1 GhostLambda: Lambda-Calculus with Ghost Code

In this section we describe a language where one can annotate programs with such *ghost code*. We start by formalizing *ghostλ-calculus*, a tiny language of simply typed λ-calculus enriched with ghost variables and ghost expressions. We then define ghost code *erasure*, which transforms a well-typed ghostLambda term to a term of standard λ-calculus. Finally we state and proof a few basic preservation properties of such translation.

1.1 gλ-calculus syntax and semantics

The syntax and small-step operational semantics of *ghost-λ* is summarized below.

Syntax

$\mathtt{t}$	$::=$		TERMS:
		$()$	*unit*
		$\mathtt{x}^{\mathfrak{B}}_{\tau}$	*variable*
		$\lambda \mathtt{x}^{\mathfrak{B}}_{\tau}.\mathtt{t}$	*abstraction*
$\mathtt{v}$	$::=$		VALUES:
		$()$	*unit*
		$\lambda \mathtt{x}^{\mathfrak{B}}_{\tau}.\mathtt{t}$	*abstraction*
$\mathfrak{B}$	$::=$		GHOST STATUS :
		$\top \ (*true*)$	*ghost code*
		$\bot \ (*false*)$	*physical code*
τ	$::=$		TYPES:
		$\mathtt{unit}$	*unit type*
		$\tau^{\mathfrak{B}} \to \tau$	*function type*

Evaluation

$$(\lambda x^{b}_{\tau}.\mathtt{t})v \ \xrightarrow{\epsilon}\ \mathtt{t}[x^{b}_{\tau} \leftarrow v] \qquad \text{(E-AppFun)}$$

$$\mathtt{ghost\ t} \ \xrightarrow{\epsilon}\ \mathtt{t} \qquad \text{(E-DeGhost)}$$

$$\frac{\mathtt{t}_1 \to \mathtt{t}_1'}{\mathtt{t}_1\mathtt{t}_2 \to \mathtt{t}_1'\mathtt{t}_2} \qquad \text{(T-AppLeft)}$$

$$\frac{\mathtt{t}_2 \to \mathtt{t}_2'}{\mathtt{v}_1\mathtt{t}_2 \to \mathtt{v}_1\mathtt{t}_2'} \qquad \text{(T-AppLeft)}$$

Figure 1: ***ghost-λ* syntax and semantics**

Free variables, scope and equivalence of terms

Admit.

1.2 Typing Relation

$$\frac{}{\vdash_{g\lambda} () : (unit, \bot)} \quad \text{(T-UNIT)}$$

$$\frac{}{\vdash_{g\lambda} x_\tau^{\mathfrak{B}} : (\tau, \mathfrak{B})} \quad \text{(T-VAR)}$$

$$\frac{\vdash_{g\lambda} \mathtt{t} : (\tau, \mathfrak{B})}{\vdash_{g\lambda} \mathtt{ghost\ \ t} : (\tau, \top)} \quad \text{(T-GHOST)}$$

$$\frac{\vdash_{g\lambda} \mathtt{t} : (\tau_2, \mathfrak{B}_2)}{\vdash_{g\lambda} \lambda x_{\tau_1}^{\mathfrak{B}_1}.\mathtt{t} : (\tau_1^{\mathfrak{B}_1} \to \tau_2, \mathfrak{B}_2)} \quad \text{(T-ABS)}$$

$$\frac{\vdash_{g\lambda} \mathtt{t_1} : (\tau_2^{\mathfrak{B}_2} \to \tau_1, \mathfrak{B}_1) \qquad \vdash_{g\lambda} \mathtt{t_2} : (\tau_2, \mathfrak{B}'_2) \qquad \mathfrak{B}_2 \Rightarrow \mathfrak{B}'_2}{\vdash_{g\lambda} \mathtt{t_1\ t_2} : (\tau_1, \mathfrak{B}_1 \vee (\neg\mathfrak{B}_2 \wedge \mathfrak{B}'_2))} \quad \text{(T-APP)}$$

1.2.1 Properties of typing

Lemma 1.1 [INVERSION OF TYPING RELATION].

1. *if* $\vdash_{g\lambda}$ *ghost* $t : (\tau_1, \mathfrak{B}_1)$ *then* $\mathfrak{B}_1 = \top$ *and* $\vdash_{g\lambda} t : (\tau_1, \mathfrak{B}_2)$.
2. *if* $\vdash_{g\lambda} \lambda x_{\tau_2}^{\mathfrak{B}_2}.t : (\tau_1, \mathfrak{B}_1)$ *then* $\tau_1 = \tau_2^{\mathfrak{B}_2} \to \tau_{11}$ *for some* τ_{11} *with* $\vdash_{g\lambda} t : (\tau_{11}, \mathfrak{B}_1)$.
3. *If* $\vdash_{g\lambda} t_1\ t_2 : (\tau_1, \mathfrak{B}_1)$ *then there exist* τ_{11}, τ_2, $\mathfrak{B}_2$ *and* $\mathfrak{B}'_2$ *such that* $\vdash_{g\lambda} t_1 : (\tau_2^{\mathfrak{B}_2} \to \tau_{11}, \mathfrak{B}_1)$ *and* $\vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}'_2)$ *with* $\vDash \mathfrak{B}_1 \vee (\neg\mathfrak{B}_2 \wedge \mathfrak{B}'_2) \wedge (\mathfrak{B}_2 \Rightarrow \mathfrak{B}'_2)$.

   In particular, if $\vdash_{g\lambda} t_1\ t_2 : (\tau_1, \bot)$ *then* $\vDash \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}'_2$.

Proof. Straightforward from definition of the typing relation. □

Lemma 1.2 [PROGRESS]. *Admit.*

Lemma 1.3 [PRESERVATION]. *Admit.*

Theorem 1.1 [SOUNDNESS]. *Admit.*

1.3 Ghost Code Erasure

Once we formally defined the simply typed lambda-calculus enriched with ghost expressions, our goal is to show that terms which are not ghost themselves have the same computational behaviour as their translation to lambda-calculus, which preserves the structure of terms except for ghost sub-expressions.

Therefore we need to define at first a type-erasure and a term-erasure translations from gλ-calculus to standard simply typed λ-calculus.

1.3.1 Target language

Standard Simply typed lambda calculus. ...

1.3.2 Ghost Erasure

Definition 1.2 (Type-Erasure). We define type-erasure by induction on the structure of types :

$\mathcal{E}_\top(\tau) = \texttt{unit}$
$\mathcal{E}_\bot(\texttt{unit}) = \texttt{unit}$
$\mathcal{E}_\bot(\tau_2^{\mathfrak{B}_2} \to \tau_1) = \mathcal{E}_{\mathfrak{B}_2}(\tau_2) \to \mathcal{E}_\bot(\tau_1)$.

Definition 1.3 (Term-Erasure). Let $\texttt{t}$ be a term such that $\vdash_{g\lambda} \texttt{t} : (\tau, \mathfrak{B})$ holds. We define term-erasure function by induction on the structure of $\texttt{t}$

$\mathcal{E}_\top(\texttt{t}) = \texttt{()}$
$\mathcal{E}_\bot(\texttt{()}) = \texttt{()}$
$\mathcal{E}_\bot(x_\tau^\bot) = x_{\mathcal{E}_\bot(\tau)}$
$\mathcal{E}_\bot(\lambda x_{\tau_2}^{\mathfrak{B}_2}.t) = \lambda x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)}.\mathcal{E}_\bot(t)$
$\mathcal{E}_\bot(\texttt{t}_1\ \texttt{t}_2) = \mathcal{E}_\bot(\texttt{t}_1)\ \mathcal{E}_{\mathfrak{B}_2}(\texttt{t}_2)$ where $\vdash_{g\lambda} \texttt{t}_2 : (\tau_2, \mathfrak{B}_2)$.

As it can be seen, the erasure function is a morphism that preserve the structure of operational (not ghost) terms and their types ($\sim \mathcal{E}_\bot(\star)$), and sends ghost expressions and types to $\texttt{()}$ and $\texttt{unit}$ respectively ($\sim \mathcal{E}_\top(\star)$).

1.4 Properties of ghost erasure

Now that we defined the erasure-translation of $g\lambda$-calculus to λ-calculus, our concern is to show that evaluation result of well-typed operational terms as well as their typing are preserved under erasure. First off we need to state and prove a few basic lemmas.

1.4.1 Evaluation Preservation

Lemma 1.4 [Substitution under erasure].
If $\vdash_{g\lambda} t_1 : (\tau_1, \bot)$ *and* $\vdash_{g\lambda} v_2 : (\tau_2, \mathfrak{B}_2)$ *hold,*
then $\mathcal{E}_\bot(t_1[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow v_2]) = \mathcal{E}_\bot(t_1)[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)]$

Proof. By induction on the structure of $\mathtt{t}_1$.

Case $\mathtt{t}_1 = x_{\tau_2}^{\mathfrak{B}_2}$:
In that case, we can deduce that $\mathfrak{B}_2 = \bot$. Therefore:

$$\begin{aligned}\mathcal{E}_\bot(x_{\tau_2}^{\bot}[x_{\tau_2}^{\bot} \hookleftarrow \mathtt{v}_2]) = \mathcal{E}_\bot(\mathtt{v}_2) = x_{\mathcal{E}_\bot(\tau_2)}[x_{\mathcal{E}_\bot(\tau_2)} \hookleftarrow \mathcal{E}_\bot(\mathtt{v}_2)]\\ = \mathcal{E}_\bot(x_{\tau_2}^{\bot})[x_{\mathcal{E}_\bot(\tau_2)} \hookleftarrow \mathcal{E}_\bot(\mathtt{v}_2)]\end{aligned}$$

Case $\mathtt{t}_1 = y_{\tau_2'}^{\bot} \neq x_{\tau_2}^{\mathfrak{B}_2}$:

$$\begin{aligned}\mathcal{E}_\bot(y_{\tau_2'}^{\bot}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2]) = \mathcal{E}_\bot(y_{\tau_2'}^{\bot}) = y_{\mathcal{E}_\bot(\tau_2')}\\ = y_{\mathcal{E}_\bot(\tau_2')}[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)] = \mathcal{E}_\bot(y_{\tau_2'}^{\bot})[x_{\mathcal{E}_{\mathfrak{B}_2(\tau_2)}} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)]\end{aligned}$$

Case $\mathtt{t}_1 = \lambda y_{\tau_2'}^{\mathfrak{B}_2'}.t_{11}$ with $y_{\tau_2'}^{\mathfrak{B}_2'} \notin \mathtt{FV}(\mathtt{v}_2)$ and $\neq x_{\tau_2}^{\mathfrak{B}_2}$:

$$\begin{aligned}\mathcal{E}_\bot((\lambda y_{\tau_2'}^{\mathfrak{B}_2'}.t_{11})[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2]) = \mathcal{E}_\bot[\lambda y_{\tau_2'}^{\mathfrak{B}_2'}.(t_{11}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2])]\\ = \lambda y_{\mathcal{E}_{\mathfrak{B}_2'}(\tau_2')}.\mathcal{E}_\bot(\mathtt{t}_{11}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2])]\\ \overset{Ind.Hyp.}{=} \lambda y_{\mathcal{E}_{\mathfrak{B}_2'}(\tau_2')}.\mathcal{E}_\bot(\mathtt{t}_{11})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)]\\ = (\lambda y_{\mathcal{E}_{\mathfrak{B}_2'}(\tau_2')}.\mathcal{E}_\bot(\mathtt{t}_{11}))[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)]\\ = \mathcal{E}_\bot(\lambda y_{\tau_2'}^{\mathfrak{B}_2'}.t_{11})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)]\end{aligned}$$

Case $\mathtt{t}_1 = \mathtt{t}_{11}\mathtt{t}_{12}$

$$\begin{aligned}\mathcal{E}_\bot(\mathtt{t}_{11}\mathtt{t}_{12}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2]) = \mathcal{E}_\bot(\mathtt{t}_{11}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2]\ \mathtt{t}_{12}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2])\\ = \mathcal{E}_\bot(\mathtt{t}_{11}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2])\ \mathcal{E}_\bot(\mathtt{t}_{12}[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_2])\\ \overset{Ind.Hyp.}{=} (\mathcal{E}_\bot(\mathtt{t}_{11})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)])(\mathcal{E}_\bot(\mathtt{t}_{12})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)])\\ = \mathcal{E}_\bot(\mathtt{t}_{11}\mathtt{t}_{12})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2}(\mathtt{v}_2)])\end{aligned}$$

□

Lemma 1.5 [One-step evaluation under erasure]. *For any closed $g\lambda$-term t such that $\vdash_{g\lambda} t : (\tau, \bot)$ holds, if $t \to_{g\lambda} t'$ for some term t', then either $\mathcal{E}_\bot(t) \to_\lambda \mathcal{E}_\bot(t')$ or $\mathcal{E}_\bot(t) = \mathcal{E}_\bot(t')$.*

Proof. By induction on the evaluation relation of $\mathtt{t} \rightarrow_{g\lambda} \mathtt{t}'$.

Case E-AppAbs: $\quad \mathtt{t} = (\lambda x_{\tau_2}^{\mathfrak{B}_2}.\mathtt{t}_1)\mathtt{v}_1$ with $(\lambda x_{\tau_2}^{\mathfrak{B}_2}.\mathtt{t}_1)\mathtt{v}_1 \xrightarrow{\epsilon}_{g\lambda} \mathtt{t}_1[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_1]$
$\vdash_{g\lambda} (\lambda x_{\tau_2}^{\mathfrak{B}_2}.\mathtt{t}_1)\mathtt{v}_1 : (\tau_1, \mathfrak{B}_1), \quad \vdash_{g\lambda} \mathtt{v}_1 : (\tau_2, \mathfrak{B}_2'),$
$\mathfrak{B}_1 = \bot, \quad \models \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}_2'$

$$
\begin{array}{ll}
\mathcal{E}_{\bot}[(\lambda x_{\tau_2}^{\mathfrak{B}_2}.\mathtt{t}_1)\mathtt{v}_1] & \\
= \lambda x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)}.\mathcal{E}_{\bot}(\mathtt{t}_1))\mathcal{E}_{\mathfrak{B}_2'}(\mathtt{v}_1) & (\text{as } \mathfrak{B}_1 = \bot) \\
\xrightarrow{\epsilon}_{\lambda} \mathcal{E}_{\bot}(\mathtt{t}_1)[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \hookleftarrow \mathcal{E}_{\mathfrak{B}_2'}(\mathtt{v}_1)] & (\text{head red.}) \\
= \mathcal{E}_{\bot}(\mathtt{t}_1[x_{\tau_2}^{\mathfrak{B}_2} \hookleftarrow \mathtt{v}_1]) & (\text{by Substitution under erasure lemma})
\end{array}
$$

Case E-DeGhost:
Trivially verified, as for any instance of $\vdash_{g\lambda}$ *ghost*$\mathtt{t}_1 : (\tau_1, \mathfrak{B}_1)$, $\mathfrak{B}_1 = \top$.

Case E-AppLeft: $\quad \mathtt{t} = \mathtt{t}_1\mathtt{t}_2, \quad \mathtt{t}' = \mathtt{t}_1\mathtt{t}_2'$, with $\mathtt{t}_1 \rightarrow_{g\lambda} \mathtt{t}_1'$
$\vdash_{g\lambda} \mathtt{t}_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1), \quad \vdash_{g\lambda} \mathtt{t}_2 : (\tau_2, \mathfrak{B}_2'),$
$\mathfrak{B}_1 = \bot, \quad \models \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}_2'$

As $\mathfrak{B}_1 = \bot$, we can apply induction hypothesis on $\mathtt{t}_1$ which gives $\mathcal{E}_{\bot}(\mathtt{t}_1) \rightarrow_{\lambda} \mathcal{E}_{\bot}(\mathtt{t}_1')$. Then, applying E-AppRight rule, we obtain:

$$\mathcal{E}_{\bot}(\mathtt{t}) = \mathcal{E}_{\bot}(\mathtt{t}_1)\mathcal{E}_{\bot}(\mathtt{t}_2) \rightarrow_{\lambda} \mathcal{E}_{\bot}(\mathtt{t}_1')\mathcal{E}_{\bot}(\mathtt{t}_2) = \mathcal{E}_{\bot}(\mathtt{t}').$$

Case E-AppRight: $\quad \mathtt{t} = \mathtt{v}_1\mathtt{t}_2, \quad \mathtt{t}' = \mathtt{v}_1\mathtt{t}_2'$, with $\mathtt{t}_2 \rightarrow_{g\lambda} \mathtt{t}_2'$
$\vdash_{g\lambda} \mathtt{v}_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1), \quad \vdash_{g\lambda} \mathtt{t}_2 : (\tau_2, \mathfrak{B}_2'),$
$\mathfrak{B}_1 = \bot, \quad \models \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}_2'$

If $\mathfrak{B}_2 = \mathfrak{B}_2' = \top$, then

$$\mathcal{E}_{\bot}(\mathtt{t}) = \mathcal{E}_{\bot}(\mathtt{v}_1)\mathcal{E}_{\top}(\mathtt{t}_2) = (\mathcal{E}_{\bot}(\mathtt{v}_1))() = \mathcal{E}_{\bot}(\mathtt{v}_1)\mathcal{E}_{\top}(\mathtt{t}_2') = \mathcal{E}_{\bot}(\mathtt{t}').$$

Otherwise, $\mathfrak{B}_2 = \mathfrak{B}_2' = \bot$. By induction hypothesis, $\mathcal{E}_{\bot}(\mathtt{t}_2) \rightarrow_{\lambda} \mathcal{E}_{\bot}(\mathtt{t}_2')$. Then, applying E-AppRight rule, we obtain:

$$\mathcal{E}_{\bot}(\mathtt{t}) = \mathcal{E}_{\bot}(\mathtt{v}_1)\mathcal{E}_{\bot}(\mathtt{t}_2) \rightarrow_{\lambda} \mathcal{E}_{\bot}(\mathtt{v}_1)\mathcal{E}_{\bot}(\mathtt{t}_2') = \mathcal{E}_{\bot}(\mathtt{t}').$$

□

Now we can prove the main theorem.

Theorem 1.4 [Value preservation under erasure]. *For any closed $g\lambda$-term t such that $\vdash_{g\lambda} t : (\tau, \bot)$ holds, if $t \rightarrow_{g\lambda}^{*} v$ for some value v, then $\mathcal{E}(t) \rightarrow_{\lambda}^{*} \mathcal{E}(v)$.*

Proof. By induction on the length of the evaluation of $\mathtt{t} \rightarrow_{g\lambda}^{*} \mathtt{v}$.

We already have proved the base case : indeed, if $\mathtt{t} \rightarrow_{g\lambda} \mathtt{v}$ then by the one-step evaluation lemma, $\mathcal{E}_{\bot}(\mathtt{t}) \rightarrow_{\lambda}^{0|1} \mathcal{E}_{\bot}(\mathtt{v})$.

Now, assume that $\mathtt{t} \rightarrow^{1}_{g\lambda} \mathtt{t}' \rightarrow^{n}_{g\lambda} \mathtt{v}$ for some arbitrary $n \in \mathbb{N}$. By the progress of typing, $\vdash_{g\lambda} \mathtt{t}' : (\tau, \perp)$, so we can apply induction hypothesis on $\mathtt{t}'$ which gives $\mathcal{E}(\mathtt{t}') \rightarrow^{*}_{\lambda} \mathcal{E}(\mathtt{v})$. By the one-step evaluation lemma again, we have $\mathcal{E}_{\perp}(\mathtt{t}) \rightarrow^{0|1}_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}')$. That is, $\mathcal{E}_{\perp}(\mathtt{t}) \rightarrow^{*}_{\lambda} \mathcal{E}_{\perp}(\mathtt{v})$. □

1.4.2 Typing Erasure

Lemma 1.6 [Typing relation under erasure].
If $\vdash_{g\lambda} t : (\tau, \perp)$ *then* $\vdash_{\lambda} \mathcal{E}_{\perp}(t) : \mathcal{E}_{\perp}(\tau)$.

Proof. By induction on a derivation of the statement $\vdash_{g\lambda} \mathcal{E}_{\perp}(\mathtt{t}) : \mathcal{E}_{\perp}(\tau)$. For a given derivation, we proceed by case analysis on the final typing rule used in the proof.

Case T-Unit: $\vdash_{g\lambda} () : (\mathtt{unit}, \perp)$

Immediately by definition of $\mathcal{E}_{\perp}$.

Case T-Var: $\vdash_{g\lambda} x^{\perp}_{\tau} : (\tau, \perp)$

$\mathcal{E}_{\perp}(x^{\perp}_{\tau}) = x_{\mathcal{E}_{\perp}(\tau)}$ gives immediately $\vdash_{\lambda} x_{\mathcal{E}_{\perp}(\tau)} : \mathcal{E}(\tau)$.

Case T-Abs: $\vdash_{g\lambda} \lambda x^{\mathfrak{B}_2}_{\tau_2}.\mathtt{t}_1 : (\tau_2^{2} \rightarrow \tau_1, \perp)$ with $\vdash_{g\lambda} \mathtt{t}_1 : (\tau_1, \perp)$

By induction hypothesis $\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1) : \mathcal{E}_{\perp}(\tau_1)$. There are two cases to consider, depending on whether the parameter of the abstraction is ghost or not. If $\mathfrak{B}_2 = \top$ then $\mathcal{E}_{\perp}(\lambda x^{\top}_{\tau_2}.\mathtt{t}_1) = \lambda x_{\mathtt{unit}}.\mathcal{E}(\mathtt{t}_1)$ and therefore

$$\frac{\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1) : \mathcal{E}_{\perp}(\tau_2)}{\vdash_{\lambda} \lambda x_{\mathtt{unit}}.\mathcal{E}_{\perp}(\mathtt{t}_1) : \mathtt{unit} \rightarrow \mathcal{E}_{\perp}(\tau_1)} \quad \text{(T-Abs)}$$

Otherwise $\mathfrak{B}_2 = \perp$ and again by the rule T-Abs we obtain :

$$\frac{\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1) : \mathcal{E}_{\perp}(\tau_1)}{\vdash_{\lambda} \lambda x_{\mathcal{E}_{\perp}(\tau_2)}.\mathcal{E}_{\perp}(\mathtt{t}_1) : \mathcal{E}_{\perp}(\tau_2) \rightarrow \mathcal{E}_{\perp}(\tau_1)} \quad \text{(T-Abs)}$$

Case T-App: $\vdash_{g\lambda} \mathtt{t}_1\ \mathtt{t}_2 : (\tau_1, \perp)$ with sub-derivations:
$\vdash_{g\lambda} \mathtt{t}_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1)$
$\vdash_{g\lambda} \mathtt{t}_2 : (\tau_2, \mathfrak{B}'_2)$,

As $\vdash_{g\lambda} \mathtt{t}_1\ \mathtt{t}_2 : (\tau_1, \perp)$, the inversion lemma gives as By inversion that $\vDash \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}'_2$. That is, we have two cases to consider.

If $\mathfrak{B}_2 = \mathfrak{B}'_2 = \perp$ then by induction hypotheses
$\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1) : \mathcal{E}_{\perp}(\tau_2) \rightarrow \mathcal{E}_{\perp}(\tau_1)$ and $\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_2) : \mathcal{E}_{\perp}(\tau_2)$. By T-App rule,

$$\frac{\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1) : \mathcal{E}_{\perp}(\tau_2) \rightarrow \mathcal{E}_{\perp}(\tau_1) \qquad \vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_2) : \mathcal{E}_{\perp}(\tau_2)}{\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1\ \mathtt{t}_2) : \mathcal{E}(\tau_1)} \quad \text{(T-App)}$$

If $\mathfrak{B}_2 = \mathfrak{B}'_2 = \top$, then by definition of $\mathcal{E}$ we have $\mathcal{E}(\mathtt{t}_2) = ()$ and $\mathcal{E}_{\mathfrak{B}'_2}(\tau_2) = \mathtt{unit}$. By induction hypothesis on $\mathtt{t}_1$, $\vdash_{\lambda} \mathcal{E}_{\perp}(\mathtt{t}_1) : \mathtt{unit} \rightarrow \mathcal{E}_{\perp}(\tau_1)$.

Applying T-App rule gives us

$$\dfrac{\dfrac{}{\vdash_\lambda \mathcal{E}_\perp(\mathtt{t}_1\ \mathtt{t}_2) : \mathcal{E}(\tau_1)} \quad \dfrac{}{\vdash_\lambda \mathtt{()} : \mathtt{unit}}\ \text{(T-Unit)}}{\vdash_\lambda \mathcal{E}_\perp(\mathtt{t}_1\ \mathtt{t}_2) : \mathcal{E}_\perp(\tau_1)}\ \text{(T-App)}$$

The case of (T-Ghost) as well as any other valid derivation where a typed term is marked as ghost do not satisfy lemma's requirement, so these cases are trivially verified. □