




Linearity and Uniqueness: An Entente Cordiale (appendix)

Daniel Marshall¹ , Michael Vollmer¹ , and Dominic Orchard^{1,2} 

¹ University of Kent, Canterbury, UK
{dm635,m.vollmer,d.a.orchard}@kent.ac.uk

² University of Cambridge, UK

Table of Contents

Linearity and Uniqueness: An Entente Cordiale (appendix)	1
<i>Daniel Marshall, Michael Vollmer, and Dominic Orchard</i>	
A Collected rules	1
A.1 Typing	1
A.2 Heap semantics	2
Runtime typing	2
Additional definitions	2
Multi-reduction rules	4
Single-reduction congruences	4
Single-reduction β -rules	5
B Proofs	5
B.1 Admissibility of substitution	5
B.2 Admissibility of weakening	12
B.3 Conservation	14
B.4 Uniqueness	22
B.5 Value lemma	34
B.6 Progress	36
B.7 Soundness of heap model wrt. equational theory	41

A Collected rules

A.1 Typing

Definition 1 (All non-linear assumptions). A context Γ is denoted as containing only non-linear assumptions by writing $[\Gamma]$ in proofs, where $[\emptyset]$ and $[\Gamma] \implies [\Gamma], x : [A]$.

Definition 2 (Context addition). The partial operation $+$ on contexts is the union of two contexts as long as they are disjoint in their linear assumptions and any variables occurring in both contexts are both non-linear assumptions, i.e.

$$\Gamma_1 + \Gamma_2 = \Gamma_1 \cup \Gamma_2 \text{ iff } \forall x \in \text{dom}(\Gamma_1) \cap \text{dom}(\Gamma_2) \implies \exists A. \Gamma_1(x) = \Gamma_2(x) = [A]$$

$$\begin{array}{c}
\frac{}{\overline{[I]}, x : A \vdash x : A} \text{VAR} \quad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \multimap B} \text{ABS} \quad \frac{\Gamma_1 \vdash t_1 : A \multimap B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1 t_2 : B} \text{APP} \\
\\
\frac{\Gamma_1 \vdash t_1 : A \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A \otimes B} \otimes_I \quad \frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : C} \otimes_E \\
\\
\frac{}{\overline{[I]} \vdash \text{unit} : 1} 1_I \quad \frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : B} 1_E \\
\\
\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B} \text{DER} \quad \frac{\overline{[I]} \vdash t : A}{\overline{[I]} \vdash !t : !A} !_I \quad \frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : B} !_E \\
\\
\frac{\Gamma \vdash t : *A}{\Gamma \vdash \&t : !A} \text{BORROW} \quad \frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : *A \vdash t_2 : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B} \text{COPY} \quad \frac{\emptyset \vdash t : A}{\overline{[I]} \vdash *t : *A} \text{NEC}
\end{array}$$

Primitives

$$\begin{array}{c}
\frac{}{\overline{[I]} \vdash \text{newArray} : \mathbb{N} \multimap *(Array \ \mathbb{F})} \text{new} \\
\\
\frac{}{\overline{[I]} \vdash \text{readArray} : *(Array \ \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(Array \ \mathbb{F})} \text{read} \\
\\
\frac{}{\overline{[I]} \vdash \text{writeArray} : *(Array \ \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(Array \ \mathbb{F})} \text{write} \\
\\
\frac{}{\overline{[I]} \vdash \text{deleteArray} : *(Array \ \mathbb{F}) \multimap 1} \text{del}
\end{array}$$

A.2 Heap semantics

Runtime typing In order to prove type preservation and progress, conservation, and uniqueness we need to weave in typing into the heap model so we also need a way of typing runtime array primitives (which extend the term syntax with an atom a). Contexts are extended to include their typing as:

$$\Gamma ::= \dots \mid \Gamma, a : Array \ A$$

and with typing rules:

$$\frac{}{\overline{[I]}, a : Array \ A \vdash a : Array \ A} \text{arr} \quad \frac{}{\overline{[I]}, a : Array \ A \vdash *a : *(Array \ A)} \text{*array}$$

Context operations are therefore also extended:

Definition A1 (All non-linear assumptions (runtime typing)). A context Γ is denoted as containing only non-linear assumptions by writing $[I]$ in the typing rules, which is defined inductively as

$$\frac{}{\overline{[\emptyset]}} \quad \frac{[I]}{\overline{[I]}, x : [A]} \quad \frac{[I]}{\overline{[I]}, a : Array \ A}$$

Additional definitions

Definition A2 (Reference counts in a heap). Given a heap H then $[H]_r$ converts all array references to have reference count r , defined:

$$\begin{aligned} [\emptyset]_r &= \emptyset \\ [(H, a \mapsto_{r'} \mathbf{arr})]_r &= ([H]_r), a \mapsto_r \mathbf{arr} \\ [(H, x \mapsto_{r'} t)]_r &= ([H]_r), x \mapsto_r t \end{aligned}$$

Note that variable assignments in the heap are ignored (although in the heap semantics this operation is applied only to heaps with just array references).

This can also be thought of as a predicate on the image of the function.

Definition A3 (Array references in a term). By induction:

$$\begin{array}{llll} \text{arrRefs}(x) & = \emptyset & \text{arrRefs}(\text{unit}) & = \emptyset \\ \text{arrRefs}(\lambda x. t) & = \text{arrRefs}(t) & \text{arrRefs}(a) & = \{a\} \\ \text{arrRefs}(t_1 \ t_2) & = \text{arrRefs}(t_1) \cup \text{arrRefs}(t_2) & \text{arrRefs}(*t) & = \text{arrRefs}(t) \\ \text{arrRefs}(!t) & = \text{arrRefs}(t) & \text{arrRefs}(\text{newArray}) & = \emptyset \\ \text{arrRefs}(\text{let } !x = t_1 \text{ in } t_2) & = \text{arrRefs}(t_1) \cup \text{arrRefs}(t_2) & \text{arrRefs}(\text{readArray}) & = \emptyset \\ \text{arrRefs}((t_1, t_2)) & = \text{arrRefs}(t_1) \cup \text{arrRefs}(t_2) & \text{arrRefs}(\text{writeArray}) & = \emptyset \\ \text{arrRefs}(\text{let } (x, y) = t_1 \text{ in } t_2) & = \text{arrRefs}(t_1) \cup \text{arrRefs}(t_2) & \text{arrRefs}(\text{deleteArray}) & = \emptyset \\ \text{arrRefs}(\&t) & = \text{arrRefs}(t) & & \\ \text{arrRefs}(\text{copy } t_1 \text{ as } x \text{ in } t_2) & = \text{arrRefs}(t_1) \cup \text{arrRefs}(t_2) & & \end{array}$$

Definition A4 (Heap copy). The heap copy $\text{copy}(H)$ copies all array references (ignoring variable assignments in the heap; this function is only ever called on sub-heaps with just array references) into fresh array references along with a substitution (return as a pair). Defined:

$$\begin{aligned} \text{copy}(\emptyset) &= (\emptyset, \emptyset) \\ \text{copy}(H, x \mapsto_r t) &= \text{copy}(H) \\ \text{copy}(H, a \mapsto_r \mathbf{arr}) &= ((H', a' \mapsto_1 \mathbf{arr}), \theta \cup \{a \mapsto a'\}) \\ &\text{where } (H', \theta) = \text{copy}(H) \ \wedge \ a' \notin \text{dom}(H) \end{aligned}$$

Definition 3 (Heap-context compatibility). A heap H is compatible with a typing context Γ if H contains assignments for every variable in the context and the typing contexts of the terms in the heap are also compatible with the heap. The relation is defined inductively as:

$$\begin{array}{c} \frac{H \bowtie \Gamma}{H, a \mapsto_r \mathbf{arr} \bowtie \Gamma, a : \text{Array } A} \text{REF} \quad \frac{H \bowtie (\Gamma_1 + \Gamma_2) \quad \Gamma_2 \vdash t : A \quad x \notin \text{dom}(H)}{(H, x \mapsto_r (\Gamma_2 \vdash t : A)) \bowtie (\Gamma_1, x : A)} \text{LIN} \\[10pt] \frac{}{\emptyset \bowtie \emptyset} \text{EMPTY} \quad \frac{H \bowtie (\Gamma_1 + [\Gamma_2]) \quad [\Gamma_2] \vdash t : A \quad x \notin \text{dom}(H)}{(H, x \mapsto_\omega ([\Gamma_2] \vdash t : A)) \bowtie (\Gamma_1, x : [A])} \omega \end{array}$$

Definition 4 (Usage context extraction). For a context Γ or heap H we can extract usage information denoted $\overline{\Gamma}$ or \overline{H} defined as:

$$\begin{aligned} \overline{\emptyset} &= \emptyset & \overline{(\Gamma, x : [A])} &= \overline{\Gamma}, x : \omega & \overline{(\Gamma, a : A)} &= \overline{\Gamma} & \overline{(\Gamma, x : A)} &= \overline{\Gamma}, x : 1 \\ \overline{\emptyset} &= \emptyset & \overline{(H, x \mapsto_r (\Gamma \vdash t : A))} &= \overline{H}, x : r & \overline{(H, a \mapsto_r t)} &= \overline{H} \end{aligned}$$

Definition A5 (Usage context approximation). We say that a usage context Δ is approximated by another Δ' if $\Delta \sqsubseteq \Delta'$ which is defined as follows:

$$\frac{\Delta \sqsubseteq \Delta' \quad r \leq r'}{\overline{\Delta}, x : r \sqsubseteq \overline{\Delta'}, x : r'}$$

where $1 \leq 1$ and $1 \leq \omega$ and $\omega \leq \omega$.

Multi-reduction rules

$$\frac{}{H \vdash t \Rightarrow H \vdash t \mid \emptyset \mid \emptyset} \text{REFL} \quad \frac{H \vdash t_1 \rightsquigarrow H' \vdash t_2 \mid \Gamma_1 \mid \Delta_1 \quad H' \vdash t_2 \Rightarrow H'' \vdash t_3 \mid \Gamma_2 \mid \Delta_2}{H \vdash t_1 \Rightarrow H'' \vdash t_3 \mid \Gamma_1, \Gamma_2 \mid \Delta_1 + \Delta_2} \text{EXT}$$

Single-reduction congruences

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash t_1 t_2 \rightsquigarrow H' \vdash t'_1 t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{APP}} \quad \frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash p t \rightsquigarrow H' \vdash p t' \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{let } (x, y) = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } (x, y) = t'_1 \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{LET}\otimes}$$

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{let unit} = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let unit} = t'_1 \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{LETunit}}$$

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{let } !x = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } !x = t'_1 \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{LET}!}$$

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash \& t \rightsquigarrow H' \vdash \& t' \mid \Gamma \mid \Delta} \rightsquigarrow_{\&}$$

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 \rightsquigarrow H' \vdash \text{copy } t'_1 \text{ as } x \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{copy}}$$

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash \text{copy } !t \text{ as } x \text{ in } t_2 \rightsquigarrow H' \vdash \text{copy } !t' \text{ as } x \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{copy}!}$$

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash *t \rightsquigarrow H' \vdash *t' \mid \Gamma \mid \Delta} \rightsquigarrow_*$$

Single-reduction β -rules

$$\begin{array}{c}
\frac{}{H, x \mapsto_{\omega} t \vdash x \rightsquigarrow H, x \mapsto_{\omega} t \vdash t \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR}\omega} \quad \frac{}{H, x \mapsto_1 t \vdash x \rightsquigarrow H \vdash t \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR}1} \\
\\
\frac{\Gamma \vdash t' : A}{H \vdash (\lambda x. t) t' \rightsquigarrow H, x \mapsto_1 (\Gamma \vdash t' : A) \vdash t \mid x : A \mid \emptyset} \rightsquigarrow_{\beta} \\
\\
\frac{\Gamma_1 \vdash t'_1 : A \quad \Gamma_2 \vdash t''_1 : B}{H \vdash \text{let } (x, y) = (t'_1, t''_1) \text{ in } t_3 \rightsquigarrow H, x \mapsto_1 (\Gamma_1 \vdash t'_1 : A), y \mapsto_1 (\Gamma_2 \vdash t''_1 : B) \vdash t_3 \mid \emptyset, x : A, y : B \mid \emptyset} \rightsquigarrow_{\otimes\beta} \\
\\
\frac{}{H \vdash \text{let unit} = \text{unit in } t \rightsquigarrow H \vdash t \mid \emptyset \mid \emptyset} \rightsquigarrow_{\beta\text{unit}} \\
\\
\frac{[\Gamma] \vdash t_1 : A}{H \vdash \text{let } !x = !t_1 \text{ in } t_2 \rightsquigarrow H, x \mapsto_{\omega} ([\Gamma] \vdash t_1 : A) \vdash t_2 \mid x : [A] \mid \emptyset} \rightsquigarrow_{! \beta} \\
\\
\frac{\text{dom}(H) \equiv \text{arrRefs}(v)}{H, H' \vdash \&(*v) \rightsquigarrow ([H]_{\omega}), H' \vdash !v \mid \emptyset \mid \emptyset} \rightsquigarrow_{\&\beta} \\
\\
\frac{\Gamma \vdash v : A \quad \text{dom}(H') \equiv \text{arrRefs}(v) \quad (H'', \theta) \equiv \text{copy}(H')}{H, H' \vdash \text{copy} !v \text{ as } x \text{ in } t_2 \rightsquigarrow H, H', H'', x \mapsto_1 (\Gamma \vdash *(\theta(v)) : *A) \vdash t_2 \mid x : *A \mid \emptyset} \rightsquigarrow_{\text{copy}\beta} \\
\\
\frac{a \# H}{H \vdash \text{newArray } n \rightsquigarrow H, a \mapsto_1 \mathbf{arr} \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{newArray}} \\
\\
\frac{}{H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash \text{readArray } (*a) i \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash (v, *a) \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{readArray}} \\
\\
\frac{}{H, a \mapsto_r \mathbf{arr} \vdash \text{writeArray } (*a) i v \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{writeArray}} \\
\\
\frac{}{H, a \mapsto_r \mathbf{arr} \vdash \text{deleteArray } (*a) \rightsquigarrow H \vdash \text{unit} \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{deleteArray}}
\end{array}$$

B Proofs

B.1 Admissibility of substitution

Lemma 1 (Linear substitution). *Given $\Gamma' \vdash t' : S$ and $\Gamma, s : S \vdash t : T$ then $\Gamma' + \Gamma \vdash [t'/s]t : T$.*

Proof. By induction on the structure of $\Gamma, s : S \vdash t : T$.

– (var)

$$\frac{}{[\Gamma], x : A \vdash x : A} \text{VAR}$$

Here, $s = t = x$ necessarily because this is the only linear variable in the scope ($s \notin \text{dom}([\Gamma])$), and thus $S = T$. We need to show that $\Gamma' + [\Gamma] \vdash [t'/s]s : S$.

Then since $[t'/s]s = t'$, and we have $\Gamma' \vdash t' : S$, we can apply the lemma of admissibility of weakening (Lemma 3) to derive the goal $\Gamma' + [\Gamma] \vdash [t'/s]s : S$.

– (abs)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \multimap B} \text{ ABS}$$

By induction, we have $\Gamma' + \Gamma, x : A \vdash [t'/s]t : B$. We need to show that $\Gamma' + \Gamma \vdash [t'/s](\lambda x. t) : A \multimap B$.

Applying the ABS rule to the induction hypothesis we get $\Gamma' + \Gamma \vdash \lambda x. [t'/s]t : A \multimap B$.

Then by the definition of substitution we have $\Gamma' + \Gamma \vdash [t'/s](\lambda x. t) : A \multimap B$ as required.

– (app)

$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1 t_2 : B} \text{ APP}$$

We need to show that $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s](t_1 t_2) : B$.

By the definition of context addition and the fact that s is linear, we either have $s \in \Gamma_1$ or $s \in \Gamma_2$ but not both. So there are two cases to consider.

- If $s \in \Gamma_1$, then by induction on the first premise, we have $\Gamma' + \Gamma_1 \vdash [t'/s]t_1 : A \multimap B$. Applying the APP rule to this and the second premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s]t_1 t_2 : B$.
Since $s \notin \Gamma_2$, $[t'/s]t_1 [t'/s]t_2$ (which we need) is equivalent to $[t'/s]t_1 t_2$ (which we have).
- If $s \in \Gamma_2$, then by induction on the second premise, we have $\Gamma' + \Gamma_2 \vdash [t'/s]t_2 : A$. Applying the APP rule to this and the first premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash t_1 [t'/s]t_2 : B$.
Since $s \notin \Gamma_1$, $[t'/s]t_1 [t'/s]t_2$ (which we need) is equivalent to $t_1 [t'/s]t_2$ (which we have).

– (derelection)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B} \text{ DER}$$

We know that $x \neq s$ as x here is non-linear whilst s is a linear assumption, thus we assume $\Gamma = \Gamma'', s : S$

By induction, we have $\Gamma' + (\Gamma'', x : A) \vdash [t'/s]t : B$. We need to show that $(\Gamma' + \Gamma''), x : [A] \vdash [t'/s]t : B$.

Applying the DER rule to the induction hypothesis we get $\Gamma' + \Gamma'', x : [A] \vdash [t'/s]t : B$, whose context, if it is defined is then equal to $(\Gamma' + \Gamma''), x : [A]$ and we are done.

– (bangIntro)

$$\frac{[\Gamma] \vdash t : A}{[\Gamma] \vdash !t : !A} !_I$$

This case is trivial, since by the premise Γ contains only non-linear assumptions, so cannot contain any linear variables.

– (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : B} !_E$$

We need to show that $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s]\text{let } !x = t_1 \text{ in } t_2 : B$.

By the definition of context addition and the fact that s is linear, we either have $s \in \Gamma_1$ or $s \in \Gamma_2$ but not both. So there are two cases to consider:

- If $s \in \Gamma_1$, then by induction on the first premise, we have $\Gamma' + \Gamma_1 \vdash [t'/s]t_1 : !A$. Applying the $!_E$ rule to this and the second premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{let } !x = [t'/s]t_1 \text{ in } t_2 : B$. Since $s \notin \Gamma_2$, $[t'/s]\text{let } !x = t_1 \text{ in } t_2$ (which we need) is equivalent to $\text{let } !x = [t'/s]t_1 \text{ in } t_2$ (which we have).
- If $s \in \Gamma_2$, then by induction on the second premise, we have $\Gamma' + \Gamma_2, x : [A] \vdash [t'/s]t_2 : B$. Applying the $!_E$ rule to this and the first premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } [t'/s]t_2 : B$. Since $s \notin \Gamma_1$, $[t'/s]\text{let } !x = t_1 \text{ in } t_2$ (which we need) is equivalent to $\text{let } !x = t_1 \text{ in } [t'/s]t_2$ (which we have).

– (pairIntro)

$$\frac{\Gamma_1 \vdash t_1 : A \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A \otimes B} \otimes_I$$

We need to show that $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s](t_1, t_2) : A \otimes B$.

By the definition of context addition and the fact that s is linear, we either have $s \in \Gamma_1$ or $s \in \Gamma_2$ but not both. So there are two cases to consider:

- If $s \in \Gamma_1$, then by induction on the first premise, we have $\Gamma' + \Gamma_1 \vdash [t'/s]t_1 : A$. Applying the \otimes_I rule to this and the second premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash ([t'/s]t_1, t_2) : A \otimes B$. Since $s \notin \Gamma_2$, $[t'/s](t_1, t_2)$ (which we need) is equivalent to $([t'/s]t_1, t_2)$ (which we have).
- If $s \in \Gamma_2$, then by induction on the second premise, we have $\Gamma' + \Gamma_2, x : [A] \vdash [t'/s]t_2 : B$. Applying the \otimes_I rule to this and the first premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash (t_1, [t'/s]t_2) : A \otimes B$. Since $s \notin \Gamma_1$, $[t'/s](t_1, t_2)$ (which we need) is equivalent to $(t_1, [t'/s]t_2)$ (which we have).

– (pairElim)

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : C} \otimes_E$$

We need to show that $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s]\text{let } (x, y) = t_1 \text{ in } t_2 : C$.

By the definition of context addition and the fact that s is linear, we either have $s \in \Gamma_1$ or $s \in \Gamma_2$ but not both. So there are two cases to consider:

- If $s \in \Gamma_1$, then by induction on the first premise, we have $\Gamma' + \Gamma_1 \vdash [t'/s]t_1 : A \otimes B$. Applying the \otimes_E rule to this and the second premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = [t'/s]t_1 \text{ in } t_2 : C$. Since $s \notin \Gamma_2$, $[t'/s]\text{let } (x, y) = t_1 \text{ in } t_2$ (which we need) is equivalent to $\text{let } (x, y) = [t'/s]t_1 \text{ in } t_2$ (which we have).
- If $s \in \Gamma_2$, then by induction on the second premise, we have $\Gamma' + \Gamma_2, x : A, y : B \vdash [t'/s]t_2 : C$. Applying the \otimes_E rule to this and the first premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } [t'/s]t_2 : C$. Since $s \notin \Gamma_1$, $[t'/s]\text{let } (x, y) = t_1 \text{ in } t_2$ (which we need) is equivalent to $\text{let } (x, y) = t_1 \text{ in } [t'/s]t_2$ (which we have).

– (unitIntro)

$$\overline{[I] \vdash \text{unit} : 1} \quad 1_I$$

Holds trivial since there is no linear assumption in the context, so the premise is false.

– (unitElim)

$$\frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : B} \quad 1_E$$

We need to show that $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s]\text{let unit} = t_1 \text{ in } t_2 : B$.

By the definition of context addition and the fact that s is linear, we either have $s \in \Gamma_1$ or $s \in \Gamma_2$ but not both. So there are two cases to consider:

- If $s \in \Gamma_1$, then by induction on the first premise, we have $\Gamma' + \Gamma_1 \vdash [t'/s]t_1 : 1$. Applying the 1_E rule to this and the second premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{let unit} = [t'/s]t_1 \text{ in } t_2 : B$. Since $s \notin \Gamma_2$, $[t'/s]\text{let unit} = t_1 \text{ in } t_2$ (which we need) is equivalent to $\text{let unit} = [t'/s]t_1 \text{ in } t_2$ (which we have).
- If $s \in \Gamma_2$, then by induction on the second premise, we have $\Gamma' + \Gamma_2 \vdash [t'/s]t_2 : B$. Applying the 1_E rule to this and the first premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } [t'/s]t_2 : B$. Since $s \notin \Gamma_1$, $[t'/s]\text{let unit} = t_1 \text{ in } t_2$ (which we need) is equivalent to $\text{let unit} = t_1 \text{ in } [t'/s]t_2$ (which we have).

– (borrow)

$$\frac{\Gamma \vdash t : *A}{\Gamma \vdash \&t : !A} \text{ BORROW}$$

By induction, we have $\Gamma' + \Gamma \vdash [t'/s]t : *A$. We need to show that $\Gamma' + \Gamma \vdash [t'/s]\&t : !A$.

Applying the BORROW rule to the induction hypothesis we get $\Gamma' + \Gamma \vdash \&[t'/s]t : !A$.

$[t'/s]\&t$ is equivalent to $\&[t'/s]t$, which we have.

– (copy)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : *A \vdash t_2 : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B} \text{ COPY}$$

We need to show that $\Gamma' + \Gamma_1 + \Gamma_2 \vdash [t'/s]\text{copy } t_1 \text{ as } x \text{ in } t_2 : B$.

By the definition of context addition and the fact that s is linear, we either have $s \in \Gamma_1$ or $s \in \Gamma_2$ but not both. So there are two cases to consider:

- If $s \in \Gamma_1$, then by induction on the first premise, we have $\Gamma' + \Gamma_1 \vdash [t'/s]t_1 : !A$. Applying the COPY rule to this and the second premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{copy } [t'/s]t_1 \text{ as } x \text{ in } t_2 : !B$. Since $s \notin \Gamma_2$, $[t'/s]\text{copy } t_1 \text{ as } x \text{ in } t_2$ (which we need) is equivalent to $\text{copy } [t'/s]t_1 \text{ as } x \text{ in } t_2$ (which we have).
- If $s \in \Gamma_2$, then by induction on the second premise, we have $\Gamma' + \Gamma_2, x : *A \vdash [t'/s]t_2 : !B$. Applying the COPY rule to this and the first premise gives $\Gamma' + \Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } [t'/s]t_2 : !B$. Since $s \notin \Gamma_1$, $[t'/s]\text{copy } t_1 \text{ as } x \text{ in } t_2$ (which we need) is equivalent to $\text{copy } t_1 \text{ as } x \text{ in } [t'/s]t_2$ (which we have).

– (nec)

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A} \text{ NEC}$$

Holds trivially since the typing here contains no linear assumptions and thus its context does not match the form of the lemma with linear assumption $s : S$.

– (primitives) The primitive operations have typing:

$$\begin{array}{l} \overline{[\Gamma] \vdash \text{newArray} : \mathbb{N} \multimap *(\text{Array } \mathbb{F})} \text{ new} \\ \overline{[\Gamma] \vdash \text{readArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ read} \\ \overline{[\Gamma] \vdash \text{writeArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})} \text{ write} \\ \overline{[\Gamma] \vdash \text{deleteArray} : *(\text{Array } \mathbb{F}) \multimap 1} \text{ del} \end{array}$$

The lemma holds trivially since the typing here contains no linear assumptions and thus its context does not match the form of the lemma with linear assumption $s : S$.

□

Lemma 2 (Non-linear substitution). *Given $[Γ'] ⊢ t' : S$ and $Γ, s : [S] ⊢ t : T$ then $[Γ'] + Γ ⊢ [t'/s]t : T$.*

Proof. By induction on the structure of $Γ, s : S ⊢ t : T$.

– (var)

$$\frac{}{[Γ], x : A ⊢ x : A} \text{VAR}$$

Two cases:

1. $x = s$. Not possible as the typing doesn't match; s is a non-linear variable.
2. $x ≠ s$ therefore $x ∈ \text{dom}(Γ)$, i.e. the typing is:

$$[Γ'', s : S], x : A ⊢ x : A \text{VAR}$$

Thus since $[t'/s]x = x$ we get the goal here by re-deriving the variable typing as follows (since it include arbitrary weakening in its definition):

$$[Γ + Γ'], x : A ⊢ x : A \text{VAR}$$

satisfying the goal here.

– (abs)

$$\frac{Γ, x : A ⊢ t : B}{Γ ⊢ λx.t : A → B} \text{ABS}$$

By induction, we have $[Γ'] + Γ, x : A ⊢ [t'/s]t : B$. We need to show that $[Γ'] + Γ ⊢ [t'/s]λx.t : A → B$.

Applying the ABS rule to the induction hypothesis we get $[Γ'] + Γ ⊢ λx.[t'/s]t : A → B$.

Then by the definition of substitution we have $[Γ'] + Γ ⊢ [t'/s]λx.t : A → B$.

– (app)

$$\frac{Γ_1 ⊢ t_1 : A → B \quad Γ_2 ⊢ t_2 : A}{Γ_1 + Γ_2 ⊢ t_1 t_2 : B} \text{APP}$$

We need to show that $[Γ'] + Γ_1 + Γ_2 ⊢ [t'/s]t_1 t_2 : B$.

By the definition of context addition we have either $s ∈ Γ_1$ or $s ∈ Γ_2$ or both, so there are three cases to consider.

1. If $s ∈ Γ_1$, then by induction on the first premise, we have $[Γ'] + Γ_1 ⊢ [t'/s]t_1 : A → B$. Applying the APP rule to this and the second premise gives $[Γ'] + Γ_1 + Γ_2 ⊢ [t'/s]t_1 t_2 : B$.
Since $s ∉ Γ_2$, $[t'/s]t_1 [t'/s]t_2$ (which we need) is equivalent to $[t'/s]t_1 t_2$ (which we have).
2. If $s ∈ Γ_2$, then by induction on the second premise, we have $[Γ'] + Γ_2 ⊢ [t'/s]t_2 : A$. Applying the APP rule to this and the first premise gives $[Γ'] + Γ_1 + Γ_2 ⊢ t_1 ([t'/s]t_2) : B$.
Since $s ∉ Γ_1$, $[t'/s]t_1 [t'/s]t_2$ (which we need) is equivalent to $t_1 [t'/s]t_2$ (which we have).

3. If $s \in \Gamma_2$ and $s \in \Gamma_1$ then by induction on the both premise, we have $[I'] + \Gamma_2 \vdash [t'/s]t_2 : A$ and $[I'] + \Gamma_1 \vdash [t'/s]t_1 : A \multimap B$. Applying the APP rule to this and the first premise gives $[I'] + [I'] + \Gamma_1 + \Gamma_2 \vdash ([t'/s]t_1) ([t'/s]t_2) : B$.

By the definition of context addition $[I'] + [I'] = [I']$ so we have the goal here.

– (dereliction)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B} \text{DER}$$

There are two cases to consider depending on the location of s :

1. $x \equiv s$ Therefore the goal is $[I'] + \Gamma \vdash [t'/s]t : B$ which follows by applying Lemma 1 on the premise
2. $x \neq s$, i.e., $x \in \text{dom}(\Gamma)$ i.e., $\Gamma = \Gamma'', s : [S]$ then we induct on the premise to get $([I'] + \Gamma''), x : A \vdash [t'/s]t : B$ on which we apply dereliction again to get the goal.

– (bangIntro)

$$\frac{[I] \vdash t : A}{[I] \vdash !t : !A} !_I$$

Let $\Gamma = \Gamma_1, s : S$ thus we can induct on the premise yielding $[I'] + [\Gamma_1] \vdash [t'/s]t : A$ upon which we can apply promotion again yielding:

$$\frac{[I'] + [\Gamma_1] \vdash [t'/s]t : A}{[I'] + [\Gamma_1] \vdash !([t'/s]t) : !A} !_I$$

which satisfies the goal since syntactic substitution is defined $[t'/s]!t = !([t'/s]t)$.

– (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : B} !_E$$

We need to show that $[I'] + \Gamma_1 + \Gamma_2 \vdash [t'/s](\text{let } !x = t_1 \text{ in } t_2) : B$.

The reasoning is almost identical to that of the application case where (app) where we inductively apply the substitution and we rely on the property that $[I'] + [I'] = [I']$ in the case that s appears in both subterms.

– (pairIntro)

$$\frac{\Gamma_1 \vdash t_1 : A \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A \otimes B} \otimes_I$$

As above for (bangElim) and (app), same proof pattern.

– (pairElim)

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : C} \otimes_E$$

Same dual induction as for (app), (bangElim), (pairIntro).

– (unitIntro)

$$\frac{}{[\Gamma] \vdash \text{unit} : 1} 1_I$$

Let $[\Gamma] = [\Gamma_1], s : [S], [\Gamma_2]$ then since $[t'/s]\text{unit} = \text{unit}$ then we can get the goal $\Gamma' + [\Gamma] \vdash [t'/s]\text{unit} : 1$ simply by reapplying this typing rule with the $\Gamma' + [\Gamma]$ context.

– (unitElim)

$$\frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : B} 1_E$$

Same proof pattern as in (app), (pairIntro) etc.

– (borrow)

$$\frac{\Gamma \vdash t : *A}{\Gamma \vdash \&t : !A} \text{BORROW}$$

By induction, we have $[\Gamma'] + \Gamma \vdash [t'/s]t : *A$. We need to show that $[\Gamma'] + \Gamma \vdash [t'/s]\&t : !A$. Applying the BORROW rule to the induction hypothesis we get $[\Gamma'] + \Gamma \vdash \&[t'/s]t : !A$. $[t'/s]\&t$ is equivalent to $\&[t'/s]t$, which we have.

– (copy)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : *A \vdash t_2 : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B} \text{COPY}$$

Same pattern as (app), (pairElim) etc.

– (nec)

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A} \text{NEC}$$

with $\Gamma = \Gamma_0, s : S$ to match the form of this lemma.

The (closed) premise implies that $[t'/s]t = t$. Then the typing rule NEC can be reapplied to yield:

$$\frac{\emptyset \vdash [t'/s]t : A}{[\Gamma_0 + \Gamma'] \vdash *([t'/s]t) : *A} \text{NEC}$$

where $[\Gamma_0 + \Gamma'] = [\Gamma_0] + [\Gamma']$ and where $*([t'/s]t) = [t'/s](*t)$ by the definition of substitution, matching the goal of this lemma.

Holds trivially since the context is empty.

– (primitives - newArray):

$$\frac{}{[\Gamma] \vdash \text{newArray} : \mathbb{N} \multimap *(\text{Array } \mathbb{F})} \text{new}$$

with the assumption that $\Gamma = \Gamma_0, s : S$. Then we can reapply the typing rule here to get:

$$\frac{}{[\Gamma_0 + \Gamma'] \vdash \text{newArray} : \mathbb{N} \multimap *(\text{Array } \mathbb{F})} \text{new}$$

satisfying the lemma.

– (remaining primitives)

$$\frac{}{[I] \vdash \text{readArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ read}$$

$$\frac{}{[I] \vdash \text{writeArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})} \text{ write}$$

$$\frac{}{[I] \vdash \text{deleteArray} : *(\text{Array } \mathbb{F}) \multimap 1} \text{ del}$$

each has the same proof structure as for `newArray` where we simply reapply the typing with the different context (which is unconstrained in these rules).

□

B.2 Admissibility of weakening

Lemma 3 (Weakening is admissible). *Given $\Gamma \vdash t : A$ then $\Gamma, [I'] \vdash t : A$.*

Proof. By induction on typing.

– (var)

$$\frac{}{[I], x : A \vdash x : A} \text{ VAR}$$

Typing of variables naturally includes weakening since we can introduce any additional context thus we get here:

$$\frac{}{[I, I'], x : A \vdash x : A} \text{ VAR}$$

which gives the goal by the definition of the all-non-linear predicate (Def. 1).

– (abs)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \multimap B} \text{ ABS}$$

By induction.

– (app)

$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1 t_2 : B} \text{ APP}$$

By induction on either premise to add the weakening context, which is then preserved by $+$.

– (derelect)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B} \text{ DER}$$

By induction.

– (bangIntro)

$$\frac{[I] \vdash t : A}{[I] \vdash !t : !A} !_I$$

By induction via the definition of the all-non-linear predicate (Def. 1).

– (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : B} !_E$$

By induction on either premise to add the weakening context, which is then preserved by $+$.

– (pairIntro)

$$\frac{\Gamma_1 \vdash t_1 : A \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A \otimes B} \otimes_I$$

By induction on either premise to add the weakening context, which is then preserved by $+$.

– (pairElim)

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : C} \otimes_E$$

By induction on either premise to add the weakening context, which is then preserved by $+$.

– (unitElim)

$$\frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : B} 1_E$$

By induction on either premise to add the weakening context, which is then preserved by $+$.

– (borrow)

$$\frac{\Gamma \vdash t : *A}{\Gamma \vdash \&t : !A} \text{BORROW}$$

By induction.

– (copy)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : *A \vdash t_2 : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B} \text{COPY}$$

By induction on either premise to add the weakening context, which is then preserved by $+$.

– (nec)

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A} \text{NEC}$$

The context in the premise is empty so we can naturally introduce any additional context in the succedent, giving the goal by the definition of the all-non-linear predicate (Def. 1).

– (newArray)

$$\overline{[\Gamma] \vdash \text{newArray} : \mathbb{N} \multimap *(\text{Array } \mathbb{F})} \text{ new}$$

By re-applying the rule and adding in the extra weakened assumptions, where $[\Gamma, \Gamma'] = [\Gamma], [\Gamma']$.

– (readArray)

$$\overline{[\Gamma] \vdash \text{readArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ read}$$

As above, by re-applying the rule and adding in the extra weakened assumptions, where $[\Gamma, \Gamma'] = [\Gamma], [\Gamma']$.

– (writeArray)

$$\overline{[\Gamma] \vdash \text{writeArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})} \text{ write}$$

As above, by re-applying the rule and adding in the extra weakened assumptions, where $[\Gamma, \Gamma'] = [\Gamma], [\Gamma']$.

– (deleteArray)

$$\overline{[\Gamma] \vdash \text{deleteArray} : *(\text{Array } \mathbb{F}) \multimap 1} \text{ del}$$

As above, by re-applying the rule and adding in the extra weakened assumptions, where $[\Gamma, \Gamma'] = [\Gamma], [\Gamma']$.

□

B.3 Conservation

Theorem 4 (Conservation). *For a well-typed term $\Gamma \vdash t : A$ and all Γ_0 and H such that $H \bowtie (\Gamma_0 + \Gamma)$ and a reduction $H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma_1 \mid \Delta$ we have:*

$$\exists \Gamma'. \Gamma' \vdash t' : A \quad \wedge \quad H' \bowtie (\Gamma_0 + \Gamma') \quad \wedge \quad (\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma_1})$$

The first conjunct gives regular type preservation, linked with heap compatibility in the second conjunct. The last conjunct expresses the core of conservation: that resource usage accrued in this reduction, given by Δ , plus remaining resources given in the heap H' are approximated by the original resources given in the heap H plus the specification of the resources from any variable bindings Γ_1 encountered along the way. The context Γ_0 gives the bindings not described by Γ , which is key to the inductive proof of this result.

Proof. By induction on the structure of typing and reductions.

– (var)

$$\overline{[\Gamma], x : A \vdash x : A} \text{ VAR}$$

With two possible reductions and heap compatibilities:

- With heap compatibility derivation:

$$\frac{H' \bowtie \Gamma_0 + [\Gamma] + \Gamma' \quad \Gamma' \vdash t : A}{H', x \mapsto_1 (\Gamma' \vdash t : A) \bowtie (\Gamma_0 + [\Gamma], x : A)} \text{ LIN}$$

and the reduction:

$$\overline{H', x \mapsto_1 (\Gamma' \vdash t : A) \vdash x \rightsquigarrow H' \vdash t' \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR1}}$$

The resulting typing judgment is then given by $\Gamma' \vdash t : A$, which by Lemma 3 (weakening) can be weakened to $[\Gamma], \Gamma' \vdash t : A$ to give the typing result here (first conjunct) and then the required concluding heap compatibility derivation (second conjunct) of $H' \bowtie \Gamma_0 + [\Gamma] + \Gamma'$ is given by the first premise of the incoming heap compatibility above.

Finally, the third goal conjunct is:

$$\begin{aligned} \overline{(\overline{H'}) + x : 1} &\sqsubseteq \overline{(\overline{H'}, x \mapsto_1 (\Gamma' \vdash t : A)), \emptyset} \\ \implies \overline{\overline{H'}, x : 1} &\sqsubseteq \overline{\overline{H'}, x : 1} \end{aligned}$$

which follows by reflexivity of the preorder.

- With heap compatibility derivation:

$$\frac{H_1 \bowtie \Gamma_0 + [\Gamma] + [\Gamma'] \quad [\Gamma'] \vdash t' : A}{H_1, x \mapsto_\omega (\Gamma' \vdash t : A) \bowtie (\Gamma_0 + [\Gamma], x : A)} \text{ LIN}$$

and thus the only possible reduction is:

$$\overline{H_1, x \mapsto_\omega ([\Gamma'] \vdash t' : A) \vdash x \rightsquigarrow H_1, x \mapsto_\omega ([\Gamma'] \vdash t' : A) \vdash t' \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR}\omega}$$

The resulting typing judgment is now given by $[Γ] + [Γ'], x : [A] ⊢ t' : A$ obtained by weakening (Lemma 3) on $[Γ'] ⊢ t' : A$ to introduce the assumption of $x : [A]$ (note, now non-linear) and the previous context $[Γ]$. This then allows the final heap to be compatible with typing. The final heap is $H' = H_1, x \mapsto_\omega ([Γ'] ⊢ t' : A)$ and then the required heap compatibility derivation is derived uses the premises of the incoming heap compatibility above but using ω and relying on idempotence of context addition:

$$\frac{\frac{H_1 \bowtie \Gamma_0 + [Γ] + [Γ']}{H_1 \bowtie \Gamma_0 + [Γ] + [Γ'] + [Γ']} \text{ idem.} \quad [Γ'] ⊢ t' : A}{H_1, x \mapsto_\omega ([Γ'] ⊢ t' : A) \bowtie (\Gamma_0 + [Γ] + [Γ'], x : [A])} \omega$$

Finally, the third goal conjunct is:

$$\begin{aligned} & \frac{(\overline{H_1, x \mapsto_\omega ([Γ'] ⊢ t : A)}) + x : 1 \sqsubseteq (\overline{H_1, x \mapsto_\omega ([Γ'] ⊢ t : A)}) + \emptyset}{\overline{H_1, x : \omega + x : 1} \sqsubseteq \overline{H, x : \omega}} \\ \implies & \overline{H_1, x : \omega} \sqsubseteq \overline{H, x : \omega} \end{aligned}$$

which follows by reflexivity of the preorder and by absorption as $\omega + 1 = \omega$.

– (abs)

$$\frac{\Gamma, x : A ⊢ t : B}{\Gamma ⊢ \lambda x. t : A \multimap B} \text{ ABS}$$

Has no reduction so the case is trivial here.

– (app)

$$\frac{\Gamma_1 ⊢ t_1 : A \multimap B \quad \Gamma_2 ⊢ t_2 : A}{\Gamma_1 + \Gamma_2 ⊢ t_1 t_2 : B} \text{ APP}$$

For which there are three possible reductions (one of which refines the typing)

- $t_1 = \lambda x. t'_1$ thus the typing is:

$$\frac{\frac{\Gamma_1, x : A ⊢ t'_1 : B}{\Gamma_1 ⊢ \lambda x. t'_1 : A \multimap B} \text{ ABS} \quad \Gamma_2 ⊢ t_2 : A}{\Gamma_1 + \Gamma_2 ⊢ (\lambda x. t'_1) t_2 : B} \text{ APP}$$

and we have compatibility $H \bowtie \Gamma_0 + (\Gamma_1 + \Gamma_2)$ with one possible reduction:

$$\overline{H ⊢ (\lambda x. t'_1) t_2} \rightsquigarrow \overline{H, x \mapsto_1 (\Gamma_2 ⊢ t_2 : A) ⊢ t'_1 \mid x : A \mid \emptyset} \rightsquigarrow^\beta$$

Therefore the resulting typing judgment is $\Gamma_1, x : A ⊢ t'_1 : B$ and we construct the goal compatibility judgment as follows, using the incoming compatibility assumption:

$$\frac{H \bowtie \Gamma_0 + \Gamma_1 + \Gamma_2}{(H, x \mapsto_1 (\Gamma_2 ⊢ t_2 : A)) \bowtie \Gamma_0 + (\Gamma_1, x : A)} \text{ LIN}$$

(where $x \notin \text{dom}(\Gamma_0)$ since it was bound in the term here).

The remaining goal is then:

$$\begin{aligned} & \frac{(\overline{(H, x \mapsto_1 (\Gamma_2 ⊢ t_2 : A))}) + \emptyset \sqsubseteq (\overline{H, x : A})}{\overline{H, x : 1} \sqsubseteq \overline{H, x : 1}} \end{aligned}$$

which follows by reflexivity of the preorder.

- otherwise application, evaluating the left

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash t_1 t_2 \rightsquigarrow H' \vdash t'_1 t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{APP}}$$

with compatibility $H \bowtie (\Gamma_0 + (\Gamma_1 + \Gamma_2))$.

Apply this theorem inductively on $\Gamma_1 \vdash t_1 : A \rightarrow B$ with Γ_0 (in the recursive call) instantiated to $\Gamma_0 + \Gamma_2$ which yields $\Gamma'_1 \vdash t'_1 : A \rightarrow B$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_2) + \Gamma'_1)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$ (which subsequently provides the third goal immediately).

Thus the resulting typing judgment for this case is the application:

$$\frac{\Gamma'_1 \vdash t'_1 : A \rightarrow B \quad \Gamma_2 \vdash t_2 : A}{\Gamma'_1 + \Gamma_2 \vdash (\lambda x. t'_1) t_2 : B} \text{APP}$$

Therefore the resulting context is $\Gamma' = \Gamma'_1 + \Gamma_2$ for which the heap compatibility judgment $H' \bowtie (\Gamma_0 + (\Gamma'_1 + \Gamma_2))$ follows from the inductive call via associativity and commutativity of context $+$.

- or $t_1 = p$ (primitive partially-applied value) and t_2 is not a value thus we have the reduction

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash p t_2 \rightsquigarrow H' \vdash p t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

with compatibility $H \bowtie (\Gamma_0 + (\Gamma_1 + \Gamma_2))$.

Applying the theorem inductively on $\Gamma_2 \vdash t_2 : A$ (second typing premise) with the Γ_0 parameter of the recursive call instantiated to $\Gamma_0 + \Gamma_1$ which yields $\Gamma'_2 \vdash t'_2 : A$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_1) + \Gamma'_2)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$ which then provides the third goal conjunct here immediately.

The result typing judgment is the application:

$$\frac{\Gamma_1 \vdash p : A \rightarrow B \quad \Gamma'_2 \vdash t'_2 : A}{\Gamma_1 + \Gamma'_2 \vdash p t'_2 : B} \text{APP}$$

Therefore the resulting context is $\Gamma' = \Gamma_1 + \Gamma'_2$ for which the heap compatibility judgment $H' \bowtie (\Gamma_0 + (\Gamma_1 + \Gamma'_2))$ follows from the inductive call via associativity and commutativity of context $+$.

- $t_1 = \text{newArray}$ and $t_2 = n$ There is only one possible reduction:

$$\frac{a \# H}{H \vdash \text{newArray } n \rightsquigarrow H, a \mapsto_1 \mathbf{arr} \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{newArray}}$$

But here $\Gamma = \Delta = \emptyset$ and there is no new grading information on the heap, so the goal follows by reflexivity of the preorder.

- $t_2 = \text{readArray } (*a)$ and $t_2 = n$. There is only one possible reduction:

$$\frac{H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash \text{readArray } (*a) i \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash (v, *a) \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{readArray}}$$

But here $\Gamma = \Delta = \emptyset$ and there is no new grading information on the heap, so the goal follows by reflexivity of the preorder.

- $t_2 = \text{writeArray } (*a) n$ and $t_2 = f$. There is only one possible reduction:

$$\frac{H, a \mapsto_r \mathbf{arr} \vdash \text{writeArray } (*a) i v \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{writeArray}}$$

But here $\Gamma = \Delta = \emptyset$ and there is no new grading information on the heap, so the goal follows by reflexivity of the preorder.

- $t_2 = \text{deleteArray}$ and $t_2 = *a$. There is only one possible reduction:

$$\overline{H, a \mapsto_{\text{red}} \mathbf{arr} \vdash \text{deleteArray}(*a) \rightsquigarrow H \vdash \text{unit} \mid \emptyset \mid \emptyset} \rightsquigarrow^{\text{deleteArray}}$$

But here $\Gamma = \Delta = \emptyset$ and there is no new grading information on the heap, so the goal follows by reflexivity of the preorder.

– (der)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B} \text{DER}$$

with compatibility $H \bowtie (\Gamma, x : [A])$, which must have derivation (only possible one):

$$\frac{H \bowtie \Gamma_0 + \Gamma + [F_1] \quad [F_1] \vdash t'_1 : A}{(H, x \mapsto_{\omega}([F_1] \vdash t'_1 : A)) \bowtie \Gamma_0 + (\Gamma, x : [A])} \omega$$

We can re-derive heap compatibility from the premises here as:

$$\frac{H \bowtie \Gamma_0 + \Gamma + [F_1] \quad [F_1] \vdash t'_1 : A}{(H, x \mapsto_{\omega}([F_1] \vdash t'_1 : A)) \bowtie \Gamma_0 + (\Gamma, x : A)} \text{LIN}$$

This provides heap compatibility which can be used with the premise of typing to inductively apply conservation, yielding some output judgment $\Gamma' \vdash t' : B$ the goal $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{T_1})$ and the heap compatibility with $H' \bowtie (\Gamma_0 + \Gamma')$ which provides the goals here.

– (bangIntro)

$$\frac{[F] \vdash t : A}{[F] \vdash !t : !A} !_I$$

Has no reduction so the case is trivial here.

– (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : B} !_E$$

For which there are two possible reductions:

- $t_1 = !t'_1$ thus the typing is

$$\frac{\frac{[F_1] \vdash t'_1 : A}{[F_1] \vdash !t'_1 : !A} !_I \quad \Gamma_2, x : [A] \vdash t_2 : B}{[F_1] + \Gamma_2 \vdash \text{let } !x = !t'_1 \text{ in } t_2 : B} !_E$$

and we have compatibility $H \bowtie \Gamma_0 + ([F_1] + \Gamma_2)$ with one possible reduction:

$$\overline{H \vdash \text{let } !x = !t'_1 \text{ in } t_2 \rightsquigarrow H, x \mapsto_{\omega} t'_1 \vdash t_2 \mid x : [A] \mid \emptyset} \rightsquigarrow^{\beta}$$

Therefore the outgoing heap is $H' = H, x \mapsto_{\omega}([F_1] \vdash t'_1 : A)$ and the resulting typing judgment is $\Gamma_2, x : [A] \vdash t_2 : B$ and we can construct the goal compatibility judgment as follows, using the incoming compatibility assumption:

$$\frac{H \bowtie (\Gamma_0 + [F_1]) + \Gamma_2 \quad [F_1] \vdash t'_1 : A}{(H, x \mapsto_{\omega}([F_1] \vdash t'_1 : A)) \bowtie \Gamma_0 + (\Gamma_2, x : [A])} \omega$$

(where $x \notin \text{dom}(\Gamma_0)$ since it was bound in the term here)
 The remaining goal is then:

$$\begin{aligned} & \frac{((\overline{H}, x \mapsto_{\omega} ([\Gamma_1] \vdash t'_1 : A)) + \emptyset)}{(\overline{H}, x : \omega)} \sqsubseteq \frac{((\overline{H}), x : [\overline{A}])}{(\overline{H}, x : \omega)} \\ \implies & \quad \quad \quad \sqsubseteq \end{aligned}$$

which follows by reflexivity of the preorder.

- otherwise, the reduction can proceed by evaluating t_1 :

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{let } !x = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } !x = t'_1 \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{LET!}}$$

with compatibility $H \bowtie (\Gamma_0 + (\Gamma_1 + \Gamma_2))$.

Apply this lemma inductively on $\Gamma_1 \vdash t_1 : !A$ with Γ_0 (in the recursive call) instantiated to $\Gamma_0 + \Gamma_2$ which yields $\Gamma'_1 \vdash t'_1 : !A$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_2) + \Gamma'_1)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$ (which subsequently provides the third goal immediately).

- (pairIntro) with typing:

$$\frac{\Gamma_1 \vdash t_1 : A \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A \otimes B} \otimes_I$$

Has no reduction so the case is trivial here.

- (pairElim) with typing:

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : C} \otimes_E$$

For which there are two possible reductions (one of which refines the typing):

- $t_1 = (t'_1, t''_1)$ thus the typing is:

$$\frac{\frac{\Gamma_1 \vdash t'_1 : A \quad \Gamma_2 \vdash t''_1 : B}{\Gamma_1 + \Gamma_2 \vdash (t'_1, t''_1) : A \otimes B} \otimes_I \quad \Gamma_3, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 + \Gamma_3 \vdash \text{let } (x, y) = (t'_1, t''_1) \text{ in } t_2 : C} \otimes_E$$

with $H \bowtie \Gamma_0 + (\Gamma_1 + \Gamma_2 + \Gamma_3)$ with one possible reduction:

$$\frac{\Gamma_1 \vdash t'_1 : A \quad \Gamma_2 \vdash t''_1 : B}{H \vdash \text{let } (x, y) = (t'_1, t''_1) \text{ in } t_3 \rightsquigarrow H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A), y \mapsto_{\rightarrow 1} (\Gamma_2 \vdash t''_1 : B) \vdash t_3 \mid \emptyset, x : A, y : B \mid \emptyset} \rightsquigarrow_{\otimes \beta}$$

Therefore the resulting typing judgment is $\Gamma_3, x : A, y : B \vdash t_2 : C$ and we construct the goal compatibility judgment as follows, using the incoming compatibility assumption:

$$\frac{\frac{H \bowtie \Gamma_0 + (\Gamma_1 + \Gamma_2 + \Gamma_3) \quad \Gamma_1 \vdash t'_1 : A}{H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A) \bowtie \Gamma_0 + (\Gamma_2 + \Gamma_3), x : A} \text{LIN} \quad \Gamma_2 \vdash t''_1 : B}{H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A), y \mapsto_{\rightarrow 1} (\Gamma_2 \vdash t''_1 : B) \bowtie \Gamma_0 + (\Gamma_3, x : A, y : B)} \text{LIN}$$

(where $x \notin \text{dom}(\Gamma_0)$ and $y \notin \text{dom}(\Gamma_0)$ since they were bound in the terms here).
 The remaining goal is then:

$$\frac{(\overline{H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A), y \mapsto_{\rightarrow 1} (\Gamma_2 \vdash t''_1 : B)}) + \emptyset}{\overline{H}, x : 1, y : 1} \sqsubseteq \frac{\overline{H}, (x : A, y : B)}{\overline{H}, x : 1, y : 1} \sqsubseteq$$

which follows by reflexivity of the preorder.

- otherwise evaluate the left subterm:

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{let } (x, y) = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } (x, y) = t'_1 \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{LET}\otimes}$$

with compatibility $H \bowtie \Gamma_0 + (\Gamma_1 + \Gamma_2)$.

Apply this lemma inductively on $\Gamma_1 \vdash t_1 : A \otimes B$ with Γ_0 (in the recursive call) instantiated to $\Gamma_0 + \Gamma_2$ which yields $\Gamma'_1 \vdash t'_1 : A \otimes B$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_2) + \Gamma'_1)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$.

Thus the resulting conclusion here is the typing: $\Gamma'_1 + \Gamma_2 \vdash \text{let } (x, y) = t'_1 \text{ in } t_2 : C$ with heap compatibility from the induction by commutativity of associativity of $+$ $H' \bowtie (\Gamma_0 + (\Gamma'_1 + \Gamma_2))$ and since output heap and usage matches the induction then the last conjunct is from the inductive evidence $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$.

- (unitIntro) with typing:

$$\frac{}{[\Gamma] \vdash \text{unit} : 1} 1_I$$

Has no reduction so the case is trivial here.

- (unitElim) with typing:

$$\frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : B} 1_E$$

Has two possible reductions.

- $t_1 = \text{unit}$

Then the typing is:

$$\frac{\frac{}{[\Gamma_1] \vdash \text{unit} : 1} 1_I \quad \Gamma_2 \vdash t_2 : B}{[\Gamma_1] + \Gamma_2 \vdash \text{let unit} = \text{unit in } t_2 : B} 1_E$$

with $H \bowtie \Gamma_0 + ([\Gamma_1] + \Gamma_2)$ with one possible reduction:

$$\frac{}{H \vdash \text{let unit} = \text{unit in } t_2 \rightsquigarrow H \vdash t_2 \mid \emptyset \mid \emptyset} \rightsquigarrow_{\beta \text{unit}}$$

The resultant typing is then obtained via weakening (Lemma 3) on the second premise to get $[\Gamma_1], \Gamma_2 \vdash t_2 : B$ such that the heap compatibility is just as before: $H \bowtie \Gamma_0 + ([\Gamma_1] + \Gamma_2)$.

Since in the reduction both the opened up binders Γ_1 and usage context Δ here are both \emptyset , so the goal here follows by reflexivity of the preorder as $\overline{H} \sqsubseteq \overline{H}$.

- $t_1 \neq \text{unit}$.

Then we have $H \bowtie \Gamma_0 + (\Gamma_1 + \Gamma_2)$ with one possible reduction:

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{let unit} = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let unit} = t'_1 \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{LETunit}}$$

Apply the lemma inductively on $\Gamma_1 \vdash t_1 : 1$ with Γ_0 (in the recursive call) instantiated to $\Gamma_0 + \Gamma_2$ which yields $\Gamma'_1 \vdash t'_1 : 1$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_2) + \Gamma'_1)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$.

The goal typing is then $\Gamma'_1 + \Gamma_2 \vdash \text{let unit} = t'_1 \text{ in } t_2 : B$ with heap compatibility from the inductive evidence by commutativity and associativity of $+$ $H' \bowtie ((\Gamma_0 + \Gamma_2) + \Gamma'_1)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma})$ from the inductive evidence provides the final conjunct goal (as the usage context and opened binders in this reduction are the same as the inductive premise).

– (borrow) with typing:

$$\frac{\Gamma \vdash t : *A}{\Gamma \vdash \&t : !A} \text{BORROW}$$

There are two possible reductions:

- t is a value $*v$ and we reduce by:

$$\frac{\text{dom}(H) \equiv \text{arrRefs}(t)}{H, H' \vdash \&(*v) \rightsquigarrow ([H]_{\omega}), H' \vdash !v \mid \emptyset \mid \emptyset} \rightsquigarrow_{\&\beta}$$

with compatibility $H, H' \bowtie (\Gamma_0 + \Gamma)$.

By the value lemma (Lemma B2) on the typing $\Gamma \vdash *v : *A$ and by the fact that we only have unique arrays, we know that $v = a$ and so the typing here can be refined to:

$$\frac{\overline{[I_1], a : \text{Array } A \vdash a : \text{Array } A}^{\text{arr}}}{[I_1], a : \text{Array } A \vdash *a : *(\text{Array } A)} \text{BORROW}$$

with heap compatibility then $H, H' \bowtie (\Gamma_0 + [I_1], a : \text{Array } A)$. Furthermore $\text{dom}(H) \equiv \text{arrRefs}(t)$ then refines to $\text{dom}(H) \equiv a$.

Therefore the incoming heap is of the form $(H', a \mapsto_r \mathbf{arr}) \bowtie (\Gamma_0 + [I_1], a : \text{Array } A)$ with the only possible derivation:

$$\frac{H' \bowtie \Gamma_0 + [I_1]}{(H', a \mapsto_r \mathbf{arr}) \bowtie (\Gamma_0 + [I_1], a : \text{Array } A)} \text{REF}$$

The resulting typing judgment for this reduction is then derived as:

$$\frac{\overline{[I_1], a : \text{Array } A \vdash a : \text{Array } A}^{\text{arr}}}{[I_1], a : \text{Array } A \vdash !a : !(\text{Array } A)} !_I$$

(using Definition A1 on the all-non-linear predicate redefined for runtime contexts which treats a as non-linear here, enabling the promotion).

Thus, the goal heap compatibility is $([H]_{\omega}), H' \bowtie (\Gamma_0 + [I_1], a : \text{Array } A)$ where $([H]_{\omega}) = ([\emptyset, a \mapsto_r \mathbf{arr}]_{\omega}) = a \mapsto_{\omega} \mathbf{arr}$, by Definition A2. This goal heap compatibility judgment is then derived.

$$\frac{H' \bowtie \Gamma_0 + [I_1]}{H', a \mapsto_{\omega} \mathbf{arr} \bowtie (\Gamma_0 + [I_1], a : \text{Array } A)} \text{REF}$$

Lastly the final goal conjunct specialises to $(\overline{H', a \mapsto_{\omega} \mathbf{arr}}) \sqsubseteq (\overline{H', a \mapsto_r \mathbf{arr}})$ which holds by reflexivity of the preorder since reference counts are ignored by usage context extraction, so this just becomes $\overline{H'} \sqsubseteq \overline{H'}$.

- t is not a value, and thus there is one possible reduction:

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash \&t \rightsquigarrow H' \vdash \&t' \mid \Gamma \mid \Delta} \rightsquigarrow_{\&}$$

with compatibility $H \bowtie \Gamma_0 + \Gamma$. Here, the goal follows straightforwardly from applying the lemma inductively on $\Gamma \vdash t : *A$.

– (copy) with typing:

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : *A \vdash t_2 : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B} \text{ COPY}$$

Here, there are three possible reductions.

- $t_1 = !v$ for some value v .

Then the typing is as follows:

$$\frac{\frac{[T_1] \vdash v : A}{[T_1] \vdash !v : !A} !_I \quad \Gamma_2, x : *A \vdash t_2 : !B}{[T_1] + \Gamma_2 \vdash \text{copy } !v \text{ as } x \text{ in } t_2 : !B} \text{ COPY}$$

with heap compatibility $H \bowtie \Gamma_0 + [T_1] + \Gamma_2$.

We can then reduce with:

$$\frac{[T_1] \vdash v : A \quad \text{dom}(H') \equiv \text{arrRefs}(v) \quad (H'', \theta) \equiv \text{copy}(H')}{H, H' \vdash \text{copy } !v \text{ as } x \text{ in } t_2 \rightsquigarrow H, H', H'', x \mapsto_1 ([T_1] \vdash *(\theta(v)) : *A) \vdash t_2 \mid x : *A \mid \emptyset} \rightsquigarrow_{\text{copy}\beta}$$

Here, H' and H'' contain no grading information, as they are made up solely of the array references present in the overall heap. Thus, we only need to consider H . The resulting typing judgment is $\Gamma_2, x : *A \vdash t_2 : !B$ by the second premise of the initial typing and we construct the goal compatibility judgment as follows, using the incoming compatibility assumption:

$$\frac{H \bowtie \Gamma_0 + [T_1] + \Gamma_2 \quad [T_1] \vdash *v : *A \quad x \notin \text{dom}(H)}{H, x \mapsto_1 ([T_1] \vdash *v : *A) \bowtie (\Gamma_0 + \Gamma_2), x : *A} \text{ LIN}$$

and the grading goal is then given by:

$$\frac{((H, x \mapsto_1 ([T_1] \vdash *v : *A)) + \emptyset) \sqsubseteq (\overline{H}, x : *A)}{\overline{H}, x : 1 \sqsubseteq \overline{H}, x : 1}$$

which follows by reflexivity.

- $t_1 = !t$ for some t .

Then the typing is as follows:

$$\frac{\frac{[T_1] \vdash t : A}{[T_1] \vdash !t : !A} !_I \quad \Gamma_2, x : *A \vdash t_2 : !B}{[T_1] + \Gamma_2 \vdash \text{copy } !t \text{ as } x \text{ in } t_2 : !B} \text{ COPY}$$

We can reduce:

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash \text{copy } !t \text{ as } x \text{ in } t_2 \rightsquigarrow H' \vdash \text{copy } !t' \text{ as } x \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{copy}!}$$

with compatibility $H \bowtie \Gamma_0 + ([T_1] + \Gamma_2)$.

Apply this lemma inductively on $[T_1] \vdash t : A$ with Γ_0 (in the recursive call) instantiated to $\Gamma_0 + \Gamma_2$ which yields $[T'_1] \vdash t'_1 : A$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_2) + [T'_1])$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, [T])$. The resultant typing for this case is then: $[T'_1] + \Gamma_2 \vdash \text{copy } !t' \text{ as } x \text{ in } t_2 : !B$ by reapplying typing rule COPY. The goal compatibility follows then by commutativity and associativity of $+$ $H' \bowtie \Gamma_0 + ([T'_1] + \Gamma_2)$ and the third goal follows immediately from the inductive evidence.

- Otherwise, there is only one reduction:

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma \mid \Delta}{H \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 \rightsquigarrow H' \vdash \text{copy } t'_1 \text{ as } x \text{ in } t_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{copy}}$$

with compatibility $H \bowtie \Gamma_0 + ([\Gamma_1] + \Gamma_2)$.

Apply this lemma inductively on $\Gamma_1 \vdash t_1 : !A$ with Γ_0 (in the recursive call) instantiated to $\Gamma_0 + \Gamma_2$ which yields $[\Gamma'_1] \vdash t'_1 : !A$ with H' such that $H' \bowtie ((\Gamma_0 + \Gamma_2) + \Gamma'_1)$ and $(\overline{H'} + \Delta) \sqsubseteq (\overline{H}, \overline{\Gamma_1})$ (which subsequently provides the third goal immediately) with the outgoing typing $[\Gamma'_1] + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B$ and compatibility from the inductive evidence via commutativity and associativity of $+$.

- (nec) with typing:

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A} \text{ NEC}$$

The only possible reduction is

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash *t \rightsquigarrow H' \vdash *t' \mid \Gamma \mid \Delta} \rightsquigarrow_*$$

then by induction we get $\emptyset \vdash t' : A$ from which we can rebuild the typing $[\Gamma] \vdash *t' : *A$ and the results of the induction give the goal here as the heap and output contexts and unchanged.

- (newArray), (readArray), (writeArray), (deleteArray) have no reduction as they are unapplied primitives.

□

B.4 Uniqueness

Lemma B1 (Single-step uniqueness). *For a well-typed term $\Gamma \vdash t_1 : *A$ and all Γ_0 and heaps H such that $H \bowtie (\Gamma_0 + \Gamma)$ and given a single-step reduction $H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta$ then for all $a \in \text{arrRefs}(t'_1)$ (array references in t'_1) we have:*

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t'' . a \mapsto_1 t'' \in H' \\ \wedge a \notin \text{dom}(H) &\implies \exists t'' . a \mapsto_1 t'' \in H' \end{aligned}$$

i.e., any array references contributing to the final term that in the incoming heap start unique stay unique, and any new array references contributing to the final term are unique.

Proof. By induction on typing.

- (var)

$$\frac{}{[\Gamma], x : *A \vdash x : *A} \text{ VAR}$$

Then we also have a heap H such that $H \bowtie ([\Gamma], x : *A)$.

There are two possibilities:

- By inversion of heap-compatibility implies that there exists a subheap H_1 such that $H = H_1, x \mapsto_1 (\Gamma' \vdash t' : *A)$.

$$\frac{H_1 \bowtie ([\Gamma] + \Gamma') \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{(H_1, x \mapsto_1 (\Gamma' \vdash t' : *A)) \bowtie ([\Gamma], x : *A)} \text{ LIN}$$

and reduction:

$$\frac{}{H_1, x \mapsto_1 t' \vdash x \rightsquigarrow H_1 \vdash t' \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR1}}$$

Then we have that $a \in \text{arrRefs}(t')$ we have:

$$(a \mapsto_1 t_2 \in H_1, x \mapsto_1 t' \implies \exists t_3. a \mapsto_1 t_3 \in H_1)$$

trivially (with $t_3 = t_2$) as no array references were modified or manipulated here (just change in variables), and

$$(a \notin \text{dom}(H_1)) \implies \exists t_3. a \mapsto_1 t_3 \in H_1$$

trivially since the premise must always be false.

- By inversion of heap-compatibility implies that there exists a subheap H_1 such that $H = H_1, x \mapsto_\omega (\Gamma' \vdash t' : *A)$.

$$\frac{H_1 \bowtie ([\Gamma] + \Gamma') \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{(H_1, x \mapsto_\omega (\Gamma' \vdash t' : *A)) \bowtie ([\Gamma], x : *A)} \text{LIN}$$

and reduction:

$$\frac{}{H_1, x \mapsto_\omega t' \vdash x \rightsquigarrow H_1, x \mapsto_\omega t' \vdash t' \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR1}}$$

Then we have that $a \in \text{arrRefs}(t')$ we have:

$$\begin{aligned} (a \in \text{dom}(H_1, x \mapsto_\omega t')) &\implies (a \mapsto_1 t_2 \in H_1 \implies \exists t_3. a \mapsto_1 t_3 \in H_1, x \mapsto_\omega t') \\ \wedge (a \notin \text{dom}(H_1, x \mapsto_\omega t')) &\implies \exists t_3. a \mapsto_1 t_3 \in H_1, x \mapsto_\omega t' \end{aligned}$$

which both follow trivially the first as essentially the identity proof as no array objects are manipulated and the second as the premise is always false.

– (abs) Trivial as it doesn't match the typing

– (app)

$$\frac{\Gamma_1 \vdash t_1 : A \multimap *B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1 t_2 : *B} \text{APP}$$

with a heap H such that $H \bowtie (\Gamma_1 + \Gamma_2)$ and two reductions (beta and application congruence on the left) then several possible reductions following from primitive applications:

1.

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma_1 \mid \Delta_1}{H \vdash t_1 t_2 \rightsquigarrow H' \vdash t'_1 t_2 \mid \Gamma_1 \mid \Delta_1} \rightsquigarrow_{\text{APP}}$$

Then the goal follows by induction.

2. Alternatively $t_1 = \lambda x. t'_1$ such that typing is:

$$\frac{\frac{\Gamma_1, x : A \vdash t'_1 : *B}{\Gamma_1 \vdash t'_1 : A \multimap *B} \text{ABS} \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash (\lambda x. t'_1) t_2 : *B} \text{APP}$$

and we have reduction:

$$\frac{\Gamma_2 \vdash t_2 : A}{H \vdash (\lambda x. t'_1) t_2 \rightsquigarrow H, x \mapsto_1 (\Gamma_2 \vdash t_2 : A) \vdash t'_1 \mid x : A \mid \emptyset} \rightsquigarrow_\beta$$

Then for all $a \in \text{arrRefs}(t_1'')$ we have that:

$$a \mapsto_1 t' \in H \implies \exists t''. a \mapsto_1 t'' \in H, x \mapsto_1 t_2$$

trivially as the right hand side does not refer to array references and the heap is preserved, and:

$$a \notin \text{dom}(H) \implies \exists t''. a \mapsto_1 t'' \in H'$$

is trivially true as the antecedent is always false.

3. $t_1 = \text{newArray}$ therefore $A = \mathbb{N}$

Therefore we induct on the second argument:

- t_2 is a value and therefore by the value lemma (Lemma B2) $t_2 = n$ and thus the typing is:

$$\frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \text{newArray } n : *(\text{Array } \mathbb{F})} \text{TyDerivedNewArray}$$

with $H \bowtie (\Gamma_0 + \Gamma)$.

Thus there is a reduction as follows:

$$\frac{a \# H}{H \vdash \text{newArray } n \rightsquigarrow H, a \mapsto_1 \mathbf{arr} \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{newArray}}$$

where $a \notin \text{dom}(H)$ but we have that $\exists t''. a \mapsto_1 t'' \in H, a \mapsto_1 \mathbf{arr}$ satisfying the goal.

- t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t_2' \mid \Gamma \mid \Delta}{H \vdash \text{newArray } t_2 \rightsquigarrow H' \vdash \text{newArray } t_2' \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

Then the result holds by induction

4. $t_1 = \text{readArray}$ therefore $A = *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})$

and there is a reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t_2' \mid \Gamma \mid \Delta}{H \vdash \text{readArray } t_2 \rightsquigarrow H' \vdash \text{readArray } t_2' \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

Therefore the single-step uniqueness result holds by induction

5. $t_1 = \text{readArray } (*a)$ therefore $A = \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})$

- t_2 is a value and therefore by the value lemma on $\Gamma_2 \vdash t_2 : \mathbb{N}$ (Lemma B2) implies $t_2 = n$ and thus the typing is refined at runtime as follows:

$$\frac{\frac{[I_1], a : \text{Array } \mathbb{F} \vdash a : (\text{Array } \mathbb{F}) \text{ arr}}{[I_1], a : \text{Array } \mathbb{F} \vdash *a : *(\text{Array } \mathbb{F})} *_{\text{array}} \quad \Gamma_2 \vdash i : \mathbb{N}}{[I_1] + \Gamma_2, a : \text{Array } \mathbb{F} \vdash \text{readArray } a \ i : \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{TyDerivedReadArray}$$

with $H' \bowtie (\Gamma_0 + [I_1] + \Gamma_2, a : \text{Array } \mathbb{F})$, and by the heap compatibility rule for array references there exists some H such that $H' = H, a \mapsto_r \mathbf{arr}$.

Then there is a reduction as follows:

$$\frac{H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash \text{readArray } (*a) \ i \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash (v, *a) \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{readArray}}$$

where $a \in \text{dom}(H)$ and thus if that $\exists t''. a \mapsto_1 t'' \in H, a \mapsto_r \mathbf{arr}$ i.e. $r = 1$ then we have $\exists t''. a \mapsto_r t'' \in H, a \mapsto_1 \mathbf{arr}$ for the output heap satisfying the goal for this array reference for all other $a' \in \text{arrRefs}(t)$ we have:

$$\begin{aligned} a' \mapsto_1 t' \in H, a \mapsto_r (\mathbf{arr}[i] = t) &\implies \exists t''. a' \mapsto_1 t'' \in H, a \mapsto_r (\mathbf{arr}[i] = t) \\ \wedge a' \notin \text{dom}(H, a \mapsto_r (\mathbf{arr}[i] = t)) &\implies \exists t''. a' \mapsto_1 t'' \in H, a \mapsto_r (\mathbf{arr}[i] = t) \end{aligned}$$

follows trivially since the rest of the heap is preserved.

- t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{readArray}(*a) t_2 \rightsquigarrow H' \vdash \text{readArray}(*a) t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

And therefore the single-step uniqueness result holds by induction.

6. $t_1 = \text{writeArray}$ therefore $A = *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})$ with reduction

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{writeArray} t_2 \rightsquigarrow H' \vdash \text{writeArray} t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

Therefore the single-step uniqueness result holds by induction.

7. $t_1 = \text{writeArray}(*a)$ therefore $A = \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})$ with reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{writeArray}(*a) t_2 \rightsquigarrow H' \vdash \text{writeArray}(*a) t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

Therefore the single-step uniqueness result holds by induction.

8. $t_1 = \text{writeArray}(*a) i$ therefore $A = \mathbb{F} \otimes *(\text{Array } \mathbb{F})$

- t_2 is a value and therefore by the value lemma on $\Gamma_2 \vdash t_2 : \mathbb{F}$ (Lemma B2) implies $t_2 = f$ and thus the typing is refined at runtime as follows:

$$\frac{\frac{[\Gamma_1], a : \text{Array } \mathbb{F} \vdash a : (\text{Array } \mathbb{F}) \text{ arr}}{[\Gamma_1], a : \text{Array } \mathbb{F} \vdash *a : *(\text{Array } \mathbb{F})} \text{ *array} \quad \Gamma_2 \vdash i : \mathbb{N} \quad \Gamma_3 \vdash f : \mathbb{F}}{[\Gamma_1] + \Gamma_2 + \Gamma_3, a : \text{Array } \mathbb{F} \vdash \text{writeArray } a \ i \ f : *(\text{Array } \mathbb{F})} \text{ TyDERIVEDWRITEARRAY}$$

with $H' \bowtie (\Gamma_0 + [\Gamma_1] + \Gamma_2 + \Gamma_3, a : \text{Array } \mathbb{F})$, and by the heap compatibility rule for array references there exists some H such that $H' = H, a \mapsto_r \mathbf{arr}$.

Then there is a reduction as follows:

$$\overline{H, a \mapsto_r \mathbf{arr} \vdash \text{writeArray}(*a) i \ v \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{writeArray}}$$

thus for all $a' \in \text{arrRefs}(v)$ (array references in v) we have:

$$\begin{aligned} a' \mapsto_1 t' \in H, a \mapsto_r \mathbf{arr} &\implies \exists t''. a' \mapsto_1 t'' \in H, a \mapsto_r (\mathbf{arr}[i] = v) \\ \wedge a' \notin \text{dom}(H, a \mapsto_r \mathbf{arr}) &\implies \exists t''. a' \mapsto_1 t'' \in H, a \mapsto_r (\mathbf{arr}[i] = v) \end{aligned}$$

where the first conjunct follows trivially as the only a' is a and if $r = 1$ the result is direct (identity) otherwise the premise is false. The second conjunct is trivially since the premise as we have the only a' here as a so the premise is always false.

- t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{writeArray}(*a) \ i \ t_2 \rightsquigarrow H' \vdash \text{writeArray}(*a) \ i \ t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

Therefore the single-step uniqueness result holds by induction.

9. $t_1 = \text{deleteArray}$ therefore $A = *(\text{Array } \mathbb{F}) \multimap 1$

But the typing does not match the lemma as the result type is therefore 1.

- (derelect)

$$\frac{\Gamma, x : A \vdash t : *B}{\Gamma, x : [A] \vdash t : *B} \text{DER}$$

and a single-reduction $H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta$ and we have heap compatibility $H \bowtie (\Gamma_0 + \Gamma, x : [A])$ which by inversion means there exists a heap H_1 such that $H = H_1, x \mapsto_{\omega} (\Gamma' \vdash t' : A)$.

$$\frac{H_1 \bowtie (\Gamma + [\Gamma']) \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{H_1, x \mapsto_{\omega} (\Gamma' \vdash t' : *A) \bowtie (\Gamma, x : [A])} \omega$$

We can thus rebuild the heap compatibility from these premises for the typing premise:

$$\frac{H'_1 \bowtie (\Gamma + [\Gamma']) \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{H'_1, x \mapsto_{\omega} (\Gamma' \vdash t' : *A) \bowtie (\Gamma, x : A)} \text{LIN}$$

On which we can induct with $\Gamma, x : A \vdash t : *B$ and the same reduction $H \vdash t \rightsquigarrow H' \vdash *v \mid \Gamma \mid \Delta$ to get that for all $a \in \text{arrRefs}(v)$ (array references in v) we have:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H' \end{aligned}$$

which satisfies the goal here.

- (bangIntro)

$$\frac{[\Gamma] \vdash t : A}{[\Gamma] \vdash !t : !A} !_I$$

Trivially satisfies the theorem since the typing cannot conclude with a type of form $*A$.

- (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : *B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : *B} !_E$$

with a heap H such that $H \bowtie (\Gamma_1 + \Gamma_2)$ and reduction:

$$H \vdash \text{let } !x = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash t' \mid \Gamma' \mid \Delta$$

which has two possible derivations:

- 1.

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta_1}{H \vdash \text{let } !x = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } !x = t'_1 \text{ in } t_2 \mid \Gamma' \mid \Delta_1} \rightsquigarrow_{\text{LET!}}$$

Then induction provides the goal.

2. Alternatively $t_1 = !t'_1$ such that the typing is:

$$\frac{\frac{[F_1] \vdash t'_1 : A}{[F_1] \vdash !t'_1 : !A} !_I \quad \Gamma_2, x : [A] \vdash t_2 : *B}{[F_1] + \Gamma_2 \vdash \text{let } !x = !t'_1 \text{ in } t_2 : *B} !_E$$

and we have reduction:

$$\frac{[F_1] \vdash t'_1 : A}{H \vdash \text{let } !x = !t'_1 \text{ in } t_2 \rightsquigarrow H, x \mapsto_\omega ([F_1] \vdash t'_1 : A) \vdash t_2 \mid x : [A] \mid \emptyset} \rightsquigarrow_{! \beta}$$

Which trivially satisfies the goal since all array references are then in H and we get the conditions trivially (since no array references are manipulated) similar to the β proof above.

- (pairIntro) Trivially satisfies the theorem since the typing cannot conclude with a type of form $*A$.
- (pairElim)

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : *C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : *C} \otimes_E$$

Two possible reductions:

1.

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta}{H \vdash \text{let } (x, y) = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } (x, y) = t'_1 \text{ in } t_2 \mid \Gamma' \mid \Delta} \rightsquigarrow_{\text{LET!}}$$

and induction provides the goal.

2. otherwise $t_1 = (t_3, t_4)$ such that the typing is:

$$\frac{\frac{\Gamma_3 \vdash t_3 : A \quad \Gamma_4 \vdash t_4 : B}{\Gamma_3 + \Gamma_4 \vdash (t_3, t_4) : (A \otimes B)} !_I \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_3 + \Gamma_4 + \Gamma_2 \vdash \text{let } (x, y) = (t_3, t_4) \text{ in } t_2 : C} !_E$$

and we have reduction:

$$\frac{\Gamma_1 \vdash t'_1 : A \quad \Gamma_2 \vdash t''_1 : B}{H \vdash \text{let } (x, y) = (t'_1, t''_1) \text{ in } t_3 \rightsquigarrow H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A), y \mapsto_{\rightarrow 1} (\Gamma_2 \vdash t''_1 : B) \vdash t_3 \mid \emptyset, x : A, y : B \mid \emptyset} \rightsquigarrow_{! \beta}$$

The goal is then straightforward: for all $a \in \text{arrRefs}(t_3)$ we have:

$$\begin{aligned} a \mapsto_{\rightarrow 1} t' \in H &\implies \exists t''. a \mapsto_{\rightarrow 1} t'' \in H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A), y \mapsto_{\rightarrow 1} (\Gamma_2 \vdash t''_1 : B) \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_{\rightarrow 1} t'' \in H, x \mapsto_{\rightarrow 1} (\Gamma_1 \vdash t'_1 : A), y \mapsto_{\rightarrow 1} (\Gamma_2 \vdash t''_1 : B) \end{aligned}$$

where the first conjunct holds trivially since there is no manipulation of the array references here and the second conjunct is always true.

- (unitIntro) Trivially satisfies the theorem since the typing cannot conclude with a type of form $*A$.
- (unitElim) Following essentially the same structure as the tensor proof where array reference counting is not used so induction provides the goal.
- (return), (bind) All trivially satisfy the theorem since the typing cannot conclude with a type of form $*A$.

– (nec)

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A} \text{ NEC}$$

with reduction

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash *t \rightsquigarrow H' \vdash *t' \mid \Gamma \mid \Delta} \rightsquigarrow_*$$

Then by induction on the premise term we get the required result here since the output heap is preserved between the two.

– (newArray) (readArray) (writeArray) (deleteArray)
All trivial as they don't match the typing and don't reduce.

□

Theorem 5 (Uniqueness). *For a well-typed term $\Gamma \vdash t : *A$ and all Γ_0 and H such that $H \bowtie (\Gamma_0 + \Gamma)$ and given a multi-reduction to a value $H \vdash t \Rightarrow H' \vdash *v \mid \Gamma' \mid \Delta$, for all $a \in \text{arrRefs}(v)$ (array references in v) we have:*

$$a \mapsto_1 t' \in H \Rightarrow \exists t''. a \mapsto_1 t'' \in H' \quad \wedge \quad a \notin \text{dom}(H) \Rightarrow \exists t''. a \mapsto_1 t'' \in H'$$

i.e., any array references contributing to the final term that are unique in the incoming heap stay unique in the resulting term, and any new array references contributing to the final term are also unique.

Proof. By induction on typing $\Gamma \vdash t : *A$.

– (var)

$$\frac{}{[\Gamma], x : *A \vdash x : *A} \text{ VAR}$$

Then we also have a heap H such that $H \bowtie ([\Gamma], x : *A)$ which by inversion of heap-compatibility implies that there exists a subheap H_1 such that $H = H_1, x \mapsto_1 (\Gamma' \vdash t' : *A)$.

$$\frac{H_1 \bowtie ([\Gamma] + \Gamma') \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{(H_1, x \mapsto_1 (\Gamma' \vdash t' : *A)) \bowtie ([\Gamma], x : *A)} \text{ LIN}$$

and reduction:

$$H \vdash x \Rightarrow H' \vdash *v \mid \Gamma' \mid \Delta$$

which necessarily must be derived by following since there is only one possible (single) reduction for variables:

$$\frac{\frac{H_1, x \mapsto_1 t' \vdash x \rightsquigarrow H_1 \vdash t' \mid \emptyset \mid x : 1}{H_1, x \mapsto_1 t' \vdash x \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid x : 1 + \Delta_2} \rightsquigarrow_{\text{VAR1}} \quad H_1 \vdash t' \Rightarrow H' \vdash *v \mid \Gamma_2 \mid \Delta_2}{H_1, x \mapsto_1 t' \vdash x \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid x : 1 + \Delta_2} \text{ EXT}$$

Then by induction on the second premise with $H_1 \bowtie ([\Gamma] + \Gamma')$ (i.e., instantiating Γ_0 to $[\Gamma]$) $\Gamma' \vdash t' : *A$ we have that $a \in \text{arrRefs}(v)$ (array references in v) we have:

$$(a \in \text{dom}(H_1, x \mapsto_1 t')) \Rightarrow (a \mapsto_1 t'' \in H_1 \Rightarrow \exists t'''. a \mapsto_1 t''' \in H') \\ \wedge (a \notin \text{dom}(H_1, x \mapsto_1 t')) \Rightarrow \exists t'''. a \mapsto_1 t''' \in H'$$

and since $H = H_1, x \mapsto_1 t'$ then the above satisfies the goal here too.

The case for the reduction $H_1, x \mapsto_\omega t' \vdash x \rightsquigarrow H_1, x \mapsto_\omega t' \vdash t' \mid \emptyset \mid x : 1$ is the same.

– (abs) Is trivial since the typing cannot conclude with a type of the form $*A$.

– (app)

$$\frac{\Gamma_1 \vdash t_1 : A \multimap *B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1 t_2 : *B} \text{APP}$$

with a heap H such that $H \bowtie (\Gamma_1 + \Gamma_2)$ and reduction:

$$H \vdash t_1 t_2 \Rightarrow H' \vdash *v \mid \Gamma' \mid \Delta$$

which has two possible derivations on terms:

1.

$$\frac{\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma'_1 \mid \Delta_1}{H \vdash t_1 t_2 \rightsquigarrow H' \vdash t'_1 t_2 \mid \Gamma'_1 \mid \Delta_1} \rightsquigarrow^{\text{APP}} \quad H' \vdash t'_1 t_2 \Rightarrow H'' \vdash *v \mid \Gamma''_2 \mid \Delta_2}{H \vdash t_1 t_2 \Rightarrow H'' \vdash *v \mid \Gamma''_1 + \Gamma''_2 \mid \Delta_1 + \Delta_2} \text{EXT}$$

By conservation on the single reduction for t_1 we get type preservation which gives us $\Gamma'_1 \vdash t'_1 : A \multimap *B$ and thus $\Gamma'_1 + \Gamma_2 \vdash t'_1 t_2 : *B$, and also that $H' \bowtie (\Gamma_0 + \Gamma'_1 + \Gamma_2)$. Thus we can induct on the multi-reduction here to get that $a \in \text{arrRefs}(v)$ then:

$$\begin{aligned} (a \mapsto_1 t' \in H' &\implies \exists t''. a \mapsto_1 t'' \in H'') \\ \wedge (a \notin \text{dom}(H')) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

which provides the goal.

2. Alternatively $t_1 = \lambda x. t'_1$ such that typing is:

$$\frac{\frac{\Gamma_1, x : A \vdash t'_1 : *B}{\Gamma_1 \vdash t'_1 : A \multimap *B} \text{ABS} \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash (\lambda x. t'_1) t_2 : *B} \text{APP}$$

and we have reduction:

$$\frac{\Gamma_2 \vdash t_2 : A}{H \vdash (\lambda x. t'_1) t_2 \rightsquigarrow H, x \mapsto_1 (\Gamma_2 \vdash t_2 : A) \vdash t'_1 \mid x : A \mid \emptyset} \rightsquigarrow^\beta \quad H, x \mapsto_1 (\Gamma_2 \vdash t_2 : A) \vdash t'_1 \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid \Delta_2}{H \vdash (\lambda x. t'_1) t_2 \Rightarrow H'' \vdash *v \mid \Gamma_2, x : A \mid \Delta_2} \text{EXT}$$

From the incoming heap compatibility $H \bowtie (\Gamma_1 + \Gamma_2)$ we can construct the following heap compatibility:

$$\frac{H \bowtie (\Gamma_1 + \Gamma_2) \quad \Gamma_2 \vdash t_2 : A \quad x \notin \text{dom}(H)}{H, x \mapsto_1 (\Gamma_2 \vdash t_2 : A) \bowtie (\Gamma_1, x : A)} \text{LIN}$$

and thus we can induct on the second multi reduction with this heap compatibility and the typing $\Gamma_1, x : A \vdash t'_1 : *B$ to get that $x \in \text{arrRefs}(v)$ then: $x \mapsto_1 t' \in H''$ which satisfies the goal here.

3. Alternatively $t_1 = p$ and then there are two classes of possible reduction: congruence on a partial primitive application (dealt with first) or primitive beta rules

(a) $t_1 = p$ and t_2 is not a value therefore the reduction is:

$$\frac{\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma'_1 \mid \Delta_1}{H \vdash p t_2 \rightsquigarrow H' \vdash p t'_2 \mid \Gamma'_1 \mid \Delta_1} \rightsquigarrow^{\text{prim}} \quad H' \vdash p t'_2 \Rightarrow H'' \vdash *v \mid \Gamma''_2 \mid \Delta_2}{H \vdash p t_2 \Rightarrow H'' \vdash *v \mid \Gamma''_1 + \Gamma_2 \mid \Delta_1 + \Delta_2} \text{EXT}$$

By single-step uniqueness on the first premise we have:

$$a \mapsto_1 t' \in H \implies \exists t''. a \mapsto_1 t'' \in H' \quad \wedge \quad a \notin \text{dom}(H) \implies \exists t''. a \mapsto_1 t'' \in H'$$

By conservation on the first premise we get that $\Gamma_1 + \Gamma'_2 \vdash p \ t_2 : *B$ and $H' \bowtie (\Gamma_0 + \Gamma_1 + \Gamma'_2)$ which then enables induction of multi-step uniqueness on the tail multi-reduction to get:

$$a \mapsto_1 t' \in H' \implies \exists t'' . a \mapsto_1 t'' \in H'' \quad \wedge \quad a \notin \text{dom}(H') \implies \exists t'' . a \mapsto_1 t'' \in H''$$

which composes with the single-step uniqueness (transitivity of implication) to give the goal here.

(b) $t_1 = \text{newArray}$ and $t_2 = n$

$$\frac{\dots}{\Gamma \vdash \text{newArray } n : *(\text{Array } \mathbb{F})} \text{NEWARRAY}$$

and given $H \bowtie \Gamma_0 + \Gamma$ then there is reduction:

$$\frac{\frac{a \# H}{H \vdash \text{newArray } n \rightsquigarrow H, a \mapsto_1 \text{arr} \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{newArray}} \frac{H, a \mapsto_1 \text{arr} \vdash *a \rightsquigarrow H, a \mapsto_1 \text{arr} \vdash *a \mid \emptyset \mid \emptyset}{H \vdash \text{newArray } n \Rightarrow H, a \mapsto_1 \text{arr} \vdash *a \mid \emptyset \mid \emptyset} \text{REFL} \quad \text{EXT}$$

where $a \notin \text{dom}(H)$ but we have that $\exists t'' . a \mapsto_1 t'' \in H, a \mapsto_1 \text{arr}$ satisfying the goal.

(c) $t_1 = \text{readArray } (*a)$ and $t_2 = n$

$$\frac{\dots}{\Gamma \vdash \text{readArray } (*a) \ n : *(\text{Array } \mathbb{F})} \text{READARRAY}$$

and given $H \bowtie \Gamma_0 + \Gamma$ then there is reduction:

$$\frac{H, a \mapsto_r (\text{arr}[i] = v) \vdash \text{readArray } (*a) \ i \rightsquigarrow H, a \mapsto_r (\text{arr}[i] = v) \vdash (v, *a) \mid \emptyset \mid \emptyset}{H, a \mapsto_r (\text{arr}[i] = t) \vdash \text{readArray } (*a) \ n \Rightarrow H' \vdash (v, *a) \mid \Gamma_2 \mid \Delta_2} \rightsquigarrow_{\text{readArray}} \text{reflexive...} \quad \text{EXT}$$

Then applying Lemma ?? we have:

$$\begin{aligned} a' \mapsto_1 t' \in H, a \mapsto_r (\text{arr}[i] = t) &\implies \exists t'' . a' \mapsto_1 t'' \in H, a \mapsto_r (\text{arr}[i] = t) \\ \wedge \ a \notin \text{dom}(H, a \mapsto_r (\text{arr}[i] = t)) &\implies \exists t'' . a' \mapsto_1 t'' \in H, a \mapsto_r (\text{arr}[i] = t) \end{aligned}$$

and by induction on the second premise we have for all $a \in \text{arrRefs}(v)$:

$$\begin{aligned} a' \mapsto_1 t' \in H, a \mapsto_r (\text{arr}[i] = t) &\implies \exists t'' . a' \mapsto_1 t'' \in H' \\ \wedge \ a' \notin \text{dom}(H, a \mapsto_r (\text{arr}[i] = t)) &\implies \exists t'' . a' \mapsto_1 t'' \in H' \end{aligned}$$

Composing these two pieces of evidence together by instantiation and transitivity provides the goal.

(d) $t_1 = \text{writeArray } (*a) \ n$ and $t_2 = f$

$$\frac{\dots}{\Gamma \vdash \text{writeArray } (*a) \ n \ v : *(\text{Array } \mathbb{F})} \text{WRITEARRAY}$$

and given $H \bowtie \Gamma_0 + \Gamma$ then there is reduction:

$$\frac{H, a \mapsto_r \text{arr} \vdash \text{writeArray } (*a) \ i \ v \rightsquigarrow H, a \mapsto_r (\text{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset}{H \vdash \text{writeArray } (*a) \ n \ v \Rightarrow H, a \mapsto_r (\text{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{writeArray}} \text{REFL} \quad \text{EXT}$$

where the goal follows by Lemma B1 on the first premise as the second premise is necessarily just the reflexive multi-reduction rule.

– (derelect)

$$\frac{\Gamma, x : A \vdash t : *B}{\Gamma, x : [A] \vdash t : *B} \text{DER}$$

and a multi-reduction $H \vdash t \Rightarrow H' \vdash *v \mid \Gamma \mid \Delta$ and we have heap compatibility $H \bowtie (\Gamma_0 + \Gamma, x : [A])$ which by inversion means there exists a heap H_1 such that $H = H_1, x \mapsto_\omega (\Gamma' \vdash t' : A)$.

$$\frac{H_1 \bowtie (\Gamma + [\Gamma']) \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{H_1, x \mapsto_\omega (\Gamma' \vdash t' : *A) \bowtie (\Gamma, x : [A])} \omega$$

We can thus rebuild the heap compatibility from these premises for the typing premise:

$$\frac{H'_1 \bowtie (\Gamma + [\Gamma']) \quad \Gamma' \vdash t' : *A \quad x \notin \text{dom}(H_1)}{H'_1, x \mapsto_\omega (\Gamma' \vdash t' : *A) \bowtie (\Gamma, x : A)} \text{LIN}$$

On which we can induct with $\Gamma, x : A \vdash t : *B$ and the multi-reduction $H \vdash t \Rightarrow H' \vdash *v \mid \Gamma \mid \Delta$ to get that for all $a \in \text{arrRefs}(v)$ (array references in v) we have:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H' \end{aligned}$$

which satisfies the goal here.

– (bangIntro)

$$\frac{[\Gamma] \vdash t : A}{[\Gamma] \vdash !t : !A} !_I$$

Trivially satisfies the theorem since the typing cannot conclude with a type of form $*A$.

– (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : *B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : *B} !_E$$

with a heap H such that $H \bowtie (\Gamma_1 + \Gamma_2)$ and reduction:

$$H \vdash \text{let } !x = t_1 \text{ in } t_2 \Rightarrow H' \vdash *v \mid \Gamma' \mid \Delta$$

which has two possible derivations:

1.

$$\frac{\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma_1 \mid \Delta_1}{H \vdash \text{let } !x = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let } !x = t'_1 \text{ in } t_2 \mid \Gamma_1 \mid \Delta_1} \rightsquigarrow_{\text{APP}} \quad H' \vdash \text{let } !x = t'_1 \text{ in } t_2 \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid \Delta_2}{H \vdash \text{let } !x = t_1 \text{ in } t_2 \Rightarrow H'' \vdash *v \mid \Gamma_1 + \Gamma_2 \mid \Delta_1 + \Delta_2} \text{EXT}$$

By conservation on the single reduction for t_1 we get type preservation which gives us $\Gamma_1 \vdash t'_1 : !A$ and thus $\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t'_1 \text{ in } t_2 : *B$, and also that $H' \bowtie (\Gamma_0 + \Gamma')$. Thus we can induct on the multi-reduction here to get

$$\begin{aligned} a \mapsto_1 t' \in H' &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H') &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

which along with Lemma B1 applied to the single reduction here gives us:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H' \end{aligned}$$

which we can combine together via instantiation and transitivity to get the goal:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

2. Alternatively $t_1 = !t'_1$ such that the typing is:

$$\frac{\frac{[T_1] \vdash t'_1 : A}{[T_1] \vdash !t'_1 : !A} !_I \quad \Gamma_2, x : [A] \vdash t_2 : *B}{[T_1] + \Gamma_2 \vdash \text{let } !x = !t'_1 \text{ in } t_2 : *B} !_E$$

and we have reduction:

$$\frac{[T_1] \vdash t'_1 : A}{H \vdash \text{let } !x = !t'_1 \text{ in } t_2 \rightsquigarrow H, x \mapsto_\omega([T_1] \vdash t'_1 : A) \vdash t_2 \mid x : [A] \mid \emptyset} \rightsquigarrow^{\beta} \frac{H, x \mapsto_\omega([T_1] \vdash t'_1 : A) \vdash t_2 \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid \Delta_2}{H \vdash \text{let } !x = !t'_1 \text{ in } t_2 \Rightarrow H'' \vdash *v \mid \Gamma_2, x : [A] \mid \Delta_2} \text{EXT}$$

From the incoming heap compatibility $H \bowtie ([T_1] + \Gamma_2)$ we can construct the following heap compatibility:

$$\frac{H \bowtie ([T_1] + \Gamma_2) \quad [T_1] \vdash t'_1 : A \quad x \notin \text{dom}(H)}{H, x \mapsto_1([T_1] \vdash t'_1 : A) \bowtie (\Gamma_2, x : [A])} \text{LIN}$$

and thus we can induct on the second multi reduction with this heap compatibility and the typing $\Gamma_2, x : [A] \vdash t_2 : *B$ to get that $a \in \text{arrRefs}(v)$ (array references in v) we have:

$$\begin{aligned} a \mapsto_1 t' \in H, x \mapsto_\omega([T_1] \vdash t'_1 : A) &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H, \text{var } x \mid - \triangleright \text{Inf}([G1] - t'_1 : A)) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

From which we can get the goal since heap assignment to x is not an array reference assignment so we have:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

- (pairIntro) Trivially satisfies the theorem since the typing cannot conclude with a type of form $*A$.
- (pairElim)

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : *C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : *C} \otimes_E$$

Proceeds in essentially the same way as the above. There are two cases that split into a beta and a congruence case, but neither head reduction manipulates the array references itself so apply induction and Lemma B1 to get the outgoing uniqueness condition as in the (app) rule.

- (unitIntro) Trivially satisfies the theorem since the typing cannot conclude with a type of form $*A$.
- (unitElim)

$$\frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : *B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : *B} 1_E$$

with a heap H such that $H \bowtie (\Gamma_1 + \Gamma_2)$ and reduction:

$$H \vdash \text{let unit} = t_1 \text{ in } t_2 \Rightarrow H' \vdash *v \mid \Gamma' \mid \Delta$$

which has two possible derivations:

1.

$$\frac{H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma_1 \mid \Delta_1}{H \vdash \text{let unit} = t_1 \text{ in } t_2 \rightsquigarrow H' \vdash \text{let unit} = t'_1 \text{ in } t_2 \mid \Gamma_1 \mid \Delta_1} \rightsquigarrow^{\text{APP}} \frac{H' \vdash \text{let unit} = t'_1 \text{ in } t_2 \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid \Delta_2}{H \vdash \text{let unit} = t_1 \text{ in } t_2 \Rightarrow H'' \vdash *v \mid \Gamma_1 + \Gamma_2 \mid \Delta_1 + \Delta_2} \text{EXT}$$

By conservation on the single reduction for t_1 we get type preservation which gives us $\Gamma_1 \vdash t'_1 : 1$ and thus $\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t'_1 \text{ in } t_2 : *B$, and also that $H' \bowtie (\Gamma_0 + \Gamma')$. Thus we can induct on the multi-reduction here to get $a \in \text{arrRefs}(v)$:

$$\begin{aligned} a \mapsto_1 t' \in H' &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

Applying this by transitivity (and instantiation) with Lemma B1 gives us the goal:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

2. Alternatively $t_1 = \text{unit}$ such that the typing is:

$$\frac{\frac{[\Gamma_1] \vdash \text{unit} : 1}{\Gamma_2 \vdash \text{let unit} = \text{unit in } t_2 : *B}^{1_I} \quad \Gamma_2 \vdash t_2 : *B}{\Gamma_2 \vdash \text{let unit} = \text{unit in } t_2 : *B}^{1_E}$$

and we have reduction:

$$\frac{\frac{H \vdash \text{let unit} = \text{unit in } t \rightsquigarrow H \vdash t \mid \emptyset \mid \emptyset \rightsquigarrow^{\beta_{\text{unit}}} H \vdash t \Rightarrow H' \vdash t'' \mid \Gamma_2 \mid \Delta_2}{H \vdash \text{let unit} = \text{unit in } t \Rightarrow H' \vdash *v \mid \Gamma_2 \mid \Delta_2}^{\text{EXT}}}{H \vdash \text{let unit} = \text{unit in } t \Rightarrow H' \vdash *v \mid \Gamma_2 \mid \Delta_2}^{\text{EXT}}$$

Thus we arrive at the goal by induction on the second premise here.

- (return), (bind)
All trivially satisfy the theorem since the typing cannot conclude with a type of form $*A$.
- (nec)

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A}^{\text{NEC}}$$

Then there is a possible multi-reduction:

$$\frac{\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash *t \rightsquigarrow H' \vdash *t' \mid \Gamma \mid \Delta}^{\rightsquigarrow_*} \quad H' \vdash *t' \Rightarrow H'' \vdash *v \mid \Gamma_2 \mid \Delta_2}{H \vdash *t \rightsquigarrow H'' \vdash *v \mid \Gamma_2 \mid \Delta_2}^{\text{EXT}}$$

By Lemma B1 on the single reduction and induction on the second premise, we then get

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H' \end{aligned}$$

and

$$\begin{aligned} a \mapsto_1 t' \in H' &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H') &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

which we can combine together via instantiation and transitivity to get the goal:

$$\begin{aligned} a \mapsto_1 t' \in H &\implies \exists t''. a \mapsto_1 t'' \in H'' \\ \wedge a \notin \text{dom}(H) &\implies \exists t''. a \mapsto_1 t'' \in H'' \end{aligned}$$

- (newArray) (readArray) (writeArray) (deleteArray) are all values and have no reduction, so the case is trivial here.

□

B.5 Value lemma

Lemma B2 (Value lemma). *Given $\Gamma \vdash v : A$ then depending on the type, the shape of v can be inferred:*

- $A = A' \rightarrow B$ then $v = \lambda x.t$ or a partially applied primitive term p
- $A = !A'$ then $v = !t$
- $A = A' \otimes B'$ then $v = (t_1, t_2)$
- $A = 1$ then $v = \text{unit}$
- $A = *A'$ then $v = *v'$
- $A = \mathbb{N}$ then $v = n$
- $A = \mathbb{F}$ then $v = f$
- $A = \text{Array } \mathbb{F}$ then $v = a$

Proof. Recall the value sub terms grammar is:

$$v ::= (t_1, t_2) \mid \text{unit} \mid *t \mid !t \mid \lambda x.t \mid i \mid a \mid p \quad (\text{value terms sub-grammar})$$

where p are partially-applied primitives, e.g., `newArray`, `readArray`, `readArray (*a)`. i.e.,

$$p ::= \text{newArray} \mid \text{readArray} \mid \text{readArray } (*a) \\ \mid \text{writeArray} \mid \text{writeArray } (*a) \mid \text{writeArray } (*a) i \mid \text{deleteArray}$$

We then proceed by case analysis on the type A to match the structure of the lemma. In each case we must consider what possible values can be assigned the type A and by which rules.

In all cases, there exists a derivation based on dereliction, e.g., for the case where $A = A' \multimap B$

$$\frac{\Gamma, x : A' \vdash t : A' \multimap B}{\Gamma, x : [A'] \vdash t : A' \multimap B} \text{DER}$$

in which case we can apply induction on the premise to get the result since the term is preserved between the premise and the conclusion. We elide handling this separately each time in the cases that follow as the reasoning through dereliction is the same each time.

- $A = A' \rightarrow B$ then there are two classes of possible typing:

- Abstract term:

$$\frac{\Gamma, x : A' \vdash t : B}{\Gamma \vdash \lambda x.t : A' \multimap B} \text{ABS}$$

thus $v = \lambda x.t$ as in the lemma statement.

- Primitive term p formed by an application of zero or more values to a primitive operation, of which there are then all seven possibilities:

1.

$$\frac{}{[\Gamma] \vdash \text{newArray} : \mathbb{N} \multimap *(\text{Array } \mathbb{F})} \text{new}$$

thus $v = \text{newArray}$

2.

$$\frac{}{[\Gamma] \vdash \text{readArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{read}$$

thus $v = \text{readArray}$

3.

$$\frac{\frac{\overline{[\Gamma] \vdash \text{readArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ read}}{\overline{[\Gamma], a : \text{Array } A \vdash *a : *(\text{Array } A)} \text{ *array}}}{[\Gamma], a : \text{Array } A \vdash \text{readArray}(*a) : \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ APP}$$

 thus $v = \text{readArray}(*a)$

4.

$$\overline{[\Gamma] \vdash \text{writeArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})} \text{ write}$$

 thus $v = \text{writeArray}$

5.

$$\frac{\frac{\overline{[\Gamma] \vdash \text{writeArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})} \text{ write}}{\overline{[\Gamma], a : \text{Array } A \vdash *a : *(\text{Array } A)} \text{ *array}}}{[\Gamma], a : \text{Array } A \vdash \text{writeArray}(*a) : \mathbb{N} \multimap \mathbb{F} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ APP}$$

 thus $v = \text{writeArray}(*a)$

$$\frac{\frac{\overline{[\Gamma] \vdash \text{writeArray} : *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})} \text{ write}}{\overline{[\Gamma], a : \text{Array } A \vdash *a : *(\text{Array } A)} \text{ *array}}}{\frac{\overline{[\Gamma], a : \text{Array } A \vdash \text{writeArray}(*a) : \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ APP} \quad \emptyset \vdash i : \mathbb{N}}{[\Gamma], a : \text{Array } A \vdash \text{writeArray}(*a) i : \mathbb{F} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{ APP}}$$

 thus $v = \text{writeArray}(*a)$

7.

$$\overline{[\Gamma] \vdash \text{deleteArray} : *(\text{Array } \mathbb{F}) \multimap 1} \text{ del}$$

 thus $v = \text{deleteArray}$

 – $A = !A'$ then there is only one possible non-derelection typing of a value at that type:

$$\frac{[\Gamma] \vdash t : A'}{[\Gamma] \vdash !t : !A'} !_I$$

 thus $v = !t$ as in the lemma statement.

 – $A = A' \otimes B'$ then there is only one possible non-derelection typing of a value at that type:

$$\frac{\Gamma_1 \vdash t_1 : A' \quad \Gamma_2 \vdash t_2 : B'}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A' \otimes B'} \otimes_I$$

 thus $v = (t_1, t_2)$ as in the lemma statement.

 – $A = 1$ then there is only one possible non-derelection typing of a value at that type:

$$\overline{[\Gamma] \vdash \text{unit} : 1} 1_I$$

 thus $v = \text{unit}$ as in the lemma statement.

- $A = *A'$ then there is only one possible non-derelection typing of a value at that type:

$$\frac{\emptyset \vdash t : A'}{[I] \vdash *t : *A'} \text{ NEC}$$

thus $v = *t$ as in the lemma statement.

- $A = \mathbb{N}$ then $v = n$ Trivial case on typing of constants which is elided in this paper for brevity (but covered by the core type theory of Granule for example).
- $A = \mathbb{F}$ then $v = f$ Trivial case on typing of constants which is elided in this paper for brevity (but covered by the core type theory of Granule for example).
- $A = \text{Array } \mathbb{F}$ then $v = a$ then the only possible typing that is a value is given by:

$$\overline{[I], a : \text{Array } A \vdash a : \text{Array } A} \text{ arr}$$

(and since the only type of arrays is \mathbb{F} currently).

□

B.6 Progress

Theorem 6 (Progress). *Values of the heap model v are given by:*

$$v ::= (t_1, t_2) \mid \text{unit} \mid *t \mid !t \mid \lambda x.t \mid i \mid a \mid p \quad (\text{value terms sub-grammar})$$

where p are partially-applied primitives, e.g., `newArray`, `readArray`, `readArray(*a)`. Given $\Gamma \vdash t : A$, then t is either a value, or if $H \bowtie \Gamma_0 + \Gamma$ there exists a heap H' , term t' , usage context Δ , and context Γ' such that $H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma' \mid \Delta$.

Proof. By induction on typing.

- (var)

$$\overline{[I], x : A \vdash x : A} \text{ VAR}$$

with $H \bowtie [I], x : A$ which by inversion of heap compatibility could imply either $H = H', x \mapsto_1 t'$ or $H = H', x \mapsto_\omega t'$, for which each has a reduction:

1. $H', x \mapsto_1 t'$

$$\overline{H, x \mapsto_1 t \vdash x \rightsquigarrow H \vdash t \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR1}}$$

2. $H', x \mapsto_\omega t'$

$$\overline{H, x \mapsto_\omega t \vdash x \rightsquigarrow H, x \mapsto_\omega t \vdash t \mid \emptyset \mid x : 1} \rightsquigarrow_{\text{VAR}\omega}$$

- (abs)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B} \text{ ABS}$$

A value.

- (app)

$$\frac{\Gamma_1 \vdash t_1 : A \multimap B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1 t_2 : B} \text{ APP}$$

By induction on the first premise, there are two possibilities.

1. t_1 is a value and therefore by the value lemma there are a number of choices:

- $t_1 = \lambda x. t'_1$. Therefore we can reduce by \rightsquigarrow_β .

- $t_1 = \text{newArray}$ therefore $A = \mathbb{N}$

Therefore we induct on the second argument:

- * t_2 is a value and therefore by the value lemma (Lemma B2) $t_2 = n$ and thus the typing is:

$$\frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \text{newArray } n : *(\text{Array } \mathbb{F})} \text{TyDerivedNewArray}$$

with $H \bowtie (\Gamma_0 + \Gamma)$.

Thus there is a reduction as follows:

$$\frac{a \# H}{H \vdash \text{newArray } n \rightsquigarrow H, a \mapsto_{\mathbf{r}} \mathbf{arr} \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{newArray}}$$

- * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{newArray } t_2 \rightsquigarrow H' \vdash \text{newArray } t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

- $t_1 = \text{readArray}$ therefore $A = *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})$

- * t_2 is a value and therefore by the value lemma on $*(\text{Array } \mathbb{F})$ (Lemma B2) $t_2 = (*a)$ and thus $t_1 t_2 = \text{readArray } (*a)$ which is also a value.

- * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{readArray } t_2 \rightsquigarrow H' \vdash \text{readArray } t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

- $t_1 = \text{readArray } (*a)$ therefore $A = \mathbb{N} \multimap \mathbb{F} \otimes *(\text{Array } \mathbb{F})$

- * t_2 is a value and therefore by the value lemma on $\Gamma_2 \vdash t_2 : \mathbb{N}$ (Lemma B2) implies $t_2 = n$ and thus the typing is refined at runtime as follows:

$$\frac{\frac{[I_1], a : \text{Array } \mathbb{F} \vdash a : (\text{Array } \mathbb{F}) \text{ arr}}{[I_1], a : \text{Array } \mathbb{F} \vdash *a : *(\text{Array } \mathbb{F})} \text{*array} \quad \Gamma_2 \vdash i : \mathbb{N}}{[I_1] + \Gamma_2, a : \text{Array } \mathbb{F} \vdash \text{readArray } a \ i : \mathbb{F} \otimes *(\text{Array } \mathbb{F})} \text{TyDerivedReadArray}$$

with $H' \bowtie (\Gamma_0 + [I_1] + \Gamma_2, a : \text{Array } \mathbb{F})$, and by the heap compatibility rule for array references there exists some H such that $H' = H, a \mapsto_{\mathbf{r}} \mathbf{arr}$.

Then there is a reduction as follows:

$$\overline{H, a \mapsto_{\mathbf{r}} (\mathbf{arr}[i] = v) \vdash \text{readArray } (*a) \ i \rightsquigarrow H, a \mapsto_{\mathbf{r}} (\mathbf{arr}[i] = v) \vdash (v, *a) \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{readArray}}$$

- * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{readArray } (*a) \ t_2 \rightsquigarrow H' \vdash \text{readArray } (*a) \ t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

- $t_1 = \text{writeArray}$ therefore $A = *(\text{Array } \mathbb{F}) \multimap \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})$

- * t_2 is a value therefore by the value lemma (Lemma B2) $t_2 = (*a)$ and thus $t_1 t_2 = \text{writeArray } (*a)$ which is also a value.

- * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{writeArray } t_2 \rightsquigarrow H' \vdash \text{writeArray } t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

- $t_1 = \text{writeArray } (*a)$ therefore $A = \mathbb{N} \multimap \mathbb{F} \multimap *(\text{Array } \mathbb{F})$
 - * t_2 is a value therefore by the value lemma (Lemma B2) $t_2 = n$ and thus $t_1 t_2 = \text{writeArray } (*a) n$ which is also a value.
 - * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{writeArray } (*a) t_2 \rightsquigarrow H' \vdash \text{writeArray } (*a) t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

- $t_1 = \text{writeArray } (*a) i$ therefore $A = \mathbb{F} \otimes *(\text{Array } \mathbb{F})$
 - * t_2 is a value and therefore by the value lemma on $\Gamma_2 \vdash t_2 : \mathbb{F}$ (Lemma B2) implies $t_2 = f$ and thus the typing is refined at runtime as follows:

$$\frac{\frac{[I_1], a : \text{Array } \mathbb{F} \vdash a : (\text{Array } \mathbb{F}) \text{ arr}}{[I_1], a : \text{Array } \mathbb{F} \vdash *a : *(\text{Array } \mathbb{F})} \text{*array} \quad \Gamma_2 \vdash i : \mathbb{N} \quad \Gamma_3 \vdash f : \mathbb{F}}{[I_1] + \Gamma_2 + \Gamma_3, a : \text{Array } \mathbb{F} \vdash \text{writeArray } a i f : *(\text{Array } \mathbb{F})} \text{TYDERIVEDWRITEARRAY}$$

with $H' \bowtie (\Gamma_0 + [I_1] + \Gamma_2 + \Gamma_3, a : \text{Array } \mathbb{F})$, and by the heap compatibility rule for array references there exists some H such that $H' = H, a \mapsto_r \mathbf{arr}$.

Then there is a reduction as follows:

$$\frac{H, a \mapsto_r \mathbf{arr} \vdash \text{writeArray } (*a) i v \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset}{H, a \mapsto_r \mathbf{arr} \vdash \text{writeArray } (*a) i v \rightsquigarrow H, a \mapsto_r (\mathbf{arr}[i] = v) \vdash *a \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{writeArray}}$$

- * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{writeArray } (*a) i t_2 \rightsquigarrow H' \vdash \text{writeArray } (*a) i t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

- $t_1 = \text{deleteArray}$ therefore $A = *(\text{Array } \mathbb{F}) \multimap 1$
 - * t_2 is a value and therefore by the value lemma (Lemma B2) $t_2 = a$ thus the typing is refined at runtime as follows:

$$\frac{\frac{[I], a : \text{Array } \mathbb{F} \vdash a : (\text{Array } \mathbb{F}) \text{ arr}}{[I], a : \text{Array } \mathbb{F} \vdash *a : *(\text{Array } \mathbb{F})} \text{*array}}{[I], a : \text{Array } \mathbb{F} \vdash \text{deleteArray } : 1} \text{TYDERIVEDDELETEARRAY}$$

with $H' \bowtie (\Gamma_0 + [I], a : \text{Array } \mathbb{F})$, and by the heap compatibility rule for array references there exists some H such that $H' = H, a \mapsto_r \mathbf{arr}$.

There there is a reduction as follows:

$$\frac{H, a \mapsto_r \mathbf{arr} \vdash \text{deleteArray } (*a) \rightsquigarrow H \vdash \text{unit} \mid \emptyset \mid \emptyset}{H, a \mapsto_r \mathbf{arr} \vdash \text{deleteArray } (*a) \rightsquigarrow H \vdash \text{unit} \mid \emptyset \mid \emptyset} \rightsquigarrow_{\text{deleteArray}}$$

- * t_2 is not a value and thus has a reduction, therefore we can build the compound reduction:

$$\frac{H \vdash t_2 \rightsquigarrow H' \vdash t'_2 \mid \Gamma \mid \Delta}{H \vdash \text{deleteArray } t_2 \rightsquigarrow H' \vdash \text{deleteArray } t'_2 \mid \Gamma \mid \Delta} \rightsquigarrow_{\text{prim}}$$

2. otherwise, by induction on the premise we then have that there exists heap H' , term t'_1 , usage context Δ , and context Γ' such that $H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta$. Therefore we can reduce by $\rightsquigarrow_{\text{APP}}$.

– (derelict)

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A] \vdash t : B} \text{DER}$$

Goal achieved immediately by induction on the premise.

– (bangIntro)

$$\frac{[\Gamma] \vdash t : A}{[\Gamma] \vdash !t : !A} !_I$$

A value.

– (bangElim)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : [A] \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let } !x = t_1 \text{ in } t_2 : B} !_E$$

By induction on the first premise, there are two possibilities.

1. t_1 is a value and therefore by the value lemma $t_1 = !t'_1$. This refines the typing as follows:

$$\frac{\frac{[\Gamma_1] \vdash t'_1 : A}{[\Gamma_1] \vdash !t'_1 : !A} !_I \quad \Gamma_2, x : [A] \vdash t_2 : B}{[\Gamma_1] + \Gamma_2 \vdash \text{let } !x = !t'_1 \text{ in } t_2 : B} !_E$$

Therefore we can reduce by $\rightsquigarrow_{! \beta}$.

2. Otherwise by induction on the premise we then have that there exists heap H' , term t'_1 , usage context Δ , and context Γ' such that $H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta$. Therefore we can reduce by $\rightsquigarrow_{\text{LET}!}$.

– (pairIntro)

$$\frac{\Gamma_1 \vdash t_1 : A \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash (t_1, t_2) : A \otimes B} \otimes_I$$

A value.

– (pairElim)

$$\frac{\Gamma_1 \vdash t_1 : A \otimes B \quad \Gamma_2, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 \vdash \text{let } (x, y) = t_1 \text{ in } t_2 : C} \otimes_E$$

By induction on the first premise, there are two possibilities.

1. t_1 is a value and therefore by the value lemma $t_1 = (t'_1, t''_1)$. This refines the typing as follows:

$$\frac{\frac{\Gamma_1 \vdash t'_1 : A \quad \Gamma_2 \vdash t''_1 : B}{\Gamma_1 + \Gamma_2 \vdash (t'_1, t''_1) : A \otimes B} \otimes_I \quad \Gamma_3, x : A, y : B \vdash t_2 : C}{\Gamma_1 + \Gamma_2 + \Gamma_3 \vdash \text{let } (x, y) = (t'_1, t''_1) \text{ in } t_2 : C} \otimes_E$$

Therefore we can reduce by $\rightsquigarrow_{\otimes \beta}$.

2. Otherwise by induction on the premise we then have that there exists heap H' , term t'_1 , usage context Δ , and context Γ' such that $H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta$. Therefore we can reduce by $\rightsquigarrow_{\text{LET} \otimes}$.

– (unitIntro)

$$\frac{}{[\Gamma] \vdash \text{unit} : 1} 1_I$$

A value.

– (unitElim)

$$\frac{\Gamma_1 \vdash t_1 : 1 \quad \Gamma_2 \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \text{let unit} = t_1 \text{ in } t_2 : B} 1_E$$

By induction on the first premise, there are two possibilities.

1. t is a value and therefore by the value lemma $t = \text{unit}$. Therefore we can reduce by $\rightsquigarrow_{\beta \text{unit}}$.
2. Otherwise by induction on the premise we then have that there exists heap H' , term t'_1 , usage context Δ , and context Γ' such that $H \mid - t > H' \mid - t'_1; G'; D$. Therefore we can reduce by $\rightsquigarrow_{\text{LETunit}}$.

– (borrow)

$$\frac{\Gamma \vdash t : *A}{\Gamma \vdash \&t : !A} \text{BORROW}$$

By induction on the premise, there are two possibilities.

1. t is a value and therefore by the value lemma $t = *t'$. Therefore we can reduce by $\rightsquigarrow_{\&\beta}$.
2. Otherwise by induction on the premise we then have that there exists heap H' , term t' , usage context Δ , and context Γ' such that $H \mid - t > H' \mid - t'; G'; D$. Therefore we can reduce by $\rightsquigarrow_{\&}$.

– (copy)

$$\frac{\Gamma_1 \vdash t_1 : !A \quad \Gamma_2, x : *A \vdash t_2 : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } t_1 \text{ as } x \text{ in } t_2 : !B} \text{COPY}$$

By induction on the first premise, there are two possibilities.

1. t_1 is a value and therefore by the value lemma $t_1 = !t$. Then there are again two possibilities.
 - (a) t is itself a value. This refines the typing as follows:

$$\frac{\frac{[\Gamma_1] \vdash v : A}{[\Gamma_1] \vdash !v : !A} !_I \quad \Gamma_2, x : *A \vdash t_2 : !B}{[\Gamma_1] + \Gamma_2 \vdash \text{copy } !v \text{ as } x \text{ in } t_2 : !B} \text{COPY}$$

Therefore we can reduce using $\rightsquigarrow_{\text{copy}\beta}$.

- (b) Otherwise by induction on the premise we then have that there exists heap H' , term t' , usage context Δ , and context Γ' such that $H \mid - t > H' \mid - t'; G'; D$. Therefore we can reduce by $\rightsquigarrow_{\text{copy}!}$.
2. Otherwise by induction on the premise we then have that there exists heap H' , term t'_1 , usage context Δ , and context Γ' such that $H \vdash t_1 \rightsquigarrow H' \vdash t'_1 \mid \Gamma' \mid \Delta$. Therefore we can reduce by $\rightsquigarrow_{\text{copy}}$.

– (nec)

$$\frac{\emptyset \vdash t : A}{[\Gamma] \vdash *t : *A} \text{NEC}$$

Then by induction either t is a value so $*t$ is a value or there is a reduction which we can plug into the following reduction:

$$\frac{H \vdash t \rightsquigarrow H' \vdash t' \mid \Gamma \mid \Delta}{H \vdash *t \rightsquigarrow H' \vdash *t' \mid \Gamma \mid \Delta} \rightsquigarrow_*$$

– (array)

$$\frac{}{[\Gamma], a : \text{Array } A \vdash a : \text{Array } A} \text{arr}$$

A value.

– (uniqueArray)

$$\frac{}{[\Gamma], a : \text{Array } A \vdash *a : *(\text{Array } A)} \text{*array}$$

A value.

□

B.7 Soundness of heap model wrt. equational theory

Theorem 7 (Soundness with respect to the equational theory). *For all t_1, t_2 such that $\Gamma \vdash t_1 : A$ and $\Gamma \vdash t_2 : A$ and $t_1 \equiv t_2$ and given H such that $H \bowtie \Gamma$, there exists a value (irreducible term) v and $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2$ such that there are full β -reductions to the same value*

$$H \vdash t_1 \Rightarrow_\beta H' \vdash v \mid \Gamma_1 \mid \Delta_1 \quad \wedge \quad H \vdash t_2 \Rightarrow_\beta H' \vdash v \mid \Gamma_2 \mid \Delta_2$$

Proof. – (unitR)

$$\frac{}{\text{copy } t \text{ as } x \text{ in } \&x \equiv t} \text{EQUNITR}$$

With the following typing derivation for the LHS:

$$\frac{\Gamma_1 \vdash t : !A \quad \frac{x : *A \vdash x : *A}{x : *A \vdash \&x : !A} \text{BORROW}}{\Gamma_1 \vdash \text{copy } t \text{ as } x \text{ in } \&x : !A} \text{COPY}$$

and from the premise we have $H \bowtie \Gamma_1$.

By conservation, we have that a well-typed term reduces to a value and so (by a value lemma) we can assume that $H \vdash t$ reduces to a value $H \vdash !t'$ for some t' (*) and then reduction under the ! yields $H'' \vdash !v$ (**). This gives the reduction of the RHS to a value.

We then have the following reduction sequence (where we elide the output binding context and usage context as they are not needed in the proof):

$$\begin{aligned} & H \vdash \text{copy } t \text{ as } x \text{ in } \&x \\ (*) & \rightsquigarrow^* H' \vdash \text{copy } !t' \text{ as } x \text{ in } \&x \\ (**) & \rightsquigarrow^* H' \vdash \text{copy } !v \text{ as } x \text{ in } \&x \\ & \rightsquigarrow_{\text{copy}\beta} \rightarrow H', x \mapsto_1 *v \vdash \&x \\ & \rightsquigarrow_{\text{copy}} + \rightsquigarrow_{\text{VAR1}} \rightarrow H', x \mapsto_1 *v \vdash \&*v \\ & \rightsquigarrow_{\&\beta} \rightarrow H', x \mapsto_1 *t' \vdash !v \end{aligned}$$

which matches the result of deep reducing t (using the full beta) rules replaying (*) and (**) to get the same result of $!v$.

– (unitL)

$$\frac{}{\text{copy } \&v \text{ as } x \text{ in } t' \equiv [v/x]t'} \text{EQUNITL}$$

With the following typing derivation for the LHS:

$$\frac{\frac{\Gamma_1 \vdash v : *A}{\Gamma_1 \vdash \&v : !A} \text{BORROW} \quad \Gamma_2, x : *A \vdash t' : !B}{\Gamma_1 + \Gamma_2 \vdash \text{copy } \&v \text{ as } x \text{ in } t' : !B} \text{COPY}$$

and from the premise we have $H \bowtie \Gamma_1 + \Gamma_2$.

By the value lemma (Lemma B2) we know that $v = *v'$ therefore we know that can perform the following reduction: (where we elide the output binding context and usage context as they are not needed in the proof):

$$\begin{aligned} & H \vdash \text{copy } \&(*v') \text{ as } x \text{ in } t' \\ \rightsquigarrow_{\text{copy}! + \rightsquigarrow_{\&\beta}} & [H']_{\omega}, H'' \vdash \text{copy } !v' \text{ as } x \text{ in } t' \\ \rightsquigarrow_{\text{copy}\beta} & [H']_{\omega}, H'', H''', x \mapsto_1 *(\theta(v)) \vdash t' \end{aligned}$$

where $H = H', H''$ and $\text{dom}(H') \equiv \text{arrRefs}(v)$ and $(H''', \theta) \equiv \text{copy}(H')$.

Then we have to sub-cases that refine this reduction

- $v' = a$ (an array reference) then we can assume that that $H = H', a \mapsto_r \mathbf{arr}$ and we specialise the above reduction to get:

$$\begin{aligned} & H', a \mapsto_r \mathbf{arr} \vdash \text{copy } \&(*a) \text{ as } x \text{ in } t' \\ \rightsquigarrow_{\text{copy}! + \rightsquigarrow_{\&\beta}} & H', a \mapsto_{\omega} \mathbf{arr} \vdash \text{copy } !v' \text{ as } x \text{ in } t' \\ \rightsquigarrow_{\text{copy}\beta} & H', a \mapsto_{\omega} \mathbf{arr}, a' \mapsto_{\omega} \mathbf{arr}, x \mapsto_1 *a' \vdash t' \end{aligned}$$

Since t' cannot contain an a as it is a non-runtime term, then the resulting term here (in its heap context) reduces down to a value that is equal $[*a/x]t'$ by full beta reduction; we have essentially just α -renamed the array reference here from a' to a .

- v' is not an array reference, therefore $\text{arrRefs}(v') = \emptyset$ and $\theta = \emptyset$ and thus we have $H'' = \emptyset$, so the reduction specialises to:

$$\begin{aligned} & H', H'' \vdash \text{copy } \&(*v') \text{ as } x \text{ in } t' \\ \rightsquigarrow_{\text{copy}! + \rightsquigarrow_{\&\beta}} & H', H'' \vdash \text{copy } !v' \text{ as } x \text{ in } t' \\ \rightsquigarrow_{\text{copy}\beta} & H', H'', x \mapsto_{\omega} *v' \vdash t' \end{aligned}$$

which is equal to $[*v'/x]t'$ after full reduction.

– (assoc)

$$\frac{x \# t_3}{\text{copy } t_1 \text{ as } x \text{ in } (\text{copy } t_2 \text{ as } y \text{ in } t_3) \equiv \text{copy } (\text{copy } t_1 \text{ as } x \text{ in } t_2) \text{ as } y \text{ in } t_3} \text{EQASSOC}$$

Thus we assume a reduction from t_1 to $!t'_1$ and then to $!v_1$ and likewise a reduction from t_2 to $!t'_2$ and then to $!v_2$ then we reduce as follows on the LHS:

$$\begin{aligned} & H \vdash \text{copy } t_1 \text{ as } x \text{ in } (\text{copy } t_2 \text{ as } y \text{ in } t_3) \\ \rightsquigarrow * & H' \vdash \text{copy } !v_1 \text{ as } x \text{ in } (\text{copy } t_2 \text{ as } y \text{ in } t_3) \\ \rightsquigarrow & H', x \mapsto_1 *v_1 \vdash \text{copy } t_2 \text{ as } y \text{ in } t_3 \\ \rightsquigarrow * & H'' \vdash \text{copy } !v_2 \text{ as } y \text{ in } t_3 \\ \rightsquigarrow & H'', y \mapsto_1 *v_2 \vdash t_3 \end{aligned}$$

and on the RHS:

$$\begin{aligned} & H \vdash \text{copy } (\text{copy } t_1 \text{ as } x \text{ in } t_2) \text{ as } y \text{ in } t_3 \\ \rightsquigarrow * & H' \vdash \text{copy } (\text{copy } !v_1 \text{ as } x \text{ in } t_2) \text{ as } y \text{ in } t_3 \\ \rightsquigarrow & H', x \mapsto_1 *v_1 \vdash \text{copy } t_2 \text{ as } y \text{ in } t_3 \\ \rightsquigarrow * & H'' \vdash \text{copy } !v_2 \text{ as } y \text{ in } t_3 \\ \rightsquigarrow & H'', y \mapsto_1 *v_2 \vdash t_3 \end{aligned}$$

matching in both sides. □