC} (v_1 : r_1 \cap \dots \cap r_m)\ (v_2 : c_{11} \cap \dots \cap c_{m1}) : c_{12} \cap \dots \cap c_{m2}} \text{SIMULATE}\cap$$

Merge casts

$$\frac{isValue\ v : cv_1 \cap \dots \cap cv_n \quad v : c'_1 \cap \dots \cap c'_m = mergeIC(v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2)}{v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2 \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_m} \text{MERGEIC}\cap$$

$$\frac{isValue\ v : T_1 \Rightarrow^l T_2 \quad v : c'_1 \cap \dots \cap c'_m = mergeCI(v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n)}{v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_m} \text{MERGECI}\cap$$

$$\frac{isValue\ v : cv_1 \cap \dots \cap cv_n \quad v : c''_1 \cap \dots \cap c''_j = mergeII(v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m)}{v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m \longrightarrow_{\cap CC} v : c''_1 \cap \dots \cap c''_j} \text{MERGEII}\cap$$

Evaluate intersection casts

$$\frac{\neg(\forall i \in 1..n . isCastValue\ c_i) \quad c_1 \longrightarrow_{\cap IC} cv_1 \ \dots \ c_n \longrightarrow_{\cap IC} cv_n}{v : c_1 \cap \dots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \dots \cap cv_n} \text{EVALUATE}\cap$$

Transition from cast values to values

$$\frac{v : blame\ T'_1\ T_1\ l_1\ {}^{m_1} \cap \dots \cap blame\ T'_n\ T_n\ l_n\ {}^{m_n}}{\longrightarrow_{\cap CC} blame_{(T_1 \cap \dots \cap T_n)}\ l_1} \text{PROPAGATEBLAME}\cap$$

$$\frac{v : \emptyset\ T_1\ {}^{m_1} \cap \dots \cap \emptyset\ T_n\ {}^{m_n} \longrightarrow_{\cap CC} v}{\text{REMOVEEMPTY}\cap}$$

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

$$\boxed{\vdash_{\cap IG} c : T} \text{ Typing}$$

$$\frac{\vdash_{\cap IG} c : T_1 \quad T_1 \sim T_2}{\vdash_{\cap IG} (c : T_1 \Rightarrow^l T_2^{\text{ }^n}) : T_1} \text{ T-SINGLEC} \qquad \frac{}{\vdash_{\cap IG} \emptyset T^{\text{ }^n} : T} \text{ T-EMPTYC}$$

$$\frac{}{\vdash_{\cap IG} \textit{blame } T_I T_F l^{\text{ }^n} : T_F} \text{ T-BLAMEC}$$

Figure 6: Intersection Casts Type System ($\vdash_{\cap IG}$)

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

Push blame to top level

$$\frac{}{\text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2} \longrightarrow_{\cap IC} \text{blame } T_I \ T_2 \ l_1 \ ^{n_1}} \text{PUSHBLAMEC}$$

Evaluate inside casts

$$\frac{\neg(\text{isCastValue } c) \quad c \longrightarrow_{\cap IC} c'}{c : T_1 \Rightarrow^l T_2 \ ^n \longrightarrow_{\cap IC} c' : T_1 \Rightarrow^l T_2 \ ^n} \text{EVALUATEC}$$

Detect success or failure of casts

$$\frac{\text{isCastValue1 } c \vee \text{isEmptyCast } c}{c : T \Rightarrow^l T \ ^n \longrightarrow_{\cap IC} c} \text{IDENTITYC}$$

$$\frac{\text{isCastValue1 } c \vee \text{isEmptyCast } c}{c : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2} \longrightarrow_{\cap IC} c} \text{SUCCEEDC}$$

$$\frac{\begin{array}{c} \text{isCastValue1 } c \vee \text{isEmptyCast } c \\ \neg(\text{same ground } G_1 \ G_2) \quad \text{initialType}(c) = T_I \end{array}}{c : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2} \longrightarrow_{\cap IC} \text{blame } T_I \ G_2 \ l_2 \ ^{n_1}} \text{FAILC}$$

Mediate the transition between the two disciplines

$$\frac{\begin{array}{c} \text{isCastValue1 } c \vee \text{isEmptyCast } c \\ G \text{ is ground type of } T \quad \neg(\text{ground } T) \end{array}}{c : T \Rightarrow^l \text{Dyn } ^n \longrightarrow_{\cap IC} c : T \Rightarrow^l G \ ^n : G \Rightarrow^l \text{Dyn } ^n} \text{GROUND C}$$

$$\frac{\begin{array}{c} \text{isCastValue1 } c \vee \text{isEmptyCast } c \\ G \text{ is ground type of } T \quad \neg(\text{ground } T) \end{array}}{c : \text{Dyn } \Rightarrow^l T \ ^n \longrightarrow_{\cap IC} c : \text{Dyn } \Rightarrow^l G \ ^n : G \Rightarrow^l T \ ^n} \text{EXPAND C}$$

Figure 7: Intersection Casts Semantics ($\longrightarrow_{\cap IC}$)

$[e]_e = e$ Erase identity casts

$$[x]_e = x$$

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

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

$$[n]_e = n$$

$$[true]_e = true$$

$$[false]_e = false$$

$$[e_1 + e_2]_e = [e_1]_e + [e_2]_e$$

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

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

$$\frac{[c_1]_e = \emptyset \ T_1^{n_1} \ \dots \ [c_n]_e = \emptyset \ T_n^{n_n}}{[e : c_1 \cap \dots \cap c_n]_e = [e]_e}$$

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

$[c]_c = c$ Erase identity casts

$$[c : T \Rightarrow^l T^n]_c = [c]_c$$

$$[c : T_1 \Rightarrow^l T_2^n]_c = [c]_c : T_1 \Rightarrow^l T_2^n$$

$$[blame \ T_I \ T_F \ l^n]_c = blame \ T_I \ T_F \ l^n$$

$$[\emptyset \ T^n]_c = \emptyset \ T^n$$

Figure 8: Identity Cast Erasure

2 Proofs

Theorem 1 (Conservative Extension). *Depends on Lemma 1. If e is fully static and T is a static type, then $\Gamma \vdash_{\text{NS}} e : T \iff \Gamma \vdash_{\text{NG}} e : T$.*

Proof. First we will prove that if $\vdash_{\text{NS}} e : T$ then $\vdash_{\text{NG}} e : T$. We proceed by induction on the length of the derivation tree of \vdash_{NS} .

Base case:

- $e = x$. If $\Gamma \vdash_{\text{NS}} x : T_i$, then $x : T_1 \cap \dots \cap T_n \in \Gamma$ such that $T_i \in \{T_1, \dots, T_n\}$. Therefore, by rule $\cap E$ of \vdash_{NG} , $\Gamma \vdash_{\text{NG}} e : T_i$.

Induction step:

- $e = \lambda x . T_1 \cap \dots \cap T_n . e'$. There are two possibilities:
 - Using the rule $\rightarrow I$. If $\Gamma \vdash_{\text{NS}} \lambda x . T_1 \cap \dots \cap T_n . e' : T_1 \cap \dots \cap T_n \rightarrow T$, then $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\text{NS}} e' : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\text{NG}} e' : T$. Therefore, by rule $\rightarrow I$, $\Gamma \vdash_{\text{NG}} \lambda x . T_1 \cap \dots \cap T_n . e' : T_1 \cap \dots \cap T_n \rightarrow T$.
 - Using the rule $\rightarrow I'$. If $\Gamma \vdash_{\text{NS}} \lambda x . T_1 \cap \dots \cap T_n . e' : T_i \rightarrow T$, then $\Gamma, x : T_i \vdash_{\text{NS}} e' : T$. By the induction hypothesis, $\Gamma, x : T_i \vdash_{\text{NG}} e' : T$. Therefore, by rule $\rightarrow I'$, $\Gamma \vdash_{\text{NG}} \lambda x . T_1 \cap \dots \cap T_n . e' : T_i \rightarrow T$.
- $e = e_1 e_2$. If $\Gamma \vdash_{\text{NS}} e_1 e_2 : T$ then $\Gamma \vdash_{\text{NS}} e_1 : T_1 \cap \dots \cap T_n \rightarrow T$ and $\Gamma \vdash_{\text{NS}} e_2 : T_1 \cap \dots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\text{NG}} e_1 : T_1 \cap \dots \cap T_n \rightarrow T$ and $\Gamma \vdash_{\text{NG}} e_2 : T_1 \cap \dots \cap T_n$. By the definition of \triangleright , $T_1 \cap \dots \cap T_n \rightarrow T \triangleright T_1 \cap \dots \cap T_n \rightarrow T$. By the definition of consistency ($T \sim T$), $T_1 \cap \dots \cap T_n \sim T_1 \cap \dots \cap T_n$. Therefore, by rule $\rightarrow E$, $\Gamma \vdash_{\text{NG}} e_1 e_2 : T$.
- $e = e$. If $\Gamma \vdash_{\text{NS}} e : T_1 \cap \dots \cap T_n$ then $\Gamma \vdash_{\text{NS}} e : T_1$ and ... and $\Gamma \vdash_{\text{NS}} e : T_n$. By the induction hypothesis, $\Gamma \vdash_{\text{NG}} e : T_1$ and ... and $\Gamma \vdash_{\text{NG}} e : T_n$. Therefore, by rule $\cap E$, $\Gamma \vdash_{\text{NG}} e : T_1 \cap \dots \cap T_n$.

Now we will prove that if $\vdash_{\text{NG}} e : T$ then $\vdash_{\text{NS}} e : T$. We proceed by induction on the length of the derivation tree of \vdash_{NG} .

Base case:

- $e = x$. If $\Gamma \vdash_{\text{NG}} x : T_i$, then $x : T_1 \cap \dots \cap T_n \in \Gamma$ such that $T_i \in \{T_1, \dots, T_n\}$. Therefore, by rule $\cap E$ of \vdash_{NS} , $\Gamma \vdash_{\text{NS}} e : T_i$.

Induction step:

- $e = \lambda x . T_1 \cap \dots \cap T_n . e'$. There are two possibilities:
 - Using the rule $\rightarrow I$. If $\Gamma \vdash_{\text{NG}} \lambda x . T_1 \cap \dots \cap T_n . e' : T_1 \cap \dots \cap T_n \rightarrow T$, then $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\text{NG}} e' : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\text{NS}} e' : T$. Therefore, by rule $\rightarrow I$, $\Gamma \vdash_{\text{NS}} \lambda x . T_1 \cap \dots \cap T_n . e' : T_1 \cap \dots \cap T_n \rightarrow T$.

– Using the rule $\rightarrow I'$. If $\Gamma \vdash_{\cap G} \lambda x . T_1 \cap \dots \cap T_n . e' : T_i \rightarrow T$, then $\Gamma, x : T_i \vdash_{\cap G} e' : T$. By the induction hypothesis, $\Gamma, x : T_i \vdash_{\cap S} e' : T$. Therefore, by rule $\rightarrow I'$, $\Gamma \vdash_{\cap S} \lambda x . T_1 \cap \dots \cap T_n . e' : T_i \rightarrow T$.

- $e = e_1 e_2$. If $\Gamma \vdash_{\cap G} e_1 e_2 : T$ then $\Gamma \vdash_{\cap G} e_1 : PM, PM \triangleright T_1 \cap \dots \cap T_n \rightarrow T$, $\Gamma \vdash_{\cap G} e_2 : T'_1 \cap \dots \cap T'_n$ and $T'_1 \cap \dots \cap T'_n \sim T_1 \cap \dots \cap T_n$. By the definition of \triangleright , $PM = T_1 \cap \dots \cap T_n \rightarrow T$, therefore $\Gamma \vdash_{\cap G} e_1 : T_1 \cap \dots \cap T_n \rightarrow T$. By Lemma 1, $T'_1 \cap \dots \cap T'_n = T_1 \cap \dots \cap T_n$, and therefore $\Gamma \vdash_{\cap G} e_2 : T_1 \cap \dots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap S} e_1 : T_1 \cap \dots \cap T_n \rightarrow T$ and $\Gamma \vdash_{\cap S} e_2 : T_1 \cap \dots \cap T_n$. Therefore, by rule $\rightarrow E$, $\Gamma \vdash_{\cap S} e_1 e_2 : T$.
- $e = e$. If $\Gamma \vdash_{\cap G} e : T_1 \cap \dots \cap T_n$ 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, by rule $\cap E$, $\Gamma \vdash_{\cap S} e : T_1 \cap \dots \cap T_n$.

□

Theorem 2 (Conservative Extension). *Depends on Lemmas 5 and 7. If e is fully static, T is a static type and $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$, then $e \rightarrow_{\cap S} v \iff e' \rightarrow_{\cap CC} v$.*

Proof. Since $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$ and e is fully static, then by Lemma 5 and by the definition of $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$, the expression e equals e' , except that e' contains identity casts. Therefore, $[e']_e = e$. Then, by Lemma 7, if $e \rightarrow v$ and $e' \rightarrow_{\cap CC} v'$, then $v = v'$. □

Theorem 3 (Monotonicity w.r.t. precision). *If $\Gamma \vdash_{\cap G} e : T$ and $e' \sqsubseteq e$ then $\exists T' . \Gamma \vdash_{\cap G} e' : T'$ and $T' \sqsubseteq T$.*

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

Base case:

- $e = x$. If $\Gamma \vdash_{\cap G} x : T_i$ and $x \sqsubseteq x$, then $\Gamma \vdash_{\cap G} x : T_i$ and $T_i \sqsubseteq T_i$.
- $e = n$. If $\Gamma \vdash_{\cap G} n : Int$ and $n \sqsubseteq n$, then $\Gamma \vdash_{\cap G} n : Int$ and $Int \sqsubseteq Int$.
- $e = true$. If $\Gamma \vdash_{\cap G} true : Bool$ and $true \sqsubseteq true$, then $\Gamma \vdash_{\cap G} true : Bool$ and $Bool \sqsubseteq Bool$.
- $e = false$. If $\Gamma \vdash_{\cap G} false : Bool$ and $false \sqsubseteq false$, then $\Gamma \vdash_{\cap G} false : Bool$ and $Bool \sqsubseteq Bool$.

Induction step:

- $e = \lambda x : T_1 \cap \dots \cap T_n . e_1$. If $\Gamma \vdash_{\cap G} \lambda x : T_1 . e_1 : T_1 \rightarrow T_2$ and $\lambda x : T'_1 . e'_1 \sqsubseteq \lambda x : T_1 . e_1$, then $\Gamma \vdash_{\cap G} e_1 : T_2, T'_1 \sqsubseteq T_1$ and $e'_1 \sqsubseteq e_1$. By the induction hypothesis, $\exists T'_2 . \Gamma \vdash_{\cap G} e'_1 : T'_2$ and $T'_2 \sqsubseteq T_2$. As $\Gamma \vdash_{\cap G} \lambda x : T'_1 . e'_1 : T'_1 \rightarrow T'_2$, and by the definition of \sqsubseteq , $T'_1 \rightarrow T'_2 \sqsubseteq T_1 \rightarrow T_2$, then it is proved.

- $e = e_1 \ e_2$. If $\Gamma \vdash_{\cap G} e_1 \ e_2 : T$ and $e'_1 \ e'_2 \sqsubseteq e_1 \ e_2$ then $\Gamma \vdash_{\cap G} e_1 : PM$, $PM \triangleright T_{11} \cap \dots \cap T_{1n} \rightarrow T$, $\Gamma \vdash_{\cap G} e_2 : T_{21} \cap \dots \cap T_{2n}$, and $T_{21} \cap \dots \cap T_{2n} \sim T_{11} \cap \dots \cap T_{1n}$, $e'_1 \sqsubseteq e_1$ and $e'_2 \sqsubseteq e_2$. By the induction hypothesis, $\exists PM' . \Gamma \vdash_{\cap G} e'_1 : PM'$ and $PM' \sqsubseteq PM$ and $PM' \triangleright T'_{11} \cap \dots \cap T'_{1n} \rightarrow T'$ and $\exists T'_{21}, \dots, T'_{2n} . \Gamma \vdash_{\cap G} e'_2 : T'_{21} \cap \dots \cap T'_{2n}$ and $T'_{21} \cap \dots \cap T'_{2n} \sqsubseteq T_{21} \cap \dots \cap T_{2n}$ and $T'_{21} \cap \dots \cap T'_{2n} \sim T'_{11} \cap \dots \cap T'_{1n}$. By the definition of \sqsubseteq and \triangleright , $T'_{11} \cap \dots \cap T'_{1n} \rightarrow T' \sqsubseteq T_{11} \cap \dots \cap T_{1n} \rightarrow T$, and therefore, $T' \sqsubseteq T$. As $\Gamma \vdash_{\cap G} e'_1 \ e'_2 : T'$, it is proved.
- $e = e$. If $\Gamma \vdash_{\cap G} e : T_1 \cap \dots \cap T_n$ and $e' \sqsubseteq e$, then $\Gamma \vdash_{\cap G} e : T_1$ and \dots and $\Gamma \vdash_{\cap G} e : T_n$. By the induction hypothesis, $\exists T'_1 . \Gamma \vdash_{\cap G} e' : T'_1$ and $T'_1 \sqsubseteq T_1$ and \dots and $\exists T'_n . \Gamma \vdash_{\cap G} e' : T'_n$ and $T'_n \sqsubseteq T_n$. Then, $\Gamma \vdash_{\cap G} e' : T'_1 \cap \dots \cap T'_n$ and by the definition of \sqsubseteq , $T'_1 \cap \dots \cap T'_n \sqsubseteq T_1 \cap \dots \cap T_n$, then it is proved.

□

Theorem 4 (Subject reduction of $\rightarrow_{\cap CC}$). *Depends on Lemmas 2 and 3. If $\Gamma \vdash_{\cap CC} e : T$ and $e \rightarrow_{\cap CC} e'$ then $\Gamma \vdash_{\cap CC} e' : T$.*

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

Base case:

- $e = (v_1 : cv_1 \cap \dots \cap cv_n) \ v_2$ and $isValue \ v_1 : cv_1 \cap \dots \cap cv_n$ and $\exists i \in 1..n . isArrowCompatible(cv_i)$ and $(cv'_1, \dots, cv'_m) = filter \ isArrowCompatible(cv_1, \dots, cv_n)$ and $((c_{11}, c_{12}, r_1), \dots, (c_{m1}, c_{m2}, r_m)) = map \ simulateArrow \ (cv'_1, \dots, cv'_m)$. If $\Gamma \vdash_{\cap CC} (v_1 : cv_1 \cap \dots \cap cv_n) \ v_2 : T_{12} \cap \dots \cap T_{n2}$, then $\Gamma \vdash_{\cap CC} v_1 : cv_1 \cap \dots \cap cv_n : T_1 \cap \dots \cap T_n$ with $\vdash_{\cap IC} cv_1 : T_1$ and \dots and $\vdash_{\cap IC} cv_n : T_n$, such that $\exists i \in 1..n . T_i = T_{i1} \rightarrow T_{i2}$ and $\Gamma \vdash_{\cap CC} v_1 : T'_1 \cap \dots \cap T'_l$ and $I_1 = initialType(cv_1)$ and \dots and $I_n = initialType(cv_n)$ such that either $T'_1 \cap \dots \cap T'_l =_{\cap} I_1 \cap \dots \cap I_n$ or *unique types in $[I_1, \dots, I_n] \subset unique \ types \ in \ [T'_1, \dots, T'_l]$* and $\Gamma \vdash_{\cap CC} v_2 : T_{11} \cap \dots \cap T_{n1}$. For the sake of simplicity lets elide cast labels and blame labels. As $\vdash_{\cap IC} cv'_1 : T_{11} \rightarrow T_{12}$ and \dots 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 \dots and $cv'_m = cv''_m : T'_{m1} \rightarrow T'_{m2} \Rightarrow T_{m1} \rightarrow T_{m2}$. By the definition of $simulateArrow$, $c_{11} : \emptyset \ T_{11} : T_{11} \Rightarrow T'_{11}$ and \dots and $c_{m1} = \emptyset \ T_{m1} : T_{m1} \Rightarrow T'_{m1}$ and $c_{12} : \emptyset \ T'_{12} : T'_{12} \Rightarrow T_{12}$ and \dots and $c_{m2} = \emptyset \ T'_{m2} : T'_{m2} \Rightarrow T_{m2}$ and $initialType(r_1) = I_1$ and \dots and $initialType(r_m) = I_m$ and $\vdash_{\cap IC} r_1 : T'_{11} \rightarrow T'_{12}$ and \dots and $\vdash_{\cap IC} r_m : T'_{m1} \rightarrow T'_{m2}$. Therefore $\Gamma \vdash_{\cap CC} v_1 : r_1 \cap \dots \cap r_m : T'_{11} \rightarrow T'_{12} \cap \dots \cap T'_{m1} \rightarrow T'_{m2}$ and $\Gamma \vdash_{\cap CC} v_2 : c_{11} \cap \dots \cap c_{m1} : T'_{11} \cap \dots \cap T'_{m1}$ and therefore $\Gamma \vdash_{\cap CC} (v_1 : r_1 \cap \dots \cap r_m) \ (v_2 : c_{11} \cap \dots \cap c_{m1}) : T'_{12} \cap \dots \cap T'_{m2}$. Therefore, $\Gamma \vdash_{\cap CC} (v_1 : r_1 \cap \dots \cap r_m) \ (v_2 : c_{11} \cap \dots \cap c_{m1}) : c_{12} \cap \dots \cap c_{m2} : T_{12} \cap \dots \cap T_{m2}$, such that *unique types in $[T_{12}, \dots, T_{m2}] \subset unique \ types \ in \ [T_{12}, \dots, T_{n2}]$* . By rule $Simulate\cap$, $(v_1 : cv_1 \cap \dots \cap cv_n) \ v_2 \rightarrow_{\cap CC} (v_1 : r_1 \cap \dots \cap r_m) \ (v_2 : c_{11} \cap \dots \cap c_{m1}) : c_{12} \cap \dots \cap c_{m2}$, therefore it is proved.
- $e = v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2$ and $isValue \ v : cv_1 \cap \dots \cap cv_n$ and $v : c'_1 \cap \dots \cap c'_m = mergeIC(v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2)$. If $\Gamma \vdash_{\cap CC} v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2 : T_2$ then $\Gamma \vdash_{\cap CC} v : cv_1 \cap \dots \cap cv_n :$

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

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

- $e = v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m$ and $\text{isValue } v : cv_1 \cap \dots \cap cv_n$ and $v : c'_1 \cap \dots \cap c'_j = \text{mergeII}(v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m)$. If $\Gamma \vdash_{\cap CC} v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m : F'_1 \cap \dots \cap F'_m$ then $\vdash_{\cap IC} c'_1 : F'_1$ and $\text{initialType}(c'_1) = I'_1$ and ... and $\vdash_{\cap IC} c'_m : F'_m$ and $\text{initialType}(c'_m) = I'_m$ and $\Gamma \vdash_{\cap CC} v : cv_1 \cap \dots \cap cv_n : F_1 \cap \dots \cap F_n$ and $\vdash_{\cap IC} cv_1 : F_1$ and $\text{initialType}(cv_1) = I_1$ and ... and $\vdash_{\cap IC} cv_n : F_n$ and $\text{initialType}(cv_n) = I_n$ and $\Gamma \vdash_{\cap CC} v : T_1 \cap \dots \cap T_l$ such that either $T_1 \cap \dots \cap T_l =_{\cap} I_1 \cap \dots \cap I_n$ or *unique types in* $[I_1, \dots, I_n] \subset \text{unique types in } [T_1, \dots, T_l]$. There are two possibilities:

- $F_1 \cap \dots \cap F_n =_{\cap} I'_1 \cap \dots \cap I'_m$. By the definition of mergeII, $\vdash_{\cap IC} c'_1 : F'_1$ and ... and $\vdash_{\cap IC} c'_j : F'_j$ such that $F'_1 \cap \dots \cap F'_j =_{\cap} F'_1 \cap \dots \cap F'_m$ and $\text{initialType}(c'_1) = I'_1$ and ... and $\text{initialType}(c'_j) = I'_j$ such that $I'_1 \cap \dots \cap I'_j =_{\cap} I_1 \cap \dots \cap I_n$. Therefore $\Gamma \vdash_{\cap CC} v : c'_1 \cap \dots \cap c'_j : F'_1 \cap \dots \cap F'_j$. By rule MergeII \cap , $v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_j$. Therefore it is proved.

- *unique types in* $[I'_1, \dots, I'_m] \subset \text{unique types in } [F_1, \dots, F_n]$. By the definition of mergeII, $\vdash_{\cap IC} c'_1 : F'_1$ and $\text{initialType}(c'_1) = I'_1$ and ... and $\vdash_{\cap IC} c'_j : F'_j$ and $\text{initialType}(c'_j) = I'_j$ such that *unique types in* $[I'_1, \dots, I'_j] \subset \text{unique types in } [T_1, \dots, T_l]$ and *unique types in* $[F'_1, \dots, F'_j] \subset \text{unique types in } [F_1, \dots, F_m]$. Therefore, $\Gamma \vdash_{\cap CC} v : c'_1 \cap \dots \cap c'_j : F'_1 \cap \dots \cap F'_j$. By rule MergeII \cap , $v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_j$. As $\Gamma \vdash_{\cap CC} v : cv_1 \cap \dots \cap cv_n : c'_1 \cap \dots \cap c'_m : F'_1 \cap \dots \cap F'_j$, it is proved.

- $e = v : c_1 \cap \dots \cap c_n$ and $\neg(\forall i \in 1..n . \text{isCastValue } c_i)$. If $\Gamma \vdash_{\cap CC} v : c_1 \cap \dots \cap c_n : T_1 \cap \dots \cap T_n$ then $\vdash_{\cap IC} c_1 : T_1$ and ... and $\vdash_{\cap IC} c_n : T_n$, $\Gamma \vdash_{\cap CC} v : I_1 \cap \dots \cap I_n$, with $I_1 = \text{initialType}(c_1)$ and ... and $I_n = \text{initialType}(c_n)$. By rule Evaluate \cap , $c_1 \longrightarrow_{\cap IC} cv_1$ and ... and $c_n \longrightarrow_{\cap IC} cv_n$. By Lemmas 2 and 3, $\vdash_{\cap IC} cv_1 : T_1$ and ... and $\vdash_{\cap IC} cv_n : T_n$ and $\text{initialType}(cv_1) = I_1$ and ... and $\text{initialType}(cv_n) = I_n$. Therefore $\Gamma \vdash_{\cap CC} v : cv_1 \cap \dots \cap cv_n : T_1 \cap \dots \cap T_n$.

- $e = v : \text{blame } T'_1 T_1 l_1^{m_1} \cap \dots \cap \text{blame } T'_n T_n l_n^{m_n}$. If $\Gamma \vdash_{\cap CC} v : \text{blame } T'_1 T_1 l_1^{m_1} \cap \dots \cap \text{blame } T'_n T_n l_n^{m_n} : T_1 \cap \dots \cap T_n$ and by rule $\text{PropagateBlame} \cap v : \text{blame } T'_1 T_1 l_1^{m_1} \cap \dots \cap \text{blame } T'_n T_n l_n^{m_n} \rightarrow_{\cap CC} \text{blame}_{(T_1 \cap \dots \cap T_n)} l_1$, and $\Gamma \vdash_{\cap CC} \text{blame}_{(T_1 \cap \dots \cap T_n)} l_1 : T_1 \cap \dots \cap T_n$, then it is proved.
- $e = v : \emptyset T_1^{m_1} \cap \dots \cap \emptyset T_n^{m_n}$. If $\Gamma \vdash_{\cap CC} v : \emptyset T_1^{m_1} \cap \dots \cap \emptyset T_n^{m_n} : T_1 \cap \dots \cap T_n$, then $\vdash_{\cap IC} \emptyset T_1^{m_1} : T_1$ and $\text{initialType}(\emptyset T_1^{m_1}) = T_1$ and ... and $\vdash_{\cap IC} \emptyset T_n^{m_n} : T_n$ and $\text{initialType}(\emptyset T_n^{m_n}) = T_n$ and $\Gamma \vdash_{\cap CC} v : T_1 \cap \dots \cap T_n$. By rule $\text{RemoveEmpty} \cap, v : \emptyset T_1^{m_1} \cap \dots \cap \emptyset T_n^{m_n} \rightarrow_{\cap CC} v$, therefore it is proved.

Induction step:

- $e =$

□

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:

- $T_1 = \text{Int}$.
 - If $T_1 = T_2$, then by the definition of \sim , $T_1 \sim T_2$.
 - If $T_1 \sim T_2$, then by the definition of \sim , $T_1 = T_2$.
- $T_1 = \text{Bool}$.
 - If $T_1 = T_2$, then by the definition of \sim , $T_1 \sim T_2$.
 - If $T_1 \sim T_2$, then by the definition of \sim , $T_1 = T_2$.
- $T_1 = \text{Dyn}$. This case is not considered due to the assumption that T_1 is a static type.

Induction step:

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

- If $T_1 \sim T_2$, then $\exists T_{21}, \dots, T_{2n} . T_2 = T_{21} \cap \dots \cap T_{2n}$ 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, by the definition of $=$, $T_1 = T_2$.

□

Lemma 2 (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:

- $c = \text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2}$. $\vdash_{\cap IC} \text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2} : T_2$ and by rule PushBlameC, $\text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2} \longrightarrow_{\cap IC} \text{blame } T_I \ T_2 \ l_1 \ ^{n_1}$. As $\vdash_{\cap IC} \text{blame } T_I \ T_2 \ l_1 \ ^{n_1} : T_2$, then it is proved.
- $c = c' : T \Rightarrow^l T \ ^n$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$. If $\vdash_{\cap IC} c' : T \Rightarrow^l T \ ^n : T$, then $\vdash_{\cap IC} c' : T$. By rule IdentityC, $c' : T \Rightarrow^l T \ ^n \longrightarrow_{\cap IC} c'$. Therefore it is proved.
- $c = c' : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2}$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$. If $\vdash_{\cap IC} c' : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2} : G$, then $\vdash_{\cap IC} c' : G$. By rule SucceedC, $c' : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2} \longrightarrow_{\cap IC} c'$. Therefore it is proved.
- $c = c' : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2}$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$ and $\neg(\text{same ground } G_1 \ G_2)$ and $\text{initialType}(c') = T_I$. If $\vdash_{\cap IC} c' : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2} : G_2$, and by rule FailC, $c' : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2} \longrightarrow_{\cap IC} \text{blame } T_I \ G_2 \ l_2 \ ^{n_1}$ and $\vdash_{\cap IC} \text{blame } T_I \ G_2 \ l_2 \ ^{n_1} : G_2$, it is proved.
- $c = c' : T \Rightarrow^l \text{Dyn } ^n$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$ and G is ground type of T and $\neg(\text{ground } T)$. If $\vdash_{\cap IC} c' : T \Rightarrow^l \text{Dyn } ^n : \text{Dyn}$ then $\vdash_{\cap IC} c' : T$. By rule GroundC, $c' : T \Rightarrow^l \text{Dyn } ^n \longrightarrow_{\cap IC} c' : T \Rightarrow^l G \ ^n : G \Rightarrow^l \text{Dyn } ^n$. As $\vdash_{\cap IC} c' : T \Rightarrow^l G \ ^n : G \Rightarrow^l \text{Dyn } ^n : \text{Dyn}$, it is proved.
- $c = c' : \text{Dyn } \Rightarrow^l T \ ^n$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$ and G is ground type of T and $\neg(\text{ground } T)$. If $\vdash_{\cap IC} c' : \text{Dyn } \Rightarrow^l T \ ^n : T$ then $\vdash_{\cap IC} c' : \text{Dyn}$. By rule ExpandC, $c' : \text{Dyn } \Rightarrow^l T \ ^n \longrightarrow_{\cap IC} c' : \text{Dyn } \Rightarrow^l G \ ^n : G \Rightarrow^l T \ ^n$. As $\vdash_{\cap IC} c' : \text{Dyn } \Rightarrow^l G \ ^n : G \Rightarrow^l T \ ^n : T$, it is proved.

Induction step:

- $c = c' : T_1 \Rightarrow^l T_2 \ ^n$ and $\neg(\text{isCastValue } c)$. If $\vdash_{\cap IC} c' : T_1 \Rightarrow^l T_2 \ ^n : T_2$ 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, $c' : T_1 \Rightarrow^l T_2 \ ^n \longrightarrow_{\cap IC} c'' : T_1 \Rightarrow^l T_2 \ ^n$. As $\vdash_{\cap IC} c'' : T_1 \Rightarrow^l T_2 \ ^n : T_2$ it is proved.

□

Lemma 3 (Initial type preservation of $\longrightarrow_{\cap IC}$). *If $\text{initialType}(c) = T$ for some T and $c \longrightarrow_{\cap IC} c'$ then $\text{initialType}(c') = T$.*

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

Base cases:

- $c = \text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2}$. By the definition of initialType , $\text{initialType}(\text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2}) = T_I$. By rule PushBlameC, $\text{blame } T_I \ T_F \ l_1 \ ^{n_1} : T_1 \Rightarrow^{l_2} T_2 \ ^{n_2} \longrightarrow_{\cap IC} \text{blame } T_I \ T_2 \ l_1 \ ^{n_1}$. Since $\text{initialType}(\text{blame } T_I \ T_2 \ l_1 \ ^{n_1}) = T_I$, it is proved.
- $c = c' : T \Rightarrow^l T \ ^n$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$. By the definitions of initialType , $\text{initialType}(c' : T \Rightarrow^l T \ ^n) = \text{initialType}(c')$. By rule IdentityC, $c' : T \Rightarrow^l T \ ^n \longrightarrow_{\cap IC} c'$. Therefore it is proved.
- $c = c' : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2}$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$. By the definition of initialType , $\text{initialType}(c' : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2}) = \text{initialType}(c')$. By rule SucceedC, $c' : G \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G \ ^{n_2} \longrightarrow_{\cap IC} c'$. Therefore it is proved.
- $c = c' : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2}$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$ and $\neg(\text{same ground } G_1 \ G_2)$ and $\text{initialType}(c') = T_I$. By the definition of initialType , $\text{initialType}(c' : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2}) = T_I$. By rule FailC, $c' : G_1 \Rightarrow^{l_1} \text{Dyn } ^{n_1} : \text{Dyn } \Rightarrow^{l_2} G_2 \ ^{n_2} \longrightarrow_{\cap IC} \text{blame } T_I \ G_2 \ l_2 \ ^{n_1}$. Since $\text{initialType}(\text{blame } T_I \ G_2 \ l_2 \ ^{n_1}) = T_I$, it is proved.
- $c = c' : T \Rightarrow^l \text{Dyn } ^n$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$ and G is ground type of T and $\neg(\text{ground } T)$. By the definition of initialType , $\text{initialType}(c' : T \Rightarrow^l \text{Dyn } ^n) = \text{initialType}(c')$. By rule GroundC, $c' : T \Rightarrow^l \text{Dyn } ^n \longrightarrow_{\cap IC} c' : T \Rightarrow^l G \ ^n : G \Rightarrow^l \text{Dyn } ^n$. Since $\text{initialType}(c' : T \Rightarrow^l G \ ^n : G \Rightarrow^l \text{Dyn } ^n) = \text{initialType}(c')$, it is proved.
- $c = c' : \text{Dyn } \Rightarrow^l T \ ^n$ and $\text{isCastValue1 } c' \vee \text{isEmptyCast } c'$ and G is ground type of T and $\neg(\text{ground } T)$. By the definition of initialType , $\text{initialType}(c' : \text{Dyn } \Rightarrow^l T \ ^n) = \text{initialType}(c')$. By rule ExpandC, $c' : \text{Dyn } \Rightarrow^l T \ ^n \longrightarrow_{\cap IC} c' : \text{Dyn } \Rightarrow^l G \ ^n : G \Rightarrow^l T \ ^n$. Since $\text{initialType}(c' : \text{Dyn } \Rightarrow^l G \ ^n : G \Rightarrow^l T \ ^n) = \text{initialType}(c')$, it is proved.

Induction step:

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

□

Lemma 4 (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:

- $e = x$. If $\Gamma \vdash_{\cap G} x : T_i$, then there is a binding $x : T' \in \Gamma$, such that $T_i \subseteq T'$. Therefore, there must have been at some point in the typing derivation, the application of the rules $(\rightarrow I)$ or $(\rightarrow I')$. If e is annotated with only static types, then both rules introduce the binding $x : T'$ in Γ , such that T' is a static type. Therefore, T_i is also a static type.

Induction step:

- $e = \lambda x : T_1 \cap \dots \cap T_n . e'$. There are two possibilities:
 - Using the rule $\rightarrow I$. If e is annotated with only static types, then $T_1 \cap \dots \cap T_n$ is a static type. By rule $(\rightarrow I)$, $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\cap G} e : T$. By the induction hypothesis, T is a static type. Therefore $T_1 \cap \dots \cap T_n \rightarrow T$ is a static type.
 - Using the rule $\rightarrow I'$. If e is annotated with only static types, then $T_1 \cap \dots \cap T_n$ is a static type. By rule $(\rightarrow I')$, $\Gamma, x : T_i \vdash_{\cap G} e : T$. Since $T_1 \cap \dots \cap T_n$ is a static type, then so is T_i . By the induction hypothesis, T is a static type, therefore so is $T_i \rightarrow T$.
- $e = e_1 e_2$. If e 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 \triangleright , $T_1 \cap \dots \cap T_n \rightarrow T$ is also a static type. Therefore, T is a static type.
- $e = e$. If e annotated with only static types, then by the induction hypothesis, $T_1 \dots T_n$ are static types. Therefore $T_1 \cap \dots \cap T_n$ is a static type.

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

Base cases:

- $e = x$. If $\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T_i$, then there is a binding $x : T' \in \Gamma$, such that $T_i \subseteq T'$. Therefore, there must have been at some point in the typing derivation, the application of the rule for the term $\lambda x : T_1 \cap \dots \cap T_n . e'$. If e is annotated with only static types, then the rule introduces the binding $x : T'$ in Γ , such that T' is a static type. Therefore, T_i is also a static type.

Induction step:

- $e = \lambda x : T_1 \cap \dots \cap T_n . e'$. If e is annotated with only static types, then $T_1 \cap \dots \cap T_n$ is a static type. By the induction hypothesis, T is a static type. Therefore $T_1 \cap \dots \cap T_n \rightarrow T$ is a static type.
- $e = e_1 e_2$. If e 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 \triangleright , $T_1 \cap \dots \cap T_n \rightarrow T$ is also a static type. Therefore, T is a static type.

□

Lemma 5 (Static program compilation only adds identity casts). *Depends on Lemmas 1 and 4. 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 $e : c_1 \cap \dots \cap c_n$ such that $c_1 = \emptyset T_1^0 : T_1 \Rightarrow T_1^0$ and ... and $c_n = \emptyset T_n^0 : T_n \Rightarrow T_n^0$ for some T_1, \dots, T_n .

Proof. We proceed by structural induction on e .

Base cases:

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

Induction step:

- $e = \lambda x : T_1 \cap \dots \cap T_n . e'$. By rule, $\Gamma \vdash_{\cap CC} e' \rightsquigarrow e'' : T$. By the induction hypothesis, e'' either doesn't contain casts or contains only identity casts. By rule, $\Gamma \vdash_{\cap CC} (\lambda x : T_1 \cap \dots \cap T_n . e') \rightsquigarrow (\lambda x : T_1 \cap \dots \cap T_n . e'') : T_1 \cap \dots \cap T_n \rightarrow T$. As the rule doesn't introduce new casts, then it is proved.
- $e = e_1 e_2$. By rule, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e'_1 : PM$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e'_2 : T'_1 \cap \dots \cap T'_n$. By the induction hypothesis, both e'_1 as well as e'_2 either only have identity casts or no casts at all. By Lemma 4, PM and $T'_1 \cap \dots \cap T'_n$ are static types. Therefore, by the definition of \triangleright , $PM = T'_1 \cap \dots \cap T'_n \rightarrow T$ and by Lemma 1, $T'_1 \cap \dots \cap T'_n = T_1 \cap \dots \cap T_n$. Therefore by the definition of *getInstances* and *addCasts*, only identity casts are introduced.

□

Lemma 6 (Elimination of identity casts in c). *For any cast c , such that $\vdash_{\cap IC} c : T_F$, $initialType(c) = T_I$ and $c \longrightarrow_{\cap IC} cv$:*

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

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

Base cases:

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

Induction step:

- $c = c' : T_1 \Rightarrow^l T_2^n$. There are two cases:

- $T_1 \neq T_2$. As $\vdash_{\cap IC} c' : T_1 \Rightarrow^l T_2^n : T_2$ and $initialType(c' : T_1 \Rightarrow^l T_2^n) = initialType(c')$, 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 $[c]_c = [c']_c : T_1 \Rightarrow^l T_2^n$, $\vdash_{\cap IC} [c]_c : T_2$ and $initialType([c]_c) = initialType([c']_c) = initialType(c') = initialType(c)$.
- $T_1 = T_2$. As $\vdash_{\cap IC} c' : T_1 \Rightarrow^l T_1^n : T_1$ and $initialType(c' : T_1 \Rightarrow^l T_1^n) = initialType(c')$ 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 structural induction on c .

Base cases:

- $c = blame\ T_I\ T_F\ l_1^{n_1} : T_1 \Rightarrow^{l_2} T_2^{n_2}$. There are two cases:
 - $T_1 \neq T_2$. As $[c]_c = blame\ T_I\ T_F\ l_1^{n_1} : T_1 \Rightarrow^{l_2} T_2^{n_2}$ and by rule Push-BlameC, $blame\ T_I\ T_F\ l_1^{n_1} : T_1 \Rightarrow^{l_2} T_2^{n_2} \rightarrow_{\cap IC} blame\ T_I\ T_2\ l_1^{n_1}$ 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, $c = blame\ T_I\ T_1\ l_1^{n_1} : T_1 \Rightarrow^{l_2} T_1^{n_2}$. By rule Push-BlameC, $blame\ T_I\ T_1\ l_1^{n_1} : T_1 \Rightarrow^{l_2} T_1^{n_2} \rightarrow_{\cap IC} blame\ T_I\ T_1\ l_1^{n_1}$. Since $[c]_c = blame\ T_I\ T_1\ l_1^{n_1}$, and it is already a value, it is proved.
- $c = c' : T \Rightarrow^l T^n$ and $isCastValue1\ c' \vee isEmptyCast\ c'$. By rule IdentityC, $c' : T \Rightarrow^l T^n \rightarrow_{\cap IC} c'$. 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.
- $c = c' : G \Rightarrow^{l_1} Dyn^{n_1} : Dyn \Rightarrow^{l_2} G^{n_2}$ and $isCastValue1\ c' \vee isEmptyCast\ c'$. By rule SucceedC, $c' : G \Rightarrow^{l_1} Dyn^{n_1} : Dyn \Rightarrow^{l_2} G^{n_2} \rightarrow_{\cap IC} c'$. As c' is already a value, then it doesn't contain identity casts, so $[c]_c = c' : G \Rightarrow^{l_1} Dyn^{n_1} : Dyn \Rightarrow^{l_2} G^{n_2}$. Therefore, $[c]_c \rightarrow_{\cap IC} c'$.
- $c = c' : G_1 \Rightarrow^{l_1} Dyn^{n_1} : Dyn \Rightarrow^{l_2} G_2^{n_2}$ and $isCastValue1\ c' \vee isEmptyCast\ c'$ and $\neg(same\ ground\ G_1\ G_2)$ and $initialType(c') = T_I$. By rule FailC, $c' : G_1 \Rightarrow^{l_1} Dyn^{n_1} : Dyn \Rightarrow^{l_2} G_2^{n_2} \rightarrow_{\cap IC} blame\ T_I\ G_2\ l_2^{n_1}$. As c' is already a value, then it doesn't contain identity casts, so $[c]_c = c' : G_1 \Rightarrow^{l_1} Dyn^{n_1} : Dyn \Rightarrow^{l_2} G_2^{n_2}$. Therefore, $[c]_c \rightarrow_{\cap IC} blame\ T_I\ G_2\ l_2^{n_1}$.
- $c = c' : T \Rightarrow^l Dyn^n$ and $isCastValue1\ c' \vee isEmptyCast\ c'$ and G is ground type of T and $\neg(ground\ T)$. By rule GroundC, $c' : T \Rightarrow^l Dyn^n \rightarrow_{\cap IC} c' : T \Rightarrow^l G : G \Rightarrow^l Dyn^n$. As c' is a value, it doesn't contain identity casts, therefore $[c]_c = c' : T \Rightarrow^l Dyn^n$. Therefore $[c]_c \rightarrow_{\cap IC} c' : T \Rightarrow^l G : G \Rightarrow^l Dyn^n$.
- $c = c' : Dyn \Rightarrow^l T^n$ and $isCastValue1\ c' \vee isEmptyCast\ c'$ and G is ground type of T and $\neg(ground\ T)$. By rule ExpandC, $c' : Dyn \Rightarrow^l T^n \rightarrow_{\cap IC} c' : Dyn \Rightarrow^l G : G \Rightarrow^l T^n$. As c' is a value, it doesn't contain identity casts, therefore $[c]_c = c' : Dyn \Rightarrow^l T^n$. Therefore $[c]_c \rightarrow_{\cap IC} c' : Dyn \Rightarrow^l G : G \Rightarrow^l T^n$.

Induction step:

- $c = c' : T_1 \Rightarrow^l T_2^{\text{ } n}$ and $\neg(\text{isCastValue} c')$. There are two cases:
 - $T_1 \neq T_2$. By rule EvaluateC, $c' \rightarrow_{\cap IC} c''$. By the induction hypothesis, $[c']_e \rightarrow_{\cap IC} c''$. As $[c]_e$ equals $[c']_e : T_1 \Rightarrow^l T_2^{\text{ } n}$, then by rule EvaluateC, $[c]_e \rightarrow_{\cap IC} c'' : T_1 \Rightarrow T_2^{\text{ } n}$.
 - $T_1 = T_2$. By the induction hypothesis, as $c' \rightarrow_{\cap IC} cv'$, then $[c']_e \rightarrow_{\cap IC} cv'$. By rule EvaluateC, $c' : T_1 \Rightarrow^l T_1^{\text{ } n} \rightarrow_{\cap IC} cv' : T_1 \Rightarrow^l T_1^{\text{ } n}$. However, as $cv' : T_1 \Rightarrow^l T_1^{\text{ } n}$ is not a value, the rule IdentityC must be applied, therefore $c' : T_1 \Rightarrow^l T_1^{\text{ } n} \rightarrow_{\cap IC} cv'$. As $[c]_e \rightarrow_{\cap IC} cv'$, then it is proved.

□

Lemma 7 (Elimination of identity casts in e). *Depends on Lemma 6. For any expression e , such that $\Gamma \vdash_{\cap CC} e : T$, and $e \rightarrow_{\cap CC} v$:*

1. $\Gamma \vdash_{\cap CC} [e]_e : T$.
2. $[e]_e \rightarrow_{\cap CC} v$.

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

Base cases:

- $e = x$. As x doesn't contain casts, then $[e]_e = x$. Therefore it is proved.
- $e = n$. As n doesn't contain casts, then $[e]_e = n$. Therefore it is proved.
- $e = \text{true}$. As true doesn't contain casts, then $[e]_e = \text{true}$. Therefore it is proved.
- $e = \text{false}$. As false doesn't contain casts, then $[e]_e = \text{false}$. Therefore it is proved.
- $e = \text{blame}_T l$. As $\text{blame}Tl$ doesn't contain casts, then $[e]_e = \text{blame}Tl$. Therefore it is proved.

Induction step:

- $e = \lambda x : T_1 \cap \dots \cap T_n . e'$. There are two possibilities:
 - Using the rule $\rightarrow I$. If $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \dots \cap T_n . e' : T_1 \cap \dots \cap T_n \rightarrow T$, then $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\cap CC} e' : T$. By the induction hypothesis, $\Gamma, x : T_1 \cap \dots \cap T_n \vdash_{\cap CC} [e']_e : T$. As $[e]_e = \lambda x : T_1 \cap \dots \cap T_n . [e']_e$, then $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \dots \cap T_n \rightarrow T$.
 - Using the rule $\rightarrow I'$. If $\Gamma \vdash_{\cap CC} \lambda x : T_1 \cap \dots \cap T_n . e' : T_i \rightarrow T$, 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 $[e]_e = \lambda x : T_1 \cap \dots \cap T_n . [e']_e$, then $\Gamma \vdash_{\cap CC} [e]_e : T_i \rightarrow T$.
- $e = e_1 e_2$. If $\Gamma \vdash_{\cap CC} e_1 e_2 : T$, then $\Gamma \vdash_{\cap CC} e_1 : PM, PM \triangleright T_1 \cap \dots \cap T_n \rightarrow T$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \dots \cap T'_n$ and $T'_1 \cap \dots \cap T'_n \sim T_1 \cap \dots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e_1]_e : PM$ and $\Gamma \vdash_{\cap CC} [e_2]_e : T'_1 \cap \dots \cap T'_n$. As $[e]_e = [e_1]_e [e_2]_e$, therefore $\Gamma \vdash_{\cap CC} [e]_e : T$.

- $e = e$. If $\Gamma \vdash_{\cap CC} e : T_1 \cap \dots \cap T_n$, 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 $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \dots \cap T_n$.
- $e = e' : T_1 \Rightarrow^l T_2$. 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 $[e]_e = [e']_e : T_1 \Rightarrow^l T_1$, then $\Gamma \vdash_{\cap CC} [e]_e : T_1$.
- $e = e' : c_1 \cap \dots \cap c_n$. If $\Gamma \vdash_{\cap CC} e' : c_1 \cap \dots \cap c_n : T_1 \cap \dots \cap T_n$, then $\Gamma \vdash_{\cap CC} e' : T$, $\vdash_{\cap IC} c_1 : T_1$ and ... and $\vdash_{\cap IC} c_n : T_n$ and $\text{initialType}(c_1) \cap \dots \cap \text{initialType}(c_n) =_{\cap} T$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} [e']_e : T$. We now have 2 possibilities:
 - $\neg(\forall i \in 1..n . \text{isEmptyCast } [c_i]_e)$: For all casts c_i , with $i \in 1..n$, that don't contain identity casts, then $[c_i]_e = c_i$, therefore $\vdash_{\cap IC} [c_i]_e : T_i$ and $\text{initialType}([c_i]_e) = \text{initialType}(c_i)$. For the remaining casts, by Lemma 6, $\vdash_{\cap IC} [c_i]_e : T_i$ and $\text{initialType}([c_i]_e) = \text{initialType}(c_i)$. Therefore, with $[e]_e = [e']_e : [c_1]_e \cap \dots \cap [c_n]_e$, $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \dots \cap T_n$.
 - $\forall i \in 1..n . \text{isEmptyCast } [c_i]_e$: As all casts are empty casts, then for all casts $[c_i]_e$, by Lemma 6 and by rule T-EmptyC, $\vdash_{\cap IC} [c_i]_e : T_i$ and $\text{initialType}([c_i]_e) = T_i$. Therefore $[e]_e = [e']_e$. We now have two possibilities:
 - * If T is not an intersection type, then $T_1 = \dots = T_n = T$ and by idempotence of \cap , we have that $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \dots \cap T_n$.
 - * If T is an intersection type, then $T = T_1 \cap \dots \cap T_n$. Therefore $\Gamma \vdash_{\cap CC} [e]_e : T_1 \cap \dots \cap T_n$.

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

Base cases:

- $e = (v_1 : cv_1 \cap \dots \cap cv_n) v_2$ and $\text{isValue}((v_1 : cv_1 \cap \dots \cap cv_n) v_2)$ and $\exists i \in 1..n . \text{isArrowCompatible } cv_i$. As $v_1 : cv_1 \cap \dots \cap cv_n$ and v_2 are values, then e doesn't contain identity casts. Therefore $[e]_e = e$.
- $e = v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2$ and $\text{isValue } v : cv_1 \cap \dots \cap cv_n$ and $v : c'_1 \cap \dots \cap c'_m = \text{mergeIC}(v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2)$. There are 2 possibilities:
 - If $T_1 \neq T_2$ and as $v : cv_1 \cap \dots \cap cv_n$ doesn't contain identity casts, then $[e]_e = e$, therefore it is proved.
 - If $T_1 = T_2$ and as $v : cv_1 \cap \dots \cap cv_n$ doesn't contain identity casts, then $[e]_e = v : cv_1 \cap \dots \cap cv_n$. By rule MergeIC \cap , $v : cv_1 \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2 \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_m$. By rule Evaluate \cap , $v : cv_1 : T_1 \Rightarrow^l T_2^{m_1} \cap \dots \cap cv_n : T_1 \Rightarrow^l T_2^{m_n} \longrightarrow_{\cap CC} v : cv_1 \cap \dots \cap cv_n$, with $cv_1 : T_1 \Rightarrow^l T_2^{m_1} \longrightarrow_{\cap IC} cv_1$ and ... and $cv_n : T_1 \Rightarrow^l T_2^{m_n} \longrightarrow_{\cap IC} cv_n$ by rule IdentityC. As $[e]_e$ is already a value, it is proved.

- $e = v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n$ and $isValue\ v : T_1 \Rightarrow^l T_2$ and $v : c'_1 \cap \dots \cap c'_n = mergeCI(v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n)$. There are 2 possibilities:
 - $v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n$ doesn't contain identity casts, then $[e]_e = e$, therefore it is proved.
 - $v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n$ contain identity casts. By rule MergeCI \cap , $v : T_1 \Rightarrow^l T_2 : c_1 \cap \dots \cap c_n \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_n$. By rule Evaluate \cap , $v : T_1 \Rightarrow^l T_2 : c'_1 \cap \dots \cap c'_n \longrightarrow_{\cap CC} v : cv'_1 \cap \dots \cap cv'_n$, with $c'_1 \longrightarrow_{\cap IC} cv'_1$ and ... and $c'_n \longrightarrow_{\cap IC} cv'_n$. For all casts c_i that don't contain identity casts, then $[c_i]_e = c_i$, therefore for those casts, the property is proved. For all casts c_i that contain identity casts, $mergeCI$ will generate casts c'_i that will evaluate to cv'_i . By Lemma 6, casts $[c_i]_e$ will generate casts c''_i that will evaluate to cv'_i , therefore it is proved.
- $e = v : cv_1 \cap \dots \cap cv_n : c_1 \cap \dots \cap c_m$ and $isValue\ v : cv_1 \cap \dots \cap cv_n$ and $v : c'_1 \cap \dots \cap c'_j = mergeII(v : cv_1 \cap \dots \cap cv_n : c_1 \cap \dots \cap c_m)$. There are 2 possibilities:
 - $v : cv_1 \cap \dots \cap cv_n : c_1 \cap \dots \cap c_m$ doesn't contain identity casts, then $[e]_e = e$, therefore it is proved.
 - $v : cv_1 \cap \dots \cap cv_n : c_1 \cap \dots \cap c_m$ contain identity casts. By rule MergeII \cap , $v : cv_1 \cap \dots \cap cv_n : c_1 \cap \dots \cap c_m \longrightarrow_{\cap CC} v : c'_1 \cap \dots \cap c'_j$. By rule Evaluate \cap , $v : c'_1 \cap \dots \cap c'_j \longrightarrow_{\cap CC} v' : cv'_1 \cap \dots \cap cv'_j$, with $c'_1 \longrightarrow_{\cap IC} cv'_1$ and ... and $c'_j \longrightarrow_{\cap IC} cv'_j$. For all casts cv_i and c_i that will be joined into c'_i by function mergeII, and that don't contain identity casts, then $[c'_i]_e = c'_i$, therefore for those casts, the property is proved. For all casts cv_i and c_i that will be joined into c'_i by function mergeII, and that contain identity casts, $c'_i \longrightarrow_{\cap IC} cv'_i$ and by Lemma 6, $[c'_i]_e \longrightarrow_{\cap IC} cv'_i$, therefore it is proved.
- $e = v : c_1 \cap \dots \cap c_n$ and $\neg(\forall i \in 1..n . isCastValue\ c_i)$. By rule Evaluate \cap , $v : c_1 \cap \dots \cap c_n \longrightarrow_{\cap CC} v : cv_1 \cap \dots \cap cv_n$, with $c_1 \longrightarrow_{\cap IC} cv_1$ and ... and $c_n \longrightarrow_{\cap IC} cv_n$. With $[e]_e = v : [c_1]_e \cap \dots \cap [c_n]_e$, by Lemma 6, $[c_1]_e \longrightarrow_{\cap IC} cv_1$ and ... and $[c_n]_e \longrightarrow_{\cap IC} cv_n$. Therefore, by rule Evaluate \cap , $v : [c_1]_e \cap \dots \cap [c_n]_e \longrightarrow_{\cap CC} v : cv_1 \cap \dots \cap cv_n$.
- $e = v : blame\ I_1\ F_1\ l_1^{m_1} \cap \dots \cap blame\ I_n\ F_n\ l_n^{m_n}$. As $[e]_e = e$, then it is proved.
- $e = v : \emptyset\ T_1^{m_1} \cap \dots \cap \emptyset\ T_n^{m_n}$. As $[e]_e = e$, then it is proved.

Induction step:

- $e =$

□