

# 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 *ghost* $\lambda$ -calculus, a tiny language of simply typed  $\lambda$ -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  $\lambda$ -calculus. Finally we state and proof a few basic preservation properties of such translation.

## 1.1 g $\lambda$ -calculus syntax and semantics

The syntax and small-step operational semantics of *ghost* $\lambda$  is summarized below.

**Free variables, scope and equivalence of terms**

*Admit.*

## 1.2 Typing Relation

$$\begin{array}{c}
\frac{}{\vdash_{g\lambda} () : (unit, \perp)} \quad (T\text{-UNIT}) \\
\\
\frac{}{\vdash_{g\lambda} x_{\tau}^{\mathfrak{B}} : (\tau, \mathfrak{B})} \quad (T\text{-VAR}) \\
\\
\frac{\vdash_{g\lambda} t : (\tau, \mathfrak{B})}{\vdash_{g\lambda} \text{ghost } t : (\tau, \top)} \quad (T\text{-GHOST}) \\
\\
\frac{\vdash_{g\lambda} t : (\tau_2, \mathfrak{B}_2)}{\vdash_{g\lambda} \lambda x_{\tau_1}^{\mathfrak{B}_1}. t : (\tau_1^{\mathfrak{B}_1} \rightarrow \tau_2, \mathfrak{B}_2)} \quad (T\text{-ABS}) \\
\\
\frac{\vdash_{g\lambda} t_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1) \quad \vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}'_2) \quad \mathfrak{B}_2 \Rightarrow \mathfrak{B}'_2}{\vdash_{g\lambda} t_1 t_2 : (\tau_1, \mathfrak{B}_1 \vee (\neg \mathfrak{B}_2 \wedge \mathfrak{B}'_2))} \quad (T\text{-APP})
\end{array}$$

### 1.2.1 Properties of typing

*Lemma 1.1* [INVERSION OF TYPING RELATION].

1. if  $\vdash_{g\lambda} \text{ghost } t : (\tau_1, \mathfrak{B}_1)$  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  $\tau_1 = \tau_2^{\mathfrak{B}_2} \rightarrow \tau_{11}$  for some  $\tau_{11}$  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  $\tau_{11}$ ,  $\tau_2$ ,  $\mathfrak{B}_2$  and  $\mathfrak{B}'_2$  such that  $\vdash_{g\lambda} t_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_{11}, \mathfrak{B}_1)$  and  $\vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}'_2)$  with  $\models \mathfrak{B}_1 \vee (\neg \mathfrak{B}_2 \wedge \mathfrak{B}'_2) \wedge (\mathfrak{B}_2 \Rightarrow \mathfrak{B}'_2)$ .

## Syntax

|                    |   |  |
|--------------------|---|--|
| $t ::=$            | $()$<br>$x_\tau^{\mathfrak{B}}$<br>$\lambda x_\tau^{\mathfrak{B}}.t$<br>$tt$<br>$\text{ghost } t$ | <b>TERMS:</b><br><i>unit</i><br><i>variable</i><br><i>abstraction</i><br><i>application</i><br><i>ghost term</i> |
| $v ::=$            | $()$<br>$\lambda x_\tau^{\mathfrak{B}}.t$   | <b>VALUES:</b><br><i>unit</i><br><i>abstraction</i>  |
| $\mathfrak{B} ::=$ | $\top$ ( <i>*true*</i> )<br>$\perp$ ( <i>*false*</i> )  | <b>GHOST STATUS :</b><br><i>ghost code</i><br><i>physical code</i>   |
| $\tau ::=$         | $\text{unit}$<br>$\tau^{\mathfrak{B}} \rightarrow \tau$   | <b>TYPES:</b><br><i>unit type</i><br><i>function type</i>  |

## Evaluation

$$(\lambda x_\tau^b.t)v \xrightarrow{\epsilon} t[x_\tau^b \leftarrow v] \quad (\text{E-APPFUN})$$

$$\text{ghost } t \xrightarrow{\epsilon} t \quad (\text{E-DEGHOST})$$

$$\frac{t_1 \rightarrow t'_1}{t_1 t_2 \rightarrow t'_1 t_2} \quad (\text{T-APPLEFT})$$

$$\frac{t_2 \rightarrow t'_2}{v_1 t_2 \rightarrow v_1 t'_2} \quad (\text{T-APPLEFT})$$

Figure 1: *ghost*- $\lambda$  syntax and semantics

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

*Proof.* Straightforward from definition of the typing relation.  $\square$

*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  $g\lambda$ -calculus to standard simply typed  $\lambda$ -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 :

$$\begin{aligned}\mathcal{E}_\top(\tau) &= \text{unit} \\ \mathcal{E}_\perp(\text{unit}) &= \text{unit} \\ \mathcal{E}_\perp(\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1) &= \mathcal{E}_{\mathfrak{B}_2}(\tau_2) \rightarrow \mathcal{E}_\perp(\tau_1).\end{aligned}$$

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

$$\begin{aligned}\mathcal{E}_\top(t) &= () \\ \mathcal{E}_\perp(()) &= () \\ \mathcal{E}_\perp(x_\tau^\perp) &= x_{\mathcal{E}_\perp(\tau)} \\ \mathcal{E}_\perp(\lambda x_{\tau_2^{\mathfrak{B}_2}}.t) &= \lambda x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)}. \mathcal{E}_\perp(t) \\ \mathcal{E}_\perp(t_1 t_2) &= \mathcal{E}_\perp(t_1) \mathcal{E}_{\mathfrak{B}_2}(t_2) \quad \text{where } \vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}_2).\end{aligned}$$

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

### 1.4 Properties of ghost erasure

Now that we defined the erasure-translation of  $g\lambda$ -calculus to  $\lambda$ -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  $\vdash_{g\lambda} t_1 : (\tau_1, \perp)$  and  $\vdash_{g\lambda} v_2 : (\tau_2, \mathfrak{B}_2)$  hold,*

*then  $\mathcal{E}_\perp(t_1[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) = \mathcal{E}_\perp(t_1)[x_{\mathcal{E}_\perp(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)]$*

*Proof.* By induction on the structure of  $t_1$ .

*Case  $t_1 = x_{\tau_2}^{\mathfrak{B}_2}$ :*

In that case, we can deduce that  $\mathfrak{B}_2 = \perp$ . Therefore:

$$\begin{aligned} \mathcal{E}_\perp(x_{\tau_2}^\perp[x_{\tau_2}^\perp \leftarrow v_2]) &= \mathcal{E}_\perp(v_2) = x_{\mathcal{E}_\perp(\tau_2)}[x_{\mathcal{E}_\perp(\tau_2)} \leftarrow \mathcal{E}_\perp(v_2)] \\ &= \mathcal{E}_\perp(x_{\tau_2}^\perp)[x_{\mathcal{E}_\perp(\tau_2)} \leftarrow \mathcal{E}_\perp(v_2)] \end{aligned}$$

*Case  $t_1 = y_{\tau'_2}^\perp \neq x_{\tau_2}^{\mathfrak{B}_2}$ :*

$$\begin{aligned} \mathcal{E}_\perp(y_{\tau'_2}^\perp[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) &= \mathcal{E}_\perp(y_{\tau'_2}^\perp) = y_{\mathcal{E}_\perp(\tau'_2)} \\ &= y_{\mathcal{E}_\perp(\tau'_2)}[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)] = \mathcal{E}_\perp(y_{\tau'_2}^\perp)[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)] \end{aligned}$$

*Case  $t_1 = \lambda y_{\tau'_2}^{\mathfrak{B}'_2}.t_{11}$  with  $y_{\tau'_2}^{\mathfrak{B}'_2} \notin \text{FV}(v_2)$  and  $\neq x_{\tau_2}^{\mathfrak{B}_2}$ :*

$$\begin{aligned} \mathcal{E}_\perp((\lambda y_{\tau'_2}^{\mathfrak{B}'_2}.t_{11})[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) &= \mathcal{E}_\perp[\lambda y_{\tau'_2}^{\mathfrak{B}'_2}.(t_{11}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2])] \\ &= \lambda y_{\mathcal{E}_{\mathfrak{B}'_2}(\tau'_2)}. \mathcal{E}_\perp(t_{11}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) \\ &\stackrel{\text{Ind.Hyp.}}{=} \lambda y_{\mathcal{E}_{\mathfrak{B}'_2}(\tau'_2)}. \mathcal{E}_\perp(t_{11})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)] \\ &= (\lambda y_{\mathcal{E}_{\mathfrak{B}'_2}(\tau'_2)}. \mathcal{E}_\perp(t_{11}))[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)] \\ &= \mathcal{E}_\perp(\lambda y_{\tau'_2}^{\mathfrak{B}'_2}.t_{11})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)] \end{aligned}$$

*Case  $t_1 = t_{11}t_{12}$*

$$\begin{aligned} \mathcal{E}_\perp(t_{11}t_{12}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) &= \mathcal{E}_\perp(t_{11}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2] \ t_{12}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) \\ &= \mathcal{E}_\perp(t_{11}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) \ \mathcal{E}_\perp(t_{12}[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_2]) \\ &\stackrel{\text{Ind.Hyp.}}{=} (\mathcal{E}_\perp(t_{11})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)])(\mathcal{E}_\perp(t_{12})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)]) \\ &= \mathcal{E}_\perp(t_{11}t_{12})[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}_2}(v_2)] \end{aligned}$$

□

*Lemma 1.5 [ONE-STEP EVALUATION UNDER ERASURE]. For any closed  $g\lambda$ -term  $t$  such that  $\vdash_{g\lambda} t : (\tau, \perp)$  holds, if  $t \rightarrow_{g\lambda} t'$  for some term  $t'$ , then either  $\mathcal{E}_\perp(t) \rightarrow_\lambda \mathcal{E}_\perp(t')$  or  $\mathcal{E}_\perp(t) = \mathcal{E}_\perp(t')$ .*

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

*Case E-APPABS:*  $t = (\lambda x_{\tau_2}^{\mathfrak{B}_2}. t_1) v_1$  with  $(\lambda x_{\tau_2}^{\mathfrak{B}_2}. t_1) v_1 \xrightarrow{e}_{g\lambda} t_1[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_1]$   
 $\vdash_{g\lambda} (\lambda x_{\tau_2}^{\mathfrak{B}_2}. t_1) v_1 : (\tau_1, \mathfrak{B}_1), \quad \vdash_{g\lambda} v_1 : (\tau_2, \mathfrak{B}'_2),$   
 $\mathfrak{B}_1 = \perp, \quad \models \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}'_2$

$$\begin{aligned} & \mathcal{E}_\perp[(\lambda x_{\tau_2}^{\mathfrak{B}_2}. t_1) v_1] \\ &= \lambda x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)}. \mathcal{E}_\perp(t_1) \mathcal{E}_{\mathfrak{B}'_2}(v_1) \quad (\text{as } \mathfrak{B}_1 = \perp) \\ &\xrightarrow{e}_\lambda \mathcal{E}_\perp(t_1)[x_{\mathcal{E}_{\mathfrak{B}_2}(\tau_2)} \leftarrow \mathcal{E}_{\mathfrak{B}'_2}(v_1)] \quad (\text{head red.}) \\ &= \mathcal{E}_\perp(t_1[x_{\tau_2}^{\mathfrak{B}_2} \leftarrow v_1]) \quad (\text{by Substitution under erasure lemma}) \end{aligned}$$

*Case E-DEGHOST:*

Trivially verified, as for any instance of  $\vdash_{g\lambda} ghost t_1 : (\tau_1, \mathfrak{B}_1), \mathfrak{B}_1 = \top$ .

*Case E-APPLEFT:*  $t = t_1 t_2, \quad t' = t_1 t'_2$ , with  $t_1 \rightarrow_{g\lambda} t'_1$   
 $\vdash_{g\lambda} t_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1), \quad \vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}'_2),$   
 $\mathfrak{B}_1 = \perp, \quad \models \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}'_2$

As  $\mathfrak{B}_1 = \perp$ , we can apply induction hypothesis on  $t_1$  which gives  $\mathcal{E}_\perp(t_1) \rightarrow_\lambda \mathcal{E}_\perp(t'_1)$ . Then, applying E-APPRIGHT rule, we obtain:

$$\mathcal{E}_\perp(t) = \mathcal{E}_\perp(t_1) \mathcal{E}_\perp(t_2) \rightarrow_\lambda \mathcal{E}_\perp(t'_1) \mathcal{E}_\perp(t_2) = \mathcal{E}_\perp(t').$$

*Case E-APPRIGHT:*  $t = v_1 t_2, \quad t' = v_1 t'_2$ , with  $t_2 \rightarrow_{g\lambda} t'_2$   
 $\vdash_{g\lambda} v_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1), \quad \vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}'_2),$   
 $\mathfrak{B}_1 = \perp, \quad \models \mathfrak{B}_2 \Leftrightarrow \mathfrak{B}'_2$

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

$$\mathcal{E}_\perp(t) = \mathcal{E}_\perp(v_1) \mathcal{E}_\top(t_2) = (\mathcal{E}_\perp(v_1))() = \mathcal{E}_\perp(v_1) \mathcal{E}_\top(t'_2) = \mathcal{E}_\perp(t').$$

Otherwise,  $\mathfrak{B}_2 = \mathfrak{B}'_2 = \perp$ . By induction hypothesis,  $\mathcal{E}_\perp(t_2) \rightarrow_\lambda \mathcal{E}_\perp(t'_2)$ . Then, applying E-APPRIGHT rule, we obtain:

$$\mathcal{E}_\perp(t) = \mathcal{E}_\perp(v_1) \mathcal{E}_\perp(t_2) \rightarrow_\lambda \mathcal{E}_\perp(v_1) \mathcal{E}_\perp(t'_2) = \mathcal{E}_\perp(t').$$

□

Now we can prove the main theorem.

**Theorem 1.4 [VALUE PRESERVATION UNDER ERASURE].** *For any closed  $g\lambda$ -term  $t$  such that  $\vdash_{g\lambda} t : (\tau, \perp)$  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  $t \rightarrow_{g\lambda}^* v$ .

We already have proved the base case : indeed, if  $t \rightarrow_{g\lambda} v$  then by the one-step evaluation lemma,  $\mathcal{E}_\perp(t) \rightarrow_\lambda^{01} \mathcal{E}_\perp(v)$ .

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

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

*Case T-UNIT:*  $\vdash_{g\lambda} () : (\text{unit}, \perp)$

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

*Case T-VAR:*  $\vdash_{g\lambda} x_{\tau}^{\perp} : (\tau, \perp)$

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

*Case T-ABS:*  $\vdash_{g\lambda} \lambda x_{\tau_2}^{\mathfrak{B}_2}. t_1 : (\tau_2^{\perp} \rightarrow \tau_1, \perp)$  with  $\vdash_{g\lambda} t_1 : (\tau_1, \perp)$

By induction hypothesis  $\vdash_{\lambda} \mathcal{E}_{\perp}(t_1) : \mathcal{E}_{\perp}(\tau_1)$ . There are two cases to consider, depending on whether the parameter of the abstraction is ghost or not. If  $\mathfrak{B}_2 = \top$  then  $\mathcal{E}_{\perp}(\lambda x_{\tau_2}^{\top}. t_1) = \lambda x_{\text{unit}}. \mathcal{E}(t_1)$  and therefore

$$\frac{\vdash_{\lambda} \mathcal{E}_{\perp}(t_1) : \mathcal{E}_{\perp}(\tau_2)}{\vdash_{\lambda} \lambda x_{\text{unit}}. \mathcal{E}_{\perp}(t_1) : \text{unit} \rightarrow \mathcal{E}_{\perp}(\tau_1)} \quad (\text{T-ABS})$$

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

$$\frac{\vdash_{\lambda} \mathcal{E}_{\perp}(t_1) : \mathcal{E}_{\perp}(\tau_1)}{\vdash_{\lambda} \lambda x_{\mathcal{E}_{\perp}(\tau_2)}. \mathcal{E}_{\perp}(t_1) : \mathcal{E}_{\perp}(\tau_2) \rightarrow \mathcal{E}_{\perp}(\tau_1)} \quad (\text{T-ABS})$$

*Case T-APP:*  $\vdash_{g\lambda} t_1 t_2 : (\tau_1, \perp)$  with sub-derivations:

$$\vdash_{g\lambda} t_1 : (\tau_2^{\mathfrak{B}_2} \rightarrow \tau_1, \mathfrak{B}_1)$$

$$\vdash_{g\lambda} t_2 : (\tau_2, \mathfrak{B}'_2),$$

As  $\vdash_{g\lambda} t_1 t_2 : (\tau_1, \perp)$ , the inversion lemma gives as By inversion that  $\mathfrak{B}_2 \Leftrightarrow \mathfrak{B}'_2$ . That is, we have two cases to consider.

If  $\mathfrak{B}_2 = \mathfrak{B}'_2 = \perp$  then by induction hypotheses  $\vdash_{\lambda} \mathcal{E}_{\perp}(t_1) : \mathcal{E}_{\perp}(\tau_2) \rightarrow \mathcal{E}_{\perp}(\tau_1)$  and  $\vdash_{\lambda} \mathcal{E}_{\perp}(t_2) : \mathcal{E}_{\perp}(\tau_2)$ . By T-APP rule,

$$\frac{\vdash_{\lambda} \mathcal{E}_{\perp}(t_1) : \mathcal{E}_{\perp}(\tau_2) \rightarrow \mathcal{E}_{\perp}(\tau_1) \quad \vdash_{\lambda} \mathcal{E}_{\perp}(t_2) : \mathcal{E}_{\perp}(\tau_2)}{\vdash_{\lambda} \mathcal{E}_{\perp}(t_1 t_2) : \mathcal{E}(\tau_1)} \quad (\text{T-APP})$$

If  $\mathfrak{B}_2 = \mathfrak{B}'_2 = \top$ , then by definition of  $\mathcal{E}$  we have  $\mathcal{E}(\tau_2) = ()$  and  $\mathcal{E}_{\mathfrak{B}'_2}(\tau_2) = \text{unit}$ . By induction hypothesis on  $t_1$ ,  $\vdash_{\lambda} \mathcal{E}_{\perp}(t_1) : \text{unit} \rightarrow \mathcal{E}_{\perp}(\tau_1)$ .

Applying T-APP rule gives us

$$\frac{\frac{}{\vdash_{\lambda} \mathcal{E}_{\perp}(t_1 \ t_2) : \mathcal{E}(\tau_1)} \quad \frac{}{\vdash_{\lambda} () : \text{unit}}}{\vdash_{\lambda} \mathcal{E}_{\perp}(t_1 \ t_2) : \mathcal{E}_{\perp}(\tau_1)} \begin{array}{l} \text{(T-UNIT)} \\ \text{(T-APP)} \end{array}$$

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.  $\square$

TODO: Extensions: - If-then-else - rec - let in - match - constructors -  
ref global mono - types polymorphes - operators

Potential difficulties: - polymorphism - schemes - non interference

$\rightarrow(E, E')$

Angles de typage Angles d'effacement