A short note explaining the bug in the APAL term assignment formulation of FILL

Harley Eades III

August 2014

In this short note I give the details of Bierman's counterexample [1] to cut elimination of the term assignment formulation of FILL first given in [4]. I first reformulate his counterexample into our definition of FILL, and then comment on the reason for the counterexample. Following this I reformulate the counterexample in the dependency tracking system proposed by Braüner and de Paiva in [2] and revised by the same authors in [3]. In this reformulation we will see that the rule proposed by Bellin but communicated by Bierman in [1] is the proper left rule for par.

1 The Parl Inference Rule

The existing Parl inference rule is as follows:

$$\frac{\Gamma, x : A \vdash d_i : C_i \quad \Gamma', y : B \vdash f_j : D_j}{\Gamma, \Gamma', z : A \parr B \vdash \mathsf{let}\, z\, \mathsf{be}\, x \parr - \,\mathsf{in}\, d_i : C_i \mid \mathsf{let}\, z\, \mathsf{be}\, - \parr y\, \mathsf{in}\, f_j : D_j}\ \textsc{Parl}$$

In the terms $\mathsf{let}\, t_1\, \mathsf{be}\, x \parr * \,\mathsf{in}\, t_2$ and $\mathsf{let}\, t_1\, \mathsf{be}\, * \parr y\, \mathsf{in}\, t_2$ the variables x and y in the patterns are bound in t_2. So when applying the Parl rule we bind both the free variable x in d_i, and the free variable y in each f_i. Now notice that we do this binding even when the variables are not free in the respective terms. Furthermore, as a result of binding these pattern variables we carry along the newly introduced free variable z. It is this global binding across the entire righthand size context along with introducing the free variable z in each term that results in the counterexample of Bierman.

2 Bierman's Counterexample

First lets recall the cut-elimination commuting conversion that is the locus of the counterexample. The following cut:

$$\frac{\Gamma \vdash d : C \mid g_k : D \qquad \dfrac{x : C, w : A, \Gamma' \vdash e_i : D' \qquad z : B, \Gamma'' \vdash t_i : D''}{x : C, y : A \parr B, \Gamma', \Gamma'' \vdash \mathsf{let}\, y\, \mathsf{be}\, w \parr * \,\mathsf{in}\, e_i : D' \mid \mathsf{let}\, y\, \mathsf{be}\, * \parr z\, \mathsf{in}\, t_i : D''}\ \textsc{Parl}}{\Gamma, y : A \parr B, \Gamma', \Gamma'' \vdash [d/x](\mathsf{let}\, y\, \mathsf{be}\, w \parr * \,\mathsf{in}\, e_i) : D' \mid [d/x](\mathsf{let}\, y\, \mathsf{be}\, * \parr z\, \mathsf{in}\, t_j) : D'' \mid g_k : D}\ \textsc{Cut}$$

Converts into the following:

$$\frac{\dfrac{\Gamma \vdash d : C \mid g_k : D \qquad x : C, w : A, \Gamma' \vdash e_i : D'}{w : A, \Gamma' \vdash [d/x]e_i : D'}\ \textsc{Cut} \qquad z : B, \Gamma'' \vdash t_i : D''}{\Gamma, y : A \parr B, \Gamma', \Gamma'' \vdash \mathsf{let}\, y\, \mathsf{be}\, w \parr * \,\mathsf{in}\, [d/x]e_i : D' \mid \mathsf{let}\, y\, \mathsf{be}\, * \parr z\, \mathsf{in}\, t_j : D'' \mid \mathsf{let}\, y\, \mathsf{be}\, w \parr * \,\mathsf{in}\, g_k : D}\ \textsc{Parl}$$

Notice that in the above cut, the Parl rule commutes with Cut. So again, we bind w as a pattern variable in each e_i, and z in each t_i regardless of whether or not these are actually free in any of the terms. In addition, we introduce z into each of these terms.

Next we give Bierman's counterexample. The following uses the first rule given above.

$$
\dfrac{\dfrac{}{v : A \vdash v : A}\;{\scriptstyle\textsc{Ax}}}{v : A \vdash v : A \mid \circ : \perp}\;{\scriptstyle\textsc{Pr}}
\qquad
\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\;{\scriptstyle\textsc{Ax}} \quad \dfrac{}{y : B \vdash y : B}\;{\scriptstyle\textsc{Ax}}}{x : A, y : B \vdash x \otimes y : A \otimes B}\;{\scriptstyle\textsc{Tr}} \quad \dfrac{}{w : C \vdash w : C}\;{\scriptstyle\textsc{Ax}}}{x : A, z : B \parr C \vdash \mathsf{let}\ z\ \mathsf{be}\ y \parr - \ \mathsf{in}\ v \otimes y : A \otimes B \mid \mathsf{let}\ z\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w : C}\;{\scriptstyle\textsc{Parl}}
$$

$$
\dfrac{\dfrac{\text{(above)}}{v : A, z : B \parr C \vdash \mathsf{let}\ z\ \mathsf{be}\ y \parr -\ \mathsf{in}\ v \otimes y : A \otimes B \mid \mathsf{let}\ z\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w : C \mid \circ : \perp}\;{\scriptstyle\textsc{Cut}}}{\dfrac{v : A, z : B \parr C \vdash (\mathsf{let}\ z\ \mathsf{be}\ y \parr -\ \mathsf{in}\ v \otimes y) \parr (\mathsf{let}\ z\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w) : ((A \otimes B) \parr C) \mid \circ : \perp}{v : A \vdash \lambda z.((\mathsf{let}\ z\ \mathsf{be}\ y \parr -\ \mathsf{in}\ v \otimes y) \parr (\mathsf{let}\ z\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w)) : (B \parr C) \multimap ((A \otimes B) \parr C) \mid \circ : \perp}\;{\scriptstyle\textsc{Impr}}}\;{\scriptstyle\textsc{Parr}}
$$

Next we use the second derived rule above to commute the cut in the previous derivation past the Parl rule:

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{v : A \vdash v : A}\;{\scriptstyle\textsc{Ax}}}{v : A \vdash v : A \mid \circ : \perp}\;{\scriptstyle\textsc{Pr}} \quad \dfrac{\dfrac{}{x : A \vdash x : A}\;{\scriptstyle\textsc{Ax}} \quad \dfrac{}{y : B \vdash y : B}\;{\scriptstyle\textsc{Ax}}}{x : A, y : B \vdash x \otimes y : A \otimes B}\;{\scriptstyle\textsc{Tr}}}{y : B, v : A \vdash v \otimes y : A \otimes B \mid \circ : \perp}\;{\scriptstyle\textsc{Cut}} \quad \dfrac{}{w : C \vdash w : C}\;{\scriptstyle\textsc{Ax}}}{v : A, {\color{blue}z} : B \parr C \vdash \mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ y \parr -\ \mathsf{in}\ v \otimes y : A \otimes B \mid \mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w : C \mid \mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ y \parr -\ \mathsf{in}\ \circ : \perp}\;{\scriptstyle\textsc{Parl}}
$$

$$
\dfrac{\dfrac{\text{(above)}}{v : A, {\color{blue}z} : B \parr C \vdash ((\mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ y \parr -\ \mathsf{in}\ v \otimes y) \parr (\mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w)) : (A \otimes B) \parr C \mid \mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ y \parr -\ \mathsf{in}\ \circ : \perp}\;{\scriptstyle\textsc{Parr}}}{v : A \vdash \lambda z.((\mathsf{let}\ z\ \mathsf{be}\ y \parr -\ \mathsf{in}\ v \otimes y) \parr (\mathsf{let}\ z\ \mathsf{be}\ - \parr w\ \mathsf{in}\ w)) : (B \parr C) \multimap ((A \otimes B) \parr C) \mid \mathsf{let}\ {\color{blue}z}\ \mathsf{be}\ y \parr -\ \mathsf{in}\ \circ : \perp}\;{\scriptstyle\textsc{Impr}}
$$

Now notice that as a result of the rule Parl rule a fresh free variable z – colored blue when it is considered free – is introduced, and then let-bound in every term in the righthand side context. Furthermore, we bind y and w in terms which do not depend on them, for example, we bind y in $\circ$. Furthermore, we introduce z into each of these terms, especially, the rightmost term. Thus, the application of the Impr rule is in error, because z occurs in the right most term.

3 Bierman's Counterexample in the Dependency-Relation Formalization

Next we give Bierman's counterexample in the dependency-relation formalization. To obtain the derivations we simply erase all the terms:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\;{\scriptstyle\textsc{Dax}}}{A \vdash A \mid \perp}\;{\scriptstyle\textsc{Dpr}} \quad \dfrac{\dfrac{\dfrac{}{A \vdash A}\;{\scriptstyle\textsc{Dax}} \quad \dfrac{}{B \vdash B}\;{\scriptstyle\textsc{Dax}}}{A, B \vdash A \otimes B}\;{\scriptstyle\textsc{Dtr}} \quad \dfrac{}{C \vdash C}\;{\scriptstyle\textsc{Dax}}}{A, (B \parr C) \vdash (A \otimes B) \mid C}\;{\scriptstyle\textsc{Dparl}}}{A, B \parr C \vdash A \otimes B \mid C \mid \perp}\;{\scriptstyle\textsc{Dcut}}}{A, B \parr C \vdash ((A \otimes B) \parr C) \mid \perp}\;{\scriptstyle\textsc{Dparr}}}{A \vdash (B \parr C) \multimap ((A \otimes B) \parr C) \mid \perp}\;{\scriptstyle\textsc{Dimpr}}
$$

Now if we compute the dependencies of the previous derivation up to the application of the Dimpr rule we will see that the only formula on the righthand side of the premise of the Dimpr rule with dependencies is $((A \otimes B) \parr C)$. The set of dependencies is as follows:

$$
\{(B \parr C, (A \otimes B) \parr C), (A, (A \otimes B) \parr C), (A', A''), (B', B''), (C', C'')\}
$$

We can easily see that $\perp$ has no dependencies and rightfully so. If we commute the cut just as before we obtain the following derivation:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\;{\scriptstyle\textsc{Dax}}}{A \vdash A \mid \perp}\;{\scriptstyle\textsc{Dpr}} \quad \dfrac{\dfrac{}{A \vdash A}\;{\scriptstyle\textsc{Dax}} \quad \dfrac{}{B \vdash B}\;{\scriptstyle\textsc{Dax}}}{A, B \vdash A \otimes B}\;{\scriptstyle\textsc{Dtr}}}{B, A \vdash A \otimes B \mid \perp}\;{\scriptstyle\textsc{Dcut}} \quad \dfrac{}{C \vdash C}\;{\scriptstyle\textsc{Dax}}}{A, B \parr C \vdash A \otimes B \mid C \mid \perp}\;{\scriptstyle\textsc{Dparl}}}{A, B \parr C \vdash (A \otimes B) \parr C \mid \perp}\;{\scriptstyle\textsc{Dparr}}}{A \vdash (B \parr C) \multimap ((A \otimes B) \parr C) \mid \perp}\;{\scriptstyle\textsc{Dimpr}}
$$

As we can see without the terms, this derivation is nearly identical to the previous, and hence, we obtain the exact same dependency set. However, we can use the dependency-relation formalization as a guiding principle in fixing the term formalization.

4 The Fix

Consider the DPARL rule in the dependency-relation formalization:

$$
\dfrac{\begin{array}{c}\Gamma_1, A \vdash \Delta_1 \\ \Gamma_3, B \vdash \Delta_2\end{array}}{\Gamma_1, \Gamma_3, A \parr B \vdash \Delta_1, \Delta_2}\ \textsc{Dparl} \qquad \mathit{Dep}(\tau) = \{(A \parr B, A), (A \parr B, B)\} \star (\mathit{Dep}(\tau_1) \cup \mathit{Dep}(\tau_2))
$$

If anything in Δ_1 and Δ_2 depend on A or B then this will be accounted for in $\mathit{Dep}(\tau_1)$ and $\mathit{Dep}(\tau_2)$. Thus, in the term formalization when binding pattern variables across the righthand side of the sequent we should do so if and only if there is a dependency. In fact, if a formula on the righthand side depends on a formula in the lefthand side, then the variable associated with that hypnosis must be free in the term associated with the formula on the right. This evidence suggests that to fix the term formalization we must modify the PARL rule.

The new PARL rule as follows:

$$
\dfrac{\Gamma, x : A \vdash d_i : C_i \quad \Gamma', y : B \vdash f_j : D_j}{\Gamma, \Gamma', z : A \parr B \vdash \text{let-pat}\, z\, (x \parr -)\, d_i : C_i \mid \text{let-pat}\, z\, (- \parr y)\, f_j : D_j}\ \textsc{NParl}
$$

The previous rule depends on a function which we define as follows:

$$
\begin{array}{l}
\text{let-pat}\, z\, (x \parr -)\, e = e \\
\quad \text{where } x \notin \mathsf{FV}(e) \\[1em]
\text{let-pat}\, z\, (- \parr y)\, e = e \\
\quad \text{where } y \notin \mathsf{FV}(e) \\[1em]
\text{let-pat}\, z\, p\, e = \mathsf{let}\, z\, \mathsf{be}\, p\, \mathsf{in}\, e
\end{array}
$$

Note that in the definition of let-pat $z\, p\, e$ the final case is a catchall case. Now the new PARL rule only pattern matches on z in the righthand side if there is a dependency between the variables in the pattern and the term in the pattern match. A similar rule to the above was proposed by Bellin in the conclusion of [1].

This rule recovers from the counterexample. The first derivation given in the counter example above is unchanged, so we only give the second:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v : A \vdash v : A}\,\textsc{Ax}}{v : A \vdash v : A \mid \circ : \perp}\,\textsc{Pr} \quad \dfrac{\dfrac{}{x : A \vdash x : A}\,\textsc{Ax} \quad \dfrac{}{y : B \vdash y : B}\,\textsc{Ax}}{x : A, y : B \vdash x \otimes y : A \otimes B}\,\textsc{Tr}}{y : B, v : A \vdash v \otimes y : A \otimes B \mid \circ : \perp}\,\textsc{Cut} \quad \dfrac{}{w : C \vdash w : C}\,\textsc{Ax}}{v : A, z : B \parr C \vdash \mathsf{let}\, z\, \mathsf{be}\, y \parr -\, \mathsf{in}\, v \otimes y : A \otimes B \mid \mathsf{let}\, z\, \mathsf{be}\, - \parr w\, \mathsf{in}\, w : C \mid \circ : \perp}\,\textsc{Parl}}{v : A, z : B \parr C \vdash ((\mathsf{let}\, z\, \mathsf{be}\, y \parr -\, \mathsf{in}\, v \otimes y) \parr (\mathsf{let}\, z\, \mathsf{be}\, - \parr w\, \mathsf{in}\, w)) : (A \otimes B) \parr C \mid \circ : \perp}\,\textsc{Parr}}{v : A \vdash \lambda z.((\mathsf{let}\, z\, \mathsf{be}\, y \parr -\, \mathsf{in}\, v \otimes y) \parr (\mathsf{let}\, z\, \mathsf{be}\, - \parr w\, \mathsf{in}\, w)) : (B \parr C) \multimap ((A \otimes B) \parr C) \mid \circ : \perp}\,\textsc{Impr}
$$

This new derivation is now correct, and mirrors that of the dependency-relation formalization.

References

[1] G.M. Bierman. A note on full intuitionistic linear logic. *Annals of Pure and Applied Logic*, 79(3):281 – 287, 1996.

[2] Torben Brauner and Valeria Paiva. Cut-elimation for full intuitionistic linear logic. BRICS 395, Computer Laboratory, University of Cambridge, 1996.

[3] Torben Brauner and Valeria Paiva. A formulation of linear logic based on dependency-relations. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, pages 129–148. Springer Berlin Heidelberg, 1998.

[4] Martin Hyland and Valeria de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, 64(3):273 – 291, 1993.

A The full specification of FILL

$\mathit{term\_var}, w, x, y, z, v$			
$\mathit{index\_var}, i, j, k$			
$\mathit{form}, A, B, C, D, E$	$::=$		
	$\mid$	I	
	$\mid$	$\perp$	
	$\mid$	$A \multimap B$	
	$\mid$	$A \otimes B$	
	$\mid$	$A \parr B$	
	$\mid$	(A)	S
$\mathit{patterns}, p$	$::=$		
	$\mid$	$*$	
	$\mid$	x	
	$\mid$	$p_1 \otimes p_2$	
	$\mid$	$p_1 \parr p_2$	
	$\mid$	$-$	
	$\mid$	(p)	S
$\mathit{term}, t, e, d, f, g, u$	$::=$		
	$\mid$	x	
	$\mid$	$*$	
	$\mid$	$\circ$	
	$\mid$	$e_1 \otimes e_2$	
	$\mid$	$e_1 \parr e_2$	
	$\mid$	$\lambda x.t$	
	$\mid$	let t be p in e	
	$\mid$	$f\ e$	
	$\mid$	let-pat $z\ p\ e$	M
	$\mid$	$[t/x]t'$	M
	$\mid$	$[t/x, e/y]t'$	M
	$\mid$	(t)	S
	$\mid$	t	M
	$\mid$	t	M
Γ	$::=$		
	$\mid$	$x : A$	
	$\mid$	$\cdot$	

	$\vert$	Γ, Γ'
	$\vert$	$x : A$
	$\vert$	A
Δ	$::=$	
	$\vert$	$t : A$
	$\vert$	$\cdot$
	$\vert$	$\Delta \mid \Delta'$
	$\vert$	Δ
	$\vert$	A
	$\vert$	Δ, Δ'
formula	$::=$	
	$\vert$	*judgement*
	$\vert$	$formula_1 \quad formula_2$
	$\vert$	$(formula)$
	$\vert$	$x \notin \mathsf{FV}(\Delta)$
InferRules	$::=$	
	$\vert$	$\Gamma \vdash \Delta$
	$\vert$	$f = e$
judgement	$::=$	
	$\vert$	*InferRules*
user_syntax	$::=$	
	$\vert$	*term_var*
	$\vert$	*index_var*
	$\vert$	*form*
	$\vert$	*patterns*
	$\vert$	*term*
	$\vert$	Γ
	$\vert$	Δ
	$\vert$	*formula*

$\boxed{\Gamma \vdash \Delta}$

$$\frac{}{x : A \vdash x : A} \quad \textsc{Ax}$$

$$\frac{\Gamma \vdash t : A \mid \Delta \quad y : A, \Gamma' \vdash f_i : B_i}{\Gamma, \Gamma' \vdash \Delta \mid [t/y]f_i : B_i} \quad \textsc{Cut}$$

$$\frac{\Gamma \vdash e_i : A_i}{\Gamma, x : I \vdash \mathsf{let}\, x\, \mathsf{be}\, * \,\mathsf{in}\, e_i : A_i} \quad \textsc{Il}$$

$$\frac{}{\cdot \vdash * : I} \quad \textsc{Ir}$$

$$\frac{\Gamma, x : A, y : B \vdash f_i : C_i}{\Gamma, z : A \otimes B \vdash \mathsf{let}\, x\, \mathsf{be}\, x \otimes y\, \mathsf{in}\, f_i : C_i} \quad \textsc{Tl}$$

$$\frac{\Gamma \vdash e : A \mid \Delta \quad \Gamma' \vdash f : B \mid \Delta'}{\Gamma, \Gamma' \vdash e \otimes f : A \otimes B \mid \Delta \mid \Delta'} \quad \textsc{Tr}$$

$$\frac{}{x :\perp\vdash \cdot}\ \textsc{Pl}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \circ :\perp \mid \Delta}\ \textsc{Pr}$$

$$\frac{\Gamma, x : A \vdash d_i : C_i \quad \Gamma', y : B \vdash f_j : D_j}{\Gamma, \Gamma', z : A \parr B \vdash \mathsf{let}\, z\, \mathsf{be}\, x \parr - \,\mathsf{in}\, d_i : C_i \mid \mathsf{let}\, z\, \mathsf{be}\, - \parr y\, \mathsf{in}\, f_j : D_j}\ \textsc{ParL}$$

$$\frac{\Gamma, x : A \vdash d_i : C_i \quad \Gamma', y : B \vdash f_j : D_j}{\Gamma, \Gamma', z : A \parr B \vdash \text{let-pat}\, z\, (x \parr -)\, d_i : C_i \mid \text{let-pat}\, z\, (- \parr y)\, f_j : D_j}\ \text{N}\textsc{ParL}$$

$$\frac{\Gamma \vdash \Delta \mid e : A \mid f : B \mid \Delta'}{\Gamma \vdash \Delta \mid e \parr f : A \parr B \mid \Delta'}\ \textsc{ParR}$$

$$\frac{\Gamma \vdash e : A \mid \Delta \quad \Gamma', x : B \vdash f_i : C_i}{\Gamma, y : A \multimap B, \Gamma' \vdash [y\, e/x] f_i : C_i \mid \Delta}\ \textsc{ImpL}$$

$$\frac{\Gamma, x : A \vdash e : B \mid \Delta \quad x \notin \mathsf{FV}(\Delta)}{\Gamma \vdash \lambda x.e : A \multimap B \mid \Delta}\ \textsc{ImpR}$$

$$\frac{}{A' \vdash A''}\ \textsc{dax}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash B', \Delta_1 \\ \Gamma_2, B'' \vdash \Delta_2\end{array}}{\Gamma_1, \Gamma_2 \vdash \Delta_2, \Delta_1}\ \textsc{dcut}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta}\ \textsc{DtL}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash A, \Delta_1 \\ \Gamma_2 \vdash B, \Delta_2\end{array}}{\Gamma_1, \Gamma_2 \vdash A \otimes B, \Delta_1, \Delta_2}\ \textsc{DtR}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, I \vdash \Delta}\ \textsc{DiL}$$

$$\frac{}{\cdot \vdash I}\ \textsc{DiR}$$

$$\frac{\begin{array}{c}\Gamma_1, A \vdash \Delta_1 \\ \Gamma_3, B \vdash \Delta_2\end{array}}{\Gamma_1, \Gamma_3, A \parr B \vdash \Delta_1, \Delta_2}\ \textsc{DparL}$$

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ \textsc{DparR}$$

$$\frac{}{\perp\vdash \cdot}\ \textsc{DpL}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \perp, \Delta}\ \textsc{DpR}$$

$$\frac{\begin{array}{c}\Gamma_1 \vdash A, \Delta_1 \\ \Gamma_2, B \vdash \Delta_2\end{array}}{\Gamma_1, \Gamma_2, A \multimap B \vdash \Delta_1, \Delta_2}\ \textsc{DimpL}$$

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \multimap B, \Delta}\ \textsc{DimpR}$$

$\boxed{f = e}$

$$\frac{}{(\lambda x.e)\ e' = [e'/x]e}\ \textsc{Eq\_Beta}$$

$$\frac{}{\lambda x.f\ x = f}\ \textsc{Eq\_Eta}$$

$$\frac{}{\mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ e = e}\ \textsc{Eq\_I}$$

$$\frac{}{\mathsf{let}\ u\ \mathsf{be}\ *\ \mathsf{in}\ [*/z]f = [u/z]f}\ \textsc{Eq\_Stp}$$

$$\frac{}{\mathsf{let}\ e \otimes t\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ u = [e/x, t/y]u}\ \textsc{Eq\_T1}$$

$$\frac{}{\mathsf{let}\ u\ \mathsf{be}\ x \otimes y\ \mathsf{in}\ [x \otimes y/z]f = [u/z]f}\ \textsc{Eq\_T2}$$

$$\frac{}{\mathsf{let}\ u \parr t\ \mathsf{be}\ x \parr -\ \mathsf{in}\ e = [u/x]e}\ \textsc{Eq\_P1}$$

$$\frac{}{\mathsf{let}\ u \parr t\ \mathsf{be}\ - \parr y\ \mathsf{in}\ e = [t/y]e}\ \textsc{Eq\_P2}$$

$$\frac{}{(\mathsf{let}\ x\ \mathsf{be}\ x \parr -\ \mathsf{in}\ x) \parr (\mathsf{let}\ u\ \mathsf{be}\ - \parr y\ \mathsf{in}\ y) = u}\ \textsc{Eq\_P3}$$