Gradual Intersection Types

Pedro Ângelo

December 7, 2017

1 Language Definition

Syntax

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

Figure 1: Gradual Intersection System

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

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

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

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

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

$\boxed{T \sim T}$ Consistency

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

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

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

$$\frac{}{(T_1 \to T_2) \rhd T_1 \to T_2} \qquad \frac{}{Dyn \rhd Dyn \to Dyn}$$

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

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

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

$$\frac{\Gamma \vdash_{\cap CC} e : T_1 \qquad T_1 \sim T_2}{\Gamma \vdash_{\cap CC} (e : T_1 \Rightarrow^l T_2) : T_2}\ \text{T-Cast} \qquad \frac{}{\Gamma \vdash_{\cap CC} blame_T\ l : T}\ \text{T-Blame}$$

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

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

$$initialType(c : T_1 \Rightarrow^l T_2\ ^n) = initialType(c)$$

$$initialType(\varnothing\ T\ ^n) = T$$

$$initialType(blame\ T_I\ T_F\ l\ ^n) = T_I$$

$$initialType(\perp\ T_I\ T_F\ ^n) = T_I$$

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

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

$$\frac{x : T_1 \cap \ldots \cap T_n \in \Gamma}{\Gamma \vdash_{\cap CC} x \rightsquigarrow x : T_i}$$

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

$$\frac{\begin{array}{c}\Gamma \vdash_{CC} e_1 \rightsquigarrow e_1' : PM \qquad PM \rhd T_1 \cap \ldots \cap T_n \rightarrow T \\ \Gamma \vdash_{CC} e_2 \rightsquigarrow e_2' : T_1' \cap \ldots \cap T_n' \qquad T_1' \cap \ldots \cap T_n' \sim T_1 \cap \ldots \cap T_n \\ e_1'' = addCasts(getInstances(PM),\ getInstances(T_1 \cap \ldots \cap T_n \rightarrow T),\ e_1') \\ e_2'' = addCasts(getInstances(T_1' \cap \ldots \cap T_n'),\ getInstances(T_1 \cap \ldots \cap T_n),\ e_2')\end{array}}{\Gamma \vdash_{CC} e_1\ e_2 \rightsquigarrow e_1''\ e_2'' : T}$$

Figure 4: Compilation to the Cast Calculus

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

Simulate casts on data types

$$\frac{\begin{array}{c} isValue\ v_1 : cv_1 \cap \ldots \cap cv_n \qquad \exists i \in 1..n\ .\ isArrowCompatible\ cv_i \\ (cv'_1, \ldots, cv'_m) = filter\ isArrowCompatible\ (cv_1, \ldots, cv_n) \\ ((c_{11}, c_{12}, r_1), \ldots, (c_{m1}, c_{m2}, r_m)) = map\ simulateArrow\ (cv'_1, \ldots, cv'_m) \end{array}}{\begin{array}{c} (v_1 : cv_1 \cap \ldots \cap cv_n)\ v_2 \longrightarrow_{\cap CC} \\ (v_1 : r_1 \cap \ldots \cap r_m)\ (v_2 : c^1_{11} \cap \ldots \cap c^m_{m1}) : c^1_{12} \cap \ldots \cap c^m_{m2} \end{array}} \text{ S{\scriptsize IMULATE}}\cap$$

Merge casts

$$\frac{isValue\ v : cv_1 \cap \ldots \cap cv_n \qquad label(c_1) = m_1\ \ldots\ label(c_n) = m_n}{\begin{array}{c} v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 \longrightarrow_{\cap CC} \\ v : (cv_1 : T_1 \Rightarrow^l T_2\ {}^{m_1}) \cap \ldots \cap (cv_n : T_1 \Rightarrow^l T_2\ {}^{m_n}) \end{array}} \text{ M{\scriptsize ERGE}1}\cap$$

$$\frac{\begin{array}{c} isIntersectionCast\ v \vee isCast\ v \\ v' : c'_1 \cap \ldots \cap c'_m = mergeCasts(v : c_1 \cap \ldots \cap c_n) \end{array}}{v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v' : c'_1 \cap \ldots \cap c'_m} \text{ M{\scriptsize ERGE}2}\cap$$

Evaluate intersection casts

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

Transition from cast values to values

$$\frac{}{\begin{array}{c} v : blame\ T'_1\ T_1\ l_1\ {}^{m_1} \cap \ldots \cap blame\ T'_n\ T_n\ l_n\ {}^{m_n} \\ \longrightarrow_{\cap CC} blame_{(T_1 \cap \ldots \cap T_n)}\ l_1 \end{array}} \text{ P{\scriptsize ROPAGATE}B{\scriptsize LAME}}\cap$$

$$\frac{}{v : \varnothing\ T_1\ {}^{m_1} \cap \ldots \cap \varnothing\ T_n\ {}^{m_n} \longrightarrow_{\cap CC} v} \text{ R{\scriptsize EMOVE}E{\scriptsize MPTY}}\cap$$

$$\frac{\begin{array}{c} \neg(\forall i \in 1..n\ .\ isStuckCast\ cv_i) \\ \exists i \in 1..n\ .\ isStuckCast\ cv_i \\ (cv'_1, \ldots, cv'_m) = filter\ (\neg isStuckCast)\ (cv_1, \ldots, cv_n) \end{array}}{v : cv_1 \cap \ldots \cap cv_n \longrightarrow_{\cap CC} v : cv'_1 \cap \ldots \cap cv'_m} \text{ R{\scriptsize EMOVE}S{\scriptsize TUCK}}\cap$$

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

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

$$\frac{\vdash_{\cap IG} c : T_1 \qquad T_1 \sim T_2}{\vdash_{\cap IG} (c : T_1 \Rightarrow^l T_2\ ^n) : T_1}\ \textsc{T-SingleC} \qquad \frac{}{\vdash_{\cap IG} \varnothing\ T\ ^n : T}\ \textsc{T-EmptyC}$$

$$\frac{}{\vdash_{\cap IG} \mathit{blame}\ T_I\ T_F\ l\ ^n : T_F}\ \textsc{T-BlameC} \qquad \frac{}{\vdash_{\cap IG} \bot\ T_I\ T_F\ ^n : T_F}\ \textsc{T-StuckC}$$

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

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

Push blame and stuck to top level

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

$$\frac{}{\perp\ T_I\ T_F\ {}^{n_1} : T_1 \Rightarrow^{l} T_2\ {}^{n_2} \longrightarrow_{\cap IC} \perp\ T_I\ T_2\ {}^{n_1}}\ \textsc{PushStuckC}$$

Evaluate inside casts

$$\frac{\neg(\mathit{isCastValue}\ c) \qquad c \longrightarrow_{\cap IC} c'}{c : T_1 \Rightarrow^{l} T_2\ {}^{n} \longrightarrow_{\cap IC} c' : T_1 \Rightarrow^{l} T_2\ {}^{n}}\ \textsc{EvaluateC}$$

Detect success or failure of casts

$$\frac{\mathit{isCastValue1}\ c \vee \mathit{isEmptyCast}\ c}{c : T \Rightarrow^{l} T\ {}^{n} \longrightarrow_{\cap IC} c}\ \textsc{IdentityC}$$

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

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

Mediate the transition between the two disciplines

$$\frac{\begin{array}{c}\mathit{isCastValue1}\ c \vee \mathit{isEmptyCast}\ c \\ G\ \mathit{is\ ground\ type\ of}\ T \qquad \neg(\mathit{ground}\ T)\end{array}}{c : T \Rightarrow^{l} \mathit{Dyn}\ {}^{n} \longrightarrow_{\cap IC} c : T \Rightarrow^{l} G : G \Rightarrow^{l} \mathit{Dyn}\ {}^{n}}\ \textsc{GroundC}$$

$$\frac{\begin{array}{c}\mathit{isCastValue1}\ c \vee \mathit{isEmptyCast}\ c \\ G\ \mathit{is\ ground\ type\ of}\ T \qquad \neg(\mathit{ground}\ T)\end{array}}{c : \mathit{Dyn} \Rightarrow^{l} T\ {}^{n} \longrightarrow_{\cap IC} c : \mathit{Dyn} \Rightarrow^{l} G : G \Rightarrow^{l} T\ {}^{n}}\ \textsc{ExpandC}$$

Trigger stuck

$$\frac{\mathit{isCastValue1}\ c \vee \mathit{isEmptyCast}\ c \qquad \mathit{initialType}(c) = T_I}{c : T_1 \Rightarrow^{l} T_2\ {}^{n} \longrightarrow_{\cap IC} \perp\ T_I\ T_2\ {}^{n}}\ \textsc{TriggerStuckC}$$

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

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

$$[\![x]\!]_e = x$$

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

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

$$[\![n]\!]_e = n$$

$$[\![true]\!]_e = true$$

$$[\![false]\!]_e = false$$

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

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

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

$$\frac{[\![c_1]\!]_c = \varnothing\ T_1\ ^{n_1}\ \ \ldots\ \ [\![c_n]\!]_c = \varnothing\ T_n\ ^{n_n}}{[\![e : c_1 \cap \ldots \cap c_n]\!]_e = [\![e]\!]_e}$$

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

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

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

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

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

$$[\![\varnothing\ T\ ^n]\!]_c = \varnothing\ T\ ^n$$

$$[\![\perp\ T_I\ T_F\ ^n]\!]_c = \perp\ T_I\ T_F\ ^n$$

Figure 8: Identity Cast Erasure

2 Proofs

Theorem 1 (Depends on Lemmas 1, 5, 7)**.** *Equivalence to the Intersection System for fully static terms*
If e is fully static, T is a static type, and $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$*:*

1. $\Gamma \vdash_{\cap S} e : T \iff \Gamma \vdash_{\cap G} e : T$
2. $e \longrightarrow_{\cap S} v \iff e' \longrightarrow_{\cap CC} v$

Proof. (1) First we will prove that if $\vdash_{\cap S} e : T$ then $\vdash_{\cap G} e : T$. We proceed by induction on the length of the derivation tree of $\vdash_{\cap S}$.

Base case:

- e $= x$. If $\Gamma \vdash_{\cap S} x : T_i$, then $x : T_1 \cap \ldots \cap T_n \in \Gamma$ such that $T_i \in \{T_1, \ldots, T_n\}$. Therefore, by rule $\cap E$ of $\vdash_{\cap G}$, $\Gamma \vdash_{\cap G} e : T_i$.

Induction step:

- e $= \lambda x \;.\; T_1 \cap \ldots \cap T_n \;.\; e'$. There are two possibilities:
  - Using the 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, by rule $\to I$, $\Gamma \vdash_{\cap G} \lambda x \;.\; T_1 \cap \ldots \cap T_n \;.\; e' : T_1 \cap \ldots \cap T_n \to T$.
  - Using the 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, by rule $\to I'$, $\Gamma \vdash_{\cap G} \lambda x \;.\; T_1 \cap \ldots \cap T_n \;.\; e' : T_i \to T$.
- e $= e_1\ e_2$. 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, by rule $\to E$, $\Gamma \vdash_{\cap G} e_1\ e_2 : T$.
- e $= e$. 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, by rule $\cap E$, $\Gamma \vdash_{\cap G} e : T_1 \cap \ldots \cap T_n$.

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

Base case:

- e $= x$. If $\Gamma \vdash_{\cap G} x : T_i$, then $x : T_1 \cap \ldots \cap T_n \in \Gamma$ such that $T_i \in \{T_1, \ldots, T_n\}$. Therefore, by rule $\cap E$ of $\vdash_{\cap S}$, $\Gamma \vdash_{\cap S} e : T_i$.

Induction step:

- e $= \lambda x \;.\; T_1 \cap \ldots \cap T_n \;.\; e'$. There are two possibilities:

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

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

(2) Since $\Gamma \vdash_{\cap CC} e \rightsquigarrow e' : T$ e e is fully static, then by Lemma 5 e 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 $e \longrightarrow v$ and $e' \longrightarrow_{\cap CC} v'$, then v = v'. ☐

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 $T_1 = T_2$, then by the definition of $\sim$, $T_1 \sim T_2$.
  - If $T_1 \sim T_2$, then by the definition of $\sim$, $T_1 = T_2$.
- $T_1 = Bool$.
  - 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 $\sim$, $T_1 = T_2$.
- $T_1 = Dyn$. 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 $\exists T_{21}, T_{22} \ . \ T_2 = T_{21} \to T_{22}$ 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 $=$, $T_1 = T_2$.

- $T_1 = T_{11} \cap \ldots \cap T_{1n}$.
  - If $T_1 = T_2$, 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 $\sim$, $T_1 \sim T_2$.
  - If $T_1 \sim T_2$, then $\exists T_{21}, \ldots, T_{2n} \ . \ T_2 = T_{21} \cap \ldots \cap T_{2n}$ 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, by the definition of $=$, $T_1 = T_2$.

$\square$

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 $= \mathit{blame}\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2}$. $\vdash_{\cap IC} \mathit{blame}\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2} : T_2$ and by rule PushBlameC, $\mathit{blame}\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2} \longrightarrow_{\cap IC} \mathit{blame}\ T_I\ T_2\ l_1\ ^{n_1}$. As $\vdash_{\cap IC} \mathit{blame}\ T_I\ T_2\ l_1\ ^{n_1} : T_2$, then it is proved.

- c $= \perp\ T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^{l} T_2\ ^{n_2}$. $\vdash_{\cap IC} \perp\ T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^{l} T_2\ ^{n_2} : T_2$ and by rule PushStuckC, $\perp\ T_I\ T_F\ ^{h}en_1 : T_1 \Rightarrow^{l} T_2\ ^{n_2} \longrightarrow_{\cap IC} \perp\ T_I\ T_2\ ^{n_1}$. As $\vdash_{\cap IC} \perp\ T_I\ T_2\ ^{n_1} : T_2$, then it is proved.

- c $= c' : T \Rightarrow^{l} T\ ^{n}$ and $\mathit{isCastValue1}\ c' \ \vee\ \mathit{isEmptyCast}\ c'$. 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 $= c' : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{n_1} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{n_2}$ and $\mathit{isCastValue1}\ c' \ \vee\ \mathit{isEmptyCast}\ c'$. If $\vdash_{\cap IC} c' : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{n_1} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{n_2} : G$, then $\vdash_{\cap IC} c' : G$. By rule SucceedC, $c' : G \Rightarrow^{l_1} \mathit{Dyn}\ ^{n_1} : \mathit{Dyn} \Rightarrow^{l_2} G\ ^{n_2} \longrightarrow_{\cap IC} c'$. Therefore it is proved.

- c $= c' : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{n_1} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{n_2}$ and $\mathit{isCastValue1}\ c' \ \vee\ \mathit{isEmptyCast}\ c'$ and $\neg(\mathit{same\ ground}\ G_1\ G_2)$ and $\mathit{initialType}(c') = T_I$. If $\vdash_{\cap IC} c' : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{n_1} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{n_2} : G_2$, and by rule FailC, $c' : G_1 \Rightarrow^{l_1} \mathit{Dyn}\ ^{n_1} : \mathit{Dyn} \Rightarrow^{l_2} G_2\ ^{n_2} \longrightarrow_{\cap IC} \mathit{blame}\ T_I\ G_2\ l_2\ ^{n_1}$ and $\vdash_{\cap IC} \mathit{blame}\ T_I\ G_2\ l_2\ ^{n_1} : G_2$, it is proved.

- c $= c' : T \Rightarrow^{l} \mathit{Dyn}\ ^{n}$ and $\mathit{isCastValue1}\ c' \ \vee\ \mathit{isEmptyCast}\ c'$ and $G\ \mathit{is\ ground\ type\ of}\ T$ and $\neg(\mathit{ground}\ T)$. If $\vdash_{\cap IC} c' : T \Rightarrow^{l} \mathit{Dyn}\ ^{n} : \mathit{Dyn}$ then $\vdash_{\cap IC} c' : T$. By rule GroundC, $c' : T \Rightarrow^{l} \mathit{Dyn}\ ^{n} \longrightarrow_{\cap IC} c' : T \Rightarrow^{l} G\ ^{n} : G \Rightarrow^{l} \mathit{Dyn}\ ^{n}$. As $\vdash_{\cap IC} c' : T \Rightarrow^{l} G\ ^{n} : G \Rightarrow^{l} \mathit{Dyn}\ ^{n} : \mathit{Dyn}$, it is proved.

- c $= c' : Dyn \Rightarrow^{l} T\ ^{n}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $G\ is\ ground\ type\ of\ T$ and $\neg(ground\ T)$. If $\vdash_{\cap IC} c' : Dyn \Rightarrow^{l} T\ ^{n} : T$ then $\vdash_{\cap IC} c' : Dyn$. By rule ExpandC, $c' : Dyn \Rightarrow^{l} T\ ^{n} \longrightarrow_{\cap IC} c' : Dyn \Rightarrow^{l} G\ ^{n} : G \Rightarrow^{l} T\ ^{n}$. As $\vdash_{\cap IC} c' : Dyn \Rightarrow^{l} G\ ^{n} : G \Rightarrow^{l} T\ ^{n} : T$, it is proved.

- c $= c' : T_1 \Rightarrow^{l} T_2\ ^{n}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $initialType(c) = T_I$. If $\vdash_{\cap IC} c' : T_1 \Rightarrow^{l} T_2\ ^{n} : T_2$, and by rule TriggerStuckC, $c' : T_1 \Rightarrow^{l} T_2\ ^{n} \longrightarrow_{\cap IC} \perp T_I\ T_2\ ^{n}$, then $\vdash_{\cap IC} \perp T_I\ T_2\ ^{n} : T_2$.

Induction step:

- c $= c' : T_1 \Rightarrow^{l} T_2\ ^{n}$ and $\neg(isCastValue\ c)$. 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 $initialType(c) = T$ *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:

- c $= blame\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2}$. By the definition of $initialType$, $initialType(blame\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2}) = T_I$. By rule PushBlameC, $blame\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2} \longrightarrow_{\cap IC} blame\ T_I\ T_2\ l_1\ ^{n_1}$. Since $initialType(blame\ T_I\ T_2\ l_1\ ^{n_1}) = T_I$, it is proved.

- c $= \perp T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^{l} T_2\ ^{n_2}$. By the definition of $initialType$, $initialType(\perp T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^{l} T_2\ ^{n_2}) = T_I$. By rule PushStuckC, $\perp T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^{l} T_2\ ^{n_2} \longrightarrow_{\cap IC} \perp T_I\ T_2\ ^{n_1}$. Since $initialType(\perp T_I\ T_2\ ^{n_1}) = T_I$, it is proved.

- c $= c' : T \Rightarrow^{l} T\ ^{n}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$. By the definitions of $initialType$, $initialType(c' : T \Rightarrow^{l} T\ ^{n}) = initialType(c')$. By rule IdentityC, $c' : T \Rightarrow^{l} T\ ^{n} \longrightarrow_{\cap IC} c'$. Therefore it is proved.

- c $= c' : G \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G\ ^{n_2}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$. By the definition of $initialType$, $initialType(c' : G \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G\ ^{n_2}) = initialType(c')$. By rule SucceedC, $c' : G \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G\ ^{n_2} \longrightarrow_{\cap IC} c'$. Therefore it is proved.

- c $= c' : G_1 \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G_2\ ^{n_2}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $\neg(same\ ground\ G_1\ G_2)$ and $initialType(c') = T_I$. By the definition of $initialType$, $initialType(c' : G_1 \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G_2\ ^{n_2}) = T_I$. By rule FailC, $c' : G_1 \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G_2\ ^{n_2} \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{n_1}$. Since $initialType(blame\ T_I\ G_2\ l_2\ ^{n_1}) = T_I$, it is proved.

- c $= c' : T \Rightarrow^{l} Dyn\ ^{n}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $G\ is\ ground\ type\ of\ T$ and $\neg(ground\ T)$. By the definition of $initialType$, $initialType(c' : T \Rightarrow^{l} Dyn\ ^{n}) = initialType(c')$. By rule GroundC, $c' : T \Rightarrow^{l} Dyn\ ^{n} \longrightarrow_{\cap IC} c' : T \Rightarrow^{l} G\ ^{n} : G \Rightarrow^{l} Dyn\ ^{n}$. Since $initialType(c' : T \Rightarrow^{l} G\ ^{n} : G \Rightarrow^{l} Dyn\ ^{n}) = initialType(c')$, it is proved.

- 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 the definition of $initialType$, $initialType(c' : Dyn \Rightarrow^l T\ ^n) = initialType(c')$. By rule ExpandC, $c' : Dyn \Rightarrow^l T\ ^n \longrightarrow_{\cap IC} c' : Dyn \Rightarrow^l G\ ^n : G \Rightarrow^l T\ ^n$. Since $initialType(c' : Dyn \Rightarrow^l G\ ^n : G \Rightarrow^l T\ ^n) = initialType(c')$, it is proved.

- c $= c' : T_1 \Rightarrow^l T_2\ ^n$ and $isCastValue1\ c' \ \vee\ isEmptyCast\ c'$ and $initialType(c') = T_I$. By the definition of $initialType$, $initialType(c' : T_1 \Rightarrow^l T_2\ ^n) = T_I$. By rule TriggerStuckC, $c' : T_1 \Rightarrow^l T_2\ ^n \longrightarrow_{\cap IC} \perp T_I\ T_2\ ^n$. Since $initialType(\perp T_I\ T_2\ ^n) = T_I$, it is proved.

Induction step:

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

$\square$

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 introduze the binding $x : T'$ in Γ, such that T' is a static type. Therefore, T_i is also a static type.

Induction step:

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

- 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 $\rhd$, $T_1 \cap \ldots \cap T_n \rightarrow T$ 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, $T_1 \ldots T_n$ are static types. Therefore $T_1 \cap \ldots \cap T_n$ 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 $\lambda x : T_1 \cap \ldots \cap T_n \ . \ e'$. 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. Therefore, T_i is also a static type.

Induction step:

- e $= \lambda x : T_1 \cap \ldots \cap T_n \ . \ e'$. If 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.

- 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 $\rhd$, $T_1 \cap \ldots \cap T_n \to T$ is also a static type. Therefore, T is a static type.

$\square$

Lemma 5 (Depends on Lemmas 1 and 4)**.** *Static program compilation only adds identity casts*
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 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 $= \lambda x : T_1 \cap \ldots \cap T_n \ . \ e'$. 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, $\Gamma \vdash_{\cap CC} (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e') \rightsquigarrow (\lambda x : T_1 \cap \ldots \cap T_n \ . \ e'') : T_1 \cap \ldots \cap T_n \to T$. As the rule doesn't introduze new casts, then it is proved.

- e $= e_1 \ e_2$. By rule, $\Gamma \vdash_{\cap CC} e_1 \rightsquigarrow e_1' : PM$ and $\Gamma \vdash_{\cap CC} e_2 \rightsquigarrow e_2' : T_1' \cap \ldots \cap T_n'$. 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 $T_1' \cap \ldots \cap T_n'$

are static types. Therefore, by the definition of $\rhd$, $PM = T'_1 \cap \ldots \cap T'_n \to T$ and by Lemma 1, $T'_1 \cap \ldots \cap T'_n = T_1 \cap \ldots \cap T_n$. Therefore by the definition of $getInstances$ and $addCasts$, only identity casts are introduzed.

$\square$

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 $= \varnothing\ T^{\ n}$. As $\vdash_{\cap IC} \varnothing\ T^{\ n} : T$, $initialType(\varnothing\ T^{\ n}) = T$ and $[\![c]\!]_c = \varnothing\ T^{\ n}$, 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 $initialType([\![c]\!]_c) = T_I$.
- c $= \perp\ T_I\ T_F\ ^{n}$. As $\vdash_{\cap IC} \perp\ T_I\ T_F\ ^{n} : T_F$, $initialType(\perp\ T_I\ T_F\ ^{n}) = T_I$ and $[\![c]\!]_c = \perp\ T_I\ T_F\ ^{n}$, 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 PushBlameC, $blame\ T_I\ T_F\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_2\ ^{n_2} \longrightarrow_{\cap IC} blame\ T_I\ T_2\ l_1\ ^{n_1}$ 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 PushBlameC, $blame\ T_I\ T_1\ l_1\ ^{n_1} : T_1 \Rightarrow^{l_2} T_1\ ^{n_2} \longrightarrow_{\cap IC} blame\ T_I\ T_1\ l_1\ ^{n_1}$. Since $[\![c]\!]_c = blame\ T_I\ T_1\ l_1\ ^{n_1}$, and it is already a value, it is proved.

- c $= \perp\ T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^l T_2\ ^{n_2}$. There are two cases:

  - $T_1 \neq T_2$. As $[\![c]\!]_c = \perp\ T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^l T_2\ ^{n_2}$ and by rule PushStuckC, $\perp\ T_I\ T_F\ ^{n_1} : T_1 \Rightarrow^l T_2\ ^{n_2} \longrightarrow_{\cap IC} \perp\ T_I\ T_2\ ^{n_1}$ it is proved.
  - $T_1 = T_2$. If $T_1 = T_2$, then by rules T-SingleC and T-StuckC, $T_F = T_1$. Therefore, $c = \perp\ T_I\ T_1\ ^{n_1} : T_1 \Rightarrow^l T_1\ ^{n_2}$. By rule PushStuckC, $\perp\ T_I\ T_1\ ^{n_1} : T_1 \Rightarrow^l T_1\ ^{n_2} \longrightarrow_{\cap IC} \perp\ T_I\ T_1\ ^{n_1}$. Since $[\![c]\!]_c = \perp\ T_I\ T_1\ ^{n_1}$, and it is already a value, it is proved.

- c $= c' : T \Rightarrow^l T\ ^n$ and $isCastValue1\ c' \lor isEmptyCast\ c'$. By rule IdentityC, $c' : T \Rightarrow^l T\ ^n \longrightarrow_{\cap IC} c'$. 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.

- c $= c' : G \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G\ ^{n_2}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$. By rule SucceedC, $c' : G \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G\ ^{n_2} \longrightarrow_{\cap IC} c'$. As c' is already a value, then it doesn't contain identity casts, so $[\![c]\!]_c = c' : G \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G\ ^{n_2}$. Therefore, $[\![c]\!]_c \longrightarrow_{\cap IC} c'$.

- c $= c' : G_1 \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G_2\ ^{n_2}$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $\neg(same\ ground\ G_1\ G_2)$ and $initialType(c') = T_I$. By rule FailC, $c' : G_1 \Rightarrow^{l_1} Dyn\ ^{n_1} : Dyn \Rightarrow^{l_2} G_2\ ^{n_2} \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{n_1}$. 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, $[\![c]\!]_c \longrightarrow_{\cap IC} blame\ T_I\ G_2\ l_2\ ^{n_1}$.

- c $= c' : T \Rightarrow^l Dyn\ ^n$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $G\ is\ ground\ type\ of\ T$ and $\neg(ground\ T)$. By rule GroundC, $c' : T \Rightarrow^l Dyn\ ^n \longrightarrow_{\cap IC} c' : T \Rightarrow^l G : G \Rightarrow^l Dyn\ ^n$. As c' is a value, it doesn't contain identity casts, therefore $[\![c]\!]_c = c' : T \Rightarrow^l Dyn\ ^n$. Therefore $[\![c]\!]_c \longrightarrow_{\cap IC} c' : T \Rightarrow^l G : G \Rightarrow^l Dyn\ ^n$.

- c $= c' : Dyn \Rightarrow^l T\ ^n$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $G\ is\ ground\ type\ of\ T$ and $\neg(ground\ T)$. By rule ExpandC, $c' : Dyn \Rightarrow^l T\ ^n \longrightarrow_{\cap IC} c' : Dyn \Rightarrow^l G : G \Rightarrow^l T\ ^n$. As c' is a value, it doesn't contain identity casts, therefore $[\![c]\!]_c = c' : Dyn \Rightarrow^l T\ ^n$. Therefore $[\![c]\!]_c \longrightarrow_{\cap IC} c' : Dyn \Rightarrow^l G : G \Rightarrow^l T\ ^n$.

- c $= c' : T_1 \Rightarrow^l T_2\ ^n$ and $isCastValue1\ c' \lor isEmptyCast\ c'$ and $initialType(c') = T_I$. By rule TriggerStuckC, $c' : T_1 \Rightarrow^l T_2\ ^n \longrightarrow_{\cap IC} \perp\ T_I\ T_2\ ^n$. As $T_1 \neq T_2$ and c' is a value, then $[\![c]\!]_c = c' : T_1 \Rightarrow^l T_2\ ^n$. Therefore, $[\![c]\!]_c \longrightarrow_{\cap IC} \perp\ T_I\ T_2\ ^n$.

Induction step:

- c $= c' : T_1 \Rightarrow^l T_2\ ^n$ and $\neg(isCastValue c')$. 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 $[\![c']\!]_c : T_1 \Rightarrow^l T_2\ ^n$, then by rule EvaluateC, $[\![c]\!]_c \longrightarrow_{\cap IC} c'' : T_1 \Rightarrow T_2\ ^n$.
  - $T_1 = T_2$. By the induction hypothesis, as $c' \longrightarrow_{\cap IC} cv'$, then $[\![c']\!]_c \longrightarrow_{\cap IC} cv'$. By rule EvaluateC, $c' : T_1 \Rightarrow^l T_1\ ^n \longrightarrow_{\cap IC} cv' : T_1 \Rightarrow^l T_1\ ^n$. However, as $cv' : T_1 \Rightarrow^l T_1\ ^n$ is not a value, the rule

IdentityC must be applied, therefore $c' : T_1 \Rightarrow^l T_1 \ {}^n \longrightarrow_{\cap IC} cv'$. As $\llbracket c \rrbracket_c \longrightarrow_{\cap IC} cv'$, then it is proved.

$\square$

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

1. $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T$.
2. $\llbracket e \rrbracket_e \longrightarrow_{\cap CC} v$.

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

Base cases:

- e $= x$. As x doesn't contain casts, then $\llbracket e \rrbracket_e = x$. Therefore it is proved.
- e $= n$. As n doesn't contain casts, then $\llbracket e \rrbracket_e = n$. Therefore it is proved.
- e $= true$. As $true$ doesn't contain casts, then $\llbracket e \rrbracket_e = true$. Therefore it is proved.
- e $= false$. As $false$ doesn't contain casts, then $\llbracket e \rrbracket_e = false$. Therefore it is proved.
- e $= blame_T \ l$. As $blameTl$ doesn't contain casts, then $\llbracket e \rrbracket_e = blameTl$. Therefore it is proved.

Induction step:

- e $= \lambda x : T_1 \cap \ldots \cap T_n \ . \ e'$. There are two possibilities:
  - Using the 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, $\Gamma, x : T_1 \cap \ldots \cap T_n \vdash_{\cap CC} \llbracket e' \rrbracket_e : T$. As $\llbracket e \rrbracket_e = \lambda x : T_1 \cap \ldots \cap T_n \ . \ \llbracket e' \rrbracket_e$, then $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T_1 \cap \ldots \cap T_n \rightarrow T$.
  - Using the 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, $\Gamma, x : T_i \vdash_{\cap CC} \llbracket e' \rrbracket_e : T$. As $\llbracket e \rrbracket_e = \lambda x : T_1 \cap \ldots \cap T_n \ . \ \llbracket e' \rrbracket_e$, then $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T_i \rightarrow T$.
- e $= e_1 \ e_2$. If $\Gamma \vdash_{\cap CC} e_1 \ e_2 : T$, then $\Gamma \vdash_{\cap CC} e_1 : PM$, $PM \rhd T_1 \cap \ldots \cap T_n \rightarrow T$, $\Gamma \vdash_{\cap CC} e_2 : T'_1 \cap \ldots \cap T'_n$ and $T'_1 \cap \ldots \cap T'_n \sim T_1 \cap \ldots \cap T_n$. By the induction hypothesis, $\Gamma \vdash_{\cap CC} \llbracket e_1 \rrbracket_e : PM$ and $\Gamma \vdash_{\cap CC} \llbracket e_2 \rrbracket_e : T'_1 \cap \ldots \cap T'_n$. As $\llbracket e \rrbracket_e = \llbracket e_1 \rrbracket_e \ \llbracket e_2 \rrbracket_e$, therefore $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T$.
- e $= e$. 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, $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T_1$ and ... and $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T_n$. Therefore $\Gamma \vdash_{\cap CC} \llbracket e \rrbracket_e : T_1 \cap \ldots \cap T_n$.
- 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 $= e' : c_1 \cap \ldots \cap c_n$. 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 $initialType(c_1) \cap \ldots \cap initialType(c_n) =_{\cap} T$. 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 6, $\vdash_{\cap IC} [\![c_i]\!]_c : T_i$ 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 6 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:

- e $= v_1 : cv_1 \cap \ldots \cap cv_n$ and $isValue(v_1 : cv_1 \cap \ldots \cap cv_n)$ and $\exists i \in 1..n \ . \ isArrowCompatible \ cv_i$. As $v_1 : cv_1 \cap \ldots \cap cv_n$ is a value, then it doesn't contain identity casts. Therefore $[\![e]\!]_e = e$.

- e $= v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2$ and $isValue \ v : cv_1 \cap \ldots \cap cv_n$ and $label(cv_1) = m_1$ and ... and $label(cv_n) = m_n$. There are 2 possibilities:
  - If $T_1 \neq T_2$ and as $v : cv_1 \cap \ldots \cap cv_n$ doesn't contain identity casts, then $[\![e]\!]_e = e$, therefore it is proved.
  - If $T_1 = T_2$ and as $v : cv_1 \cap \ldots \cap cv_n$ doesn't contain identity casts, then $[\![e]\!]_e = v : cv_1 \cap \ldots \cap cv_n$. By rule Merge1$\cap$, $v : cv_1 \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 \longrightarrow_{\cap CC} v : cv_1 : T_1 \Rightarrow^l T_2 \ ^{m_1} \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 \ ^{m_n}$. By rule EvaluateCasts$\cap$, $v : cv_1 : T_1 \Rightarrow^l T_2 \ ^{m_1} \cap \ldots \cap cv_n : T_1 \Rightarrow^l T_2 \ ^{m_n} \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$, 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 $= v : c_1 \cap \ldots \cap c_n$ and $isIntersectionCast \ v \vee isCast \ v$. There are 2 possibilities:

- $v : c_1 \cap \ldots \cap c_n$ doesn't contain identity casts, then $\llbracket e \rrbracket_e = e$, therefore it is proved.
- $v : c_1 \cap \ldots \cap c_n$ contain identity casts. By rule Merge2$\cap$, $v : c_1 \cap \ldots \cap c_n \longrightarrow_{\cap CC} v' : c'_1 \cap \ldots \cap c'_m$. By rule Evaluate$\cap$, $v' : c'_1 \cap \ldots \cap c'_m \longrightarrow_{\cap CC} v' : cv'_1 \cap \ldots \cap cv'_m$, with $c'_1 \longrightarrow_{\cap IC} cv'_1$ and ... and $c'_m \longrightarrow_{\cap IC} cv'_m$. For all casts c_i that don't contain identity casts, then $\llbracket c_i \rrbracket_c = c_i$, therefore for those casts, the property is proved. For all casts c_i that contain identity casts, *margeCasts* will generate casts c'_i that will evaluate to cv'_i. By Lemma 6, casts $\llbracket c_i \rrbracket_c$ will generate casts c''_i that will evaluate to cv'_i, therefore it is proved.

- e $= v : c_1 \cap \ldots \cap c_n$ and $\neg(\forall i \in 1..n \,.\, isCastValue\ c_i)$. 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 $\llbracket e \rrbracket_e = v : \llbracket c_1 \rrbracket_c \cap \ldots \cap \llbracket c_n \rrbracket_c$, by Lemma 6, $\llbracket c_1 \rrbracket_c \longrightarrow_{\cap IC} cv_1$ and ... and $\llbracket c_n \rrbracket_c \longrightarrow_{\cap IC} cv_n$. Therefore, by rule EvaluateCasts$\cap$, $v : \llbracket c_1 \rrbracket_c \cap \ldots \cap \llbracket c_n \rrbracket_c \longrightarrow_{\cap CC} v : cv_1 \cap \ldots \cap cv_n$.
- e $= v : blame\ I_1\ F_1\ l_1\ {}^{m_1} \cap \ldots \cap blame\ I_n\ F_n\ l_n\ {}^{m_n}$. As $\llbracket e \rrbracket_e = e$, then it is proved.
- e $= v : \varnothing\ T_1\ {}^{m_1} \cap \ldots \cap \varnothing\ T_n\ {}^{m_n}$. As $\llbracket e \rrbracket_e = e$, then it is proved.
- e $= v : cv_1 \cap \ldots \cap cv_n$ and $\neg(\forall i \in 1..n \,.\, isStuck\ c_i)$ and $\exists i \in 1..n \,.\, isStuck\ c_i$. As $\llbracket e \rrbracket_e = e$, then it is proved.

Induction step:

- e =

$\square$