Substructural Typestates
(Technical Appendix)

Filipe Militão
Carnegie Mellon University &
Universidade Nova de Lisboa
filipe.militao@cs.cmu.edu

Jonathan Aldrich
Carnegie Mellon University
jonathan.aldrich@cs.cmu.edu

Luís Caires
Universidade Nova de Lisboa
luis.caires@di.fct.unl.pt

Contents

A Abbreviations .............................................. 1
B Proofs .................................................. 3
B.1 Well-Formed Types and Environments ........... 3
B.2 Subtyping Inversion Lemma ...................... 3
B.3 Store Typing ........................................ 4
B.4 Values Inversion Lemma .......................... 6
B.5 Substitution .......................................... 9
B.6 Free Variables Lemma .............................. 12
B.7 Well-Form Lemmas .................................. 15
B.8 Substitution Lemma ................................ 15
B.9 Values Lemma ........................................ 23
B.10 Preservation ......................................... 25
B.11 Progress ............................................. 29

A. Abbreviations

We define a few convenient abbreviations that were used in the examples and can be encoded into the core language without the need for additional constructs.

- Let “compressed” expressions are encoded as:

```
! e : \ let x = e in !p end
```

and similar to the remaining constructs (such as assignment, open, etc).

- **Sequence:**

```
e_0; e_1 \equiv \ let x = e_0 in e_1 end
```

Where `x` is not free in `e_1` and `e_0`. Note that `e_0` must have a pure type since the value will be discarded. However, it can have capabilities stacked on top of it since those get automatically threaded to `e_1`.

- **Unlabeled pairs (and their generalization, tuples),** which can be encoded as function and application.

Our core language only includes `choice` labeled products, so that the programmer must pick one (and only one) of a set of available fields — all of which produce the same effect in the `Δ` environment. The alternative would be to require all fields to be used, so that it is a linear labeled pair (where the order matters). We now show how unlabeled pairs can be encoded in the language, and leave out the generalization for arbitrary (but fixed length) tuples since it is straightforward.

```
Γ; Δ_0 + e_0 : A_0 + Δ_1
Γ; Δ_1 + e_1 : A_1 + Δ_2
Γ; Δ_0 + [e_0, e_1] : [A_0, A_1] + Δ_2
```

Can be encoded as:

```
|e_0, e_1| \equiv  
let x_0 = e_0 in 
let x_1 = e_1 in 
< R > fun(x : Δ_0 \Rightarrow Δ_1). ( fun(x_0 : A_0). fun(x_1 : A_1). e_f )
end
```

where `e_0 : A_0, e_1 : A_1` and `e_f : A_f`.

- **Recursion:** We use the traditional `call-by-value Y-combinator` encoded in our core language to provide recursion without using additional typing rules or reductions.

Note that using a special construct, such as “rec x.e” would require changing the `substitution lemma` since if `rec` is a value,
then no further reductions can occur, and if it is not a value, then the substitution lemma must account for expressions, not
just values.

\begin{verbatim}
let fix = \langle A \rightarrow B \rangle \rightarrow \langle A \rightarrow B \rangle .
let r = \langle x : \text{rec } X . \langle X \rightarrow (A \rightarrow B) \rangle \rangle .
X (\langle x : \text{rec } X . \langle X \rightarrow (A \rightarrow B) \rangle \rangle ) (
\text{ref } \langle x : \text{rec } X . \langle X \rightarrow (A \rightarrow B) \rangle \rangle ) (\langle x : \text{rec } X . \langle X \rightarrow (A \rightarrow B) \rangle \rangle )
end :
\forall A.\forall B.\forall X . \langle (A \rightarrow B) \rightarrow (A \rightarrow B) \rangle
\end{verbatim}

Noting that \(x\) in line 3 has types:

\begin{verbatim}
\langle x : \text{rec } X . \langle X \rightarrow (A \rightarrow B) \rangle \rangle .\langle X \rightarrow (A \rightarrow B) \rangle
\end{verbatim}

Making the argument of \(\text{fix}\) in that line to be \(A \rightarrow B\), and \(r\) to be of type:

\begin{verbatim}
\text{rec } X . \langle X \rightarrow (A \rightarrow B) \rangle .\langle X \rightarrow (A \rightarrow B) \rangle
\end{verbatim}

which, applied to itself, yields the type of the function without the
recursive argument visible \((A \rightarrow B)\).

Therefore, to use recursion, we must create a function that takes
the recursive argument visible \((X \rightarrow \langle X \rightarrow (A \rightarrow B) \rangle \rangle\).

In the examples, we make use of the construct “\text{rec } x.e” to
define a recursive function (with body \(e\)), without having to
use the expanded notation, and that automatically threads all
location variables through its argument(s).

**Shorter delete rule.**

In the examples, we use a shorter (and more limited) delete
typing rule to avoid having to carry existential types around.

\begin{verbatim}
delete \text{example } x : \text{ref } t_0 . y . y = (\text{delete } < t_0 , x > ) \text{in } y \text{end}
\end{verbatim}

where \(x : \text{ref } t_0\) and \(y : A\) where \(t_0\) does not occur in \(A\) (and
therefore does not need to be packed to leave that scope).

Similar functionality could be achieved with the following typing
rule (that is used in the prototype):

\[
\frac{}{\Gamma ; \Delta_0 \vdash e : \text{ref } p \rightarrow \Delta_1, \text{rw } p \rightarrow A}
\]

**Girards’ encoding of existential types.**

However, this abbreviation is not used since it makes the use of
existential types slightly more complex and less clear. Nonetheless,
we leave it here as an observation on how it could be achieved.

An existential type can be encoded into an universal type by
consider the packed type to be hidden inside an universally
quantified function that is not directly usable to client:

\[
\exists X . A \rightarrow \forall R . (\forall X . (A \rightarrow R ) \rightarrow R )
\]

where \(R\) is the result of the expression that uses the packed
existential and where \(X\) cannot occur in \(R\).

**Pack** if we have:

\[
\langle A_0 , e \rangle : \exists X A_1
\]

then it can be encoded as:

\[
\langle R \rangle (\langle \text{fun } (x : \forall X . (A_1 \rightarrow R )) . (x[A_0](e)) \rangle)
\]

so that it is a polymorphic function on \(R\), i.e. the result of
opening the packed existential.

**Open** if we have:

\[
\text{open } \langle X , x \rangle = e_0 \text{in } e_1 \text{end} : A_1
\]

where \(e_0 : \exists X A_0\), then it can be encoded as:

\[
e_0[A_1](\langle X \rangle (\langle \text{fun } (x : A_0 ) . e_1 \rangle))
\]
B. Proofs

B.1 Well-Formed Types and Environments

Our well-formed definition ensures that types are properly formed (i.e. type formation), be it in the environments or just in a regular type. Therefore, each type must have all the location variables it depends on declared in the corresponding Γ environment so that all location variables must be known in the same scope as the capability that refers a certain location variable. An analogous condition must hold for type variables.

Definition 1 (Well-Formed). We have the following cases (defined by induction on the structure of the type/environment):

- \( \Gamma \text{ wf} \) (Gamma)
  \[
  \frac{\Gamma \text{ wf} \quad \Gamma \vdash \text{type} \quad \Gamma, x : A \text{ wf}}{\Gamma, \lambda \, x. \, A \text{ type}}
  \]

- \( \Gamma \vdash \Delta \text{ wf} \) (Delta)
  \[
  \frac{\Gamma \vdash \Delta \text{ wf} \quad \Gamma \vdash A \text{ type}}{\Gamma \vdash (\Delta, A) \text{ wf}}
  \]

- \( \Gamma \vdash A \text{ type} \) (Type)
  \[
  \frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash A_0 \text{ type} \quad \Gamma \vdash A_1 \text{ type}}{\Gamma \vdash (A_0 \to A_1) \text{ type}}
  \]

\[
\]

B.2 Subtyping Inversion Lemma

Lemma 1 (Subtyping Inversion Lemma). We have the following cases for types (\( A \)) and for the linear typing environment (\( \Delta \)):

- (Type) If \( A <: A' \) then one of the following holds:
  1. \( A' = A \).
  2. if \( A = \lambda x : A_0 \) then either:
     (a) \( A' = \lambda x : A_0 \), or;
     (b) \( A' = A_0 \) and \( A_0 <: A_1 \), or;
     (c) \( A' = A_0 \).
  3. if \( A = A_0 \to A_1 \) then \( A' = A_0 \to A_1 \) and \( A_1 <: A_2 \) and \( A_2 <: A_0 \).
  4. if \( A = A_0 \sqcup A_1 \) then \( A' = A_1 \) or \( A_0 <: A_1 \) and \( A_1 <: A_0 \).
  5. if \( A = [\Gamma : A] \) then either:
     (a) \( A = [\Gamma : A] \) and \( A' = [\Gamma : A] \) and \( i > 0 \).
     (b) \( A = [\Gamma : A] \) and \( A' = [\Gamma : A] \) and \( A_0 <: A_1 \).
     (c) \( A = [\Gamma : A] \) and \( A' = [\Gamma : A] \).
  6. if \( A = \text{ref} p \text{ A} \) then \( A' = \text{ref} p \text{ A} \) and \( A_0 <: A_1 \).
  7. if \( A = \exists x : A_0 \text{ A} \) then \( A' = \exists x : A_0 \text{ A} \) and \( A_0 <: A_1 \).
  8. if \( A = \forall x : A_0 \text{ A} \) then \( A' = \forall x : A_0 \text{ A} \) and \( A_0 <: A_1 \).
  9. if \( A = \forall x : A_0 \text{ A} \) then \( A' = \forall x : A_0 \text{ A} \) and \( A_0 <: A_1 \).
  10. if \( A = \text{rec} \text{ X} \text{ A} \) then \( A' = \text{rec} \text{ X} \text{ A} \) and \( A_0 <: A_1 \).
  11. if \( A = \text{rec} \text{ X} \text{ A} \) then \( A' = \text{rec} \text{ X} \text{ A} \).

- (Delta) If \( \Delta <: \Delta' \) then one of the following holds:
  1. \( \Delta = \Delta' \).
  2. if \( \Delta = \lambda x : A_0 \text{ A} \) then \( \Delta' = \lambda x : A_0 \text{ A} \) and \( A_0 <: A_1 \).
  3. if \( \Delta = \exists x : A_0 \text{ A} \) then \( \Delta' = \exists x : A_0 \text{ A} \) and \( A_0 <: A_1 \).
  4. if \( \Delta = \forall x : A_0 \text{ A} \) then \( \Delta' = \forall x : A_0 \text{ A} \) and \( A_0 <: A_1 \).
  5. if \( \Delta = [\Gamma : A] \) then \( \Delta' = [\Gamma : A] \).

Proof. We only very informally sketch the proof, without going into detail on each case since they are straightforward to show.

1. (Type) By induction on the derivation of \( A <: A' \).
   Case (str:Symmetry) Case 1 of the definition.
   Case (str:TopLem) Case 2 (a) of the definition.
   Case (str:SubVar) Case 2 (b) of the definition.
   Case (str:Top) Case 2 (c) of the definition.
   Case (str:SubType) Case 3 of the definition.
   Case (str:Loc-Exists) Case 7 of the definition.
   Case (str:Loc-Forall) Case 8 of the definition.
   Case (str:Type-Exists) Case 9 of the definition.
   Case (str:Type-Forall) Case 10 of the definition.
   Case (str:Rec) Case 5 (b) of the definition.
   Case (str:Discard) Case 5 (a) of the definition.
   Case (str:ForallRec) Case 5 (c) of the definition.
   Case (str:Stacks) Case 4 of the definition.
   Case (str:Cap) Case 6 of the definition.
   Case (str:Com) Case 12 (a) of the definition.
   Case (str:Cong) Case 12 (b) of the definition.
2. (Delta) By induction on the derivation of $\Delta <: \Delta'$.

   Case (str:Symmetry) - Case 1 of the definition.
   Case (str:Var) - Case 2 of the definition.
   Case (str:Type) - Case 3 (a), 4 (b) and 4 (c) of the definition.
   Case (str:Star), left - Case 4 of the definition.
   Case (str:Star), right - Case 5 of the definition.
   Case (str:None) - Cases 7 (for $<, \text{right}$) and 6 (for $:, \text{left}$) of the definition.
   Case (str:Alternative-R) - Case 8 of the definition.
   Case (str:Alternative-L) - Case 8 of the definition.

B.3 Store Typing

We use the notation $\Gamma$ to mean that $\Gamma$ is closed in the sense of only containing $(p : \text{loc})$ elements and nothing else. Therefore, it only lists the known location constants. Similarly, we use $\Delta$ to mean that $\Delta$ is closed, so that it only includes capabilities (of the form: $\text{rw} \; \rho \; A$ — note the location constant $\rho$). There is no inconsistency with the notation of $A$ since if such type can only depend on closed environments (in order to be well-formed), then it too must be closed or it would not be well-formed.

Definition 2 (Store Typing).

\[
\begin{align*}
\text{(str:Empty)} & \quad \Gamma; \Delta \vdash H \quad \Gamma; \Delta, A_0 \vdash H \\
\text{(str:Loc)} & \quad \Gamma; \Delta \vdash H \quad \Gamma; \Delta, A_0, A_1 \vdash H \\
\text{(str:Star)} & \quad \Gamma; \Delta, A_0 \vdash H \\
\text{(str:Alternative-R)} & \quad \Gamma; \Delta, A_0 \oplus A_1 \vdash H \\
\text{(str:Alternative-L)} & \quad \Gamma; \Delta, \rho : \text{loc}; \Delta \vdash H \\
\text{(str:Binding)} & \quad \Gamma; \Delta, \Delta_0 \vdash H \\
\end{align*}
\]

Note that, since the added capability on (str:Binding) must still be well-formed, such implies that $\Gamma$ must contain $\rho$. For the same reason, $\rho$ must also not appear in $\Delta$ or $H$. On (str:Alternative), we only need one rule because such type is assumed to be commutative.

Lemma 2 (Store Typing Inversion Lemma). If

\[
\Gamma; \Delta \vdash H
\]

then one of the following holds:

1. $\Gamma = \cdot$ and $\Delta = \cdot$ and $H = \cdot$.
2. If $\Gamma = \Gamma', \rho : \text{loc}$ then $\Gamma'; \Delta \vdash H$.
3. If $\Delta = \Delta', A_0 \oplus A_1$ then $\Gamma; \Delta', A_0, A_1 \vdash H$.
4. If $\Delta = \Delta', \text{rw} \; \rho \; A$ and $H = H', \rho \leftrightarrow v$ then $\Gamma; \Delta', \Delta_0 \vdash H'$ and $\Gamma; \Delta_0 \vdash v : A \leftrightarrow v$.
5. If $\Delta = \Delta', \text{none}$ then $\Gamma; \Delta \vdash H$.
6. If $\Delta = \Delta', A_0 \oplus A_1$ then either:
   - $\Gamma; \Delta, A_0 \oplus A_1 \vdash H$;
   - $\Gamma; \Delta, A_1 \vdash H$.

(note that $\oplus$ is commutative)

Proof. Straightforward induction on the derivation of $\Gamma; \Delta \vdash H$. □
Lemma 3 (Subtyping Store Typing). If $\Gamma; \Delta \vdash H$ and $\Delta < \Sigma$ then $\Gamma; \Delta' \vdash H$.

Proof. By induction on the derivation of $\Gamma; \Delta \vdash H$.

Case (str:Empty) We have:

- $\vdash \Delta < \Sigma$ (1)
- $\vdash \Delta$ (2)

By (Subtyping Inversion Lemma) on (2), we have that either:

- $\Delta' = \Delta$ (2.1)
- $\vdash \Delta < \Sigma$ (2.2)

Thus, we conclude by (1).

Case (str:Loc) We have:

- $\Gamma, \rho : \text{loc}; \Delta \vdash H$
- $\Delta < \Sigma$

By induction hypothesis on (1), (2).

By inversion on (str:Loc) with (1). By induction hypothesis with (3) and (2).

Thus, we conclude.

Case (str:BaseType) We have:

- $\Gamma, \Delta, \rho : A \vdash H, \rho \rightarrow v$
- $\Delta < \Sigma$

By (Subtyping Inversion Lemma) on (2), we have that either:

- $\Delta' = \Delta, \rho : A$ (2.1)
- $\vdash \Delta < \Sigma$ (2.2)

Thus, we conclude.

Case (str:BaseType) We have:

- $\Gamma, \Delta, \rho : A \vdash H$
- $\Delta < \Sigma$

By induction hypothesis on (1), (2).

By inversion on (str:BaseType) with (1).

Thus, we conclude.

Case (str:None) We have:

- $\vdash \Delta, \rho : A \vdash H$
- $\Delta < \Sigma$

By (Subtyping Inversion Lemma) on (2), we have that either:

- $\Delta' = \Delta, \rho : A$ (2.1)
- $\vdash \Delta < \Sigma$ (2.2)

Thus, we conclude by (1).

By (Subtyping Inversion Lemma) on (2.2), we have that either:

- $\Delta' = \Delta, A_0 + A_1$ (2.3)

Thus, we conclude by (1).

By rewriting hypothesis.

Thus, we conclude.

Since $\Delta$ is a set, re-ordering is allowed.

Thus, we conclude by (2.6).

By (Store Typing Inversion Lemma) on (4.3).

By (str:BaseType) on (4.5).

Thus, we conclude by (4.6).

Thus, we conclude.

Case (str:Alternative) We have:

- $\Gamma, \Delta, \rho : A \vdash H$
- $\Delta < \Sigma$

By (Subtyping Inversion Lemma) on (2), we have that either:

- $\Delta' = \Delta, A_0 \vdash A_1$ (2.1)
- $\vdash \Delta < \Sigma$ (2.2)

Thus, we conclude by (1).

By (Subtyping Inversion Lemma) on (2), we have that either:

- $\Delta' = \Delta, A_0 + A_1$ (2.3)

By sub-case hypothesis.

Thus, we conclude.

By sub-case hypothesis.

Thus, we conclude by (3).

Thus, we conclude.

Thus, we conclude.

Thus, we conclude.

Thus, we conclude by (2.2) and (str:Symmetry) with $A_0$. 

5 2013/11/3
\[ \Gamma; \Delta, A_0 \vdash H \]  
by induction hypothesis on (2.5) and (2.6).

\[ \Gamma; \Delta, A_0 @ A_1 \vdash H \]  
by (str:Alternative) on (2.7).

Thus, we conclude.

\( \circ \) \[ \Gamma; \Delta, A_1 \vdash H \]  
Analogous to the previous case, noting that @ is commutative.

\( \circ \) \[ \text{[3(b)]} \quad \Delta' = \Delta, (A_0 @ A_1) @ A_2 \]  
Thus, we conclude by (str:Alternative) on (1) with \( A_2 \).

\( \circ \) \[ \text{[7]} \quad \Delta' = \Delta, A_0 @ A_1, \text{none} \]  
Thus, we conclude by (str:None) on (1).

\( \circ \) \[ \Delta, A_0 < \Delta' \]  
By inversion on (1) we have that either:

\( \circ \) \[ \Gamma; \Delta, A_0 \vdash H \]  
by induction hypothesis on (5.1) and sub-case hypothesis.

\( \circ \) \[ \Gamma; \Delta, A_1 \vdash H \]  
Analogous to the previous case, using (5.2).

\[ \blacksquare \]

### B.4 Values Inversion Lemma

**Lemma 4** (Values Inversion Lemma). If \( v \) is a value such that:

\[ \Gamma; \Delta \vdash v : A_0 \vdash \]

then one of the following holds:

1. if \( A_0 = [] \) then:

\[ \Delta = \vdash \Gamma; \Delta \vdash v : [] \vdash \]

2. if \( A_0 = \Diamond A_1 \) then:

\[ \Delta = \vdash \Gamma; \Delta \vdash v : A_1 \vdash \]

3. if \( A_0 = A_1 : A_2 \) then:

\[ \Gamma; \Delta \vdash v : A_1 \vdash A_2 \]

4. if \( A_0 = \text{ref } \rho \) then:

\[ v = \rho \quad \rho : \text{loc} \in \Gamma \quad \Delta = \]

5. if \( A_0 = A \to A' \) then:

\[ A < A'' \quad v = \text{fun}(x : A'').e \quad \Gamma; \Delta, x : A'' \vdash e : A' \vdash \]

6. if \( A_0 = \forall t. A \) then:

\[ v = (t) e \quad \Gamma; \Delta \vdash e : A \vdash \]

7. if \( A_0 = \exists t. A \) then:

\[ v = (p, v') \quad \Gamma; \Delta \vdash v' : A[p/t] \vdash \]

8. if \( A_0 = \Gamma \Gamma \Delta \vdash v' : A_i \vdash \]

(Note that, although the record value can have more fields than those that are listed in the type, only the fields that are in the type will appear in the inversion.)

9. if \( A_0 = \forall X. A \) then:

\[ v = (X) e \quad \Gamma; \Delta \vdash e : X : A \vdash \]

10. if \( A_0 = \exists X. A \) then:

\[ v = (A', v') \quad \Gamma; \Delta \vdash v' : A[A'/X] \vdash \]

11. if \( A_0 = \sum_i 1 \# A_i \) then:

\[ v = 1_i v_i \quad \Gamma; \Delta \vdash v_i : A_i \vdash \]

for some \( i \).

12. if \( A_0 = \text{rec } X. A \) then

\[ \Gamma; \Delta \vdash v : A[\text{rec } X. A/X] \vdash \]

13. if \( \Delta = \Delta', A_1 @ A_2 \) then

\[ \Gamma; \Delta; A_1 \vdash v : A_0 \vdash \quad \Gamma; \Delta; A_2 \vdash v : A_0 \vdash \]

**Proof.** By induction on the derivation of \( \Gamma; \Delta \vdash v : A \vdash \)

**Case (\text{tr:Rex})** - We have:

\[ \Gamma; \Delta ; \rho : \text{loc} ; \vdash \rho : \text{ref } \rho \vdash \]

by hypothesis.

Thus, we conclude by case 4 of the definition.

**Case (\text{tr:Put})** - We have:

\[ \Gamma; \Delta \vdash v : A \vdash \]

by hypothesis.

\[ \Gamma; \Delta \vdash v : A_i \vdash \]

by inversion on (tr:Put).

Thus, we conclude by case 2 of the definition.

**Case (\text{tr:User})** - We have:
Thus, we conclude by case 1 of the definition.

Case (\texttt{Pure-Read}), (\texttt{Linear-Read}), (\texttt{Pure-Elim}), (\texttt{New}) - Not applicable.

Case (\texttt{Delete}), (\texttt{Assign}), (\texttt{Difference-Linear}), (\texttt{Difference-Pure}) - Not applicable.

Case (\texttt{Record}) - We have:
\[ Γ; v : [\Delta] + \] (1)
by hypothesis.
\[ Γ; v : A + \] (2)
by inversion on (\texttt{Record}).
Thus, we conclude by case 8 of the definition.

Case (\texttt{Selection}), (\texttt{Application}) - Not applicable.

Case (\texttt{Function}) - We have:
\[ Γ; f(u : A_0) : e : A_0 \Rightarrow A_1 + \] (1)
by hypothesis.
\[ Γ; e : A_1 + \] (2)
by inversion on (\texttt{Function}).
Thus, we conclude by case 5 of the definition.

Case (\texttt{Cap-Elim}) - Not applicable.

Case (\texttt{Cap-Stack}) - We have:
\[ Γ; v : A_0 : A_1 + \] (1)
by hypothesis.
\[ Γ; v : A_0 + A_1 \] (2)
by inversion on (\texttt{Cap-Stack}).
Thus, we conclude by case 3 of the definition.

Case (\texttt{Cap-Open}), (\texttt{Application}) - Not applicable.

Case (\texttt{Forall-Loc}) We have:
\[ Γ; \lambda x : A + \] (1)
by hypothesis.
\[ Γ; \lambda x : A + \] (2)
by inversion on (\texttt{Forall-Loc}) with (1).
Thus, we conclude by case 6 of the definition.

Case (\texttt{Loc-Def}) Not applicable.

Case (\texttt{Loc-Pack}) We have:
\[ Γ; (p, v) : \exists A + \] (1)
by hypothesis.
\[ Γ; v : A[p/\hat{\lambda}] + \] (2)
by inversion on (\texttt{Loc-Pack}) with (1).
Thus, we conclude by case 7 of the definition.

Case (\texttt{Loc-Open}) Not applicable.

Case (\texttt{Forall-Type}) We have:
\[ Γ; \lambda x : X + A + \] (1)
by hypothesis.
\[ Γ; \lambda x : A + \] (2)
by inversion on (\texttt{Forall-Type}) with (1).
Thus, we conclude by case 9 of the definition.

Case (\texttt{Type-Appl}) Not applicable.

Case (\texttt{Type-Pack}) We have:
\[ Γ; A_0 + (A_0 / X) : AXA + \] (1)
by hypothesis.
\[ Γ; A_0 + v : A[X / A] + \] (2)
by inversion on (\texttt{Type-Pack}) with (1).
Thus, we conclude by case 10 of the definition.

Case (\texttt{Type-Open}) Not applicable.

Case (\texttt{Tag}) We have:
\[ Γ; \lambda x : 1 : A + \] (1)
by hypothesis.
\[ Γ; v : A + \] (2)
by inversion on (\texttt{Tag}).
Thus, we conclude by case 11 of the definition.

Case (\texttt{Case}) Not applicable.

Case (\texttt{Alternative-Left}) We have:
Thus, we conclude by case 13 of the definition.

Case (\texttt{Frame}) Not applicable, \(\Delta\) environment on right is empty, otherwise direct application of induction hypothesis.

Case (\texttt{Substitution}) We have:
\[ Γ; \lambda x : A + \] (1)
by hypothesis.
\[ \Delta; v : A_0 + \] (2)
by inversion on (\texttt{Frame}).
Thus, we conclude by case 11 of the definition.

By induction hypothesis on (3) we have that one of the following holds:

1. if \(A_0 = \emptyset\) then:
\[ A = \emptyset \] (1.1)
\[ Γ; v : \emptyset + \] (1.2)
\[ \emptyset < A_1 \] (1.3)
by case 1 of the hypothesis and rewriting (4).
Then, by (Subtyping Inversion Lemma) on (1.3) we have that either:
\[ [1] A_1 = 1 \] (1.4)
and we conclude as case 1 of the definition.
\[ [5c] A_1 = 1 \] (1.5)
and we conclude as case 2 of the definition.

2. if \(A_0 \cap A_1 = \emptyset\) then:
\[ A = A \] (2.1)
\[ Γ; v : A + \] (2.2)
\[ A \cup A_1 \] (2.3)
by case 2 of the hypothesis and rewriting (4).
Then, by (Subtyping Inversion Lemma) on (2.3) we have that either:
\[ [1] A_1 = A \] (2.4)
and we conclude as case 2 of the definition through (2.3).
\[ [2c] A_1 = A \] (2.5)
and we conclude as case 2 of the definition with (2.4).
\[ [2c] (1) A_1 = 1 \] (2.6)
\[ Γ; v : \emptyset + \] (2.7)
by (\texttt{Frame}) on \(v\).
Thus, we conclude by case 2 of the definition.

3. if \(A_0 = A \Rightarrow A'\) then:
\[ v = \text{fun}(x : A)e \] (3.1)
\[ Γ; \lambda x : A + \] (3.2)
\[ A \Rightarrow A' \] (3.3)
by case 5 of the hypothesis and rewriting (4).
by (Subtyping Inversion Lemma) on (3.3) we have that:
\[ (\text{note: we omit the case } A_1 = A_0, \text{since it is immediate}) \]
\[ A_1 = A'' \Rightarrow A''' \] (3.4)
\[ A' \Rightarrow A'' \] (3.5)
\[ A'' \Rightarrow A' \] (3.6)
\[ Γ; \lambda x : A + \] (3.7)
\[ \lambda x : A + \] (3.8)
by (\texttt{Frame}) on (3.7) and (\texttt{Def}) with (2).
\[ \text{(a defocus-guarantee can never be introduced by subtyping, thus } \overline{\Delta}) \]
Thus, with (3.8), (3.6) and (3.1) we conclude by case 5 of the definition.

4. if \(A_0 \cap A' = A''\) then:
\[ Γ; \lambda x : A + A'' \] (4.1)
\[ A \Rightarrow A' \] (4.2)
by case 3 of the hypothesis and rewriting (4).
by (Subtyping Inversion Lemma) on (4.2) we have that:
\[ (\text{note: we omit the case } A_1 = A_0, \text{since it is immediate}) \]
\[ A_1 = A'' \Rightarrow A''' \] (4.3)
\[ A \Rightarrow A'' \] (4.4)
\[ A' \Rightarrow A''' \] (4.5)
\[ A'' \Rightarrow A' \] (4.6)
\[ Γ; \lambda x : A'' + A''' \] (4.7)
by (\texttt{Frame}) on (4.1) with (4.4) and (4.5).
Thus, we conclude by case 3 of the definition.
5. if $A_0 = \emptyset \cdot A$ then:
$$v = [\emptyset \cdot v']$$
(5.1)
$$\Gamma; \Delta + v' : A + \cdot$$
(5.2)
$$[\emptyset \cdot \Delta < A_1$$
(5.5)
by case 8 of the hypothesis and rewriting (4).
by (Subtyping Inversion Lemma) on (5.5) we have that either:
• $\forall \Gamma \exists \Delta. \Delta + v : A[\emptyset \cdot \Delta < A_1$ (note: we omit the case
• $\not{\exists} \Gamma \exists \Delta. \Delta + v : A \iff \emptyset$ (note: we omit the case
• $\not{\exists} \Gamma \exists \Delta. \Delta + v : A \iff \emptyset$ (note: we omit the case
• $\not{\exists} \Gamma \exists \Delta. \Delta + v : A \iff \emptyset$ (note: we omit the case

Thus, we conclude by case 2 of the definition.

6. if $A_0 = \exists \cdot A$ then:
$$v = (p, v')$$
(6.1)
$$\Gamma; \Delta + v : A[p/t] + \cdot$$
(6.2)
$$\exists \cdot A < A_1$$
(6.3)
by case 7 of the hypothesis and rewriting (4).
by (Subtyping Inversion Lemma) on (6.3) we have that:
(note: we omit the case $A_1 = A_0$, since it is immediate)
$$A_1 = \exists \cdot A$$
(6.4)
$$A < A'$$
(6.5)
$$\Gamma; \Delta + v : A'[p/t] + \cdot$$
(6.6)
by (Subtyping Inversion Lemma) on (6.2) and (6.5).
Thus, we conclude by case 7 of the definition.

7. if $A_0 = \forall \cdot A$ then:
$$v = (\lambda e)$$
(7.1)
$$\Gamma; \Delta + \lambda : A + \cdot$$
(7.2)
$$\forall \cdot A < A_1$$
(7.3)
by case 6 of the hypothesis and rewriting (4).
by (Subtyping Inversion Lemma) on (7.3) we have that:
(note: we omit the case $A_1 = A_0$, since it is immediate)
$$A_1 = \forall \cdot A$$
(7.4)
$$A < A'$$
(7.5)
$$\Gamma; \Delta + \lambda : A' + \cdot$$
(7.6)
by (Subtyping Inversion Lemma) on (7.2) and (7.5).
(note: a defocus-guarantee cannot be introduced by subtyping)
Thus, we conclude by case 6 of the definition.

8. if $A_0 = \text{ref } \rho$ then:
$$v = \rho$$
(8.1)
$$\rho : \text{loc} \in T$$
(8.2)
$$\Delta =$$
(8.3)
$$\text{ref} \rho < A_1$$
(8.4)
by case 4 of the hypothesis and rewriting (4).
(note: we omit the case $A_1 = A_0$, since it is immediate)
by (Subtyping Inversion Lemma) on (8.4) we have:
• $\forall \{11\} A1 = \text{ref } \rho$
Thus, we conclude by case 2 of the definition.

9. if $A_0 = \exists X A$, analogous to $\exists \cdot A$.

10. if $A_0 = \forall X A$, analogous to $\forall \cdot A$.

11. if $A_0 = \sum 1 \cdot A'_i$ then:
$$v = 1 \cdot v_i$$
(11.1)
$$\Gamma; \Delta + v : A'_i + \cdot$$
(11.2)
$$\sum 1 \cdot A'_i < A_1$$
(11.3)
(note: we omit the case $A_1 = \sum 1 \cdot A'_i$, since it is immediate)

by (Subtyping Inversion Lemma) on (8.4) we have that:
$$A_1 = 1 \cdot A' \oplus \sum 1 \cdot A'_i$$
(11.4)
Thus, by (11.2) we conclude by case 11 of the definition.

12. if $A_0 \equiv \text{rec } X A$ then:
$$\Gamma; \Delta + v : A[\text{rec } X A/X] + \cdot$$
(12.1)
$$\text{rec } X A < A_1$$
(12.2)
by case 12 of the hypothesis and rewriting (4).
(note: we omit the case $A_1 = A_0$, since it is immediate)
by (Subtyping Inversion Lemma) on (12.2) we have that either:
• $\forall \Gamma \exists \Delta. \Delta + v : A \iff A'$$
(12.3)
$$\Gamma; \Delta + v : A'[\text{rec } X A/X] + \cdot$$
by (Subtyping Inversion Lemma) on (12.1).
Thus, we conclude by case 12 of the definition.

• $\forall \Gamma \exists \Delta. \Delta + v : A[X/\text{rec } X A]$ Thus, we conclude by induction hypothesis on (12.1) combined with (Subtyping Inversion Lemma) on each case.

Case (refLoc) Not a value.
### B.5 Substitution

For clarity, substitution is defined on constructs that allow expressions even though our grammar (in some places) only allows values since such difference has no impact in the following definitions and is generally more readable.

#### 1. Variable Substitution, \(\text{vs.}^8\)

We define the usual capture-avoiding (i.e. up to renaming of bounded variables) substitution rules:

\[
e_0[v/x] = e_1
\]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(vs:1)</td>
<td>(\rho[v/x] = \rho)</td>
</tr>
<tr>
<td>(vs:2)</td>
<td>(x[v/x] = x)</td>
</tr>
<tr>
<td>(vs:3)</td>
<td>(x_0[v/x_1] = x_0 \quad (x_0 \neq x_1))</td>
</tr>
<tr>
<td>(vs:4)</td>
<td>((\text{fun}(x_0 : A).e_0)[v/x_1] = \text{fun}(x_0 : A).e_0[v/x_1] \quad (x_0 \neq x_1))</td>
</tr>
<tr>
<td>(vs:5)</td>
<td>([\bar{e} = e][v/x] = [\bar{e} = e[v/x]])</td>
</tr>
<tr>
<td>(vs:6)</td>
<td>((e.f)[v/x] = e[v/x].f)</td>
</tr>
<tr>
<td>(vs:7)</td>
<td>((e_0 \ e_1)[v/x] = e_0[v/x] \ e_1[v/x])</td>
</tr>
<tr>
<td>(vs:8)</td>
<td>((\text{new} e)[v/x] = \text{new} e[v/x])</td>
</tr>
<tr>
<td>(vs:9)</td>
<td>((\text{delete} e)[v/x] = \text{delete} e[v/x])</td>
</tr>
<tr>
<td>(vs:10)</td>
<td>((p \ e)[v/x] = l e[v/x])</td>
</tr>
<tr>
<td>(vs:11)</td>
<td>((e_0 := e_1)[v/x] = e_0[v/x] := e_1[v/x])</td>
</tr>
<tr>
<td>(vs:12)</td>
<td>((p, e)[v/x] = (p, e[v/x]))</td>
</tr>
<tr>
<td>(vs:13)</td>
<td>(e[p][v/x] = e[v/x][p])</td>
</tr>
<tr>
<td>(vs:14)</td>
<td>((t, e)[v/x] = t, e[v/x])</td>
</tr>
<tr>
<td>(vs:15)</td>
<td>((\text{open} \ t, x_0, e_0 \text{ in } e_1 \text{ end})[v/x_1] = \text{open} \ t, x_0 = e_0[v/x_1] \text{ in } e_1[v/x_1] \text{ end} \quad (x_0 \neq x_1))</td>
</tr>
<tr>
<td>(vs:16)</td>
<td>((A, e)[v/x] = (A, e[v/x]))</td>
</tr>
<tr>
<td>(vs:17)</td>
<td>(e[A][v/x] = e[v/x][A])</td>
</tr>
<tr>
<td>(vs:18)</td>
<td>((X \ e)[v/x] = (X) e[v/x])</td>
</tr>
<tr>
<td>(vs:19)</td>
<td>((\text{open} \ X, x_0, e_0 \text{ in } e_1 \text{ end})[v/x_1] = \text{open} \ X, x_0 = e_0[v/x_1] \text{ in } e_1[v/x_1] \text{ end} \quad (x_0 \neq x_1))</td>
</tr>
<tr>
<td>(vs:20)</td>
<td>((1#e)[v/x] = 1#e[v/x])</td>
</tr>
<tr>
<td>(vs:21)</td>
<td>((\text{case } e \text{ of } 1#x_1 \rightarrow e_1 \text{ end})[v/x] = \text{case } e[v/x] \text{ of } 1#x_1 \rightarrow e_1[v/x] \text{ end} \quad (x_1 \neq x))</td>
</tr>
<tr>
<td>(vs:22)</td>
<td>((\text{let } x_0 = e_0 \text{ in } e_1 \text{ end})[v/x_1] = \text{let } x_0 = e_0[v/x_1] \text{ in } e_1[v/x_1] \text{ end} \quad (x_0 \neq x_1))</td>
</tr>
</tbody>
</table>
2. Location Variable Substitution. (ls:*)

Similarly, we define location substitution (but here up to renaming of bounded location variables) as:

\[ e_0[p/t] = e_1 \]

\[
\begin{align*}
\text{(ls:1.1)} & \quad p[p/t] = p \\
\text{(ls:1.2)} & \quad x[p/t] = x \\
\text{(ls:1.3)} & \quad (\text{fun}(x : \text{A}) e)[p/t] = \text{fun}(x : \text{A}[p/t], e[p/t]) \\
\text{(ls:1.4)} & \quad (\text{f} = e)[p/t] = (\text{f} = e[p/t]) \\
\text{(ls:1.5)} & \quad (e_f)[p/t] = e[p/t] \\
\text{(ls:1.6)} & \quad (e_0 e_1)[p/t] = e_0[p/t] e_1[p/t] \\
\text{(ls:1.7)} & \quad (\text{new } e)[p/t] = \text{new } e[p/t] \\
\text{(ls:1.8)} & \quad (\text{delete } e)[p/t] = \text{delete } e[p/t] \\
\text{(ls:1.9)} & \quad (\text{ls} e)[p/t] = e[p/t] \\
\text{(ls:1.10)} & \quad (e_0 = e_1)[p/t] = e_0[p/t] = e_1[p/t] \\
\text{(ls:1.11)} & \quad (p_0 e)[p_1/t] = (p_0[p_1/t], e[p_1/t]) \\
\text{(ls:1.12)} & \quad e[p_0[p_1/t]] = e[p_1/t][p_0[p_1/t]] \\
\text{(ls:1.13)} & \quad ((t_0 e)[p_1/t]) = ((t_0 e)[p_1/t]) \\
\text{(ls:1.14)} & \quad \text{(open } t_0, x = e_0 \text{ in } e_1 \text{ end})[p_1/t] = \text{(open } t_0, x = e_0[p_1/t] \text{ in } e_1[p_1/t] \text{ end}) \\
\text{(ls:1.15)} & \quad (\text{A}_0 \to \text{A}_1)[p/t] = (\text{A}_0[p/t] \to \text{A}_1[p/t]) \\
\text{(ls:1.16)} & \quad (\text{A}_0 \sqsupset \text{A}_1)[p/t] = (\text{A}_0[p/t] \sqsupset \text{A}_1[p/t]) \\
\text{(ls:1.17)} & \quad (\bar{X}).e[p/t] = (X).e[p/t] \\
\text{(ls:1.18)} & \quad \text{(open } X, x = e_0 \text{ in } e_1 \text{ end})[p/t] = \text{(open } X, x = e_0[p/t] \text{ in } e_1[p/t] \text{ end}) \\
\text{(ls:1.19)} & \quad (1#)[p/t] = 1#(p[t]) \\
\text{(ls:1.20)} & \quad (\text{case } e \text{ of } 1\#X \to e_1)[p/t] = \text{case } e[p/t] \text{ of } 1\#X \to e_1[p/t] \text{ end} \\
\text{(ls:1.21)} & \quad (\text{let } x = e_0 \text{ in } e_1 \text{ end})[p/t] = \text{let } x_0 = e_0[p/t] \text{ in } e_1[p/t] \text{ end} \\
\end{align*}
\]

\[ A_0[p/t] = A_1 \]

\[
\begin{align*}
\text{(ls:2.1)} & \quad p[p/t] = p \\
\text{(ls:2.2)} & \quad x[p/t] = x \\
\text{(ls:2.3)} & \quad t_0[p_1/t] = t_0 \\
\text{(ls:2.4)} & \quad (\text{A})[p/t] = 1\text{A}[p/t] \\
\text{(ls:2.5)} & \quad (\text{A}_0 \to \text{A}_1)[p/t] = (\text{A}_0[p/t] \to \text{A}_1[p/t]) \\
\text{(ls:2.6)} & \quad (\text{A}_0 \sqsupset \text{A}_1)[p/t] = (\text{A}_0[p/t] \sqsupset \text{A}_1[p/t]) \\
\text{(ls:2.7)} & \quad (\bar{X}).e[p/t] = (X).e[p/t] \\
\text{(ls:2.8)} & \quad (\text{v}_0 \text{A})[p_1/t] = 0\text{v}_0 \text{A}[p_1/t] \\
\text{(ls:2.9)} & \quad (\text{\#0.A})[p_1/t] = 0\text{\#0.A}[p_1/t] \\
\text{(ls:2.10)} & \quad (\text{ref } p_0)[p_1/t] = \text{ref } p_0[p_1/t] \\
\text{(ls:2.12)} & \quad (\text{rw } p_0 \text{A})[p_1/t] = \text{rw } p_0[p_1/t] \text{A}[p_1/t] \\
\text{(ls:2.13)} & \quad (\text{A}_0 \times \text{A}_1)[p/t] = (\text{A}_0[p/t] \times \text{A}_1[p/t]) \\
\text{(ls:2.14)} & \quad (\text{VX}.A)[p/t] = \text{VX.A}[p/t] \\
\text{(ls:2.15)} & \quad (\text{3X}.A)[p/t] = \text{3X.A}[p/t] \\
\text{(ls:2.16)} & \quad (\text{X})[p/t] = X \\
\text{(ls:2.17)} & \quad (\text{rec } X.A)[p/t] = \text{rec } X.A[p/t] \\
\text{(ls:2.18)} & \quad (\sum \text{\#0.A})[p/t] = (\sum \text{\#0.A})[p/t] \\
\text{(ls:2.19)} & \quad (\text{A}_0 \oplus \text{A}_1)[p/t] = (\text{A}_0[p/t] \oplus \text{A}_1[p/t]) \\
\text{(ls:2.20)} & \quad (\text{none}[p/t] = \text{none} \\
\end{align*}
\]

\[ \Gamma_0[p/t] = \Gamma_1 \]

\[
\begin{align*}
\text{(ls:3.1)} & \quad \llcorner p/t \lrcorner = \cdot \\
\text{(ls:3.2)} & \quad (\text{\Gamma}, x : \text{A})[p/t] = (\text{\Gamma}[p/t], x : \text{A}[p/t]) \\
\text{(ls:3.3)} & \quad (\text{\Gamma}, \text{t}_0 : \text{loc})[p_1/t] = (\text{\Gamma}[p_1/t], \text{t}_0 : \text{loc}) \\
\text{(ls:3.4)} & \quad (\text{\Gamma}, X : \text{type})[p/t] = (\text{\Gamma}[p/t], X : \text{type}) \\
\end{align*}
\]

\[ \Delta_0[p/t] = \Delta_1 \]

\[
\begin{align*}
\text{(ls:4.1)} & \quad \llcorner p/t \lrcorner = \cdot \\
\text{(ls:4.2)} & \quad (\text{\Delta}, x : \text{A})[p/t] = (\text{\Delta}[p/t], x : \text{A}[p/t]) \\
\text{(ls:4.3)} & \quad (\text{\Delta}, A)[p/t] = (\text{\Delta}[p/t], A[p/t]) \\
\end{align*}
\]
3. Type Variable Substitution, (ts.)*

Finally, we define type substitution (up to renaming of bounded type variables) as:

\[ e_0[A/X] = e_1 \]

\[
\begin{align*}
    \text{(ts.1.1)} &\quad \rho[A/X] = \rho \\
    \text{(ts.1.2)} &\quad x[A/X] = x \\
    \text{(ts.1.3)} &\quad (\text{fun}(x : A), e)[A/X] = \text{fun}(x : A_0[A/X]), e[A/X] \\
    \text{(ts.1.4)} &\quad \{ f = e \}[A/X] = \{ f = e[A/X] \} \\
    \text{(ts.1.5)} &\quad (e.\xi)[A/X] = e[A/X].\xi \\
    \text{(ts.1.6)} &\quad (e \circ e_0)[A/X] = e_0[A/X] e[A/X] \\
    \text{(ts.1.7)} &\quad (\text{new} e)[A/X] = \text{new} e[A/X] \\
    \text{(ts.1.8)} &\quad (\text{delete} e)[A/X] = \text{delete} e[A/X] \\
    \text{(ts.1.9)} &\quad (\text{let}) e[A/X] = !e[A/X] \\
    \text{(ts.1.10)} &\quad (e_0 = e_1)[A/X] = e_0[A/X] = e_1[A/X] \\
    \text{(ts.1.11)} &\quad (\rho, e)[A/X] = (\rho, e[A/X]) \\
    \text{(ts.1.12)} &\quad e[p][A/X] = e[A/X][p] \\
    \text{(ts.1.13)} &\quad (\langle \xi \rangle e)[A/X] = \langle \xi \rangle e[A/X] \\
    \text{(ts.1.14)} &\quad (\text{open} (\xi, x) = e_0 \text{ in } e_1)[A/X] = \text{open} (\xi, x) = e_0[A/X] \text{ in } e_1[A/X] \text{ end} \\
    \text{(ts.1.15)} &\quad (A_0, e)[A/X] = A_0[A/X, e(A_1/X)] \\
    \text{(ts.1.16)} &\quad (\text{let} : \xi)[A/X] = e[A/X][\xi] \\
    \text{(ts.1.17)} &\quad (\text{case } e_0 \text{ of } \Gamma \Rightarrow e_1)[A/X] = \text{case } e_0[A/X] \text{ of } \Gamma \Rightarrow e_1[A/X] \text{ end} \\
    \text{(ts.1.18)} &\quad (\text{let } e_0 \text{ in } e_1)[A/X] = \text{let } e_0[A/X] \text{ in } e_1[A/X] \text{ end} \\
    \text{(ts.1.19)} &\quad (1 \# e)[A/X] = 1 \# e[A/X] \\
    \text{(ts.20)} &\quad (\text{case } e \text{ of } \Gamma \Rightarrow e_1)[A/X] = \text{let } x = e_0 \text{ in } e_1[A/X] \text{ end} \\
    \text{(ts.21)} &\quad (\text{let } x = e_0 \text{ in } e_1)[A/X] = \text{let } x = e_0[A/X] \text{ in } e_1[A/X] \text{ end} \\
\end{align*}
\]

\[ A_0[A_1/X] = A_2 \]

\[
\begin{align*}
    \text{(ts.2.1)} &\quad \rho[A/X] = \rho \\
    \text{(ts.2.2)} &\quad \lambda A[X] = p \\
    \text{(ts.2.3)} &\quad X(A/X) = A \\
    \text{(ts.2.4)} &\quad X_0[A/X_1] = X_0 \\
    \text{(ts.2.5)} &\quad (\lambda A_0)[A_1/X] = \lambda A_0[A_1/X] \\
    \text{(ts.2.6)} &\quad (A_0 \Rightarrow A_1)[A_2/X] = A_0[A_2/X] \Rightarrow A_1[A_2/X] \\
    \text{(ts.2.7)} &\quad (A_0 :: A_1)[A_2/X] = A_0[A_2/X] :: A_1[A_2/X] \\
    \text{(ts.2.8)} &\quad [F :: A][A_0/X] = [F :: A][A_0/X] \\
    \text{(ts.2.9)} &\quad (\forall A_0)[A_1/X] = \forall A_0[A_1/X] \\
    \text{(ts.2.10)} &\quad (\exists A_0)[A_1/X] = \exists A_0[A_1/X] \\
    \text{(ts.2.11)} &\quad (\text{ref } p)[A/X] = \text{ref } p \\
    \text{(ts.2.12)} &\quad (\text{rw } p A_0)[A_1/X] = \text{rw } p A_0[A_1/X] \\
    \text{(ts.2.13)} &\quad (A_0 \ast A_1)[A_2/X] = A_0[A_2/X] \ast A_1[A_2/X] \\
    \text{(ts.2.15)} &\quad (\forall X_0,A_0)[A_1/X_1] = \forall X_0,A_0[A_1/X_1] \\
    \text{(ts.2.16)} &\quad (\forall X_0,A_0)[A_1/X_1] = \forall X_0,A_0[A_1/X_1] \\
    \text{(ts.2.17)} &\quad (\text{rec } X_0,A_0)[A_1/X_1] = \text{rec } X_0,A_0[A_1/X_1] \\
    \text{(ts.2.18)} &\quad (\Sigma X_0 \# A_0)[A/X] = \Sigma X_0 \# A_0[A/X] \\
    \text{(ts.2.19)} &\quad (\text{let } x = e_0 \text{ in } e_1)[A/X] = \text{let } x = e_0[A/X] \text{ in } e_1[A/X] \text{ end} \\
    \text{(ts.2.20)} &\quad (\text{none} A/X) = \text{none} \\
\end{align*}
\]

\[ \Gamma_0[A/X] = \Gamma_1 \]

\[
\begin{align*}
    \text{(ts.3.1)} &\quad \{ A/X \} = \cdot \\
    \text{(ts.3.2)} &\quad (\Gamma, x : A_0)[A_1/X] = \Gamma[A_1/X], x : A_0[A_1/X] \\
    \text{(ts.3.3)} &\quad (\Gamma, \xi : \text{loc})[A_0/X] = \Gamma[A/X], \xi : \text{loc} \\
    \text{(ts.3.4)} &\quad (\Gamma, X_0 : \text{type})[A_1/X] = \Gamma[A_1/X], X_0 : \text{type} \\
\end{align*}
\]

\[ \Delta_0[A/X] = \Delta_1 \]

\[
\begin{align*}
    \text{(ts.4.1)} &\quad A[A/X] = \cdot \\
    \text{(ts.4.2)} &\quad (\Delta, x : A_0)[A_1/X] = \Delta[A_1/X], x : A_0[A_1/X] \\
    \text{(ts.4.3)} &\quad (\Delta, A_0)[A_1/X] = \Delta[A_1/X], A_0[A_1/X] \\
\end{align*}
\]
B.6 Free Variables Lemma

Lemma 5 (Free Variables Lemma). If $Γ, Δ_0, x : A_0 ⊢ e : A_1 + Δ_1$ and $x ∈ Fv(e)$ then $x ∉ Δ_1$.

$Fv(e)$ denotes “set of all free variables inside the expression $e$”.

**Proof.** We proceed by induction on the derivation of $Γ, Δ_0, x : A_0 ⊢ e : A_1 + Δ_1$.

Case (tRef), (tPure), (tUnit), (tPure-Read) - $Δ$ is empty.

Case (tLinear-Read) - We have:

- $Γ, x : A + x : A +_1$ (1)
- $x ∈ Fv(x)$ (2)

Therefore, we immediately conclude $x ∉ Δ$.

Case (tPure- Elm) - We have:

- $Γ, Δ_0, x : A_0 ⊢ e : A_1 + Δ_1$ (1)
- $x ∈ Fv(e)$ (2)

Therefore, we conclude.

Case (tRef- Elm) - We have:

- $Γ, x : A_0 ⊢ x : A + Δ_1$ (1)
- $x ∈ Fv(x)$ (2)

Therefore, we conclude.

(Note: the case when $x$ is not the one used in the (tPure- Elm) rule is a direct application of the induction hypothesis.)

Case (tNew) - We have:

- $Γ, Δ_0, x : A_0 ⊢ \text{new } v : \exists e (v : A ) + Δ_1$ (1)
- $x ∈ Fv(\text{new } v)$ (2)

Therefore, we conclude.

Case (tDelete) - We have:

- $Γ, Δ_0, x : A_0 ⊢ \text{ delete } v : \exists e (v : A ) + Δ_1$ (1)
- $x ∈ Fv(\text{delete } v)$ (2)

Therefore, we conclude.

Case (tAssign) - We have:

- $Γ, Δ_0, x : A + v_0 := v_1 : A_1 + Δ_2, rw p A_0$ (1)
- $x ∈ Fv(v_0 := v_1)$ (2)

Therefore, we conclude.

**We have the following possibilities:**

1. $x ∈ Fv(v_0) \wedge x ∉ Fv(v_1)$
   - $(x : A) ∈ Δ_1$ (1.1)
   - by hypothesis.
   - $x ∉ Δ_2, rw p A_1$ (1.2)
   - by hypothesis.
   - $x ∉ Δ_2, rw p A_0$ (1.3)
   - by induction hypothesis on (4) with (1.1).
   - Since the capability trivially obeys the restriction (since $x$ is not a type).
   - Thus, we conclude.

2. $x ∈ Fv(v_1) \wedge x ∉ Fv(v_0)$
   - $x ∉ Δ_1$ (2.1)
   - by induction hypothesis on (3) and case assumption.
   - $x ∉ Δ_2, rw p A_1$ (2.2)
   - by (2.1) and (4).
   - $x ∉ Δ_2, rw p A_0$ (2.3)
   - since the capability trivially obeys the restriction on (2.2).
   - Thus, we conclude.

3. $x ∈ Fv(v_0) \land x ∈ Fv(v_1)$
   - $x ∉ Δ_1$ (3.1)
   - by induction hypothesis on (3) and case assumption.
   - We reach a contradiction since $v_0$ is well-typed by (4) but $x ∉ Fv(v_1)$ contradicts (3.1). Thus, such case is impossible to occur in a well-typed expression.
   - Thus, we conclude.

Case (tReference-Linear) - We have:

- $Γ, Δ_0, x : A_0 ⊢ v : A + Δ_1, rw p A_1$ (1)
- $x ∈ Fv(v)$ (2)

Therefore, we conclude.

Case (tReference-Pure) - We have:

- $Γ, Δ_0, x : A_0 ⊢ v : A + Δ_1, rw p A_1$ (1)
- $x ∈ Fv(v)$ (2)

Therefore, we conclude.

Case (tReference) - We have:

- $Γ, Δ_0, x : A_0 ⊢ v : A + Δ_1$ (1)
- $x ∈ Fv(v)$ (2)

Therefore, we conclude.

Case (tRecord) - We have:

- $Γ, Δ_0, x : A_0 ⊢ [x := v] : [x : A ] + Δ_1$ (1)
- $x ∈ Fv([x := v])$ (2)

Therefore, we immediately conclude $x ∉ Δ$.

Case (tDelete-Record) - We have:

- $Γ, Δ_0, x : A_0 ⊢ v : \exists e (v : A ) + Δ_1$ (1)
- $x ∈ Fv(v)$ (2)

Therefore, we conclude.

Case (tSelect-Record) - We have:

- $Γ, Δ_0, x : A_0 ⊢ v : \exists e (v : A ) + Δ_1$ (1)
- $x ∈ Fv(v)$ (2)

Therefore, we conclude.

Case (tApplication) - We have:

- $Γ, Δ_0, x : A + v_0 := v_1 : A_1 + Δ_2$ (1)
- $x ∈ Fv(v_0 := v_1)$ (2)

Therefore, we immediately conclude $x ∉ Δ$.

**We have the following possibilities:**

1. $x ∈ Fv(v_0)$
   - $(x : A) ∈ Δ_1$ (1.1)
   - $x = \Delta_1, x : A$ (1.2)
   - $x ≠ \Delta_1, x : A$ (1.3)
   - by rewriting (1.2).
   - $x ∉ Δ_2$ (1.4)
Thus, we conclude.

\[ x \in \mathbb{F}(v_0) \land x \notin \mathbb{F}(v_1) \]
\[ x \notin \Delta_1 \]  
by induction hypothesis on (3) and case assumption.

We reach a contradiction since \( v_0 \) is well-typed by (4) but \( x \in \mathbb{F}(v_1) \) contradicts (2.1). Thus, such case is impossible to occur in a well-typed expression. Therefore, we conclude.

\[ x \in \mathbb{F}(v_1) \land x \notin \mathbb{F}(v_0) \]
\[ x \notin \Delta_1 \]  
by induction hypothesis on (3) and case assumption.

Thus, we conclude.

Case (\texttt{forall-loc}) - We have:

\[ \Gamma, \Delta, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{forall-loc}) - We have:

\[ \Gamma, \Delta, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{forall-loc}) - We have:

\[ \Gamma, \Delta, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{loc-app}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{loc-loc}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Therefore, we have the following possibilities:

1. \( \mathbb{F}(v_1) \) is not a well-typed expression. Then \( x \in \mathbb{F}(v_1) \) and \( x \notin \mathbb{F}(v_0) \) by hypothesis.

2. \( x \in \mathbb{F}(v_1) \land x \notin \mathbb{F}(v_0) \) by hypothesis.

3. \( x \in \mathbb{F}(v_1) \land x \notin \mathbb{F}(v_0) \) by induction hypothesis on (3) and case assumption.

4. \( x \notin \Delta_1 \) by hypothesis.

Thus, we conclude.

We reach a contradiction since \( v_0 \) is well-typed by (4) but \( x \in \mathbb{F}(v_1) \) contradicts (2.1). Thus, such case is impossible to occur in a well-typed expression. Therefore, we conclude.

\[ x \in \mathbb{F}(v_0) \land x \notin \mathbb{F}(v_1) \]
\[ x \notin \Delta_1 \]  
by induction hypothesis on (3) and case assumption.

Thus, we conclude.

Case (\texttt{loc-pack}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow v : A_1 + \Delta_1 \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{forall-type}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{type-app}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{type-pack}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow \forall \Gamma(x : A_1) \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{type-open}) - Analogous to (\texttt{loc-loc}).

Case (\texttt{case-elm}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow e : A_1 + \Delta_1 \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{case-stack}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow e : A_1 + \Delta_1 \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.

Case (\texttt{case-unstack}) - We have:

\[ \Gamma, A, x : A_0 \rightarrow e : A_1 + \Delta_1 \]  
by hypothesis.

\[ x \notin \Delta \]  
since it is the empty environment.

Thus, we conclude.
Thus, we conclude.

Case (rFrame) - We have:

\[(\Delta_0, x : A \vdash e : A + \Delta_1, \Delta_2) \vdash e : A + \Delta_1, \Delta_2\]

by inversion on (rFrame) with (1), note by (2) \(x\) must be in environment.

\(x \not\in \Delta_1\)

by induction hypothesis with (2) and (3).

Thus, we conclude.

Case (rSummation) - We have:

\[\Gamma, \Delta_0, x : A \vdash e : A + \Delta_1\]

by inversion on (rSummation) with (1).

\(\Gamma, \Delta_0, x : A \vdash e : A'\)

by hypothesis.

\(\Gamma, \Delta_0, x : A \vdash e : A' + \Delta_1\)

by induction hypothesis on (3).

\(\Gamma, \Delta_0, x : A \vdash e : A' + \Delta_1\)

by hypothesis.

\(\Gamma, \Delta_0, x : A \vdash e : \Delta_1\)

by inversion on (rSummation) with (1).

\(x \not\in \Delta_1\)

by induction hypothesis on (2) and (4).

Thus, we conclude.

Case (rTao) - We have:

\[\Gamma, \Delta_0, x : A \vdash \lambda v : A_1 : A + \Delta_1\]

by induction hypothesis on (3) and case assumption.

\[\Gamma, \Delta_0, x : A \vdash v : A_1 : A + \Delta_1\]

by inversion on (rTao) with (1) and (2).

\(x \not\in \Delta_1\)

by hypothesis.

Thus, we conclude.

Case (rCase) - We have:

\[\Gamma, \Delta_0, x : A \vdash \text{case } v \text{ of } \sum_{i \leq j} \lambda A_i \rightarrow e_i \text{ end} : A + \Delta_1\]

by inversion on (rCase) with (1).

\(x \not\in \Delta_1\)

by hypothesis.

\(x \not\in \Delta_1\)

by induction hypothesis on (3) and case assumption.

Thus, we conclude.

Therefore, we have the following possibilities:

1. \(x \in \text{fv}(e) \land x \not\in \text{fv}(\epsilon)\)

   \(x \not\in \Delta'\)

   by induction hypothesis on (3) and case assumption.

   by (1.1) and (4).

   Thus, we conclude.

2. \(x \not\in \text{fv}(e) \land x \not\in \text{fv}(\epsilon)\)

   \(x \not\in \Delta_1\)

   by hypothesis.

   by (2.1) and (2.2).

   Thus, we conclude.

3. \(x \in \text{fv}(e) \land x \not\in \text{fv}(\epsilon)\)

   \(x \not\in \Delta_1\)

   by induction hypothesis on (3) and sub-case hypothesis.

   We reach a contradiction since \(v\) is well-typed by (4) but \(x \not\in \text{fv}(\epsilon)\) contradicts (3.1). Thus, such case is impossible to occur in a well-typed expression.

Case (rAlternative-Left) - We have:

\[\Gamma, \Delta_0, x : A_0, A_1 \vdash e : A_1 + \Delta_1\]

by inversion on (rAlternative-Left) with (1).

\(x \not\in \Delta_1\)

by induction hypothesis with (2) and (3).

Thus, we conclude.

Case (rLet) - We have:

\[\Gamma, \Delta_0, x : A \vdash \text{let } x_0 = e_0 \text{ in } e_1 : A_1 + \Delta_2\]

by hypothesis.

\[\Gamma, \Delta_0, x : A \vdash e_0 : A_1 + \Delta_1\]

by inversion on (rLet) with (1).

\[\Gamma, \Delta_0, x : A \vdash e_1 : A_1 + \Delta_2\]

by induction hypothesis on (3) and case assumption.

We reach a contradiction since \(e_0\) is well-typed by (4) but \(x \not\in \text{fv}(e_1)\) contradicts (2.1). Thus, such case is impossible to occur in a well-typed expression.
B.7 Well-Form Lemmas

Lemma 6 (Well-Formed Type Substitution). We have:
- For location variables:
  1. If \( \Gamma, t : \text{loc} \vdash \rho : \text{loc} \in \Gamma \)
     then \( \Gamma[t/\rho] \vdash \text{wf} \).
  2. If \( \Gamma, t : \text{loc} \vdash \Delta \vdash \rho : \text{loc} \in \Gamma \)
     then \( \Gamma[t/\rho] \vdash \Delta \vdash \text{wf} \).
  3. If \( \Gamma, t : \text{loc} \vdash A \text{ type} \vdash \rho : \text{loc} \in \Gamma \)
     then \( \Gamma[t/\rho] \vdash A \text{ type} \).
- For type variables:
  1. If \( \Gamma, X \vdash A \text{ type} \)
     then \( \Gamma[X/X] \vdash A \text{ type} \).
  2. If \( \Gamma, X \vdash \Delta \vdash A \text{ type} \)
     then \( \Gamma[X/X] \vdash \Delta \vdash A \text{ type} \).
  3. If \( \Gamma, X \vdash A \text{ type} \vdash A' \text{ type} \)
     then \( \Gamma[X/X] \vdash A \Rightarrow A' \text{ type} \).

Proof. Straightforward by induction on the definition of \( \Gamma, \Delta \) and types.

Lemma 7 (Well-Formed Subtyping). We have two cases:
1. (Type) If \( \Gamma, A \vdash A \text{ type} \) and \( A \vdash A' \text{ type} \) then \( \Gamma, A, A \vdash A \Rightarrow A' \text{ type} \).
2. (Linear) If \( \Gamma, A \vdash \Delta \vdash A \text{ type} \) and \( \Delta \vdash A \Rightarrow A' \text{ type} \) then \( \Gamma, A, A \vdash A \Rightarrow A' \text{ type} \).

Proof. Straightforward by induction on the definition of \( \vdash \) for types and \( \Delta \), respectively.

B.8 Substitution Lemma

Lemma 8 (Substitution Lemma). We have the following substitution properties for both expression typing and type formation:
1. (Linear) If \( \Gamma ; A_0 \vdash v : A_0 + \Delta_1 \)
   then \( \Gamma[\rho/v] \vdash A_0 + \Delta_2 \).
2. (Pure) If \( \Gamma ; A_0 \vdash v : A_0 + \Delta_1 \)
   then \( \Gamma ; \Delta_0 \vdash v : A_0 + \Delta_2 \).
   (Note that due to the required pure types, the \( \Delta \) environments to check \( v \) must be empty)
3. (Location Variable) If \( \Gamma, t : \text{loc} \vdash \Delta \vdash A \Rightarrow \Delta_1 \)
   then \( \Gamma[t/\rho] \vdash \Delta \vdash \rho : \text{loc} \in \Gamma \).
4. (Type Variable) If \( \Gamma, A_0 \vdash e : A_0 + \Delta_1 \)
   then \( \Gamma[A_1/X] ; \Delta_0 \vdash e[A_1/X] : A_0[A_1/X] + \Delta_1[A_1/X] \).
   (replaces \( X \) in all places it may occur free)

Proof. We split the proof on each of the lemma’s sub-parts:

1. (Linear) Proof. Proceed by induction on the typing derivation of \( \Gamma, A_0, x : A_0 + e : A_1 + \Delta_2 \).

Case (Ref), (Pure), (Ref-Unif), (Pure-Read) - Not applicable since these rules require an empty \( \Delta \) environment.

Case (Ref-Linear-Read) - We have:
\( \Gamma, A_0 + e : A_1 + \Delta_2 \)
\( \Gamma, A_0 + e : A_1 + \Delta_1 \) by hypothesis.
\( \Gamma, A_0 + e : A_1 + \Delta_2 \) (note \( \Delta \)’s ending environment must be \( \Delta \) to apply (Ref-Linear-Read)).
\( A_0 + e : A_1 + \Delta_2 \) by (vs:2) with (1) and \( x \).
Thus, we conclude.

Case (Pure-Elm) - We have:
\( \Gamma, A_0 + e : A_1 + \Delta_2 \)
\( \Gamma, A_0 + e : A_1 + \Delta_1 \) by hypothesis.
\( A_0 + e : A_1 + \Delta_2 \) by inversion on (Pure-Elm) with (2).
\( A_0 + e : A_1 + \Delta_2 \) by induction hypothesis on (3) with (1).
\( A_0 + e : A_1 + \Delta_2 \) by (Pure-Elm) with (4).
Thus, we conclude.

Case (Ref-New) - We have:
\( \Gamma, A_0 + e : A_1 + \Delta_2 \)
\( \Gamma, A_0 + e : A_1 + \Delta_1 \) by hypothesis.
\( \Gamma, A_0 + e : A_1 + \Delta_2 \) by inversion on (Ref-New) with (2).
\( \Gamma, A_0 + e : A_1 + \Delta_2 \) by induction hypothesis with (1) and (3).
\( \Gamma, A_0 + e : A_1 + \Delta_2 \) (5)

2013/11/3
\[
\Gamma; \Delta \vdash \text{(new } v_0)[v/x]: \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2
\]
by (r:New) with (4).

Thus, we conclude.

Case (r:Delete) - We have:
\[
\Gamma; \Delta, x : A_0 \vdash \Delta_1
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash v_0 : \text{ref } t :: \text{rw } t \ A_1 \Delta_2
\]
by inversion on (r:Delete) with (2).

\[
\Gamma; \Delta_0, v_0[v/x] : \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2
\]
by induction hypothesis with (1) and (3).

\[
\Gamma; \Delta_0 \vdash \text{delete } v_0[v/x] : \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2
\]
by (r:Delete) with (4).

\[
\Gamma; \Delta_0 \vdash (\text{delete } v_0[v/x]) : \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2
\]
by (vs:9) with (5).

Thus, we conclude.

Case (r:Assign) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
(1)

\[
\Gamma; \Delta_1, x : A_0 \vdash v_1 : \Delta_1, \text{rw } p A_2
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash v_1 : A_1 \Delta_2 \Delta'
\]
by (Free Variables Lemma) on (3).

\[
\Gamma; \Delta_0 \vdash v_1[v/x] : A_1 \Delta_2 \Delta'
\]
by induction hypothesis with (1) and (3).

\[
\Gamma; \Delta_0 \vdash v_0[v/x] : A_1 \Delta_2 \Delta'
\]
by (r:Assign) with (1.2) and (1.3).

\[
\Gamma; \Delta_0 \vdash (\lambda x. v_1)[v/x] : A_1 \Delta_2 \Delta'
\]
by (vs:11) with (1.4).

Thus, we conclude.

We have that either:
\[
(s : A_0) \in \Delta'
\]
\[
\Gamma; \Delta' \vdash v_0[v/x] : \text{ref } p + \Delta_2, \text{rw } p A_1
\]
by inversion on (r:Assign) with (2).

(b) \( x \notin \text{fv}(v_1) \)
\[
\Gamma; \Delta' \vdash v_0[v/x] : \text{ref } p + \Delta_2, \text{rw } p A_1
\]
by (Free Variables Lemma) on (3).

\[
\Gamma; \Delta_0 \vdash v_1[v/x] : A_1 \Delta_2 \Delta'
\]
by induction hypothesis with (1) and (3).

\[
\Gamma; \Delta_0 \vdash v_0[v/x] : A_1 \Delta_2 \Delta'
\]
by (r:Assign) with (2.2) and (2.3).

Thus, we conclude.

Case (r:DeleteReference-Lineal) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash v_0 : \text{ref } p + \Delta_2, \text{rw } p A_1
\]
by inversion on (r:DeleteReference-Lineal) with (2).

\[
\Gamma; \Delta_0 \vdash v_0[v/x] : \text{ref } p + \Delta_2, \text{rw } p A_1
\]
by induction hypothesis with (1) and (3).

\[
\Gamma; \Delta_0 \vdash (\text{ref } t :: \text{rw } t \ A_1) + \Delta_2 \Delta'
\]
by (r:DeleteReference-Lineal) on (4).

\[
\Gamma; \Delta_0 \vdash (\text{ref } t :: \text{rw } t \ A_1) + \Delta_2 \Delta'
\]
by (vs:10) on (5).

Thus, we conclude.

Case (r:DeleteReference-Pure) - Analogous to (r:DeleteReference-Lineal).

Case (r:Record) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash (F :: v)[v/x] : \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2 \Delta'
\]
by (r:Record) with (2).

\[
\Gamma; \Delta_1, x : A_0 \vdash (F :: v)[v/x] : \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2 \Delta'
\]
by (r:Record) with (2).

\[
\Gamma; \Delta_1, x : A_0 \vdash (F :: v)[v/x] : \exists \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2 \Delta'
\]
by (r:Record) with (2).

Thus, we conclude.

Case (r:Select) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash v_0 : \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2 \Delta'
\]
by inversion on (r:Select) with (2).

\[
\Gamma; \Delta_0 \vdash v_0[v/x] : \{ \text{ref } t :: \text{rw } t \ A_1 \} + \Delta_2 \Delta'
\]
by induction hypothesis with (1) and (3).

Thus, we conclude.

Case (r:Application) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
(1)

\[
\Gamma; \Delta_1, x : A_0 \vdash v_0 v_1 : A_1 \Delta_2 \Delta'
\]
by hypothesis.

\[
\Gamma; \Delta_0 \vdash v_0[v/x] : A_1 \Delta_2 \Delta'
\]
by (r:Select) with (2).

\[
\Gamma; \Delta_0 \vdash (\text{ew } v_1)[v/x] : A_1 \Delta_2 \Delta'
\]
by (r:Select) with (2).

Thus, we conclude.

Case (r:Function) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash \text{fun}(x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by def. of substitution up to rename of bounded variables.

\[
\Gamma; \Delta_1, x : A_0 \vdash \text{fun}(x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by induction hypothesis with (1) and (3).

\[
\Gamma; \Delta_1, x : A_0 \vdash \text{fun}(x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by (r:Function) with (5).

\[
\Gamma; \Delta_1, x : A_0 \vdash \text{fun}(x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by (r:Function) with (5).

Thus, we conclude.

Case (r:Forall-Loc) - We have:
\[
\Gamma; \Delta_0 \vdash v : A_0 \Delta_1
\]
by hypothesis.

\[
\Gamma; \Delta_1, x : A_0 \vdash (\exists \Delta_1 x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by inversion on (r:Forall-Loc) with (2).

\[
\Gamma; \Delta_1, x : A_0 \vdash (\exists \Delta_1 x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by inversion with (r:Record) with (2).

\[
\Gamma; \Delta_1, x : A_0 \vdash (\exists \Delta_1 x : A_1) \ e : A_1 \to A_2 \to \cdot
\]
by (r:Record) on (4).

Thus, we conclude.
Thus, we conclude.

**Case (rLoc-Arr)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 \rightarrow A_1 \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + v_0[p] : A_1[p/t] + \Delta_2 \]
by hypothesis.
\[ p : \text{loc} \in \Gamma \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + v_0 : \forall x_1 A_1 + \Delta_2 \]
by inversion on (rLoc-Arr) with (2).
\[ \Gamma; \delta_1 \triangleright v_0[x] : \forall x_1 A_1 + \Delta_2 \]
by induction hypothesis on (4) and (1).
\[ \Gamma; \delta_1 \triangleright v_0[p]/v_0[x] : A_1[p/t] + \Delta_2 \]
by (rLoc-Arr) on (5) and (3).
\[ \Gamma; \delta_1 \triangleright (v_0[p]/v_0[x] : A_1[p/t] + \Delta_2 \]
by (vs.13) on (6).
Thus, we conclude.

**Case (rLoc-Pack)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + (p, v_0) : \exists \! x_1 A_1 + \Delta_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1[p/t] + \Delta_2 \]
by inversion on (rLoc-Pack) with (2).
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1[p/t] + \Delta_2 \]
by induction hypothesis on (1) and (3).
\[ \Gamma; \delta_1 \triangleright (p, v_0[x]) : \exists \! x_1 A_1 + \Delta_2 \]
by (rLoc-Pack) on (4).
\[ \Gamma; \delta_1 \triangleright ((p, v_0[x])/v_0[x] : \exists \! x_1 A_1 + \Delta_2 \]
by (vs.12) on (5).
Thus, we conclude.

**Case (rLoc-Open)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x_0 : A_0 + \text{open}(t, x_1) = v_0 \text{ in } e_1 \end{term} : A_1 + \Delta_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright x_0 : A_0 + v_0 : \exists \! x_1 A_1 + A' \]
by inversion on (rLoc-Open) with (2).
\[ \Gamma; t : \text{loc} ; x_1 : A_2 + e_1[v_0/x_1] : A_1 + \Delta_2 \]
by (Free Variables Lemma) on (3).
\[ x_0 \neq x_1 \text{ by def. of substitution up torename of bounded variables,} \]
\[ \Gamma; t : \text{loc} ; x_1 : A_2 + e_1[v_0/x_1] : A_1 + \Delta_2 \]
since \( x_0 \) cannot occur in \( e_1 \) and by (1.1) nor in \( \Gamma \) by (3).
\[ \Gamma; \delta_1 \triangleright v_0[x] : \exists \! \Delta_2 \text{ by induction hypothesis on (1) and (3).} \]
\[ \Gamma; \delta_1 \triangleright (t, x_1) = v_0[x] \text{ in } e_1[x_1] \end{term} : A_1 + \Delta_2 \]
by (rLoc-Open) on (1.3) and (1.4).
\[ \Gamma; \delta_1 \triangleright (t, x_1) = v_0 \text{ in } e_1 \end{term} : A_1 + \Delta_2 \]
by (vs.15) on (1.6) and (1.2).
Thus, we conclude.

**Case (rForall-Type)** - Analogous to (rForall-Loc) with (vs.18).
**Case (rType-Arr)** - Analogous to (rLoc-Arr) with (vs.17).
**Case (rType-Pack)** - Analogous to (rLoc-Pack) with (vs.16).
**Case (rType-Open)** - Analogous to (rLoc-Open) with (vs.19).
**Case (rCap-Elim)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 + A_2 \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + v_0 : A_1 + \Delta_2 \]
by hypothesis.
\[ \delta_1, \delta_1 : A_2 :: A_3 \]
\[ \Gamma; \delta_1, \delta_1 : A_2 :: A_3, v_0 : A_1 + e : A_1 + \Delta_2 \]
by induction hypothesis with (1) and (3).
\[ \Gamma; \delta_1, \delta_1 : A_2 :: A_3 + e[v_0/x_1] : A_1 + \Delta_2 \]
by (rCap-Elim) with (4).
Thus, we conclude.

**Case (rCap-Stack)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + e : A_1 + \Delta_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 + A_2 \]
by inversion on (rCap-Stack) with (2).
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 + A_2 \]
by induction hypothesis with (1) and (3).
\[ \Gamma; \delta_1 \triangleright e[v_0/x] : A_1 + \Delta_2 + A_2 \]
by (rCap-Stack) on (4).
Thus, we conclude.

**Case (rCap-Unstack)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + e : A_1 + \Delta_2 + A_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 + A_2 \]
by inversion on (rCap-Unstack) with (2).
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 + A_2 \]
by induction hypothesis with (1) and (3).
\[ \Gamma; \delta_1 \triangleright e[v_0/x] : A_1 + \Delta_2 + A_2 \]
by (rCap-Stacks) on (4).
Thus, we conclude.

**Case (rSubsumption)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x : A_0 + e : A_1 + \Delta_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 \]
by (Subtyping Inversion Lemma) on (1) with (7).
\[ \Gamma; \delta_1 \triangleright A_0 < A_0' \]
by (rSubsumption) on (2).
\[ \Gamma; \delta_1 \triangleright e[v_0/x] : A_2 + \Delta_2 \]
by (rSubsumption) on (1) with (8).
\[ \Gamma; \delta_1 \triangleright A_2 < A_2' \]
by induction hypothesis on (4) and (8).
\[ \Gamma; \delta_1 \triangleright A_0 < A_0' \]
by (Subtyping Inversion Lemma) on (3).
\[ \Gamma; \delta_1 \triangleright e[v_0/x] : A_1 + \Delta_2 \]
by (rSubsumption) on (9) with (10), (5) and (6).
Thus, we conclude.

**Case (rFrame)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x_0 : A_0 + A_1 + \Delta_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright x : A_0 + e : A_1 + \Delta_2 \]
by inversion on (rFrame) with (2).
\[ \Gamma; \delta_1 \triangleright e[v_0/x] : A_1 + \Delta_2 \]
by induction hypothesis with (1) and (3).
\[ \Gamma; \delta_1 \triangleright e[v_0/x] : A_1 + \Delta_2 \]
by (rFrame) on (4) with \( \Delta_1 \).
Thus, we conclude.

**Case (rTua)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
\[ \Gamma; \delta_1 \triangleright x_0 : A_0 + v_0[\#A_1] + \Delta_2 \]
by hypothesis.
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 \]
by inversion on (rTua) with (2).
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 \]
by induction hypothesis with (1) and (3).
\[ \Gamma; \delta_1 \triangleright v_0[x] : A_1 + \Delta_2 \]
by (rTua) with (4).
Thus, we conclude.

**Case (rCast)** - We have:
\[ \Gamma; \delta_0 \triangleright v : A_0 + A_1 \]
by inversion on (rCap-Elim) with (2).
Thus, we conclude.

\[ \Gamma, \Delta_1, x : A_0 + \text{case } v_0 \text{ of } I \# x_j \rightarrow e_j \text{ end } : A + A_2 \] by hypothesis.

\[ \Gamma, \Delta_1, x : A_0 + v_0 : \sum_j I_1 [\theta]^j + A' \] (3)

\[ \Gamma, \Delta_2, x : A_0 + e_j : A + A_2 \] (4)

\[ x \leq j \] by inversion (\( t \text{Case} \)) with (2).

We have that either:

\[ (a) \ x \notin \text{fv}(v_0) \]

\[ x \notin A' \] by (Free Variables Lemma) on (3).

\[ x \neq x_j \] by def. of substitution up to rename of bounded variables.

\[ \Gamma, \Delta, x : A' + e_j[v/x] : A + A_2 \] since \( x \) cannot occur in \( e_j \) and by (1.1) nor in \( \Gamma \) by (3).

\[ \Gamma, \Delta_1, x : A_0 + v_0[v/x] : \sum_j I_1 [\theta]^j + A' \] by induction hypothesis on (1) and (3).

\[ \Gamma, \Delta_1 + \text{case } v_0[v/x] \text{ of } \Gamma, \Delta_1 \rightarrow e_j[v/x] \text{ end } : A + A_2 \] by (\( \Gamma \text{Case} \)) on (5), (1.3) and (1.4).

\[ \Gamma, \Delta_1 + \text{case } v_0 \text{ of } \Gamma, \Delta_1 \rightarrow e_j \text{ end } : A + A_2 \] by (vs.21) on (1.6) and (1.2).

Thus, we conclude.

\[ \exists \] by def. of substitution up to rename of bounded variables.

\[ \Gamma, \Delta' \rightarrow x : A' + e_j[v/x] : A + A_2 \] by induction hypothesis where \( \Delta' \) is same as \( \Delta' \) without \( x \).

\[ \Gamma, \Delta_1 + v_0[v/x] : \sum_j I_1 [\theta]^j + A' \] since \( x \) cannot occur in \( e_j \) by \( x \notin \text{fv}(v_0) \).

\[ \Gamma, \Delta_1 + \text{case } v_0[v/x] \text{ of } \Gamma, \Delta_1 \rightarrow e_j[v/x] \text{ end } : A + A_2 \] (2.5)

by (\( \Gamma \text{Case} \)) on (5), (2.3) and (2.4).

\[ \Gamma, \Delta_1 + \text{case } v_0 \text{ of } \Gamma, \Delta_1 \rightarrow e_j \text{ end } : A + A_2 \] by (vs.21) on (2.1) and (2.5).

Thus, we conclude.

**Case (t:Alternative-Left)** - Immediate by applying the induction hypothesis on the inversion and then re-applying the rule.

**Case (t:Let)** - Analagous to previous cases.

2. (Pure)

\[ \Gamma, \Delta_1, x : A_0 + \text{case } v_0 \text{ of } I \# x_j \rightarrow e_j \text{ end } : A + A_2 \]

**Proof.** We proceed by induction on the typing derivation of

\[ \Gamma, \Delta_1, x : A_0 + \] by (Free Variables Lemma) on (3).

\[ \Gamma, \Delta_1, x : A_0 + v_0 : \sum_j I_1 [\theta]^j + A' \] (3)

\[ \Gamma, \Delta_2, x : A_0 + e_j : A + A_2 \] (4)

\[ x \leq j \] by inversion (\( t \text{Case} \)) with (2).

We have that either:

\[ (a) \ x \notin \text{fv}(v_0) \]

\[ x \notin A' \] by (Free Variables Lemma) on (3).

\[ x \neq x_j \] by def. of substitution up to rename of bounded variables.

\[ \Gamma, \Delta, x : A' + e_j[v/x] : A + A_2 \] by induction hypothesis where \( \Delta' \) is same as \( \Delta' \) without \( x \).

\[ \Gamma, \Delta_1, x : A_0 + v_0[v/x] : \sum_j I_1 [\theta]^j + A' \] since \( x \) cannot occur in \( e_j \) by \( x \notin \text{fv}(v_0) \).

\[ \Gamma, \Delta_1 + \text{case } v_0[v/x] \text{ of } \Gamma, \Delta_1 \rightarrow e_j[v/x] \text{ end } : A + A_2 \] (2.5)

by (\( \Gamma \text{Case} \)) on (5), (2.3) and (2.4).

\[ \Gamma, \Delta_1 + \text{case } v_0 \text{ of } \Gamma, \Delta_1 \rightarrow e_j \text{ end } : A + A_2 \] by (vs.21) on (2.1) and (2.5).

Thus, we conclude.

**Case (t:Inline-Read)** - We have:

\[ \Gamma, \Delta_1, x : A_0 + \] by hypothesis.

\[ x \neq x_j \] since \( \Gamma \) and \( A \) identitiers cannot collide.

\[ \Gamma, \Delta_1, x : A_0 + v_0[v/x] : A + A_2 \] (3)

by (vs.21) on (2.3) and (2.4).

Thus, we conclude.

**Case (t:Delete)** - We have:

\[ \Gamma, \Delta_1, x : A_0 + \] by hypothesis.

\[ x \neq x_j \] since \( \Gamma \) and \( A \) identitiers cannot collide.

\[ \Gamma, \Delta_1, x : A_0 + v_0[v/x] : A + A_2 \] (3)

by inversion on (t:Delete) with (2).

\[ \Gamma, \Delta_1 + \text{case } v_0[v/x] \text{ of } \Gamma, \Delta_1 \rightarrow e_j[v/x] \text{ end } : A + A_2 \] (4)

by (\( \Gamma \text{Case} \)) on (5), (3) and (4).

\[ \Gamma, \Delta_1 + \text{case } v_0 \text{ of } \Gamma, \Delta_1 \rightarrow e_j \text{ end } : A + A_2 \] (5)

by (vs.21) on (3) and (2.5).

Thus, we conclude.

**Case (t:Assign)** - We have:

\[ \Gamma, \Delta_1, x : A_0 + \] by hypothesis.

\[ x \neq x_j \] since \( \Gamma \) and \( A \) identitiers cannot collide.

\[ \Gamma, \Delta_1, x : A_0 + v_0[v/x] : A + A_2 \] (3)

by inversion on (t:Assign) with (2).

\[ \Gamma, \Delta_1 + \text{case } v_0[v/x] \text{ of } \Gamma, \Delta_1 \rightarrow e_j[v/x] \text{ end } : A + A_2 \] (4)

by (\( \Gamma \text{Case} \)) on (5), (3) and (4).

\[ \Gamma, \Delta_1 + \text{case } v_0 \text{ of } \Gamma, \Delta_1 \rightarrow e_j \text{ end } : A + A_2 \] (5)

by (vs.21) on (3) and (2.5).

Thus, we conclude.
Thus, we conclude.

Case (\(\text{tReferenceLinear}\)) - We have:
\[
\begin{align*}
\Gamma; A_t \vdash v_i[v/s] & : \text{ref } p + A_t; \text{rw } p, A_t & (6) \\
\text{by induction hypothesis on (4) with (1).} & \\
\Gamma; \Delta_0 \vdash v_i[v/s] & : v_i[v/s] : A_t + A_\Delta; \text{rw } p, A_t & (7) \\
\text{by (rAssume) with (5) and (6).} & \\
\Gamma; \Delta_0 + \langle v_i[v/s] : A_t + A_\Delta; \text{rw } p, A_t \rangle & (8) \\
\text{by (rFunction) on (7).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tReferencePure}\)) - Analogous to (\(\text{tReferenceLinear}\)).

Case (\(\text{tRecfun}\)) - We have:
\[
\begin{align*}
\Gamma; A_t \vdash v_i[v/s] & : \text{fun } x_i : A_0, e : A_t \rightarrow A_0 + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; x : A_t; A_t \vdash v_i[v/s] & : A_t + \text{ref } p & (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : \text{ref } p + A_t; \text{rw } p, A_t & (3) \\
\text{by induction hypothesis on (3) with (1).} & \\
\Gamma; \Delta_0 + \langle v_i[v/s] : A_t + A_\Delta; \text{rw } p \rangle & (4) \\
\text{by (rFunctionLinear) with (6).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + A_\Delta; \text{rw } p \} \rangle & (5) \\
\text{by (vs.10) on (5).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tApplication}\)) - We have:
\[
\begin{align*}
\Gamma; \Delta \vdash \text{ref } x_i : A_0, e : A_t \rightarrow A_0 + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; x : A_t; A_t \vdash v_i[v/s] & : A_t + \text{ref } p & (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : \text{ref } p + A_t; \text{rw } p, A_t & (3) \\
\text{by induction hypothesis on (3) with (1).} & \\
\Gamma; \Delta_0 + \langle v_i[v/s] : A_t + A_\Delta; \text{rw } p \rangle & (4) \\
\text{by (rFunctionLinear) with (6).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + A_\Delta; \text{rw } p \} \rangle & (5) \\
\text{by (vs.10) on (5).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tFunction}\)) - We have:
\[
\begin{align*}
\Gamma; x : A_t; A_t \vdash \text{fun } x_i : A_0, e : A_t \rightarrow A_0 + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; x : A_t; A_t \vdash v_i[v/s] & : v_i[v/s] : A_t + A_\Delta; \text{rw } p, A_t & (2) \\
\text{by inversion on (rFunction) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : v_i[v/s] : A_t + A_\Delta; \text{rw } p, A_t & (3) \\
\text{by def. of substitution up to rename of bounded variables.} & \\
\Gamma; \Delta_0 + \langle v_i[v/s] : A_t + A_\Delta; \text{rw } p, A_t \rangle & (4) \\
\text{by induction hypothesis with (3) and (1).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + A_\Delta; \text{rw } p, A_t \} \rangle & (5) \\
\text{by (rFunction) on (4) and (4).} & \\
\end{align*}
\]
Thus, we conclude.

Thus, we conclude.

Case (\(\text{tRecfun}\)) - We have:
\[
\begin{align*}
\Gamma; \Delta \vdash \text{loc } x_i : A_t + \rightarrow \forall A \rightarrow A_t + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; \Delta \vdash \text{loc } x_i : A_t + \rightarrow \forall A \rightarrow A_t + * - (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (3) \\
\text{by induction hypothesis on (3) with (2).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + \forall A \rightarrow A_t \} \rangle & (4) \\
\text{by (rLocApp) with (3) and (1).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tRecfun}\)) - We have:
\[
\begin{align*}
\Gamma; \Delta \vdash \text{loc } x_i : A_t + \rightarrow \forall A \rightarrow A_t + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; \Delta \vdash \text{loc } x_i : A_t + \rightarrow \forall A \rightarrow A_t + * - (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (3) \\
\text{by induction hypothesis on (3) with (2).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + \forall A \rightarrow A_t \} \rangle & (4) \\
\text{by (rLocApp) with (3) and (1).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tLocFun}\)) - We have:
\[
\begin{align*}
\Gamma; \Delta \vdash \text{loc } x_i : A_t + \rightarrow \forall A \rightarrow A_t + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; \Delta \vdash \text{loc } x_i : A_t + \rightarrow \forall A \rightarrow A_t + * - (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (3) \\
\text{by induction hypothesis on (3) with (2).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + \forall A \rightarrow A_t \} \rangle & (4) \\
\text{by (rLocApp) with (3) and (1).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tRecfun}\)) - We have:
\[
\begin{align*}
\Gamma; \Delta \vdash v_i[v/s] & : A_t + \forall A \rightarrow A_t + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; \Delta \vdash v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (3) \\
\text{by induction hypothesis on (3) with (2).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + \forall A \rightarrow A_t \} \rangle & (4) \\
\text{by (rLocApp) with (3) and (1).} & \\
\end{align*}
\]
Thus, we conclude.

Case (\(\text{tRecfun}\)) - We have:
\[
\begin{align*}
\Gamma; \Delta \vdash v_i[v/s] & : A_t + \forall A \rightarrow A_t + * - (1) \\
\text{by hypothesis.} & \\
\Gamma; \Delta \vdash v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (2) \\
\text{by inversion on (rRecfun) with (2).} & \\
\Gamma; \Delta_0 + v_i[v/s] & : v_i[v/s] : A_t + \forall A \rightarrow A_t & (3) \\
\text{by induction hypothesis on (3) with (2).} & \\
\Gamma; \Delta_0 + \langle\{v_i[v/s] : A_t + \forall A \rightarrow A_t \} \rangle & (4) \\
\text{by (rLocApp) with (3) and (1).} & \\
\end{align*}
\]
Thus, we conclude.
Thus, we conclude.

Case (rCap-Stack) - We have:

1. $\Gamma, v : !A' +$
2. $\Gamma, x : A' : \Delta_0 + v : A + \Delta_1$
   by hypothesis.
3. $\Delta_0 + e[v/x] : A + \Delta_1$
   by inversion on (rCap-Stack) with (2).
4. $\Delta_0 + e[v/x] : A + \Delta_1$
   by induction hypothesis with (1) and (3).
5. $\Delta_0 + e[v/x] : A + \Delta_1$
   by (rCap-Stack) with (4).

Thus, we conclude.

Case (rCap-Unstack) - We have:

1. $\Gamma, v : !A' +$
2. $\Gamma, x : A' : \Delta_0 + v : A + \Delta_1 + \Delta_2$
   by hypothesis.
3. $\Delta_0 + e[v/x] : A + \Delta_1$
   by inversion on (rCap-Unstack) with (2).
4. $\Delta_0 + e[v/x] : A + \Delta_1$
   by induction hypothesis with (1) and (3).
5. $\Delta_0 + e[v/x] : A + \Delta_1$
   by (rCap-Unstack) with (4).

Thus, we conclude.

Case (rFrame) - We have:

1. $\Gamma, v : !A' +$
2. $\Gamma, x : A' : \Delta_0 + v : A + \Delta_1 + \Delta_2$
   by hypothesis.
3. $\Delta_0 + e[v/x] : A + \Delta_1$
   by inversion on (rFrame) with (2).
4. $\Delta_0 + e[v/x] : A + \Delta_1$
   by induction hypothesis with (1) and (3).
5. $\Delta_0 + e[v/x] : A + \Delta_1$
   by (rFrame) with $\Delta_2$.

Thus, we conclude.

Case (rSubsumption) - We have:

1. $\Gamma, v : !A' +$
2. $\Gamma, x : A' : \Delta_0 + v : A + \Delta_1 + \Delta_2$
   by hypothesis.
3. $\Delta_0 < \Delta_2$
   by inversion on (rSubsumption) with (2).
4. $\Delta_0 < \Delta_2$
   by induction hypothesis with (1) and (3).
5. $\Delta_0 < \Delta_2$
   by (rSubsumption) with (2), (3), (5), and (6).

Thus, we conclude.

Case (rTco) - We have:

1. $\Gamma, v : !A' +$
2. $\Gamma, x : A' : \Delta_0 + v_0 : \#A_1 + \Delta_1$
   by hypothesis.
3. $\Delta_0 + v_0[v/x] : A + \Delta_1$
   by inversion on (rTco) with (2).
4. $\Delta_0 + v_0[v/x] : A + \Delta_1$
   by induction hypothesis with (1) and (3).
5. $\Delta_0 + v_0[v/x] : A + \Delta_1$
   by (rTco) with (4).
6. $\Gamma, \Delta_0 + v_0[v/x] : \#A_1 + \Delta_1$
   by (vs.20) on (5).

Thus, we conclude.

Case (rCase) - We have:

1. $\Gamma, v : !A' +$
2. $\Gamma, x : A' : \Delta_0 + \text{case } v_0 \text{ of } t : \pi_j \Rightarrow e_j \text{ end} : A + \Delta_1$
   by hypothesis.
3. $\Gamma, x : A' : \Delta_0 + v_0 : \sum_j \#A_j' + \#A'$
   by inversion on (rCase) with (2).
4. $\Gamma, x : A' : \Delta_0 + v_0 : \sum_j \#A_j' + \#A'$
   by def. of substitution up to rename of bounded variables.
5. $\Delta_0 + v_0[v/x] : \sum_j \#A_j' + \Delta_1'$
   by induction hypothesis on (3) and (1).

Thus, we conclude.

Case (rLoc) - Analogous to other cases such as (rLoc-Ops).

3. (Location Variable)

Proof. We proceed by induction on the typing derivation of

$$\Gamma, v : \text{loc } \Delta_0 + e : A + \Delta_1.$$

Case (rRef) - We have:

1. $\Gamma, v : \text{loc } \text{loc } \to \rho_0 : \text{ref } \rho_0 +$
2. $\rho : \text{loc } \in \Gamma$
   by hypothesis.
3. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by typing.
4. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (Well-Formed Type Substitution - Gamma) on (3), (2).
5. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (ls.3.3) on (4).
6. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (rRef) with (5).
7. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (ls.3.3), (ls.2.10) on (6).

Thus, we conclude.

Case (rPure) - We have:

1. $\Gamma, v : \text{loc } \text{ref } \rho_0 +$
2. $\rho : \text{loc } \in \Gamma$
   by hypothesis.
3. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by inversion on (rPure) with (1).
4. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by induction hypothesis with (2) and (3).
5. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (rPure) on (4).
6. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (ls.2.4) on (5).

Thus, we conclude.

Case (rUnr) - We have:

1. $\Gamma, v : \text{loc } \text{loc } \to [] +$
2. $\rho : \text{loc } \in \Gamma$
   by hypothesis.
3. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by typing.
4. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (Well-Formed Type Substitution - Gamma) on (3), (2).
5. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (rUnr) with (4).
6. $\Gamma, v : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (ls.2.7), (ls.4.1) on (5) and noting that regardless if
   $r$ occurs or not in $v$ its type remains unchanged.

Thus, we conclude.

Case (rPure-Read) - We have:

1. $\Gamma, x : A : \text{loc } \to x : A +$
2. $\rho : \text{loc } \in \Gamma$
   by hypothesis.
3. $\Gamma, x : A : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by typing.
4. $\Gamma, x : A : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (Well-Formed Type Substitution) on (3), (2).
5. $\Gamma, x : A : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (ls.3.2) on (4).
6. $\Gamma, x : A : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (rPure-Read) with (5).
7. $\Gamma, x : A : \text{loc } \text{loc } \text{ref } \rho_0 / \rho / \text{ref } \rho_0 / \rho / +$
   by (ls.3.1), (ls.2.4), (ls.1.2) on (6).

Thus, we conclude.

Case (rLinear-Read) - We have:
Thus, we conclude.

Case (rPore-Elim) - We have:
\[
\Gamma, t : \text{loc} \vdash A \rightarrow t : A + \Delta_1
\]
\[
\rho : \text{loc} \in \Gamma
\]
by hypothesis.

\[
\Gamma, t : \text{loc} \vdash A : A + \Delta_1
\]
\[
\rho : \text{loc} \in \Gamma
\]
by typing.

\[
\Gamma[t] \vdash \text{w} f \text{w}
\]
by (Well-Formed Type Substitution) with (3) and (2).

\[
\Gamma[t] \vdash A \text{ type}
\]
by (Well-Formed Delta) on (1)

\[
\Gamma[t] \vdash A[p[t]] \text{ type}
\]
by (Well-Formed Type Substitution) with (6) and (2).

\[
\Gamma[t], x : A[p[t]] \vdash x : A[p[t]] + \Delta_1
\]
by (rLinear-Rex) with (5).

\[
\Gamma[t], (\Delta_1 : A[p[t]] + \Delta_1) \vdash (v_0 \vdash v_1) : A[p[t]] + \Delta_1
\]
by (rAssump) on (6) and (7).

Thus, we conclude.

Case (rPore-Elim) - Analagous to (rPore-Elim).

Case (rRecursion) - We have:
\[
\Gamma, t : \text{loc} \vdash A \rightarrow t : A + \Delta
\]
\[
\rho : \text{loc} \in \Gamma
\]
by hypothesis.

\[
\Gamma, t : \text{loc} \vdash \text{ref } t_0 : \text{ref } t_0 : A + \Delta
\]
\[
\rho : \text{loc} \in \Gamma
\]
by inversion on (rRecursion) with (1).

\[
\Gamma[t], \Delta_0 \vdash p[t] : A[p[t]] + \Delta_0[p[t]]
\]
by induction hypothesis on (2) and (3).

\[
\Gamma[t], \Delta_0 \vdash p[t] : A[p[t]] + \Delta_0[p[t]]
\]
by (rRef) with (4).

\[
\Gamma[t], \Delta_0 \vdash p[t] : A[p[t]] + \Delta_0[p[t]]
\]
by (rRecursion) on (7).

Thus, we conclude.

Case (rApplicaton) - We have:
\[
\Gamma, t : \text{loc} \vdash A \rightarrow t : A + \Delta_2
\]
\[
\rho : \text{loc} \in \Gamma
\]
by hypothesis.

\[
\Gamma, t : \text{loc} \vdash \text{ref } t_0 : \text{ref } t_0 : A + \Delta_2
\]
\[
\rho : \text{loc} \in \Gamma
\]
by inversion on (rAssump) with (1).

\[
\Gamma[t], \Delta_1 \vdash v[p[t]] : A[p[t]] + \Delta_1[p[t]]
\]
by induction hypothesis on (3) with (2).

\[
\Gamma[t], \Delta_1 \vdash v[p[t]] : (\text{ref } p[t]/p[t]) + (\Delta_1 : p[t]/p[t])
\]
by induction hypothesis on (4) with (2).

\[
\Gamma[t], \Delta_1 \vdash v[p[t]] : (\text{ref } p[t]/p[t]) + (\Delta_1 : p[t]/p[t])
\]
by (rApplicaton) on (8).

\[
\Gamma[t], \Delta_1 \vdash v[p[t]] + v[p[t]], \Delta_1 : p[t]/p[t]
\]
by (rApplicaton) on (5) and (3), and (rApplicaton) on (6).

Thus, we conclude.
Thus, we conclude.

Case (r(forall-loc)) - We have:

\[ \Gamma, t : \text{loc}, \Delta, x : A \vdash e : A + \ldots \]
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t : \text{loc}, \Delta, x : A \vdash e : A + \ldots \]
by hypothesis.
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by inversion on (r(forall-loc)) with (1).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by induction hypothesis on (2) and (3).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by (ls.4.2) on (4).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by (r(forall-loc)) on (5).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by (ls.1.3), (ls.2.5) on (6).
Thus, we conclude.

Case (r(forall-loc)) - We have:

\[ \Gamma, t : \text{loc}, \Delta, x : A \vdash e : A + \ldots \]
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t : \text{loc}, \Delta, x : A \vdash e : A + \ldots \]
by hypothesis.
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by inversion on (r(forall-loc)) with (1).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by induction hypothesis on (2) and (3).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by (ls.3.3), (ls.2.3) with (4) on (5).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by (r(forall-loc)) on (6).
\[ \Gamma, t, \Delta, x : A \vdash e : A + \ldots \]
by (ls.1.13), (ls.2.8) with (4) on (7).
Thus, we conclude.

Case (r(loc-arr)) - We have:

\[ \Gamma, t : \text{loc}, \Delta, \alpha + \ldots \]
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t : \text{loc}, \Delta, \alpha + \ldots \]
by hypothesis.
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by inversion on (r(loc-arr)) with (1).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by induction hypothesis on (2) and (4).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by induction hypothesis on (2) and (3).
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.3.3), (ls.2.3) with (4) on (5).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (r(loc-arr)) on (6).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.1.12) on (7).
Thus, we conclude.

Case (r(loc-pack)) - We have:

\[ \Gamma, t : \text{loc}, \Delta, \alpha + \ldots \]
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by hypothesis.
\[ \Gamma, t : \text{loc}, \Delta, \alpha + \ldots \]
by inversion on (r(loc-pack)) with (1).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by induction hypothesis on (2) and (4).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.1.11), (ls.2.9) on (7), (5).
Thus, we conclude.

Case (r(loc-open)) - We have:

\[ \Gamma, t : \text{loc}, \Delta, \alpha + \ldots \]
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by hypothesis.
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.1.11), (ls.2.9) on (7), (5).
Thus, we conclude.

Case (r(loc-open)) - We have:

\[ \Gamma, t : \text{loc}, \Delta, \alpha + \ldots \]
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by hypothesis.
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by inversion on (r(loc-open)) with (1).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by induction hypothesis on (2) and (3).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by induction hypothesis on (2) and (4).
\[ \rho : \text{loc} \in \Gamma \]
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by def. of substitution up to rename of bound location variables.
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.10) on (5), (7).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.12) on (8) and (9).
\[ \Gamma, t, \Delta, \alpha + \ldots \]
by (ls.12) on (6).
Thus, we conclude.
4. (Type Variable), analogous to the (Location Variable) proof.
Therefore, by (3) and (5) we conclude.

**Case (rLoc-Open)** - Not a value.

**Case (rForall-Type)** - We have:

\[\Gamma, \Delta \vdash (X : \forall X A) +\]  
by hypothesis.  
\[\Gamma, \Delta \vdash A[X/x] +\]  
by inversion on (rLoc-Pack) with (1).

\[\Delta \subseteq \Delta_1\]  
(3)

\[\Gamma, \Delta \vdash v : A[X/x] +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta \vdash (p, v) : \exists X A +\]  
by (rLoc-Pack) on (4).

Therefore, by (3) and (5) we conclude.

**Case (rType-App)** - Not a value.

**Case (rType-Pack)** - We have:

\[\Gamma, \Delta \vdash (A_1, v : \exists X A_0) +\]  
by hypothesis.  
\[\Gamma, \Delta \vdash A_0[A_1/X] +\]  
by inversion on (rType-Pack) with (1).

\[\Delta \subseteq \Delta_1\]  
(3)

\[\Gamma, \Delta \vdash v : A_0[A_1/X] +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta \vdash (A_1, v) : \exists X A_0 +\]  
by (rType-Pack) on (4).

Therefore, by (3) and (5) we conclude.

**Case (rType-Open)** - Not a value.

**Case (rCap-Env)** - Environment not closed.

**Case (rCap-Stack)** - We have:

\[\Gamma, \Delta \vdash v : A_0 : A_1 + \Delta_1\]  
by hypothesis.  
\[\Gamma, \Delta \vdash A_0 : A_1 + \Delta_1\]  
by inversion on (rCap-Stack) with (1).

\[\Delta \subseteq \Delta_1, \Delta_{01}\]  
(3)

\[\Gamma, \Delta \vdash v : A_0 +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta, A_1 \vdash v : A_0 + A_1\]  
by (rCap-Stack) on (4).

Therefore, by (3) and (6) we conclude.

(Note that \(A_1\) is immediate since a defocus-guarantee is not a type)

**Case (rCap-Unstack)** - We have:

\[\Gamma, \Delta_0 \vdash v : A_0 + \Delta_{01}, A_1\]  
by hypothesis.  
\[\Gamma, \Delta_0 \vdash A_0 : A_1 + \Delta_{01}\]  
by inversion on (rCap-Unstack) with (1).

\[\Delta_0 \subseteq \Delta_{01}, \Delta_1\]  
(3)

\[\Gamma, \Delta \vdash v : A_0 +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta, A_1 \vdash v : A_0 + A_1\]  
by (rCap-Unstack) with (4).

\[\Delta \subseteq \Delta_{01}, A_1\]  
(6)

\[\Gamma, \Delta \vdash v : \Delta_{01} +\]  
by induction hypothesis on (5).

\[\Delta_0 \subseteq \Delta_{01}, \Delta_1\]  
(8)  
by transitivity of subtyping with (3) and (6).

Therefore, by (7) and (8) we conclude.

**Case (rFrame)** - We have:

\[\Gamma, \Delta_0, \Delta_2 \vdash v : A + \Delta_1, \Delta_2\]  
by hypothesis.  
\[\Gamma, \Delta_0 \vdash A +\]  
by inversion on (rFrame) with (1).

\[\Delta_0 \subseteq \Delta_{01}, \Delta_1\]  
(3)

\[\Gamma, \Delta \vdash A +\]  
by induction hypothesis on (2).

\[\Delta_0, \Delta_2 \subseteq \Delta_{01}, \Delta_1, \Delta_2\]  
(4)  
by induction hypothesis on (2).

\[\Gamma, \Delta \vdash v : A +\]  
by (rFrame) on (4).

Therefore, by (3) and (5) we conclude.

**Case (rSubsumption)** - We have:

\[\Gamma, \Delta_0 \vdash v : A_1 + \Delta_1\]  
by hypothesis.  
\[\Delta_0 \subsetneq \Delta_{01}\]  
(1)

\[\Gamma, \Delta_0 \vdash v : A_1 + \Delta_1\]  
by inversion on (rSubsumption) with (1).

\[\Delta_0 \subseteq \Delta_{01}, \Delta_1\]  
(3)

\[\Gamma, \Delta \vdash A_1 +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta, \Delta_1 \vdash v : A_1 +\]  
by (rSubsumption) with (so:Symmetric) and (4) on (7).

Therefore, by (9) and (10) we conclude.

**Case (rTae)** - We have:

\[\Gamma, \Delta_0 \vdash 1v : 1A +\]  
by hypothesis.  
\[\Gamma, \Delta_0 \vdash v : A +\]  
by inversion on (rTae) with (1).

\[\Delta_0 \subsetneq \Delta_{01}\]  
(3)

\[\Gamma, \Delta \vdash v : A +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta \vdash 1v : 1A +\]  
by (rTae) on (4).

Therefore, by (5) and (3) we conclude.

**Case (rCap)** - Not a value.

**Case (rAlternative-Left)** - We have:

\[\Gamma, \Delta_0, A_0 \vdash A_1 + v : A_2 + \Delta_1\]  
by hypothesis.  
\[\Gamma, \Delta_0, A_0 \vdash v : A_2 + \Delta_1\]  
by inversion on (rAlternative-Left) with (1).

\[\Delta_0 \subsetneq \Delta_1, \Delta_2\]  
(4)

\[\Gamma, \Delta_0 \vdash v : A_2 +\]  
by induction hypothesis on (2).

\[\Gamma, \Delta_2, A_2 \vdash v : A_2 +\]  
by induction hypothesis on (3).

\[\Delta_0, A_0 @ A_1 \subseteq \Delta_1, \Delta_2\]  
(8)  
by (so:Alternative-L) on (4) and (6).

Therefore, by (7) and (8) we conclude.

**Case (rLet)** Not a value.
B.10 Preservation

Theorem 1 (Preservation). If $e_0$ is a closed expression such that:

$$\Gamma \vdash A + \Delta$$

then:

$$\Gamma \vdash A \vdash \Delta$$

for some $\Delta_1$, $\Gamma_1$. 

Proof. By induction on the typing derivation of $\Gamma \vdash A + \Delta$.

Case (tRef), (tPRef), (tUnit) - are values.

Case (tPure-Read), (tLinear-Read), (tPure-Elim) - not applicable, environments not closed.

Case (tNew) - We have:

$$\Gamma \vdash A \vdash B$$

then:

$$\Gamma \vdash A \vdash B$$

by hypothesis, with (tNew).

$$\Gamma \vdash A \vdash B$$

by inversion on (tNew) with (1).

$$\Gamma \vdash A \vdash B$$

by (Values Lemma) with (4).

$$\Gamma \vdash A \vdash B$$

by inversion on (tNew) with (3).

$$\Gamma \vdash A \vdash B$$

by (Subtyping Inversion) with (6).

Thus, if we make:

$$\Gamma_1 = \rho$$

We have:

$$\Gamma_1 \vdash A \vdash B$$

by (Weakening) (6) with $\Gamma_1$.

note that weakening is only valid in the lexical environments, $\Gamma_1$

(because inversion is immediate). $\Gamma_1$

by (tNew) with (10) and (11) with $\rho$.

Thus, if we make:

$$\Delta = A \vdash B$$

We have:

$$\Gamma_1 \vdash \rho : \text{ref} \vdash \rho$$

by (tRRef) with (15).

Since $\beta$ is empty, frame is immediate.

by (tNew) with (14) on (19) with $\Delta$.

by (tNew) with (20) and (21) we conclude.

Case (tDelete) - We have:

$$\Delta_0 \vdash \Delta_1$$

by inversion on (tDelete) with (1).

$$\Delta_1 \vdash \Delta_2$$

by (Values Lemma) with (4).

$$\Delta_2 \vdash \Delta_3$$

by (Store Typing Inversion) with (11).

by making:

$$\Gamma_1 = \rho$$

by (tNew) with (22) and (23) with (7) and (12).

by hypothesis, (tDelete).

by inversion on (tDelete) with (1).

by inversion on (tDelete) with (1).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).

by (tNew) with (21) and (22) with (18) and (19).

by (tNew) with (22) and (23) with (7) and (12).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).

by (tNew) with (21) and (22) with (18) and (19).

by (tNew) with (22) and (23) with (7) and (12).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).

by (tNew) with (21) and (22) with (18) and (19).

by (tNew) with (22) and (23) with (7) and (12).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).

by (tNew) with (21) and (22) with (18) and (19).

by (tNew) with (22) and (23) with (7) and (12).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).

by (tNew) with (21) and (22) with (18) and (19).

by (tNew) with (22) and (23) with (7) and (12).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).

by (tNew) with (21) and (22) with (18) and (19).

by (tNew) with (22) and (23) with (7) and (12).

by (tNew) with (13) and (14).

by (tNew) with (16) and (17).

by (tNew) with (18) and (19).

by (tNew) with (20) and (21) with (17).
The text contains mathematical and logical expressions that seem to be part of a formal system, possibly in the context of type theory or formal logic. The expressions include symbols that are common in such fields, such as \( \Delta \), \( \Gamma \), \( \rho \), \( \gamma \), and \( t \), as well as logical and type-theoretic operations denoted by symbols like \( \vdash \), \( \Rightarrow \), and \( \Leftrightarrow \). The text appears to be a proof or derivation, possibly in a computer science or mathematics document. The specific content of the proof involves the manipulation of types and terms, and the use of inference rules to establish certain properties or conclusions. The presence of Greek letters and mathematical symbols indicates a formal and rigorous approach to the subject matter.
Therefore, by (12) and (13) we conclude.

Case (rLoc-Pack) - Is a value.

Case (rLoc-Ops) - We have:
\[ \bar{f}_1 = e - \bar{\Delta}_1 + \bar{\Delta} \]
\[ \bar{\Delta}_1 = \bar{\Delta} \]
We trivially have:
\[ \bar{f}_0, \bar{f}_1, \bar{\Delta}_1 + \bar{\Delta} \]
(12)
\[ \bar{f}_0, \bar{f}_1, \bar{\Delta}_1 + H_0 \]
(13)
by (Subtyping Store Typing) using with (2) and (6).

Therefore, by (12) and (13) we conclude.

By making:
\[ \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
\[ \Delta_0, \hat{\rho} / \Delta \]
\[ \Delta_1, \hat{\rho} / \Delta \]
for some $\Delta_1, \Gamma_1$.

Therefore, we conclude.
- $\Gamma_1; \Delta_0, A_1 \vdash H_0$

(2.1)

Thus, we conclude.

**Case (t:FRAME)** - We have:

1. **Sub-Case (t:LearCoso):**

   \[
   \begin{align*}
   \Gamma_0; \Delta_0 + & \text{let } x = e_0 \text{ in } e_2 \text{ end} : A_1 + \Delta \\
   \Gamma_0; \Delta_0 + & H_0 \\
   (H_0 \parallel \text{let } x = e_1 \text{ in } e_2 \text{ end}) & \Rightarrow (H_1 \parallel \text{let } x = e_1 \text{ in } e_2 \text{ end}) \\
   (H_0 \parallel e_0) & \Rightarrow (H_1 \parallel e_1) \\
   \end{align*}
   \]

(1)

(2)

(3)

(4)

(5)

(6)

(7)

by hypothesis.

by inversion on (t:FRAME) with (1).

by inversion on (t:LearCoso) with (3).

by inversion on (t:FRAME) with (1).

by hypothesis.

by inversion on (t:LearCoso) with (3).

by (Values Lemma) with (4).

by (Values Lemma) with (4).

by (Values Lemma) with (4).

by (Values Lemma) with (4).

Therefore, by (9) and (6) we conclude.

2. **Sub-Case (t:Lear):**

   \[
   \begin{align*}
   \Gamma_0; \Delta_0 + & \text{let } x = v \text{ in } e \text{ end} : A_1 + \Delta \\
   \Gamma_0; \Delta_0 + & H \\
   (H \parallel \text{let } x = v \text{ in } e \text{ end}) & \Rightarrow (H \parallel ev/s) \\
   \end{align*}
   \]

(1)

(2)

(3)

(4)

(5)

(6)

(7)

(8)

(9)

(10)

(11)

by hypothesis.

by inversion on (t:Lear) with (1).

by inversion on (t:Lear) with (1).

by inversion on (t:Lear) with (1).

by (Values Lemma) with (4).

by (Values Lemma) with (4).

by (Values Lemma) with (4).

by (Values Lemma) with (4).

Therefore, by (9) and (6) we conclude.
B.11 Progress

Theorem 2 (Progress). If \( e_0 \) is a closed expression such that
\[
\Gamma, \Delta_0 \vdash e_0 : A + \Delta_1
\]
then either:

(value) \( e_0 \) is a value (v), or,

(steps) if exists \( H_0 \) such that \( \Gamma, \Delta_0 \vdash H_0 \) then
\[
(H_0, e_0) \rightarrow (H_1, e_1)
\]

Proof. By induction on the typing derivation of \( \Gamma, \Delta_0 \vdash e_0 : A + \Delta_1 \).

Case (\texttt{Ref}), (\texttt{Pure}), (\texttt{Unit}), (\texttt{Linear-Read}), (\texttt{Pure-Elms}) -
are all values or the environments are not closed.

Case (\texttt{New}) - We have:
\[
\Gamma, \Delta_0 \vdash v : \exists t \cdot \langle t, A \rangle + \Delta_1
\]
by hypothesis.

Thus, by (Values Lemma) and (Values Inversion Lemma) with (2).

Case (\texttt{Delete}) - We have:
\[
\Gamma, \Delta_0 \vdash v : \exists t \cdot A + \Delta_1
\]
by hypothesis.

Thus, by (Values Lemma) and (Values Inversion Lemma) with (2).

Case (\texttt{Assign}) - We have:
\[
\Gamma, \Delta_0 + v := v_1 : A + \Delta_0, \rho \vdash A_0
\]
by hypothesis.

Thus, by (Values Lemma) and (Values Inversion Lemma) with (2).

Case (\texttt{Function}) - is a value.

Case (\texttt{Frame}) - is a value.

Case (\texttt{Subsumption}) - We have:
\[
\Delta_0 \vdash e : A + \Delta_1
\]
by hypothesis.

Thus, by (Values Lemma) and (Values Inversion Lemma) with (2).

Case (\texttt{Tag}) - is a value.

Case (\texttt{Case}) - We have:
\[
\Gamma, \Delta_0 \vdash \text{case } v \text{ of } \Gamma_1 \theta_1 \rightarrow e_1 \text{ end } : A + \Delta_1
\]
by hypothesis.

Thus, by (Values Lemma) and (Values Inversion Lemma) with (2).

Case (\texttt{Loc-App}) - We have:
\[
\Gamma, \Delta_0 \vdash v : \forall A + \Delta_1
\]
by hypothesis.

Thus, by (Values Lemma) and (Values Inversion Lemma) with (2).
Therefore the expression is a value.

- If exists $H_0$ such that $\widetilde{\Gamma}, \widetilde{\Delta}_0 \vdash A_0 + H_0$ (5)
- By (Store Typing Inversion Lemma) on (5), we have that either:
  - $\vdash \widetilde{\Gamma}, \widetilde{\Delta}_0 \vdash H_0$ (6)
- Then by induction hypothesis on (2), we conclude that:
  - $\langle H_0 \parallel e \parallel \ advis \Gamma, \Delta_0 \rangle \implies \langle H_0' \parallel e' \parallel \ advis \Gamma, \Delta_0 \rangle$ (7)
- Thus, the expression steps, since $e$ cannot be a value.
- $\vdash \widetilde{\Gamma}, \widetilde{\Delta}_0 \vdash A_0$ (8)
- Then by induction hypothesis on (3), we conclude that:
  - $\langle H_0 \parallel e \parallel \ advis \Gamma, \Delta_0 \rangle \implies \langle H_0' \parallel e' \parallel \ advis \Gamma, \Delta_0 \rangle$ (9)
- Thus, the expression steps, since $e$ cannot be a value.
- Therefore, we conclude.

Case \((\tau \text{let})\) - We have:

\[
\begin{align*}
\vdash \widetilde{\Gamma}, \widetilde{\Delta}_0 \vdash x = e_0 \text{ in } e_1 & \text{ and } A \vdash \widetilde{\Delta}_1 & \text{ (1)} \\
\vdash \widetilde{\Gamma}, \widetilde{\Delta}_0 \vdash e_0 : A_0 \vdash \widetilde{\Delta}_0 & \text{ (2)} \\
\vdash \widetilde{\Gamma}, \widetilde{\Delta}_1, x : A_0 \vdash e_1 : A_1 \vdash \widetilde{\Delta}_2 & \text{ (3)}
\end{align*}
\]

By inversion on \((\tau \text{let})\) with (1).

By induction hypothesis on (2), we have that either:

- $e_0$ is a value \((v)\); (4)
- Thus, by \((\tau \text{let})\) the expression transitions.
- If exists $H_0$ such that $\vdash \widetilde{\Gamma}, \widetilde{\Delta}_0 \vdash H_0$ (5)
- $\langle H_0 \parallel e_0 \parallel \ advis \Gamma, \Delta_0 \rangle \implies \langle H_0' \parallel e'_0 \parallel \ advis \Gamma, \Delta_0 \rangle$ (6)
- Thus, by \((\tau \text{letConstr})\) the expression (1) transitions.
- Therefore, we conclude.