Sound and Complete Bidirectional Typechecking for Higher-Rank Polymorphism with Existentials and Indexed Types: Full definitions, lemmas and proofs

(Anonymous Authors)

March 16, 2018

The first part (Sections 1–2) of this supplementary material contains rules, figures and definitions omitted in the main paper for space reasons, and a list of judgment forms (Section 2).

The remainder (Sections A–K′) includes statements of all lemmas and theorems, along with full proofs, as well as statements of theorems and a few selected lemmas.

Contents

1 Figures 7

2 List of Judgments 17

A Properties of the Declarative System 18

1 Lemma (Declarative Well-foundedness) . . . . . . . . . . . . . . . . . . . . . . . . . . 18
2 Lemma (Declarative Weakening) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
3 Lemma (Declarative Term Substitution) . . . . . . . . . . . . . . . . . . . . . . . . . . 18
4 Lemma (Reflexivity of Declarative Subtyping) . . . . . . . . . . . . . . . . . . . . . . 18
5 Lemma (Subtyping Inversion) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
6 Lemma (Subtyping Polarity Flip) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
7 Lemma (Transitivity of Declarative Subtyping) . . . . . . . . . . . . . . . . . . . . . . 19

B Substitution and Well-formedness Properties 19

8 Lemma (Substitution—Well-formedness) . . . . . . . . . . . . . . . . . . . . . . . . . . 19
9 Lemma (Uvar Preservation) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
10 Lemma (Sorting Implies Typing) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
11 Lemma (Right-Hand Substitution for Sorting) . . . . . . . . . . . . . . . . . . . . . . 19
12 Lemma (Right-Hand Substitution for Propositions) . . . . . . . . . . . . . . . . . . . 19
13 Lemma (Right-Hand Substitution for Typing) . . . . . . . . . . . . . . . . . . . . . . 19
14 Lemma (Substitution for Sorting) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
15 Lemma (Substitution for Prop Well-Formedness) . . . . . . . . . . . . . . . . . . . . 19
16 Lemma (Substitution for Type Well-Formedness) . . . . . . . . . . . . . . . . . . . . 19
17 Lemma (Substitution Stability) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
18 Lemma (Equal Domains) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

C Properties of Extension 19

19 Lemma (Declaration Preservation) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
20 Lemma (Declaration Order Preservation) . . . . . . . . . . . . . . . . . . . . . . . . . . 19
21 Lemma (Reverse Declaration Order Preservation) . . . . . . . . . . . . . . . . . . . . 19
22 Lemma (Extension Inversion) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Lemma (Deep Evar Introduction)</td>
<td>20</td>
</tr>
<tr>
<td>24</td>
<td>Lemma (Soft Extension)</td>
<td>20</td>
</tr>
<tr>
<td>26</td>
<td>Lemma (Parallel Admissibility)</td>
<td>21</td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Parallel Extension Solution)</td>
<td>21</td>
</tr>
<tr>
<td>28</td>
<td>Lemma (Parallel Variable Update)</td>
<td>21</td>
</tr>
<tr>
<td>29</td>
<td>Lemma (Substitution Monotonicity)</td>
<td>21</td>
</tr>
<tr>
<td>30</td>
<td>Lemma (Substitution Invariance)</td>
<td>21</td>
</tr>
<tr>
<td>31</td>
<td>Lemma (Split Extension)</td>
<td>21</td>
</tr>
<tr>
<td>C.1</td>
<td>Reflexivity and Transitivity</td>
<td>22</td>
</tr>
<tr>
<td>32</td>
<td>Lemma (Extension Reflexivity)</td>
<td>22</td>
</tr>
<tr>
<td>33</td>
<td>Lemma (Extension Transitivity)</td>
<td>22</td>
</tr>
<tr>
<td>C.2</td>
<td>Weakening</td>
<td>22</td>
</tr>
<tr>
<td>34</td>
<td>Lemma (Suffix Weakening)</td>
<td>22</td>
</tr>
<tr>
<td>35</td>
<td>Lemma (Suffix Weakening)</td>
<td>22</td>
</tr>
<tr>
<td>36</td>
<td>Lemma (Extension Weakening (Sorts))</td>
<td>22</td>
</tr>
<tr>
<td>37</td>
<td>Lemma (Extension Weakening (Props))</td>
<td>22</td>
</tr>
<tr>
<td>38</td>
<td>Lemma (Extension Weakening (Types))</td>
<td>22</td>
</tr>
<tr>
<td>C.3</td>
<td>Principal Typing Properties</td>
<td>22</td>
</tr>
<tr>
<td>39</td>
<td>Lemma (Principal Agreement)</td>
<td>22</td>
</tr>
<tr>
<td>40</td>
<td>Lemma (Right-Hand Subst. for Principal Typing)</td>
<td>22</td>
</tr>
<tr>
<td>41</td>
<td>Lemma (Extension Weakening for Principal Typing)</td>
<td>22</td>
</tr>
<tr>
<td>42</td>
<td>Lemma (Inversion of Principal Typing)</td>
<td>22</td>
</tr>
<tr>
<td>C.4</td>
<td>Instantiation Extends</td>
<td>22</td>
</tr>
<tr>
<td>43</td>
<td>Lemma (Instantiation Extension)</td>
<td>22</td>
</tr>
<tr>
<td>C.5</td>
<td>Equivalence Extends</td>
<td>23</td>
</tr>
<tr>
<td>44</td>
<td>Lemma (Elimeq Extension)</td>
<td>23</td>
</tr>
<tr>
<td>45</td>
<td>Lemma (Elimprop Extension)</td>
<td>23</td>
</tr>
<tr>
<td>46</td>
<td>Lemma (Checkeq Extension)</td>
<td>23</td>
</tr>
<tr>
<td>47</td>
<td>Lemma (Checkprop Extension)</td>
<td>23</td>
</tr>
<tr>
<td>48</td>
<td>Lemma (Prop Equivalence Extension)</td>
<td>23</td>
</tr>
<tr>
<td>49</td>
<td>Lemma (Equivalence Extension)</td>
<td>23</td>
</tr>
<tr>
<td>C.6</td>
<td>Subtyping Extends</td>
<td>23</td>
</tr>
<tr>
<td>50</td>
<td>Lemma (Subtyping Extension)</td>
<td>23</td>
</tr>
<tr>
<td>C.7</td>
<td>Typing Extends</td>
<td>23</td>
</tr>
<tr>
<td>51</td>
<td>Lemma (Typing Extension)</td>
<td>23</td>
</tr>
<tr>
<td>C.8</td>
<td>Unfiled</td>
<td>23</td>
</tr>
<tr>
<td>52</td>
<td>Lemma (Context Partitioning)</td>
<td>23</td>
</tr>
<tr>
<td>54</td>
<td>Lemma (Completing Stability)</td>
<td>23</td>
</tr>
<tr>
<td>55</td>
<td>Lemma (Completing Completeness)</td>
<td>23</td>
</tr>
<tr>
<td>56</td>
<td>Lemma (Confluence of Completeness)</td>
<td>23</td>
</tr>
<tr>
<td>57</td>
<td>Lemma (Multiple Confluence)</td>
<td>23</td>
</tr>
<tr>
<td>59</td>
<td>Lemma (Canonical Completion)</td>
<td>24</td>
</tr>
<tr>
<td>60</td>
<td>Lemma (Split Solutions)</td>
<td>24</td>
</tr>
<tr>
<td>D</td>
<td>Internal Properties of the Declarative System</td>
<td>24</td>
</tr>
<tr>
<td>61</td>
<td>Lemma (Interpolating With and Exists)</td>
<td>24</td>
</tr>
<tr>
<td>62</td>
<td>Lemma (Case Invertibility)</td>
<td>24</td>
</tr>
<tr>
<td>E</td>
<td>Miscellaneous Properties of the Algorithmic System</td>
<td>24</td>
</tr>
<tr>
<td>63</td>
<td>Lemma (Well-Formed Outputs of Typing)</td>
<td>24</td>
</tr>
</tbody>
</table>
H’ Decidability of Algorithmic Subtyping

H’’.1 Lemmas for Decidability of Subtyping
73 Proof of Lemma (Substitution Isn’t Large) .................................................. 95
74 Proof of Lemma (Instantiation Solves) ....................................................... 95
75 Proof of Lemma (Checkeq Solving) ........................................................... 95
76 Proof of Lemma (Prop Equiv Solving) ....................................................... 96
77 Proof of Lemma (Equiv Solving) ............................................................... 97
78 Proof of Lemma (Decidability of Propositional Judgments) ..................... 98
79 Proof of Lemma (Decidability of Equivalence) ........................................ 99
H’’.2 Decidability of Subtyping ................................................................. 100
1 Proof of Theorem (Decidability of Subtyping) ........................................ 100
H’’.3 Decidability of Matching and Coverage ............................................. 101
80 Proof of Lemma (Decidability of Expansion Judgments) ....................... 101
2 Proof of Theorem (Decidability of Coverage) ......................................... 102
H’’.4 Decidability of Typing ................................................................. 102
3 Proof of Theorem (Decidability of Typing) ............................................. 102

I’ Determinacy
81 Proof of Lemma (Determinacy of Auxiliary Judgments) ....................... 104
82 Proof of Lemma (Determinacy of Equivalence) ..................................... 105
4 Proof of Theorem (Determinacy of Subtyping) ...................................... 106
5 Proof of Theorem (Determinacy of Typing) ............................................. 106

J’ Soundness
J’’.1 Instantiation............................................................................................ 108
83 Proof of Lemma (Soundness of Instantiation) .................................... 108
84 Proof of Lemma (Soundness of Checkeq) ........................................... 109
85 Proof of Lemma (Soundness of Propositional Equivalence) ................. 110
86 Proof of Lemma (Soundness of Algorithmic Equivalence) .................. 111
J’’.2 Soundness of Checkprop ................................................................. 112
87 Proof of Lemma (Soundness of Checkprop) ........................................ 112
J’’.3 Soundness of Eliminations (Equality and Proposition) ...................... 113
88 Proof of Lemma (Soundness of Equality Elimination) ......................... 113
6 Proof of Theorem (Soundness of Algorithmic Subtyping) .................... 116
J’’.4 Soundness of Typing ........................................................................... 118
7 Proof of Theorem (Soundness of Match Coverage) .......................... 118
89 Proof of Lemma (Well-formedness of Algorithmic Typing) ............... 119
8 Proof of Theorem (Soundness of Algorithmic Typing) ....................... 120

K’ Completeness

K’’.1 Completeness of Auxiliary Judgments .............................................. 133
90 Proof of Lemma (Completeness of Instantiation) .............................. 133
91 Proof of Lemma (Completeness of Checkeq) ...................................... 136
92 Proof of Lemma (Completeness of Elimeq) ........................................... 137
93 Proof of Lemma (Substitution Upgrade) ............................................. 139
94 Proof of Lemma (Completeness of Propequiv) ................................... 140
95 Proof of Lemma (Completeness of Checkprop) ................................. 140
K’’.2 Completeness of Equivalence and Subtyping ................................ 141
96 Proof of Lemma (Completeness of Equiv) ........................................... 141
9 Proof of Theorem (Completeness of Subtyping) ............................... 144
K’’.3 Completeness of Typing .................................................................. 148
10 Proof of Theorem (Completeness of Match Coverage) .................... 148
11 Proof of Theorem (Completeness of Algorithmic Typing) ............... 149
# 1 Figures

We repeat some figures from the main paper. In Figures 5a and 12a, we include rules omitted from the main paper for space reasons.

\[
\Gamma \vdash P \text{ true} \quad \text{Under context } \Gamma, \text{ check } P
\]

\[
\Gamma \vdash \varepsilon \triangleleft A \quad \Gamma \vdash \varepsilon \Rightarrow A \quad \text{Under context } \Gamma, \text{ expression } \varepsilon \text{ checks against input type } A
\]

\[
\Gamma \vdash \sigma : A \quad \Gamma \varepsilon : C[\varepsilon] \quad \text{Under context } \Gamma, \text{ passing spine } \sigma \text{ to a function of type } A \text{ synthesizes type } C; \text{ in the } [\varepsilon] \text{ form, recover principality in } q \text{ if possible}
\]

\[
\frac{x : A \in \Gamma \quad \Gamma \vdash A \text{ type} 
\Gamma \vdash e \triangleleft A} {\Gamma \vdash \varepsilon \triangleleft A} \quad \text{DeclAnno}
\]

\[
\frac{v \in \text{chk-I} \quad \Gamma, \alpha : \kappa \vdash v \triangleleft A \quad \Gamma \vdash \lambda \alpha . e \triangleleft A \quad \Gamma \vdash \lambda \alpha . e \Rightarrow A \quad \text{Decl-\lambda_1} 
\text{for all } C', \text{ if } \Gamma \vdash s : A ! \gg C' \text{ then } C' = C
\qquad \frac{\Gamma \vdash s : A ! \gg C ! [!] 
\Gamma \vdash \cdot : A \gg A} {\Gamma \vdash \cdot : A \gg A} \quad \text{DeclEmptySpine}
\]

\[
\frac{\Gamma \vdash e \triangleleft A_k \quad \Gamma \vdash \text{inj}_k e \triangleleft A_1 + A_2} {\Gamma \vdash \text{inj}_k e \triangleleft A_1 + A_2} \quad \text{Decl+}_{k}
\]

\[
\frac{\Gamma \vdash t = \text{zero true} 
\Gamma \vdash [] \bigtriangleup \text{[Vec } t A \text{]}} {\Gamma \vdash e \triangleleft A} \quad \Gamma \vdash \Pi : A \triangleleft C \quad \Gamma \vdash \Pi \text{ covers } A 
\frac{\Gamma \vdash \Pi \text{ covers } A} {\Gamma \vdash \Pi \triangleleft C} \quad \Gamma \vdash e \Rightarrow A 
\frac{\text{mgu}(\sigma, \tau) = \bot} {\Gamma \vdash (\sigma = \tau) \vdash e \triangleleft C} \quad \text{DeclCheck\bot}
\]

\[
\frac{\text{mgu}(\sigma, \tau) = \emptyset 
\emptyset(\Gamma) \vdash \emptyset(e) \triangleleft \emptyset(C)} {\Gamma \vdash (\sigma = \tau) \vdash e \triangleleft C} \quad \text{DeclCheckUnify}
\]

Figure 5a: Declarative typing, including rules omitted from main paper
1 Figures

Figure 12a: Algorithmic typing, including rules omitted from main paper

$$\Gamma \vdash e \leftrightarrow A \vdash \Delta$$ Under input context $\Gamma$, expression $e$ checks against input type $A$, with output context $\Delta$

$$\Gamma \vdash e \Rightarrow A \vdash \Delta$$ Under input context $\Gamma$, expression $e$ synthesizes output type $A$, with output context $\Delta$

$$\Gamma \vdash s : A \Rightarrow C \vdash \Delta$$ Under input context $\Gamma$, passing spine $s$ to a function of type $A$ synthesizes type $C$; in the $[\cdot]$ form, recover principality in $q$ if possible

$$\Gamma \vdash \text{inj} \alpha \vdash \Delta \vdash \Theta \vdash \alpha : \text{pol}(\{B\}) \vdash \Delta$$

$$\Gamma \vdash (x : A) \Rightarrow [\Gamma \vdash A \Rightarrow \Theta] \vdash \Delta$$

$$\Gamma \vdash (\text{rec } x . v) \Rightarrow A \vdash \Delta$$

$$\Gamma \vdash A \Rightarrow \text{type} \quad \Gamma \vdash e \Rightarrow [\Gamma / A \Rightarrow \Delta]$$

$$\Gamma \vdash (e : A) \Rightarrow [\Delta / A \Rightarrow \Delta]$$

$$\Gamma \vdash \text{chk-}I \quad \Gamma \vdash e : [\Theta] A \Rightarrow C \vdash \Delta$$

$$\Gamma \vdash e : \forall x : \alpha . A \vdash \Delta$$

$$\Gamma \vdash e \Rightarrow (\forall x : \alpha . A) \Rightarrow \Delta$$

$$\Gamma \vdash s : A ! \Rightarrow C \vdash \Delta$$

$$\text{SpineRecover}$$

$$\text{SpinePass}$$

$$\text{Spine}$$

$$\text{EmptySpine}$$

$$\text{Nil}$$

$$\text{Cons}$$

$$\Rightarrow \text{Spine}$$

March 16, 2018
\[ \Psi \vdash t : \kappa \quad \text{Under context } \Psi, \text{ term } t \text{ has sort } \kappa \]

\[ \begin{align*}
\Psi & \vdash \alpha : \kappa \\
\Psi & \vdash 1 : \kappa \\
\Psi & \vdash t_1 \odot t_2 : \kappa \\
\Psi & \vdash \text{zero} : \kappa \\
\end{align*} \]

\[ \Psi \vdash \alpha : \kappa \quad \text{UvarSort} \]

\[ \Psi \vdash 1 : \kappa \quad \text{UnitSort} \]

\[ \Psi \vdash t_1 \odot t_2 : \kappa \quad \text{BinSort} \]

\[ \Psi \vdash \text{zero} : \kappa \quad \text{ZeroSort} \]

\[ \Psi \vdash t : \kappa \quad \text{SuccSort} \]

\[ \Psi \vdash P \quad \text{prop} \quad \text{Under context } \Psi, \text{ proposition } P \text{ is well-formed} \]

\[ \begin{align*}
\Psi & \vdash t : \kappa \\
\Psi & \vdash t : \kappa \\
\Psi & \vdash t = t' \quad \text{prop} \\
\end{align*} \]

\[ \Psi \vdash \text{EqDeclProp} \]

\[ \Psi \vdash \alpha : \kappa \quad \text{DeclUvarWF} \]

\[ \Psi \vdash 1 \quad \text{DeclUnitWF} \]

\[ \begin{align*}
\Psi & \vdash A \quad \text{type} \\
\Psi & \vdash B \quad \text{type} \\
\Psi & \vdash A \odot B \quad \text{DeclBinWF} \\
\Psi, \alpha : \kappa & \vdash A \quad \text{DeclAllWF} \\
\Psi, \alpha : \kappa & \vdash A \quad \text{DeclExistsWF} \\
\Psi & \vdash P \quad \text{prop} \\
\Psi & \vdash A \quad \text{DeclImpliesWF} \\
\Psi & \vdash P \quad \text{prop} \\
\Psi & \vdash A \quad \text{DeclWithWF} \\
\end{align*} \]

\[ \Psi \vdash \vec{A} \quad \text{types} \quad \text{Under context } \Psi, \text{ types in } \vec{A} \text{ are well-formed} \]

\[ \begin{align*}
\text{for all } A \in \vec{A}, \\
\Psi & \vdash A \quad \text{DeclTypevecWF} \\
\end{align*} \]

\[ \Psi \text{ ctx} \quad \text{Declarative context } \Psi \text{ is well-formed} \]

\[ \begin{align*}
\Psi \text{ ctx} & \quad \text{EmptyDeclCtx} \\
\Psi \text{ ctx} & \quad x \notin \text{dom}(\Psi) \\
\Psi, \alpha & \vdash \text{A \ type} \\
\Psi, \alpha & \vdash \text{ctx} \\
\Psi \text{ ctx} & \quad \text{HypDeclCtx} \\
\Psi \text{ ctx} & \quad \text{VarDeclCtx} \\
\end{align*} \]

Figure 14: Sorting; well-formedness of propositions, types, and contexts in the declarative system
1 Figures

Γ ⊢ τ : κ

Under context Γ, term τ has sort κ

\[
\frac{(u : \kappa) \in \Gamma}{\Gamma \vdash u : \kappa} \quad \text{VarSort} \quad \frac{(\hat{\alpha} : \kappa = \tau) \in \Gamma}{\Gamma \vdash \hat{\alpha} : \kappa} \quad \text{SolvedVarSort} \\
\frac{\Gamma \vdash 1 : *}{\text{UnitSort}}
\]

\[
\frac{\Gamma \vdash \tau_1 : * \quad \Gamma \vdash \tau_2 : *}{\Gamma \vdash \tau_1 \oplus \tau_2 : *} \quad \text{BinSort} \quad \frac{\Gamma \vdash \text{zero} : \mathbb{N}}{\text{ZeroSort}} \quad \frac{\Gamma \vdash \text{succ}(t) : \mathbb{N}}{\text{SuccSort}}
\]

Γ ⊢ P prop

Under context Γ, proposition P is well-formed

\[
\frac{\Gamma \vdash t : \mathbb{N} \quad \Gamma \vdash t' : \mathbb{N}}{\Gamma \vdash t = t' : \text{prop}} \quad \text{EqProp}
\]

Γ ⊢ A type

Under context Γ, type A is well-formed

\[
\frac{(u : *) \in \Gamma}{\Gamma \vdash \text{var} : \kappa} \quad \text{VarWF} \quad \frac{(\hat{\alpha} : * = \tau) \in \Gamma}{\Gamma \vdash \hat{\alpha} : \kappa} \quad \text{SolvedVarWF} \quad \frac{\Gamma \vdash 1 : *}{\text{UnitWF}}
\]

\[
\frac{\Gamma \vdash A \\text{type} \quad \Gamma \vdash B \\text{type}}{\Gamma \vdash A \oplus B \\text{type}} \quad \text{BinWF} \quad \frac{\Gamma \vdash t : \mathbb{N}}{\Gamma \vdash A \\text{type}} \quad \frac{\Gamma \vdash \text{Vec} \ t \ A \ \text{type}}{\text{VecWF}}
\]

\[
\frac{\Gamma, \alpha \in \kappa \vdash A \text{type}}{\Gamma \vdash \forall \alpha : \kappa. A \text{ type}} \quad \text{ForallWF} \quad \frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \exists \alpha : \kappa. A \text{ type}} \quad \text{ExistsWF}
\]

\[
\frac{\Gamma \vdash P \text{ prop}}{\Gamma \vdash \forall A \text{ type} \quad \text{ImpliesWF}}
\]

Γ ⊢ A p type

Under context Γ, type A is well-formed and respects principality p

\[
\frac{\Gamma \vdash A \text{ type} \quad \text{FEV}(\Gamma|A) = \emptyset}{\Gamma \vdash A \ ! \text{ type}} \quad \text{PrincipalWF} \quad \frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \ ! \text{ type}} \quad \text{NonPrincipalWF}
\]

Γ ⊢ A [p] types

Under context Γ, types in A are well-formed [with principality p]

\[
\text{for all } A \in \bar{A}, \Gamma \vdash A \text{ type} \quad \text{TypevecWF} \quad \text{for all } A \in \bar{A}, \Gamma \vdash A \ p \text{ type} \quad \text{PrincipalTypevecWF}
\]

Γ ctx

Algorithmic context Γ is well-formed

\[
\frac{\text{EmptyCtx}}{\Gamma \vdash \text{ctx} \text{ empty}} \quad \frac{x \notin \text{dom}(\Gamma)}{\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \ ! \text{ ctx}} \quad \text{HypCtx}} \quad \frac{x \notin \text{dom}(\Gamma)}{\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \ ! \text{ ctx}} \quad \text{HypCtx}} \quad \frac{\text{FEV}(\Gamma|A) = \emptyset}{\Gamma, x : A \ ! \text{ ctx}} \quad \text{HypCtx}
\]

\[
\frac{\Gamma, u \notin \text{dom}(\Gamma)}{\frac{\Gamma, u : \kappa \text{ ctx}}{\text{VarCtx}}} \quad \frac{\hat{\alpha} \notin \text{dom}(\Gamma)}{\frac{\Gamma, \hat{\alpha} : \kappa = t \text{ ctx}}{\text{SolvedCtx}}} \quad \frac{\alpha \in \Gamma, \alpha = \tau \text{ ctx}}{\text{EqnVarCtx}} \quad \frac{\text{Γ ctx} \quad \Gamma, u \notin \Gamma}{\frac{\text{MarkerCtx}}{\Gamma, \uparrow u \text{ ctx}}}
\]

Figure 15: Well-formedness of types and contexts in the algorithmic system
\[\Gamma \vdash P \text{ true} \vdash \Delta\] Under context \(\Gamma\), check \(P\), with output context \(\Delta\)

\[
\begin{align*}
\Gamma \vdash t_1 \overset{\Delta}{=} t_2 : N \to \Delta & \quad \text{CheckpropEq} \\
\Gamma \vdash t_1 = t_2 \text{ true} \vdash \Delta &
\end{align*}
\]

\[\Gamma / P \vdash \Delta\] Incorporate hypothesis \(P\) into \(\Gamma\), producing \(\Delta\) or inconsistency \(\bot\)

\[
\begin{align*}
\Gamma / t_1 \overset{\Delta}{=} t_2 : N \to \Delta & \quad \text{ElimpropEq} \\
\Gamma / t_1 = t_2 \text{ true} \vdash \Delta &
\end{align*}
\]

Figure 16: Checking and assuming propositions

\[\Gamma \vdash t_1 \overset{\Delta}{=} t_2 : \kappa \to \Delta\] Check that \(t_1\) equals \(t_2\), taking \(\Gamma\) to \(\Delta\)

\[
\begin{align*}
\Gamma \vdash u \overset{\Delta}{=} u : \kappa \vdash \Gamma & \quad \text{CheckeqVar} \\
\Gamma \vdash 1 \overset{\Delta}{=} 1 : \star \to \Gamma & \quad \text{CheckeqUnit} \\
\Gamma \vdash \tau_1 \overset{\Delta}{=} \tau'_1 : \star \to \Theta & \quad \Theta \vdash [\Theta] \tau_2 \overset{\Delta}{=} [\Theta] \tau'_2 : \star \to \Delta & \quad \text{CheckeqBin} \\
\Gamma \vdash (\tau_1 \oplus \tau_2) \overset{\Delta}{=} (\tau'_1 \oplus \tau'_2) : \star \to \Delta &
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \text{zero} \overset{\Delta}{=} \text{zero} : N \to \Gamma & \quad \text{CheckeqZero} \\
\Gamma \vdash t_1 \overset{\Delta}{=} t_2 : N \to \Delta & \quad \text{CheckeqSucc} \\
\Gamma[\alpha : \kappa] \vdash \text{zero} : \kappa \to \Delta & \quad \text{CheckeqInstL} \\
\Gamma[\alpha : \kappa] \vdash \alpha := t : \kappa \to \Delta & \quad \alpha \notin \text{FV}(t) \\
\Gamma[\alpha : \kappa] \vdash t \overset{\Delta}{=} \alpha : \kappa \to \Delta & \quad \text{CheckeqInstR} \\
\Gamma[\alpha : \kappa] \vdash \text{succ}(t) = \text{succ}(t) : N \to \Delta &
\end{align*}
\]

Figure 17: Checking equations

\(t_1 \neq t_2\) \(t_1\) and \(t_2\) have incompatible head constructors

\[
\begin{align*}
\text{zero} \neq \text{succ}(t) & \\
\text{succ}(t) \neq \text{zero} & \\
1 \neq (\tau_1 \oplus \tau_2) & \\
(\tau_1 \oplus \tau_2) \neq 1 & \\
\sigma_1 \neq \oplus_1 & \\
(\sigma_1 \oplus \tau_1) \neq (\sigma_2 \oplus \tau_2) &
\end{align*}
\]

Figure 18: Head constructor clash
Figure 19: Eliminating equations
\[ \Gamma \vdash \alpha \triangleleft^\pm B \vdash \Delta \] Under input context \( \Gamma \), type \( A \) is a subtype of \( B \), with output context \( \Delta \)

\[ \begin{align*}
A \text{ not headed by } \forall &/ \exists & \Gamma \vdash A \equiv B \vdash \Delta & \triangleleft \text{Equiv} \\
B \text{ not headed by } \forall & & \Gamma \vdash A \triangleleft^\pm B \vdash \Delta & \triangleleft \forall L \\
& & \Gamma \vdash A \triangleleft^\pm B \vdash \Delta & \triangleleft \forall R \\
& & \Gamma \vdash A \triangleleft B \vdash \Delta & \triangleleft \exists L \\
& & \Gamma \vdash A \triangleleft B \vdash \Delta & \triangleleft \exists R \\
\end{align*} \]

\[ \Gamma \vdash P \equiv Q \vdash \Delta \] Under input context \( \Gamma \), check that \( P \) is equivalent to \( Q \) with output context \( \Delta \)

\[ \begin{align*}
& \Gamma \vdash t_1 \equiv t_2 : \text{N} \vdash \Theta & \Theta \vdash [\Theta]t_1' \equiv [\Theta]t_2' : \text{N} \vdash \Delta \\
& \Gamma \vdash (t_1 = t_1') \equiv (t_2 = t_2') \vdash \Delta & \equiv \text{PropEq} \\
\end{align*} \]

\[ \Gamma \vdash A \equiv B \vdash \Delta \] Under input context \( \Gamma \), check that \( A \) is equivalent to \( B \) with output context \( \Delta \)

\[ \begin{align*}
\Gamma \vdash \alpha \equiv \alpha \vdash \Gamma & \equiv \text{Var} \\
\Gamma \vdash \exists \alpha \equiv \exists \alpha \vdash \Gamma & \equiv \text{Exvar} \\
\Gamma \vdash 1 \equiv 1 \vdash \Gamma & \equiv \text{Unit} \\
\Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \vdash \Delta & \equiv \oplus \\
\Gamma \vdash (\forall \alpha : \kappa. A) \equiv (\forall \alpha : \kappa. B) \vdash \Delta & \equiv \forall \\
\Gamma \vdash P \equiv Q \vdash \Theta & \Theta \vdash [\Theta]A \equiv [\Theta]B \vdash \Delta \\
\Gamma \vdash (P \triangleright A) \equiv (Q \triangleright B) \vdash \Delta & \equiv \triangleright \\
\]
Under input context $\Gamma$, instantiate $\alpha$ such that $\langle \alpha \rangle = \tau$ with output context $\Delta$

$\Gamma \vdash \langle \alpha \rangle : t \kappa \Delta$

InstSolve

$\Gamma_0 \vdash \tau : \kappa$

InstReach

$\beta \in \text{unsolved}(\Gamma[\langle \alpha \rangle : \kappa][\beta : \kappa])$

InstBin

$\Gamma[\langle \alpha \rangle : \kappa][\beta : \kappa] \vdash \langle \beta \rangle : \kappa \vdash \Gamma[\langle \alpha \rangle : \kappa][\beta : \kappa] : \kappa = \langle \alpha \rangle$

$\Gamma[\langle \beta \rangle : \kappa] \vdash \langle \beta \rangle : \kappa \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstSucc

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$

InstZero

$\Gamma[\langle \alpha \rangle : \kappa] \vdash \langle \alpha \rangle : \star \vdash \Theta \vdash \langle \theta \rangle [\tau_2] : \tau_2 : \star \Delta$
Figure 22: Algorithmic pattern matching
Figures

\[
\frac{\Gamma \vdash \Pi \text{ covers } \vec{A}}{
\Gamma / P \vdash \Pi \text{ covers } \vec{A}} \quad \text{Under context } \Gamma, \text{ patterns } \Pi \text{ cover the types } \vec{A}
\]

\[
\frac{\Gamma \vdash (\cdot \Rightarrow e_1) \Pi \text{ covers } \cdot}{\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } \vec{A}}{\Gamma \vdash \Pi \text{ covers } 1, \vec{A}}} \quad \text{Covers}_1
\]

\[
\frac{\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } A, \vec{A}}{\Gamma \vdash \Pi \text{ covers } (A_1 \times A_2), \vec{A}}} {\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } A_1, A_2, \vec{A}}{\Gamma \vdash \Pi \text{ covers } (A_1 \times A_2), \vec{A}}} \quad \text{Covers}_x
\]

\[
\frac{\Gamma, \alpha : \kappa \vdash \Pi \text{ covers } \vec{A}}{\Gamma \vdash \Pi \text{ covers } (\exists \alpha : \kappa. A), \vec{A}} \quad \text{Covers}_{\exists}
\]

\[
\frac{\Gamma / t = \text{zero} \vdash \Pi \text{ covers } \vec{A}}{\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } A_0, \vec{A}} \quad \text{Covers}_{\land}
\]

\[
\frac{\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } \vec{A}}{\Gamma \vdash \Pi \text{ covers } \vec{A}}} {\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } A_0, A_1, \vec{A}}{\Gamma \vdash \Pi \text{ covers } (A_1 \times A_2), \vec{A}}} \quad \text{Covers}_+ \quad \text{Covers}_{\lor}
\]

\[
\frac{\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } \vec{A}}{\Gamma \vdash \Pi \text{ covers } \vec{A}}} {\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } A, \vec{A}}{\Gamma \vdash \Pi \text{ covers } (A, \text{Vec } n A), \vec{A}}} \quad \text{Covers}_{\text{Vec}}
\]

\[
\frac{\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } \vec{A}}{\Gamma \vdash \Pi \text{ covers } \vec{A}}} {\Pi \rightsquigarrow \Pi' \quad \frac{\Gamma \vdash \Pi' \text{ covers } \vec{A}}{\Gamma \vdash \Pi \text{ covers } \text{Vec } t A, \vec{A}}} \quad \text{Covers}_{\text{Eq}}
\]

\[
\frac{\Gamma / [\Gamma] t_1 \equiv \vdash [\Gamma] t_2 : \kappa \vdash \Delta \quad \Delta \vdash [\Delta] \Pi \text{ covers } [\Delta] \vec{A}} {\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \vec{A}} \quad \text{Covers}_{\text{Eq}}
\]

\[
\frac{\Gamma / [\Gamma] t_1 \equiv \vdash [\Gamma] t_2 : \kappa \equiv \bot} {\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \vec{A}} \quad \text{Covers}_{\text{EqBot}}
\]

Figure 23: Algorithmic match coverage
For convenience, we list all the judgment forms:

<table>
<thead>
<tr>
<th>Judgment</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Psi \vdash t : \kappa$</td>
<td>Index term/monotype is well-formed</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Psi \vdash P \prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Psi \vdash A \type$</td>
<td>Type is well-formed</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Psi \vdash \vec{A} \type$</td>
<td>Type vector is well-formed</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Psi \ctx$</td>
<td>Declarative context is well-formed</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Psi \vdash A \leqCube B$</td>
<td>Declarative subtyping</td>
<td>Figure 3</td>
</tr>
<tr>
<td>$\Psi \vdash P \true$</td>
<td>Declarative truth</td>
<td>Figure 5</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A \ p$</td>
<td>Declarative checking</td>
<td>Figure 5</td>
</tr>
<tr>
<td>$\Psi \vdash e \rightarrow A \ p$</td>
<td>Declarative synthesis</td>
<td>Figure 5</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg C \ q$</td>
<td>Declarative spine typing</td>
<td>Figure 5</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg C \ [q]$</td>
<td>Declarative spine typing, recovering principality</td>
<td>Figure 5</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi :: \vec{A} \leftarrow C \ p$</td>
<td>Declarative pattern matching</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi / P \vdash \Pi :: \vec{A} \leftarrow C \ p$</td>
<td>Declarative proposition assumption</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi \covers \vec{A}$</td>
<td>Declarative match coverage</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Gamma \vdash \tau : \kappa$</td>
<td>Index term/monotype is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Gamma \vdash P \prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Gamma \vdash A \type$</td>
<td>Polytype is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Gamma \ctx$</td>
<td>Algorithmic context is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Gamma \vdash A \App \to T \Delta$</td>
<td>Applying a context, as a substitution, to a type</td>
<td>Figure 10</td>
</tr>
<tr>
<td>$\Gamma \vdash P \true \to T \Delta$</td>
<td>Check proposition</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Gamma / P \vdash \Delta$</td>
<td>Assume proposition</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Gamma \vdash s = t : \kappa \to T \Delta$</td>
<td>Check equation</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$s \not= t$</td>
<td>Head constructors clash</td>
<td>Figure 18</td>
</tr>
<tr>
<td>$\Gamma / s \not= t : \kappa \to T \Delta$</td>
<td>Assume/eliminate equation</td>
<td>Figure 19</td>
</tr>
<tr>
<td>$\Gamma \vdash A \lessOrd B \to T \Delta$</td>
<td>Algorithmic subtyping</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma / P \vdash A \lessOrd B \to T \Delta$</td>
<td>Assume/eliminate proposition</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash P \equiv Q \to T \Delta$</td>
<td>Equivalence of propositions</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash A \equiv B \to T \Delta$</td>
<td>Equivalence of types</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash \vec{a} := t : \kappa \to T \Delta$</td>
<td>Instantiate</td>
<td>Figure 21</td>
</tr>
<tr>
<td>$e \ \chk-I$</td>
<td>Checking intro form</td>
<td>Figure 4</td>
</tr>
<tr>
<td>$\Gamma \vdash e \leftarrow A \ p \to T \Delta$</td>
<td>Algorithmic checking</td>
<td>Figure 12</td>
</tr>
<tr>
<td>$\Gamma \vdash e \rightarrow A \ p \to T \Delta$</td>
<td>Algorithmic synthesis</td>
<td>Figure 12</td>
</tr>
<tr>
<td>$\Gamma \vdash s : A \ p \gg C \ q \to T \Delta$</td>
<td>Algorithmic spine typing</td>
<td>Figure 12</td>
</tr>
<tr>
<td>$\Gamma \vdash s : A \ p \gg C \ [q] \to T \Delta$</td>
<td>Algorithmic spine typing, recovering principality</td>
<td>Figure 12</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi :: \vec{A} \leftarrow C \ p \to T \Delta$</td>
<td>Algorithmic pattern matching</td>
<td>Figure 22</td>
</tr>
<tr>
<td>$\Gamma / P \vdash \Pi :: \vec{A} \leftarrow C \ p \to T \Delta$</td>
<td>Algorithmic pattern matching (assumption)</td>
<td>Figure 22</td>
</tr>
<tr>
<td>$\Gamma \ldots \Rightarrow \to T \Delta$</td>
<td>Algorithmic match coverage</td>
<td>Figure 23</td>
</tr>
<tr>
<td>$\Gamma \Rightarrow \Delta$</td>
<td>Context extension</td>
<td>Figure 13</td>
</tr>
<tr>
<td>$\Gamma \Rightarrow \Delta$</td>
<td>Apply complete context</td>
<td>Figure 11</td>
</tr>
</tbody>
</table>
Properties of the Declarative System

Lemma 1 (Declarative Well-foundedness).\[\text{Go to proof}\]
The inductive definition of the following judgments is well-founded:

(i) synthesis $\Psi \vdash e \Rightarrow B p$

(ii) checking $\Psi \vdash e \Leftarrow A p$

(iii) checking, equality elimination $\Psi / P \vdash e \Leftarrow C p$

(iv) ordinary spine $\Psi \vdash s : A p \gg B q$

(v) recovery spine $\Psi \vdash s : A p \gg B [q]$

(vi) pattern matching $\Psi \vdash \Pi :: \vec{A} \Leftarrow C p$

(vii) pattern matching, equality elimination $\Psi / P : A \Leftarrow C p$

Lemma 2 (Declarative Weakening).\[\text{Go to proof}\]

(i) If $\Psi_0, \Psi_1 \vdash t : \kappa$ then $\Psi_0, \alpha : \kappa, \Psi_1 \vdash t : \kappa$.

(ii) If $\Psi_0, \Psi_1 \vdash P \text{ prop}$ then $\Psi_0, \Psi_1 \vdash P \text{ prop}$.

(iii) If $\Psi_0, \Psi_1 \vdash P \text{ true}$ then $\Psi_0, \Psi_1 \vdash P \text{ true}$.

(iv) If $\Psi_0, \Psi_1 \vdash A \text{ type}$ then $\Psi_0, \Psi_1 \vdash A \text{ type}$.

Lemma 3 (Declarative Term Substitution).\[\text{Go to proof}\]

Suppose $\Psi \vdash t : \kappa$. Then:

1. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash t' : \kappa$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] t' : \kappa$.

2. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P \text{ prop}$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] P \text{ prop}$.

3. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \text{ type}$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] A \text{ type}$.

4. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \leq \pm B$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] A \leq \pm [t/\alpha] B$.

5. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P \text{ true}$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] P \text{ true}$.

Lemma 4 (Reflexivity of Declarative Subtyping). \[\text{Go to proof}\]
Given $\Psi \vdash A \text{ type}$, we have that $\Psi \vdash A \leq \pm A$.

Lemma 5 (Subtyping Inversion). \[\text{Go to proof}\]

- If $\Psi \vdash \exists \alpha : \kappa. A \leq^{+} B$ then $\Psi, \alpha : \kappa \vdash A \leq^{+} B$.
- If $\Psi \vdash A \leq^{-} \forall \beta : \kappa. B$ then $\Psi, \beta : \kappa \vdash A \leq^{-} B$.

Lemma 6 (Subtyping Polarity Flip). \[\text{Go to proof}\]

- If nonpos$(A)$ and nonpos$(B)$ and $\Psi \vdash A \leq^{+} B$ then $\Psi \vdash A \leq^{-} B$ by a derivation of the same or smaller size.
- If nonneg$(A)$ and nonneg$(B)$ and $\Psi \vdash A \leq^{-} B$ then $\Psi \vdash A \leq^{+} B$ by a derivation of the same or smaller size.
- If nonpos$(A)$ and nonneg$(A)$ and nonpos$(B)$ and nonneg$(B)$ and $\Psi \vdash A \leq^{\pm} B$ then $A = B$.
Lemma 7 (Transitivity of Declarative Subtyping). \textit{Go to proof} \\
Given $\Psi \vdash A$ type and $\Psi \vdash B$ type and $\Psi \vdash C$ type:

\begin{itemize}
  \item[(i)] If $D_1 : \Psi \vdash A \leq \mp B$ and $D_2 : \Psi \vdash B \leq \mp C$ then $\Psi \vdash A \leq \mp C$.
\end{itemize}

Property 1. We assume that all types mentioned in annotations in expressions have no free existential variables. By the grammar, it follows that all expressions have no free existential variables, that is, $\text{FEV}(e) = \emptyset$.

\section*{B Substitution and Well-formedness Properties}

Definition 1 (Softness). A context $\Theta$ is soft iff it consists only of $\alpha : \kappa$ and $\alpha : \kappa = \tau$ declarations.

Lemma 8 (Substitution—Well-formedness). \textit{Go to proof} \\
\begin{itemize}
  \item[(i)] If $\Gamma \vdash A \ p$ type and $\Gamma \vdash \tau \ p$ type then $\Gamma \vdash [\tau/\alpha] A \ p$ type.
  \item[(ii)] If $\Gamma \vdash \Psi \ prop$ and $\Gamma \vdash \tau \ p$ type then $\Gamma \vdash [\tau/\alpha] \Psi \ prop$. Moreover, if $\delta p !$ and $\text{FEV}([\Gamma] \Psi) = \emptyset$ then $\text{FEV}([\Gamma] [\tau/\alpha] \Psi) = \emptyset$.
\end{itemize}

Lemma 9 (Uvar Preservation). \textit{Go to proof} \\
If $\Delta \rightarrow \Omega$ then:

\begin{itemize}
  \item[(i)] If $\alpha : \kappa \in \Omega$ then $\alpha : \kappa \in [\Omega] \Delta$.
  \item[(ii)] If $\kappa \in \Omega \ p \in \Omega$ then $(\kappa : [\Omega] \ A \ p) \in [\Omega] \Delta$.
\end{itemize}

Lemma 10 (Sorting Implies Typing). \textit{Go to proof} \\
If $\Gamma \vdash t : \kappa$ then $\Gamma \vdash t$ type.

Lemma 11 (Right-Hand Substitution for Sorting). \textit{Go to proof} \\
If $\Gamma \vdash t : \kappa$ then $\Gamma \vdash [\Gamma] t : \kappa$.

Lemma 12 (Right-Hand Substitution for Propositions). \textit{Go to proof} \\
If $\Gamma \vdash P \ prop$ then $\Gamma \vdash [\Gamma] P \ prop$.

Lemma 13 (Right-Hand Substitution for Typing). \textit{Go to proof} \\
If $\Gamma \vdash A$ type then $\Gamma \vdash [\Gamma] A$ type.

Lemma 14 (Substitution for Sorting). \textit{Go to proof} \\
If $\Omega \vdash t : \kappa$ then $[\Omega] \Omega \vdash [\Omega] t : \kappa$.

Lemma 15 (Substitution for Prop Well-Formedness). \textit{Go to proof} \\
If $\Omega \vdash P \ prop$ then $[\Omega] \Omega \vdash [\Omega] P \ prop$.

Lemma 16 (Substitution for Type Well-Formedness). \textit{Go to proof} \\
If $\Omega \vdash A$ type then $[\Omega] \Omega \vdash [\Omega] A$ type.

Lemma 17 (Substitution Stability). \textit{Go to proof} \\
If $(\Omega, \Omega_Z)$ is well-formed and $\Omega_Z$ is soft and $\Omega \vdash A$ type then $[\Omega] A = [\Omega, \Omega_Z] A$.

Lemma 18 (Equal Domains). \textit{Go to proof} \\
If $\Omega_1 \vdash A$ type and $\text{dom} (\Omega_1) = \text{dom} (\Omega_2)$ then $\Omega_2 \vdash A$ type.

\section*{C Properties of Extension}

Lemma 19 (Declaration Preservation). \textit{Go to proof} \\
If $\Gamma \rightarrow \Delta$ and $u$ is declared in $\Gamma$, then $u$ is declared in $\Delta$.

Lemma 20 (Declaration Order Preservation). \textit{Go to proof} \\
If $\Gamma \rightarrow \Delta$ and $u$ is declared to the left of $v$ in $\Gamma$, then $u$ is declared to the left of $v$ in $\Delta$.

Lemma 21 (Reverse Declaration Order Preservation). \textit{Go to proof} \\
If $\Gamma \rightarrow \Delta$ and $u$ and $v$ are both declared in $\Gamma$ and $u$ is declared to the left of $v$ in $\Delta$, then $u$ is declared to the left of $v$ in $\Gamma$.

An older paper had a lemma
“Substitution Extension Invariance”
If $\Theta \vdash A$ type and $\Theta \rightarrow \Gamma$ then $[\Gamma]A = [\Gamma]([\Theta]A)$ and $[\Gamma]A = [\Theta]([\Gamma]A)$.

For the second part, $[\Gamma]A = [\Theta]([\Gamma]A)$, use Lemma 29 (Substitution Monotonicity) (i) or (iii) instead. The first part $[\Gamma]A = [\Gamma]([\Theta]A)$ hasn’t been proved in this system.

Lemma 22 (Extension Inversion). Go to proof

(i) If $D :: \Gamma_0, \alpha : \kappa, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0$ and $\Delta_1$
such that $\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ where $D' < D$.
Moreover, if $\Gamma_1$ is soft, then $\Delta_1$ is soft.

(ii) If $D :: \Gamma_0, \mu, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0$ and $\Delta_1$
such that $\Delta = (\Delta_0, \mu, \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ where $D' < D$.
Moreover, if $\Gamma_1$ is soft, then $\Delta_1$ is soft.
Moreover, if $\text{dom}(\Gamma_0, \mu, \Gamma_1) = \text{dom}(\Delta)$ then $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$.

(iii) If $D :: \Gamma_0, \alpha = \tau, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0, \tau'$, and $\Delta_1$
such that $\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]\tau = [\Delta_0]\tau'$ where $D' < D$.

(iv) If $D :: \Gamma_0, \beta : \kappa = \tau, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0, \tau'$, and $\Delta_1$
such that $\Delta = (\Delta_0, \beta : \kappa = \tau', \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]\tau = [\Delta_0]\tau'$ where $D' < D$.

(v) If $D :: \Gamma_0, x : A, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0, A'$, and $\Delta_1$
such that $\Delta = (\Delta_0, x : A', \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]A = [\Delta_0]A'$ where $D' < D$.
Moreover, if $\Gamma_1$ is soft, then $\Delta_1$ is soft.
Moreover, if $\text{dom}(\Gamma_0, x : A, \Gamma_1) = \text{dom}(\Delta)$ then $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$.

(vi) If $D :: \Gamma_0, \beta : \kappa, \Gamma_1 \rightarrow \Delta$ then either

- there exist unique $\Delta_0, \tau'$, and $\Delta_1$
such that $\Delta = (\Delta_0, \beta : \kappa = \tau', \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ where $D' < D$,
or
- there exist unique $\Delta_0$ and $\Delta_1$
such that $\Delta = (\Delta_0, \beta : \kappa, \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ where $D' < D$.

Lemma 23 (Deep Evar Introduction). Go to proof

(i) If $\Gamma_0, \Gamma_1$ is well-formed and $\alpha$ is not declared in $\Gamma_0, \Gamma_1$ then $\Gamma_0, \Gamma_1 \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1$.

(ii) If $\Gamma_0, \kappa, \Gamma_1$ is well-formed and $\Gamma \vdash t : \kappa$ then $\Gamma_0, \kappa, \Gamma_1 \rightarrow \Gamma_0, \kappa : \kappa \vdash t, \Gamma_1$.

(iii) If $\Gamma_0, \Gamma_1$ is well-formed and $\Gamma \vdash t : \kappa$ then $\Gamma_0, \Gamma_1 \rightarrow \Gamma_0, \kappa : \kappa \vdash t, \Gamma_1$.

Lemma 24 (Soft Extension). Go to proof
If $\Gamma \rightarrow \Delta$ and $\Gamma, \Theta \text{ ctx}$ and $\Theta$ is soft, then there exists $\Omega$ such that $\text{dom}(\Theta) = \text{dom}(\Omega)$ and $\Gamma, \Theta \rightarrow \Delta, \Omega$.

Definition 2 (Filling). The filling of a context $[\Gamma]$ solves all unsolved variables:
Lemma 25 (Filling Completes). If $\Gamma \rightarrow \Omega$ and $(\Gamma, \Theta)$ is well-formed, then $\Gamma, \Theta \rightarrow \Omega, \Theta$.

Proof. By induction on $\Theta$, following the definition of $\vdash$ and applying the rules for $\rightarrow$.

Lemma 26 (Parallel Admissibility). If $\Gamma_L \rightarrow \Delta_L$ and $\Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta_R$ then:

(i) $\Gamma_L, \xi : \kappa, \Gamma_R \rightarrow \Delta_L, \xi : \kappa, \Delta_R$

(ii) If $\Delta_L \vdash \tau : \kappa$ then $\Gamma_L, \xi : \kappa, \Gamma_R \rightarrow \Delta_L, \xi : \kappa = \tau', \Delta_R$.

(iii) If $\Gamma_L \vdash \tau : \kappa$ and $\Delta_L \vdash \tau' : \kappa$ type and $|\Delta_L|\tau = |\Delta_L|\tau'$, then $\Gamma_L, \xi : \kappa, \Gamma_R \rightarrow \Delta_L, \xi : \kappa = \tau', \Delta_R$.

Lemma 27 (Parallel Extension Solution). If $\Gamma_L, \xi : \kappa, \Gamma_R \rightarrow \Delta_L, \xi : \kappa = \tau_0, \Delta_R$ and $\Gamma_L \vdash \tau_1 : \kappa$ and $\Delta_L \vdash \tau_0 : \kappa = \tau_1$ then $\Gamma_L, \xi : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \xi : \kappa = \tau_2, \Delta_R$.

Lemma 28 (Parallel Variable Update). If $\Gamma_L, \xi : \kappa, \Gamma_R \rightarrow \Delta_L, \xi : \kappa = \tau_0, \Delta_R$ and $\Gamma_L \vdash \tau_1 : \kappa$ and $\Delta_L \vdash \tau_2 : \kappa$ and $|\Delta_L|\tau_0 = |\Delta_L|\tau_1 = |\Delta_L|\tau_2$ then $\Gamma_L, \xi : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \xi : \kappa = \tau_2, \Delta_R$.

Lemma 29 (Substitution Monotonicity). If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash t : \kappa$ then $|\Delta|\Gamma t = |\Delta|t$.

(ii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash P$ prop then $|\Delta|\Gamma P = |\Delta|P$.

(iii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash A$ type then $|\Delta|\Gamma A = |\Delta|A$.

Lemma 30 (Substitution Invariance). If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash t : \kappa$ and $\text{FEV}(|\Gamma|t) = \emptyset$ then $|\Delta|\Gamma t = |\Gamma|t$.

(ii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash P$ prop and $\text{FEV}(|\Gamma|P) = \emptyset$ then $|\Delta|\Gamma P = |\Gamma|P$.

(iii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash A$ type and $\text{FEV}(|\Gamma|A) = \emptyset$ then $|\Delta|\Gamma A = |\Gamma|A$.

Definition 3 (Canonical Contexts). A (complete) context $\Omega$ is canonical iff, for all $(\xi : \kappa = t)$ and $(\alpha = t) \in \Omega$, the solution $t$ is ground ($\text{FEV}(t) = \emptyset$).

Lemma 31 (Split Extension). If $\Delta \rightarrow \Omega$ and $\xi \in \text{unsolved}(\Delta)$ and $\Delta = \Omega_1[\xi : \kappa = t_1]$ and $\Omega$ is canonical (Definition 3) and $\Omega \vdash t_2 : \kappa$ then $\Delta \rightarrow \Omega_1[\xi : \kappa = t_2]$.
C.1 Reflexivity and Transitivity

Lemma 32 (Extension Reflexivity). Go to proof
If \( \Gamma \text{ ctx} \) then \( \Gamma \rightarrow \Gamma \).

Lemma 33 (Extension Transitivity). Go to proof
If \( D :: \Gamma \rightarrow \Theta \) and \( D' :: \Theta \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).

C.2 Weakening

The “suffix weakening” lemmas take a judgment under \( \Gamma \) and produce a judgment under \( (\Gamma, \Theta) \). They do not require \( \Gamma \rightarrow \Gamma, \Theta \).

Lemma 34 (Suffix Weakening). Go to proof
If \( \Gamma \vdash t : \kappa \) then \( \Gamma, \Theta \vdash t : \kappa \).

Lemma 35 (Suffix Weakening). Go to proof
If \( \Gamma \vdash A \) type then \( \Gamma, \Theta \vdash A \) type.

The following proposed lemma is false.

“Extension Weakening (Truth)”

If \( \Gamma \vdash P \text{ true} \) \( \rightarrow \Delta \) and \( \Gamma \rightarrow \Gamma' \) then there exists \( \Delta' \) such that \( \Delta \rightarrow \Delta' \) and \( \Gamma' \vdash P \text{ true} \rightarrow \Delta' \).

Counterexample: Suppose \( \check{\alpha} \vdash \check{\alpha} = 1 \text{ true} \rightarrow \check{\alpha} = 1\) and \( \check{\alpha} \rightarrow (\check{\alpha} = (1 \rightarrow 1)) \). Then there does not exist such a \( \Delta' \).

Lemma 36 (Extension Weakening (Sorts)). Go to proof
If \( \Gamma \vdash t : \kappa \) and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash t : \kappa \).

Lemma 37 (Extension Weakening (Props)). Go to proof
If \( \Gamma \vdash P \) prop and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash P \) prop.

Lemma 38 (Extension Weakening (Types)). Go to proof
If \( \Gamma \vdash A \) type and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash A \) type.

C.3 Principal Typing Properties

Lemma 39 (Principal Agreement). Go to proof
(i) If \( \Gamma \vdash A \) ! type and \( \Gamma \rightarrow \Delta \) then \( \Delta|A = [\Gamma|A] \).

(ii) If \( \Gamma \vdash P \) prop and \( \text{FEV}(P) = \emptyset \) and \( \Gamma \rightarrow \Delta \) then \( \Delta|P = [\Gamma|P] \).

Lemma 40 (Right-Hand Subst. for Principal Typing). Go to proof
If \( \Gamma \vdash A \) p type then \( \Gamma \vdash [\Gamma|A \) p type.

Lemma 41 (Extension Weakening for Principal Typing). Go to proof
If \( \Gamma \vdash A \) p type and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash A \) p type.

Lemma 42 (Inversion of Principal Typing). Go to proof
(1) If \( \Gamma \vdash (A \rightarrow B) \) p type then \( \Gamma \vdash A \) p type and \( \Gamma \vdash B \) p type.

(2) If \( \Gamma \vdash (P \supset A) \) p type then \( \Gamma \vdash P \) prop and \( \Gamma \vdash A \) p type.

(3) If \( \Gamma \vdash (A \land P) \) p type then \( \Gamma \vdash P \) prop and \( \Gamma \vdash A \) p type.

C.4 Instantiation Extends

Lemma 43 (Instantiation Extension). Go to proof
If \( \Gamma \vdash \check{\alpha} := \tau : \kappa \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).
C.5 Equivalence Extends

Lemma 44 (Elimeq Extension). \[\text{Go to proof}\]
If \(\Gamma / s \Rightarrow t : \kappa \vdash \Delta\) then there exists \(\Theta\) such that \(\Gamma, \Theta \vdash \Delta\).

Lemma 45 (Elimprop Extension). \[\text{Go to proof}\]
If \(\Gamma / P \vdash \Delta\) then there exists \(\Theta\) such that \(\Gamma, \Theta \vdash \Delta\).

Lemma 46 (Checkeq Extension). \[\text{Go to proof}\]
If \(\Gamma \vdash A \equiv B \vdash \Delta\) then \(\Gamma \vdash \Delta\).

Lemma 47 (Checkprop Extension). \[\text{Go to proof}\]
If \(\Gamma \vdash \text{true} \vdash \Delta\) then \(\Gamma \vdash \Delta\).

Lemma 48 (Prop Equivalence Extension). \[\text{Go to proof}\]
If \(\Gamma \vdash P \equiv Q \vdash \Delta\) then \(\Gamma \vdash \Delta\).

Lemma 49 (Equivalence Extension). \[\text{Go to proof}\]
If \(\Gamma \vdash A \equiv B \vdash \Delta\) then \(\Gamma \vdash \Delta\).

C.6 Subtyping Extends

Lemma 50 (Subtyping Extension). \[\text{Go to proof}\]
If \(\Gamma \vdash A <: \mathcal{T} \vdash \Delta\) then \(\Gamma \vdash \Delta\).

C.7 Typing Extends

Lemma 51 (Typing Extension). \[\text{Go to proof}\]
If \(\Gamma \vdash e \equiv A \vdash \Delta\)
or \(\Gamma \vdash e \Rightarrow A \vdash \Delta\)
or \(\Gamma \vdash s : A \vdash B \vdash q \vdash \Delta\)
or \(\Gamma \vdash \Pi : \overline{A} \equiv C \vdash p \vdash \Delta\)
or \(\Gamma / P \vdash \Pi : \overline{A} \equiv C \vdash p \vdash \Delta\)\then \(\Gamma \vdash \Delta\).

C.8 Unfiled

Lemma 52 (Context Partitioning). \[\text{Go to proof}\]
If \(\Delta, \Theta \vdash \Omega, \Omega, \Omega, \Omega Z\) then there is a \(\Psi\) such that \([\Omega, \Theta \vdash \Delta, \Theta \vdash \Omega, \Omega, \Omega Z] = [\Omega] \Delta, \Psi\).

Lemma 53 (Softness Goes Away).
If \(\Delta, \Theta \vdash \Omega, \Omega Z\) where \(\Delta \rightarrow \Omega\) and \(\Theta\) is soft, then \([\Omega, \Omega Z] \vdash \Delta, \Theta = [\Omega] \Delta\).

Proof. By induction on \(\Theta\), following the definition of \([\Omega] \Gamma\). \[\square\]

Lemma 54 (Completing Stability). \[\text{Go to proof}\]
If \(\Gamma \vdash \Omega\) then \([\Omega] \Gamma = [\Omega] \Omega\).

Lemma 55 (Completing Completeness). \[\text{Go to proof}\]
(i) If \(\Omega \vdash \Omega'\) and \(\Omega \vdash t : \kappa\) then \([\Omega] t = [\Omega'] t\).
(ii) If \(\Omega \vdash \Omega'\) and \(\Omega \vdash A\) type then \([\Omega] A = [\Omega'] A\).
(iii) If \(\Omega \vdash \Omega'\) then \([\Omega] \Omega = [\Omega'] \Omega'\).

Lemma 56 (Confluence of Completeness). \[\text{Go to proof}\]
If \(\Delta_1 \rightarrow \Omega\) and \(\Delta_2 \rightarrow \Omega\) then \([\Omega] \Delta_1 = [\Omega] \Delta_2\).

Lemma 57 (Multiple Confluence). \[\text{Go to proof}\]
If \(\Delta \rightarrow \Omega\) and \(\Omega \rightarrow \Omega'\) and \(\Delta' \rightarrow \Omega'\) then \([\Omega] \Delta = [\Omega'] \Delta'\).

March 16, 2018
Lemma 58 (Bundled Substitution for Sorting). If $\Gamma \vdash t : \kappa$ and $\Gamma \rightarrow \Omega$ then $[\Omega]\Gamma \vdash [\Omega]t : \kappa$.

Proof. 
\[
\begin{align*}
\Gamma \vdash t : \kappa & \quad \text{Given} \\
[\Omega]\Omega \vdash [\Omega]t : \kappa & \quad \text{By Lemma 14 (Substitution for Sorting)} \\
\Omega \rightarrow \Omega & \quad \text{By Lemma 32 (Extension Reflexivity)} \\
[\Omega]\Omega = [\Omega]\Gamma & \quad \text{By Lemma 56 (Confluence of Completeness)} \\
\end{align*}
\]
\[\square\]

Lemma 59 (Canonical Completion). Go to proof

If $\Gamma \rightarrow \Omega$ then there exists $\Omega_{\text{canon}}$ such that $\Gamma \rightarrow \Omega_{\text{canon}}$ and $\Omega_{\text{canon}} \rightarrow \Omega$ and $\text{dom}(\Omega_{\text{canon}}) = \text{dom}(\Gamma)$ and, for all $\tilde{\alpha} : \kappa = \tau$ and $\alpha = \tau$ in $\Omega_{\text{canon}}$, we have $\text{FEV}(\tau) = \emptyset$.

The completion $\Omega_{\text{canon}}$ is “canonical” because (1) its domain exactly matches $\Gamma$ and (2) its solutions $\tau$ have no evars. Note that it follows from Lemma 57 (Multiple Confluence) that $[\Omega_{\text{canon}}]\Gamma = [\Omega]\Gamma$.

Lemma 60 (Split Solutions). Go to proof

If $\Delta \rightarrow \Omega$ and $^\alpha \in \text{unsolved}(\Delta)$ then there exists $\Omega_1 = \Omega_{\tau}^\alpha$ such that $\Omega_1 \rightarrow \Omega$ and $\Omega_2 = \Omega_{\tilde{\alpha}}$ where $\Delta \rightarrow \Omega_2$ and $t_2 \neq t_1$ and $\Omega_1$ is canonical.

D Internal Properties of the Declarative System

Lemma 61 (Interpolating With and Exists). Go to proof

(1) If $D :: \Psi \vdash \Pi :: \tilde{A} \Leftarrow C \land P_0 \true$ then $D' :: \Psi \vdash \Pi :: \tilde{A} \Leftarrow C \land P_0 \true$.

(2) If $D :: \Psi \vdash \Pi :: \tilde{A} \Leftarrow [\tau/\alpha]C_0 \land P$ and $\Psi \vdash \tau : \kappa$ then $D' :: \Psi \vdash \Pi :: \tilde{A} \Leftarrow (\exists \alpha : \kappa. C_0) \land P$.

In both cases, the height of $D'$ is one greater than the height of $D$.

Moreover, similar properties hold for the eliminating judgment $\Psi / P \vdash \Pi :: \tilde{A} \Leftarrow C \land P$.

Lemma 62 (Case Invertibility). Go to proof

If $\Psi \vdash \text{case}(e_0, \Pi) \Leftarrow C \land P$ then $\Psi \vdash e_0 \Rightarrow A \land P$ and $\Psi \vdash \Pi :: A \Leftarrow C \land P$ and $\Psi \vdash \Pi$ covers $A$ where the height of each resulting derivation is strictly less than the height of the given derivation.

E Miscellaneous Properties of the Algorithmic System

Lemma 63 (Well-Formed Outputs of Typing). Go to proof

(Spines) If $\Gamma \vdash s : A \land P \rightarrow \Delta$ or $\Gamma \vdash s : A \land P \rightarrow \Delta$ and $\Gamma \vdash A \land P \rightarrow \Delta$ then $\Delta \vdash C \land P \rightarrow \Delta$.

(Synthesis) If $\Gamma \vdash e \Rightarrow A \land P \rightarrow \Delta$ then $A \vdash P \rightarrow \Delta$. 
\section{Decidability of Instantiation}

\begin{lemma}[Left Unsolvedness Preservation]\label{lem:left-unresolved}
\[
\text{If } \Gamma, \alpha, \Gamma_1 \vdash \alpha := \Delta : \kappa \vdash \Delta \text{ and } \beta \in \text{unsolved}(\Gamma_0) \text{ then } \beta \in \text{unsolved}(\Delta).
\]
\end{lemma}

\begin{lemma}[Left Free Variable Preservation]\label{lem:left-free-variable}
\[
\text{If } \Gamma, \alpha, \kappa, \Gamma_1 \vdash \alpha := t : \kappa \vdash \Delta \text{ and } \alpha \notin \text{FV}(\Gamma) \text{ and } \beta \in \text{unsolved}(\Gamma_0) \text{ and } \beta \notin \text{FV}(\Delta), \text{ then } \beta \notin \text{FV}(\Delta).
\]
\end{lemma}

\begin{lemma}[Instantiation Size Preservation]\label{lem:instantiation-size-preservation}
\[
\text{If } \Gamma, \alpha, \Gamma_1 \vdash \alpha := t : \kappa \vdash \Delta \text{ and } \Gamma \vdash s : \kappa' \text{ and } \alpha \notin \text{FV}(\Gamma) \text{ then } |\Gamma| = |\Delta|, \text{ where } |C| \text{ is the plain size of the term } C.
\]
\end{lemma}

\begin{lemma}[Decidability of Instantiation]\label{lem:decidability-of-instantiation}
\[
\text{If } \Gamma = \Gamma_0[\alpha : \kappa'] \text{ and } \Gamma \vdash t : \kappa \text{ such that } |\Gamma|t = t \text{ and } \alpha \notin \text{FV}(t), \text{ then:}
\]
\begin{enumerate}
\item Either there exists \( \Delta \) such that \( \Gamma_0[\alpha'] = t \vdash \kappa \vdash \Delta \), or not.
\end{enumerate}
\end{lemma}

\section{Separation}

\begin{definition}[Separation]\label{def:separation}
An algorithmic context \( \Gamma \) is separable and written \( \Gamma_L \ast \Gamma_R \) if (1) \( \Gamma = (\Gamma_L, \Gamma_R) \) and (2) for all \( (\alpha : \kappa = \tau) \in \Gamma_R \) it is the case that \( \text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R) \).
\end{definition}

Any context \( \Gamma \) is separable into, at least, \( \cdot \ast \Gamma \) and \( \Gamma \ast \cdot \).

\begin{definition}[Separation-Preserving Extension]\label{def:separation-preserving-extension}
The separated context \( \Gamma_L \ast \Gamma_R \) extends to \( \Delta_L \ast \Gamma_R \), written
\[
(\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R)
\]
if \( \Gamma_L \ast \Gamma_R \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \text{ and dom}(\Gamma_L) \subseteq \text{dom}(\Delta_L) \text{ and dom}(\Gamma_R) \subseteq \text{dom}(\Delta_R) \).

Separation-preserving extension says that variables from one half don’t “cross” into the other half. Thus, \( \Delta_L \) may add existential variables to \( \Gamma_L \), and \( \Delta_R \) may add existential variables to \( \Gamma_R \), but no variable from \( \Gamma_L \) ends up in \( \Delta_R \) and no variable from \( \Gamma_R \) ends up in \( \Delta_L \).

It is necessary to write \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \) rather than \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \), because only \( \xrightarrow{\ast} \) includes the domain conditions. For example, \( (\lambda \ast \beta) \xrightarrow{\ast} (\lambda \ast \beta) \ast \cdot \), but the variable \( \beta \) has “crossed over” to the left of \( \ast \) in the context \( (\lambda, \beta, \lambda = \lambda) \ast \cdot \).

\begin{lemma}[Transitivity of Separation]\label{lem:transitivity-of-separation}
If \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Theta_L \ast \Theta_R) \) and \( (\Theta_L \ast \Theta_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \) then \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \).
\end{lemma}

\begin{lemma}[Separation Truncation]\label{lem:separation-truncation}
If \( H \) has the form \( \alpha : \kappa \text{ or } \ast \alpha \text{ or } \ast \alpha : \Lambda \text{ or } P \text{ true } \ast \Delta \text{ and } \Gamma \ast (\Gamma_L, H) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \) then \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \) where \( \Delta_R = (\Delta_0, H, \Theta) \).
\end{lemma}

\begin{lemma}[Separation for Auxiliary Judgments]\label{lem:separation-for-auxiliary-judgments}
\begin{enumerate}
\item If \( \Gamma_L \ast \Gamma_R \vdash \sigma \equiv \tau : \kappa \vdash \Delta \) \text{ and } \text{FEV}(\sigma) \subseteq \text{dom}(\Gamma_R) \) then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \).
\item If \( \Gamma_L \ast \Gamma_R \vdash P \text{ true } \ast \Delta \) \text{ and } \text{FEV}(\Gamma) \subseteq \text{dom}(\Gamma_R) \) then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\ast} (\Delta_L \ast \Delta_R) \).
\end{enumerate}
\end{lemma}
(iii) If $\Gamma_L \vdash \gamma \downarrow \Delta$
and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset$
then $\Delta = (\Delta_L \ast (\Delta_R, \Theta))$ and $(\Gamma_L \ast (\Gamma_R, \Theta)) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(iv) If $\Gamma_L \ast \Gamma_R / P \vdash \Delta$
and $\text{FEV}(P) = \emptyset$
then $\Delta = (\Delta_L \ast (\Delta_R, \Theta))$ and $(\Gamma_L \ast (\Gamma_R, \Theta)) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(v) If $\Gamma_L \ast \Gamma_R \vdash \& := \pi : \kappa \vdash \Delta$
and $(\text{FEV}(\tau) \cup \{\&\}) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(vi) If $\Gamma_L \ast \Gamma_R \vdash P \equiv Q \vdash \Delta$
and $\text{FEV}(P) \cup \text{FEV}(Q) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(vii) If $\Gamma_L \ast \Gamma_R \vdash \lambda \equiv \beta : B \vdash \Delta$
and $\text{FEV}(\lambda) \cup \text{FEV}(\beta) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

Lemma 71 (Separation for Subtyping). Go to proof
If $\Gamma_L \ast \Gamma_R \vdash \lambda \equiv \beta : B \vdash \Delta$
and $\text{FEV}(\lambda) \subseteq \text{dom}(\Gamma_R)$
and $\text{FEV}(\beta) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

Lemma 72 (Separation—Main). Go to proof
(Spines) If $\Gamma_L \ast \Gamma_R \vdash s : A \rightarrow C \vdash \Delta$
or $\Gamma_L \ast \Gamma_R \vdash s : A \rightarrow C \lfloor q \rfloor \vdash \Delta$
and $\Gamma_L \ast \Gamma_R \vdash A : \text{type}$
and $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$ and $\text{FEV}(C) \subseteq \text{dom}(\Delta_R)$.

(Checking) If $\Gamma_L \ast \Gamma_R \vdash e \equiv C \downarrow \Delta$
and $\Gamma_L \ast \Gamma_R \vdash C : \text{type}$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(Synthesis) If $\Gamma_L \ast \Gamma_R \vdash e \Rightarrow A \vdash \Delta$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(Match) If $\Gamma_L \ast \Gamma_R \vdash \Pi :: \overline{x} \leftarrow C \downarrow \Delta$
and $\text{FEV}(\overline{x}) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

(Match Elim.) If $\Gamma_L \ast \Gamma_R / P \vdash \Pi :: \overline{x} \leftarrow C \downarrow \Delta$
and $\text{FEV}(P) = \emptyset$
and $\text{FEV}(\overline{x}) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \overrightarrow{=} (\Delta_L \ast \Delta_R)$.

H Decidability of Algorithmic Subtyping

**Definition 6.** The following connectives are large:

\[ \forall \quad \exists \quad \wedge \]
A type is large iff its head connective is large. (Note that a non-large type may contain large connectives, provided they are not in head position.)

The number of these connectives in a type $\Lambda$ is denoted by $\#\text{large}(\Lambda)$.

### H.1 Lemmas for Decidability of Subtyping

**Lemma 73** (Substitution Isn’t Large). If for all contexts $\Theta$, we have $\#\text{large}(\Theta|\Lambda) = \#\text{large}(\Lambda)$.

**Lemma 74** (Instantiation Solves). If $\Gamma \vdash \Delta : \tau \vdash \Delta$ and $|\Gamma|\tau = \tau$ and $\Delta \notin \text{FV}(|\Gamma|\tau)$ then $|\text{unsolved}(\Gamma)| = |\text{unsolved}(\Delta)| + 1$.

**Lemma 75** (Checkeq Solving). If $\Gamma \vdash s = t : \kappa \vdash \Delta$ then either $\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

**Lemma 76** (Prop Equiv Solving). If $\Gamma \vdash P \equiv Q \vdash \Delta$ then either $\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

**Lemma 77** (Equiv Solving). If $\Gamma \vdash A \equiv B \vdash \Delta$ then either $\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

**Lemma 78** (Decidability of Propositional Judgments). The following judgments are decidable, with $\Delta$ as output in (1)–(3), and $\Delta_\perp$ as output in (4) and (5).

- We assume $\sigma = [\Gamma]\sigma$ and $t = [\Gamma]\Gamma \vdash t$ in (1) and (4). Similarly, in the other parts we assume $P = [\Gamma]P$ and (in part (3)) $Q = [\Gamma]Q$.
  1. $\Gamma \vdash \sigma \triangleq t : \kappa \vdash \Delta$
  2. $\Gamma \vdash P \text{ true} \vdash \Delta$
  3. $\Gamma \vdash P \equiv Q \vdash \Delta$
  4. $\Gamma / \sigma \triangleq t : \kappa \vdash \Delta_\perp$
  5. $\Gamma / \sigma \equiv t \vdash \kappa \vdash \Delta_\perp$

**Lemma 79** (Decidability of Equivalence). Given a context $\Gamma$ and types $A$, $B$ such that $\Gamma \vdash A$ type and $\Gamma \vdash B$ type and $|\Gamma|A = A$ and $|\Gamma|B = B$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash A \equiv B \vdash \Delta$.

### H.2 Decidability of Subtyping

**Theorem 1** (Decidability of Subtyping). Given a context $\Gamma$ and types $A$, $B$ such that $\Gamma \vdash A$ type and $\Gamma \vdash B$ type and $|\Gamma|A = A$ and $|\Gamma|B = B$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash A \vdash \Delta$.

### H.3 Decidability of Matching and Coverage

**Lemma 80** (Decidability of Expansion Judgments). Given branches $\Pi$, it is decidable whether:

1. there exists $\Pi'$ such that $\Pi \equiv \Pi'$;
2. there exist $\Pi_L$ and $\Pi_R$ such that $\Pi \equiv \Pi_L \parallel \Pi_R$;
3. there exists $\Pi'$ such that $\Pi \vartriangleright \Pi'$;
4. there exists $\Pi'$ such that $\Pi \perp \Pi'$.

**Theorem 2** (Decidability of Coverage). Given a context $\Gamma$, branches $\Pi$ and types $A$, it is decidable whether $\Gamma \vdash \Pi$ covers $\bar{A}$ is derivable.
H.4 Decidability of Typing

Theorem 3 (Decidability of Typing).

(i) Synthesis: Given a context $\Gamma$, a principality $p$, and a term $e$, it is decidable whether there exist a type $A$ and a context $\Delta$ such that $\Gamma \vdash e \Rightarrow A \vdash \Delta$.

(ii) Spines: Given a context $\Gamma$, a spine $s$, a principality $p$, and a type $A$ such that $\Gamma \vdash A$ type, it is decidable whether there exist a type $B$, a principality $q$ and a context $\Delta$ such that $\Gamma \vdash s : A \vdash B \vdash q \vdash \Delta$.

(iii) Checking: Given a context $\Gamma$, a principality $p$, a term $e$, and a type $B$ such that $\Gamma \vdash B$ type, it is decidable whether there is a context $\Delta$ such that $\Gamma \vdash e \Leftrightarrow B \vdash \Delta$.

(iv) Matching: Given a context $\Gamma$, branches $\Pi$, a list of types $\vec{A}$, a type $C$, and a principality $p$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash \Pi :: \vec{A} \Leftrightarrow C \vdash \Delta$.

Also, if given a proposition $P$ as well, it is decidable whether there exists $\Delta$ such that $\Gamma / P \vdash \Pi :: \vec{A} \Leftrightarrow C \vdash \Delta$.

I Determinacy

Lemma 81 (Determinacy of Auxiliary Judgments).

(1) Elimeq: Given $\Gamma, \sigma, t, \kappa$ such that $\text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset$ and $D_1 :: \Gamma / \sigma \vdash t : \kappa \vdash \Delta_1^\downarrow$ and $D_2 :: \Gamma / \sigma \vdash t : \kappa \vdash \Delta_2^\downarrow$,

it is the case that $\Delta_1^\downarrow = \Delta_2^\downarrow$.

(2) Instantiation: Given $\Gamma, \Hat{\alpha}, t, \kappa$ such that $\Hat{\alpha} \in \text{unsolved}(\Gamma)$ and $\Gamma \vdash t : \kappa$ and $\Hat{\alpha} \notin \text{FV}(t)$ and $D_1 :: \Gamma \vdash \Hat{\alpha} := t : \kappa \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \Hat{\alpha} := t : \kappa \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

(3) Symmetric instantiation:

Given $\Gamma, \Hat{\alpha}, \Hat{\beta}, \kappa$ such that $\Hat{\alpha}, \Hat{\beta} \in \text{unsolved}(\Gamma)$ and $\Hat{\alpha} \neq \Hat{\beta}$ and $D_1 :: \Gamma \vdash \Hat{\alpha} := \Hat{\beta} : \kappa \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \Hat{\beta} := \Hat{\alpha} : \kappa \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

(4) Checkeq: Given $\Gamma, \sigma, t, \kappa$ such that $D_1 :: \Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

(5) Elimprop: Given $\Gamma, P$ such that $D_1 :: \Gamma / P \vdash \Delta_1^\downarrow$ and $D_2 :: \Gamma / P \vdash \Delta_2^\downarrow$,

it is the case that $\Delta_1 = \Delta_2$.

(6) Checkprop: Given $\Gamma, P$ such that $D_1 :: \Gamma \vdash P \text{ true } \vdash \Delta_1$ and $D_2 :: \Gamma \vdash P \text{ true } \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

Lemma 82 (Determinacy of Equivalence).

(1) Propositional equivalence: Given $\Gamma, P, Q$ such that $D_1 :: \Gamma \vdash P \equiv Q \vdash \Delta_1$ and $D_2 :: \Gamma \vdash P \equiv Q \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

(2) Type equivalence: Given $\Gamma, A, B$ such that $D_1 :: \Gamma \vdash A \equiv B \vdash \Delta_1$ and $D_2 :: \Gamma \vdash A \equiv B \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

Theorem 4 (Determinacy of Subtyping).

(i) Type equivalence: Given $\Gamma, A, B$ such that $D_1 :: \Gamma \vdash A \equiv B \vdash \Delta_1$ and $D_2 :: \Gamma \vdash A \equiv B \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$. 

(ii) Propositional equivalence: Given $\Gamma, P, Q$ such that $D_1 :: \Gamma \vdash P \equiv Q \vdash \Delta_1$ and $D_2 :: \Gamma \vdash P \equiv Q \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$. 

(iii) Checking: Given a context $\Gamma$, a principality $p$, a term $e$, and a type $B$ such that $\Gamma \vdash B$ type, it is decidable whether there is a context $\Delta$ such that $\Gamma \vdash e \Leftrightarrow B \vdash \Delta$.

Also, if given a proposition $P$ as well, it is decidable whether there exists $\Delta$ such that $\Gamma / P \vdash \Pi :: \vec{A} \Leftrightarrow C \vdash \Delta$.
(1) Subtyping: Given \( \Gamma, e, A, B \) such that \( D_1 :: \Gamma \vdash A <:= B \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash B <:= A \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

**Theorem 5** (Determinacy of Typing). \[ \text{Go to proof} \]

(1) Checking: Given \( \Gamma, e, A, p \) such that \( D_1 :: \Gamma \vdash e \iff A \vdash p \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash e \iff A \vdash p \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

(2) Synthesis: Given \( \Gamma, e \) such that \( D_1 :: \Gamma \vdash e \Rightarrow B_1 \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash e \Rightarrow B_2 \vdash \Delta_2 \), it is the case that \( B_1 = B_2 \) and \( p_1 = p_2 \) and \( \Delta_1 = \Delta_2 \).

(3) Spine judgments:

Given \( \Gamma, e, A, p \) such that \( D_1 :: \Gamma \vdash e : A \vdash p \Rightarrow C_1 \vdash q_1 \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash e : A \vdash p \Rightarrow C_2 \vdash q_2 \vdash \Delta_2 \), it is the case that \( C_1 = C_2 \) and \( q_1 = q_2 \) and \( \Delta_1 = \Delta_2 \).

The same applies for derivations of the principality-recovering judgments \( \Gamma \vdash e : A \vdash p \Rightarrow C_k \vdash [q_k] \vdash \Delta_k \).

(4) Match judgments:

Given \( \Gamma, \Pi, \bar{\alpha}, p, C \) such that \( D_1 :: \Gamma \vdash \Pi :: \bar{\alpha} \iff C \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash \Pi :: \bar{\alpha} \iff C \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

**J  Soundness**

**J.1 Soundness of Instantiation**

**Lemma 83** (Soundness of Instantiation). \[ \text{Go to proof} \]

If \( \Gamma \vdash \alpha : \tau : \kappa \vdash \Delta \) and \( \bar{\alpha} \notin \text{FV}([\Gamma] \tau) \) and \( [\Gamma] \tau = \tau \) and \( \Delta \to \Omega \) then \( [\Omega] \bar{\alpha} = [\Omega] \tau \).

**J.2 Soundness of Checkeq**

**Lemma 84** (Soundness of Checkeq). \[ \text{Go to proof} \]

If \( \Gamma \vdash \sigma = t : \kappa \vdash \Delta \to \Omega \) then \( [\Omega] \sigma = [\Omega] t \).

**J.3 Soundness of Equivalence (Propositions and Types)**

**Lemma 85** (Soundness of Propositional Equivalence). \[ \text{Go to proof} \]

If \( \Gamma \vdash \Pi \equiv \Pi' \vdash \Delta \to \Omega \) then \( [\Omega] \Pi = [\Omega] \Pi' \).

**Lemma 86** (Soundness of Algorithmic Equivalence). \[ \text{Go to proof} \]

If \( \Gamma \vdash A \equiv B \vdash \Delta \to \Omega \) then \( [\Omega] A = [\Omega] B \).

**J.4 Soundness of Checkprop**

**Lemma 87** (Soundness of Checkprop). \[ \text{Go to proof} \]

If \( \Gamma \vdash \Pi \text{ true } \to \Delta \to \Omega \) then \( \Psi \vdash [\Omega] \Pi \text{ true } \).

**J.5 Soundness of Eliminations (Equality and Proposition)**

**Lemma 88** (Soundness of Equality Elimination). \[ \text{Go to proof} \]

If \( [\Gamma] \sigma = \sigma \) and \( [\Gamma] t = t \) and \( \Gamma \vdash \sigma : \kappa \) and \( \Gamma \vdash t : \kappa \) and \( \text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset \), then:
(1) If \( \Gamma / \sigma \vdash t : \kappa \vdash \Delta \) then \( \Delta = (\Gamma, \Theta) \) where \( \Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n) \) and for all \( \Omega \) such that \( \Gamma \rightarrow \Omega \) and all \( t' \) such that \( \Omega \vdash t' : \kappa' \), it is the case that \( [\Omega, \Theta] t' = [\theta][\Omega] t' \), where \( \theta = \text{mgu}(\sigma, t) \).

(2) If \( \Gamma / \sigma \vdash t : \kappa \vdash \bot \) then \( \text{mgu}(\sigma, t) = \bot \) (that is, no most general unifier exists).

J.6 Soundness of Subtyping

Theorem 6 (Soundness of Algorithmic Subtyping). Go to proof

If \( [\Gamma] \mathcal{A} = \mathcal{A} \) and \( [\Gamma] \mathcal{B} = \mathcal{B} \) and \( \Gamma \vdash \mathcal{A} \) type and \( \Gamma \vdash \mathcal{B} \) type and \( \Delta \rightarrow \Omega \) and \( \Gamma \vdash \mathcal{A} < \mathcal{B} \vdash \Delta \) then \( [\Omega] \Delta \vdash [\Omega] \mathcal{A} \leq \mathcal{B} [\Omega] \mathcal{B} \).

J.7 Soundness of Typing

Theorem 7 (Soundness of Match Coverage). Go to proof

1. If \( \Gamma \vdash \Pi \) covers \( \bar{\mathcal{A}} \) and \( \Gamma \rightarrow \Omega \) and \( \Gamma \vdash \bar{\mathcal{A}} \) ! types and \( [\Gamma] \bar{\mathcal{A}} = \bar{\mathcal{A}} \) then \( [\Omega] \Gamma \vdash \Pi \) covers \( \bar{\mathcal{A}} \).
2. If \( \Gamma / \mathcal{P} \vdash \Pi \) covers \( \bar{\mathcal{A}} \) and \( \Gamma \rightarrow \Omega \) and \( \Gamma \vdash \bar{\mathcal{A}} \) ! types and \( [\Gamma] \bar{\mathcal{A}} = \bar{\mathcal{A}} \) and \( [\Gamma] \mathcal{P} = \mathcal{P} \) then \( [\Omega] \Gamma / \mathcal{P} \vdash \Pi \) covers \( \bar{\mathcal{A}} \).

Lemma 89 (Well-formedness of Algorithmic Typing). Go to proof

Given \( \Gamma \) ctx:

(i) If \( \Gamma \vdash e \Rightarrow \mathcal{A} \mathcal{P} \vdash \Delta \) then \( \Delta \vdash \mathcal{A} \mathcal{P} \) type.

(ii) If \( \Gamma \vdash s : \mathcal{A} \mathcal{P} \gg \mathcal{B} \mathcal{Q} \vdash \Delta \) and \( \Gamma \vdash \mathcal{A} \mathcal{P} \) type then \( \Delta \vdash \mathcal{B} \mathcal{Q} \) type.

Definition 7 (Measure). Let measure \( \mathcal{M} \) on typing judgments be a lexicographic ordering:

1. first, the subject expression \( e \), spine \( s \), or matches \( \Pi \)—regarding all types in annotations as equal in size;
2. second, the partial order on judgment forms where an ordinary spine judgment is smaller than a principality-recovering spine judgment—and with all other judgment forms considered equal in size; and,
3. third, the derivation height.

\[
\left< e/s/\Pi, \begin{array}{c} \text{ordinary spine judgment} \\ \text{recovering spine judgment} \end{array}, \text{height}(\mathcal{D}) \right>
\]

Note that this definition doesn’t take notice of whether a spine judgment is declarative or algorithmic.

This measure works to show soundness and completeness. We list each rule below, along with a 3-tuple. For example, for \( \text{Sub} \) we write \( <, =, < > \), meaning that each judgment to which we need to apply the i.h. has a subject of the same size (\( = \)), a judgment form of the same size (\( = \)), and a smaller derivation height. We write – when a part of the measure need not be considered because a lexicographically more significant part is smaller, as in the \( \text{Anno} \) rule, where the premise has a smaller subject: \( <, <, < > \).

Algorithmic rules (soundness cases):

- \( \text{Var}, \text{11}, \text{EmptySpine}, \text{Nil} \) have no premises, or only auxiliary judgments as premises.
- \( \text{Sub} \) : \( <, =, < > \)
J.7 Soundness of Typing

Theorem 8 (Soundness of Algorithmic Typing). Go to proof

Given $\Delta \rightarrow \Omega$:

(i) If $\Gamma \vdash e \leftrightarrow A \ p \vdash \Delta$ and $\Gamma \vdash A \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] e \leftrightarrow [\Omega] A \ p$.

(ii) If $\Gamma \vdash e \Rightarrow A \ p \vdash \Delta$ then $[\Omega] \Delta \vdash [\Omega] e \Rightarrow [\Omega] A \ p$.

(iii) If $\Gamma \vdash s : A \ p \Rightarrow B \ q \vdash \Delta$ and $\Gamma \vdash A \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] s : [\Omega] A \ p \Rightarrow [\Omega] B \ q$.

(iv) If $\Gamma \vdash s : A \ p \Rightarrow B \ [q] \vdash \Delta$ and $\Gamma \vdash A \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] s : [\Omega] A \ p \Rightarrow [\Omega] B \ [q]$.

(v) If $\Gamma \vdash \Pi :: \tilde{A} \leftrightarrow C \ p \vdash \Delta$ and $\Gamma \vdash \tilde{A} \ type$ and $[\Gamma] \tilde{A} = \tilde{A}$ and $\Gamma \vdash C \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] \Pi :: [\Omega] \tilde{A} \leftrightarrow [\Omega] C \ p$.

(vi) If $\Gamma / P \vdash \Pi :: \tilde{A} \leftrightarrow C \ p \vdash \Delta$ and $\Gamma \vdash P \ prop$ and $\text{FEV}(P) = \emptyset$ and $[\Gamma] P = P$ and $\Gamma \vdash \tilde{A} \ type$ and $\Gamma \vdash C \ p \ type$ then $[\Omega] \Delta / [\Omega] P \vdash [\Omega] \Pi :: [\Omega] \tilde{A} \leftrightarrow [\Omega] C \ p$. 

Declarative rules (completeness cases):

• $\text{DeclVar}$, $\text{DeclNil}$, $\text{DeclEmptySpine}$ and $\text{DeclNil}$ have no premises, or only auxiliary judgments as premises.

• $\text{DeclSub}$: $\langle =, =, < \rangle$

• $\text{DeclAnno}$: $\langle <,-,- \rangle$

• $\text{DeclSpine}$, $\text{DeclSpinePass}$ and $\text{DeclSpineRecover}$ have only an auxiliary judgment, to which we need not apply the i.h., putting it in the same class as the rules with no premises.

• $\text{DeclSpine}$: $\langle =, =, < \rangle$

• $\text{DeclSpinePass}$: $\langle =, <,- \rangle$

• $\text{DeclSpineRecover}$: $\langle =, <,- \rangle$
K Completeness

K.1 Completeness of Auxiliary Judgments

**Lemma 90** (Completeness of Instantiation). *Go to proof*

Given $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash \tau : \kappa$ and $\tau = [\Gamma] \tau$ and $\alpha \in \text{unsolved}(\Gamma)$ and $\alpha \notin \text{FV}(\tau)$:

If $[\Omega] \alpha = [\Omega] \tau$

then there are $\Delta, \Omega'$ such that $\Omega \rightarrow \Omega'$ and $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Gamma \vdash \Delta := \tau : \kappa \vdash \Delta$.

**Lemma 91** (Completeness of Checkeq). *Go to proof*

Given $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$

and $\Gamma \vdash \sigma : \kappa$ and $\Gamma \vdash \tau : \kappa$

and $[\Omega] \sigma = [\Omega] \tau$

then $\Gamma \vdash [\Gamma] \sigma \equiv [\Gamma] \tau : \kappa \vdash \Delta$

where $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$.

**Lemma 92** (Completeness of Elimeq). *Go to proof*

If $[\Gamma] \sigma = \sigma$ and $[\Gamma] t = t$ and $\Gamma \vdash \sigma : \kappa$ and $\Gamma \vdash t : \kappa$ and $\text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset$ then:

1. If $\text{mgu}(\sigma, t) = \emptyset$

   then $\Gamma \vdash \sigma \equiv t : \kappa \vdash \perp$ (that is, no most general unifier exists) then $\Gamma \vdash \sigma \equiv t : \kappa \vdash \perp$.

**Lemma 93** (Substitution Upgrade). *Go to proof*

If $\Delta$ has the form $\alpha_1 = t_1, \ldots, \alpha_n = t_n$

and, for all $u$ such that $\Gamma \vdash u : \kappa$, it is the case that $[\Gamma, \Delta] u = \theta([\Gamma] u)$, then:

(i) If $\Gamma \vdash A$ type then $[\Gamma, \Delta] A = \theta([\Gamma] A)$.

(ii) If $\Gamma \rightarrow \Omega$ then $[\Omega] \Gamma = \theta([\Omega] \Gamma)$.

(iii) If $\Gamma \rightarrow \Omega$ then $[\Omega, \Delta] [\Gamma, \Delta] = \theta([\Omega] [\Gamma])$.

(iv) If $\Gamma \rightarrow \Omega$ then $[\Omega, \Delta] e = \theta([\Omega] e)$.

**Lemma 94** (Completeness of Propequiv). *Go to proof*

Given $\Gamma \rightarrow \Omega$

and $\Gamma \vdash P$ prop and $\Gamma \vdash Q$ prop

and $[\Omega] P = [\Omega] Q$

then $\Gamma \vdash [\Gamma] P \equiv [\Gamma] Q : \Delta$

where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$.

**Lemma 95** (Completeness of Checkprop). *Go to proof*

If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$

and $\Gamma \vdash P$ prop

and $[\Gamma] P = P$

and $[\Omega] \Gamma \vdash [\Omega] P$ true

then $\Gamma \vdash P$ true $\vdash \Delta$

where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$.

K.2 Completeness of Equivalence and Subtyping

**Lemma 96** (Completeness of Eq). *Go to proof*

If $\Gamma \rightarrow \Omega$ and $\Gamma \vdash A$ type and $\Gamma \vdash B$ type
K.3 Completeness of Typing

Theorem 9 (Completeness of Subtyping). \( \text{Go to proof} \)
If \( \Gamma \rightarrow \Omega \) and \( \text{dom}(\Gamma) = \text{dom}(\Omega) \) and \( \Gamma \vdash A \) type and \( \Gamma \vdash B \) type
and \( [\Omega]\Gamma \vdash [\Omega]A \leq [\Omega]B \)
then there exist \( \Delta \) and \( \Omega' \) such that \( \Delta \rightarrow \Omega' \)
and \( \text{dom}(\Delta) = \text{dom}(\Omega') \)
and \( \Omega \rightarrow \Omega' \)
and \( \Gamma \vdash [\Gamma]A \triangleq [\Gamma]B \rightarrow \Delta \).

K.3 Completeness of Typing

Theorem 10 (Completeness of Match Coverage). \( \text{Go to proof} \)

1. If \( [\Omega]\Gamma \vdash [\Omega]\Pi \) covers \( [\Omega]\bar{A} \) and \( \Gamma \rightarrow \Omega \) and \( \Gamma \vdash \bar{A} ! \) types and \( [\Gamma]\bar{A} = \bar{A} \)
then \( \Gamma \vdash \Pi \) covers \( \bar{A} \).

2. If \( [\Omega]\Gamma / [\Omega]\Pi \vdash [\Omega]\bar{A} \) and \( \Gamma \rightarrow \Omega \) and \( \Gamma \vdash \bar{A} ! \) types and \( [\Gamma]\bar{A} = \bar{A} \) and \( [\Gamma]\Pi = \Pi \)
then \( \Gamma / \Pi \vdash \Pi \) covers \( \bar{A} \).

Theorem 11 (Completeness of Algorithmic Typing). \( \text{Go to proof} \)
Given \( \Gamma \rightarrow \Omega \) such that \( \text{dom}(\Gamma) = \text{dom}(\Omega) \):

(i) If \( \Gamma \vdash A \ p \) type and \( [\Omega]\Gamma \vdash [\Omega]e \leftrightarrow [\Omega]A \ p \) and \( p' \subseteq p \)
then there exist \( \Delta \) and \( \Omega' \) such that \( \Delta \rightarrow \Omega' \)
and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \)
and \( \Gamma \vdash e \leftarrow [\Gamma]A \ p' \rightarrow \Delta \).

(ii) If \( \Gamma \vdash A \ p \) type and \( [\Omega]\Gamma \vdash [\Omega]e \Rightarrow A \ p \)
then there exist \( \Delta, \Omega', A', \) and \( p' \subseteq p \)
such that \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \)
and \( \Gamma \vdash e \Rightarrow A' \ p' \rightarrow \Delta \) and \( A' = [\Delta]A' \) and \( A = [\Omega']A' \).

(iii) If \( \Gamma \vdash A \ p \) type and \( [\Omega]\Gamma \vdash [\Omega]s : [\Omega]A \ p \triangleright B \ q \) and \( p' \subseteq p \)
then there exist \( \Delta, \Omega', B', \) and \( q' \subseteq q \)
such that \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \)
and \( \Gamma \vdash s : [\Gamma]A \ p' \triangleright B' \ q' \rightarrow \Delta \) and \( B' = [\Delta]B' \) and \( B = [\Omega']B' \).

(iv) If \( \Gamma \vdash A \ p \) type and \( [\Omega]\Gamma \vdash [\Omega]s : [\Omega]A \ p \triangleright B \ [q] \) and \( p' \subseteq p \)
then there exist \( \Delta, \Omega', B', \) and \( q' \subseteq q \)
such that \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \)
and \( \Gamma \vdash s : [\Gamma]A \ p' \triangleright B' \ [q'] \rightarrow \Delta \) and \( B' = [\Delta]B' \) and \( B = [\Omega']B' \).

(v) If \( \Gamma \vdash \bar{A} ! \) types and \( \Gamma \vdash C \ p \) type and \( [\Omega]\Gamma \vdash [\Omega]\Pi : [\Omega]\bar{A} \leftrightarrow [\Omega]C \ p \) and \( p' \subseteq p \)
then there exist \( \Delta, \Omega', \) and \( C \)
such that \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \)
and \( \Gamma \vdash \Pi : [\Gamma]\bar{A} \leftrightarrow [\Gamma]C \ p' \rightarrow \Delta \).

(vi) If \( \Gamma \vdash \bar{A} ! \) types and \( \Gamma \vdash P \ prop \) and \( \text{FEV}(P) = \emptyset \) and \( \Gamma \vdash C \ p \) type
and \( [\Omega]\Gamma / [\Omega]P \vdash [\Omega]\Pi : [\Omega]\bar{A} \leftrightarrow [\Omega]C \ p \)
and \( p' \subseteq p \)
then there exist \( \Delta, \Omega', \) and \( C \)
such that \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \)
and \( \Gamma / [\Gamma]\Pi \vdash \Pi : [\Gamma]\bar{A} \leftrightarrow [\Gamma]C \ p' \rightarrow \Delta \).
Proofs

In the rest of this document, we prove the results stated above, with the same sectioning.

A’  Properties of the Declarative System

Lemma 1 (Declarative Well-foundedness).
The inductive definition of the following judgments is well-founded:

(i) synthesis \( \Psi \vdash e \Rightarrow B p \)
(ii) checking \( \Psi \vdash e \Leftarrow A p \)
(iii) checking, equality elimination \( \Psi / P \vdash e \Leftarrow C p \)
(iv) ordinary spine \( \Psi \vdash s : A p \gg B q \)
(v) recovery spine \( \Psi \vdash s : A p \gg B \lceil q \rceil \)
(vi) pattern matching \( \Psi \vdash \Pi :: \vec{A} \Leftarrow C p \)
(vii) pattern matching, equality elimination \( \Psi / P \vdash \Pi :: \vec{A} \Leftarrow C p \)

Proof. Let \(|e|\) be the size of the expression \(e\). Let \(|s|\) be the size of the spine \(s\). Let \(|\Pi|\) be the size of the branch list \(\Pi\). Let \(#\text{large}(A)\) be the number of “large” connectives \(\forall, \exists, \supset, \land\) in \(A\).

First, stratify judgments by the size of the term (expression, spine, or branches), and say that a judgment is at \(n\) if it types a term of size \(n\). Order the main judgment forms as follows:

\[
\begin{align*}
\text{synthesis judgment at } n &< \text{ checking judgments at } n \\
&< \text{ ordinary spine judgment at } n \\
&< \text{ recovery spine judgment at } n \\
&< \text{ match judgments at } n \\
&< \text{ synthesis judgment at } n + 1 \\
&< \ldots
\end{align*}
\]

Within the checking judgment forms at \(n\), we compare types lexicographically, first by the number of large connectives, and then by the ordinary size. Within the match judgment forms at \(n\), we compare using a lexicographic order of, first, \(#\text{large}(\vec{A})\); second, the judgment form, considering the match judgment to be smaller than the matchelim judgment; third, the size of \(\vec{A}\). These criteria order the judgments as follows:

\[
\begin{align*}
\text{synthesis judgment at } n &< \text{ (checking judgment at } n \text{ with } #\text{large}(A) = 1 \\
&< \text{ checkelim judgment at } n \text{ with } #\text{large}(A) = 1 \\
&< \text{ checking judgment at } n \text{ with } #\text{large}(A) = 2 \\
&< \text{ checkelim judgment at } n \text{ with } #\text{large}(A) = 2 \\
&< \ldots
\end{align*}
\]

\[
\begin{align*}
&< \text{ (match judgment at } n \text{ with } #\text{large}(\vec{A}) = 1 \text{ and } \vec{A} \text{ of size } 1 \\
&< \text{ match judgment at } n \text{ with } #\text{large}(\vec{A}) = 1 \text{ and } \vec{A} \text{ of size } 2 \\
&< \text{ matchelim judgment at } n \text{ with } #\text{large}(\vec{A}) = 1 \\
&< \text{ match judgment at } n \text{ with } #\text{large}(\vec{A}) = 2 \text{ and } \vec{A} \text{ of size } 1 \\
&< \text{ match judgment at } n \text{ with } #\text{large}(\vec{A}) = 2 \text{ and } \vec{A} \text{ of size } 2 \\
&< \text{ matchelim judgment at } n \text{ with } #\text{large}(\vec{A}) = 2 \\
&< \ldots
\end{align*}
\]
Proof of Lemma 1 (Declarative Well-foundedness) \( \text{lem:declarative-well-founded} \)

The class of ordinary spine judgments at 1 need not be refined, because the only ordinary spine rule applicable to a spine of size 1 is \( \text{DeclEmptySpine} \) which has no premises; rules \( \text{Decl\lorSpine} \), \( \text{Decl\toSpine} \) and \( \text{Decl\to\toSpine} \) are restricted to non-empty spines and can only apply to larger terms.

Similarly, the class of match judgments at 1 need not be refined, because only \( \text{DeclMatchEmpty} \) is applicable.

Note that we distinguish the “checkelim” form \( \Psi / P \vdash e \leftarrow A \ p \) of the checking judgment. We also define the size of an expression \( e \) to consider all types in annotations to be of the same size, that is,

\[ |(e : A)| = |e| + 1 \]

Thus, \(|\theta(e)| = |e|\), even when \( e \) has annotations. This is used for \( \text{DeclCheckUnify} \), see below.

We assume that coverage, which does not depend on any other typing judgments, is well-founded. We likewise assume that subtyping, \( \Psi \vdash A \type \), \( \Psi \vdash \tau : \kappa \), and \( \Psi \vdash P \prop \) are well-founded.

We now show that, for each class of judgments, every judgment in that class depends only on smaller judgments.

- **Synthesis judgments**
  
  **Claim:** For all \( n \), synthesis at \( n \) depends only on judgments at \( n - 1 \) or less.
  
  **Proof:** Rule \( \text{DeclVar} \) has no premises.
  
  Rule \( \text{DeclAnno} \) depends on a premise at a strictly smaller term.
  
  Rule \( \text{Decl\toE} \) depends on (1) a synthesis premise at a strictly smaller term, and (2) a recovery spine judgment at a strictly smaller term.

- **Checking judgments**
  
  **Claim:** For all \( n \geq 1 \), the checking judgment over terms of size \( n \) with type of size \( m \) depends only on
  
  (1) synthesis judgments at size \( n \) or smaller, and
  
  (2) checking judgments at size \( n - 1 \) or smaller, and
  
  (3) checking judgments at size \( n \) with fewer large connectives, and
  
  (4) checkelim judgments at size \( n \) with fewer large connectives, and
  
  (5) match judgments at size \( n - 1 \) or smaller.
  
  **Proof:** Rule \( \text{DeclSub} \) depends on a synthesis judgment of size \( n \). (1)
  
  Rule \( \text{DeclI} \) has no premises.
  
  Rule \( \text{Decl\lorI} \) depends on a checking judgment at \( n \) with fewer large connectives. (3)
  
  Rule \( \text{Decl\landI} \) depends on a checking judgment at \( n \) with fewer large connectives. (3)
  
  Rule \( \text{Decl\toI} \) depends on a checkelim judgment at \( n \) with fewer large connectives. (4)
  
  Rules \( \text{Decl\toI}, \text{DeclRec}, \text{Decl\to\toI}, \text{Decl\timesI} \) and \( \text{DeclCons} \) depend on checking judgments at size \( < n \). (2)
  
  Rule \( \text{DeclNil} \) depends only on an auxiliary judgment.
  
  Rule \( \text{DeclCase} \) depends on:
  
  - a synthesis judgment at size \( n \) (1),
  
  - a match judgment at size \( < n \) (5), and
  
  - a coverage judgment.

- **Checkelim judgments**
  
  **Claim:** For all \( n \geq 1 \), the checkelim judgment \( \Psi / P \vdash e \leftarrow A \ p \) over terms of size \( n \) depends only on checking judgments at size \( n \), with a type \( A' \) such that \( \#\text{large}(A') = \#\text{large}(A) \).
  
  **Proof:** Rule \( \text{DeclCheckI} \) has no nontrivial premises.
  
  Rule \( \text{DeclCheckUnify} \) depends on a checking judgment: Since \( |\theta(e)| = |e| \), this checking judgment is at \( n \).
  
  Since the mgu \( \theta \) is over monotypes, \( \#\text{large}(\theta(A)) = \#\text{large}(A) \).
Proof of Lemma 1 (Declarative Well-foundedness).

• Ordinary spine judgments
An ordinary spine judgment at 1 depends on no other judgments: the only spine of size 1 is the empty spine, so only DeclEmptySpine applies, and it has no premises.

Claim: For all \( n \geq 2 \), the ordinary spine judgment \( \Psi \vdash s : A \gg C q \) over spines of size \( n \) depends only on

(a) checking judgments at size \( n - 1 \) or smaller, and
(b) ordinary spine judgments at size \( n - 1 \) or smaller, and
(c) ordinary spine judgments at size \( n \) with strictly smaller \#large\( \)\( A \).

Proof. Rule Decl\( \triangleright \)Spine depends on an ordinary spine judgment of size \( n \), with a type that has fewer large connectives. (c)
Rule Decl\( \triangleleft \)Spine depends on an ordinary spine judgment of size \( n \), with a type that has fewer large connectives. (c)
Rule DeclEmptySpine has no premises.
Rule Decl\( \rightarrow \)Spine depends on a checking judgment of size \( n - 1 \) or smaller (a) and an ordinary spine judgment of size \( n - 1 \) or smaller (b).

• Recovery spine judgments
Claim: For all \( n \), the recovery spine judgment at \( n \) depends only on ordinary spine judgments at \( n \).
Proof. Rules DeclSpineRecover and DeclSpinePass depend only on ordinary spine judgments at \( n \).

• Match judgments
Claim: For all \( n \geq 1 \), the match judgment \( \Psi \vdash \Pi :: \vec{A} \leftarrow C p \) over \( \Pi \) of size \( n \) depends only on

(a) checking judgments at size \( n - 1 \) or smaller, and
(b) match judgments at size \( n - 1 \) or smaller, and
(c) match judgments at size \( n \) with smaller \( \vec{A} \), and
(d) matchelim judgments at size \( n \) with fewer large connectives in \( \vec{A} \).

Proof. Rule DeclMatchEmpty has no premises.
Rule DeclMatchSeq depends on match judgments at \( n - 1 \) or smaller (b).
Rule DeclMatchBase depends on a checking judgment at \( n - 1 \) or smaller (a).
Rules DeclMatchUnit, DeclMatch\( \times \), DeclMatch\( +k \), DeclMatchNeg, and DeclMatchWild depend on match judgments at \( n - 1 \) or smaller (b).
Rule DeclMatch\( \exists \) depends on a match judgment at size \( n \) with smaller \( \vec{A} \) (c).
Rule DeclMatch\( \land \) depends on an matchelim judgment at \( n \), with fewer large connectives in \( \vec{A} \) (d).

• Matchelim judgments
Claim: For all \( n \geq 1 \), the matchelim judgment \( \Psi/\Pi \vdash P :: \vec{A} \leftarrow C p \) over \( \Psi \) of size \( n \) depends only on match judgments with the same number of large connectives in \( \vec{A} \).
Proof. Rule DeclMatch\( \bot \) has no nontrivial premises.
Rule DeclMatchUnify depends on a match judgment with the same number of large connectives (similar to DeclCheckUnify considered above).

Lemma 2 (Declarative Weakening).

(i) If \( \Psi_0, \Psi_1 \vdash t : \kappa \) then \( \Psi_0, \Psi, \Psi_1 \vdash t : \kappa \).
(ii) If \( \Psi_0, \Psi_1 \vdash P \text{ prop} \) then \( \Psi_0, \Psi, \Psi_1 \vdash P \text{ prop} \).
(iii) If \( \Psi_0, \Psi_1 \vdash P \text{ true} \) then \( \Psi_0, \Psi, \Psi_1 \vdash P \text{ true} \).
(iv) If \( \Psi_0, \Psi_1 \vdash A \text{ type} \) then \( \Psi_0, \Psi, \Psi_1 \vdash A \text{ type} \).
Proof. By induction on the derivation. □

**Lemma 3** (Declarative Term Substitution). Suppose $\Psi \vdash t : \kappa$. Then:

1. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash t' : \kappa$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha]t' : \kappa$.
2. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P$ prop then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha]P$ prop.
3. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A$ type then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha]A$.
4. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \leq B$ then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha]A \leq [t/\alpha]B$.
5. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P$ true then $\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha]P$ true.

Proof. By induction on the derivation of the substitutee. □

**Lemma 4** (Reflexivity of Declarative Subtyping).

Given $\Psi \vdash A$ type, we have that $\Psi \vdash A \leq \pm A$.

Proof. By induction on $A$, writing $p$ for the sign of the subtyping judgment.

Our induction metric is the number of quantifiers on the outside of $A$, plus one if the polarity of $A$ and the subtyping judgment do not match up (that is, if $\neg A$ and $p = +$, or $A$ and $p = -$).

- **Case** $\text{nonpos}(A), \text{nonneg}(A), p = \pm$:
  By rule $\leq\text{Refl} \pm$.

- **Case** $A = \exists b : \kappa. B, p = +$:
  
  $\Psi, b : \kappa \vdash B \leq + B$ \quad By i.h. (one less quantifier)
  
  $\Psi, b : \kappa \vdash b : \kappa$ \quad By rule $\text{UvarSort}$
  
  $\Psi \vdash \exists b : \kappa. B \leq + \exists b : \kappa. B$ \quad By rule $\leq \exists R$

- **Case** $A = \exists b : \kappa. B, p = -$:
  
  $\Psi \vdash \exists b : \kappa. B \leq - \exists b : \kappa. B$ \quad By $\leq$ $

- **Case** $A = \forall b : \kappa. B, p = +$:
  
  $\Psi \vdash \forall b : \kappa. B \leq + \forall b : \kappa. B$ \quad By i.h. (polarities match)
  
  $\Psi \vdash \forall b : \kappa. B \leq + \forall b : \kappa. B$ \quad By $\leq +$

- **Case** $A = \forall b : \kappa. B, p = -$:
  
  $\Psi, b : \kappa \vdash B \leq - B$ \quad By i.h. (one less quantifier)
  
  $\Psi, b : \kappa \vdash b : \kappa$ \quad By rule $\text{UvarSort}$
  
  $\Psi \vdash \forall b : \kappa. B \leq - \forall b : \kappa. B$ \quad By rule $\leq \forall L$

- **Case** $A = \forall b : \kappa. B, p = -$:
  
  $\Psi, b : \kappa \vdash B \leq - B$ \quad By i.h. (one less quantifier)
  
  $\Psi, b : \kappa \vdash b : \kappa$ \quad By rule $\text{UvarSort}$
  
  $\Psi \vdash \forall b : \kappa. B \leq - \forall b : \kappa. B$ \quad By rule $\leq \forall R$

- **Lemma 5** (Subtyping Inversion).
  
  - $\text{If } \Psi \vdash \exists \alpha : \kappa. A \leq B \text{ then } \Psi, \alpha : \kappa \vdash A \leq + B$.
  
  - $\text{If } \Psi \vdash A \leq \forall \beta : \kappa. B \text{ then } \Psi, \beta : \kappa \vdash A \leq - B$.

Proof. By a routine induction on the subtyping derivations. □

**Lemma 6** (Subtyping Polarity Flip).

March 16, 2018
Properties of the Declarative System

- If \( \text{nonpos}(A) \) and \( \text{nonpos}(B) \) and \( \psi \vdash A \leq^+ B \)
  then \( \psi \vdash A \leq^- B \) by a derivation of the same or smaller size.

- If \( \text{nonneg}(A) \) and \( \text{nonneg}(B) \) and \( \psi \vdash A \leq^- B \)
  then \( \psi \vdash A \leq^+ B \) by a derivation of the same or smaller size.

- If \( \text{nonpos}(A) \) and \( \text{nonneg}(A) \) and \( \text{nonpos}(B) \) and \( \text{nonneg}(B) \)
  and \( \psi \vdash A \leq^\pm B \)
  then \( A = B \).

Proof. By a routine induction on the subtyping derivations.

Lemma 7 (Transitivity of Declarative Subtyping).

\[ \text{Given } \psi \vdash A \text{ type and } \psi \vdash B \text{ type and } \psi \vdash C \text{ type:} \]

(i) If \( D_1 : \psi \vdash A \leq^\pm B \) and \( D_2 : \psi \vdash B \leq^\pm C \)
then \( \psi \vdash A \leq^\pm C \).

Proof. By lexicographic induction on (1) the sum of head quantifiers in \( A, B, \) and \( C \), and (2) the size of the derivation.

We begin by case analysis on the shape of \( B \), and the polarity of subtyping:

- Case \( B = \forall \beta : \kappa_2, B' \), polarity = \(-\):
  We case-analyze \( D_1 \):

  \[
  \frac{
  \psi \vdash \tau : \kappa_1 \quad \psi \vdash [\tau/\alpha]A' \leq^- B
  }{
  \psi \vdash \forall \alpha : \kappa_1, A' \leq^- B}
  \]
  Subderivation
  Subderivation
  Given
  By i.h. (A lost a quantifier)
  By rule \( \leq\forall L \)

  \[
  \frac{
  \psi \vdash \tau : \kappa_1 \quad \psi \vdash [\tau/\beta]B' \leq^- C
  }{
  \psi \vdash A \leq^- \forall \beta : \kappa_2, B'}
  \]
  Subderivation
  Subderivation of \( D_2 \)
  By Lemma 3 (Declarative Term Substitution)
  By i.h. (B lost a quantifier)
  By \( \leq\forall R \)

We case-analyze \( D_2 \):

* Case \( \psi, \beta : \kappa_2 \vdash A \leq^- B' \)
  By Lemma 5 (Subtyping Inversion) on \( D_1 \)
  \( \psi \vdash \tau : \kappa_2 \)
  Subderivation
  \( \psi \vdash [\tau/\beta]B' \leq^- C \)
  Subderivation of \( D_2 \)
  \( \psi \vdash A \leq^- [\tau/\beta]B' \)
  By Lemma 3 (Declarative Term Substitution)
  \( \psi \vdash A \leq^- C \)
  By i.h. (B lost a quantifier)
  \( \psi, \beta : \kappa_2 \vdash A \leq^- B' \)

* Case \( \psi, c : \kappa_3 \vdash B \leq^- C' \)
  By Lemma 2 (Declarative Weakening)
  \( \psi, c : \kappa_3 \vdash A \leq^- B \)
  Subderivation
  \( \psi, c : \kappa_3 \vdash A \leq^- C' \)
  By i.h. (C lost a quantifier)
  \( \psi \vdash B \leq^- \forall c : \kappa_3, C' \)
  By \( \leq\forall R \)
Case $\text{nonpos}(B)$, polarity $= \vdash$:

Now we case-analyze $D_1$:

- Case $\Psi, \alpha : \tau \vdash A' \leq^+ B$
  
  $\frac{\Psi \vdash \exists \alpha : \kappa_1, A' \leq^+ B}{\Psi \vdash A \leq^+ B}$

  Subderivation

  $\Psi, \alpha : \tau \vdash A' \leq^+ B$ By Lemma 2 (Declarative Weakening) ($D_2$)

  $\Psi, \alpha : \tau \vdash A' \leq^+ C$ By i.h. (A lost a quantifier)

  $\Psi \vdash \exists \alpha : \kappa_1, A' \leq^+ C$ By $\leq^+$

- Case $\Psi \vdash A \leq^+ B$

  $\frac{\Psi \vdash A \leq^+ B}{\Psi \vdash A \leq^+ B}$

  Now we case-analyze $D_2$:

  * Case $\Psi \vdash \tau : \kappa_3$
    
    $\frac{\Psi \vdash B \leq^+ [\tau/c]C'}{\Psi \vdash B \leq^+ \exists c : \kappa_3, C'}$

    $\Psi \vdash A \leq^+ B$ Given

    $\Psi \vdash \tau : \kappa_3$ Subderivation of $D_2$

    $\Psi \vdash B \leq^+ [\tau/c]C'$ Subderivation of $D_2$

    $\Psi \vdash A \leq^+ [\tau/c]C'$ By i.h. (C lost a quantifier)

    $\Psi \vdash A \leq^+ \exists c : \kappa_3, C'$ By $\leq^+$

  * Case $\Psi \vdash B \leq^+ C$

    $\frac{\Psi \vdash B \leq^+ C}{\Psi \vdash B \leq^+ C}$

    $\Psi \vdash A \leq^+ B$ Subderivation of $D_1$

    $\Psi \vdash B \leq^+ C$ Subderivation of $D_2$

    $\Psi \vdash A \leq^+ C$ By i.h. ($D_1$ and $D_2$ smaller)

    $\text{nonpos}(A)$ Subderivation of $D_1$

    $\text{nonpos}(C)$ Subderivation of $D_2$

    $\Psi \vdash A \leq^+ C$ By $\leq^+$

Case $B = \exists \beta : \kappa_2, B'$, polarity $= \vdash$:

Now we case-analyze $D_2$:

- Case $\Psi \vdash \tau : \kappa_3$

  $\frac{\Psi \vdash B \leq^+ [\tau/\alpha]C'}{\Psi \vdash B \leq^+ \exists \alpha : \kappa_3, C'}$

  $\Psi \vdash \tau : \kappa_3$ Subderivation of $D_2$

  $\Psi \vdash B \leq^+ [\tau/\alpha]C'$ Subderivation of $D_2$

  $\Psi \vdash A \leq^+ B$ Given

  $\Psi \vdash A \leq^+ [\tau/\alpha]C'$ By i.h. (C lost a quantifier)

  $\Psi \vdash A \leq^+ C$ By rule $\leq^+$
Proof of Lemma 7 (Transitivity of Declarative Subtyping)

\[ \text{lem:declarative-transitivity} \]

– Case \( \Psi, \beta : \kappa_2 \vdash B' \leq^+ C \)
\[ \Psi \vdash \exists \beta : \kappa_2. B' \leq^+ C \] \( \leq \exists \text{L} \)

Now we case-analyze \( D_1 \):

* Case \( \Psi, \tau : \kappa_2 \)
\[ \Psi \vdash A \leq^+ \exists \beta : \kappa_2. B' \] \( \leq \exists \text{R} \)

\[ \begin{align*}
\Psi, \beta : \kappa_2 \vdash B' \leq^+ C \\
\Psi \vdash \tau : \kappa_2 \\
\Psi \vdash A \leq^+ [\tau/\beta]B' \\
\Psi \vdash [\tau/\beta]B' \leq^+ C \\
\Psi \vdash A \leq^+ C
\end{align*} \]

Subderivation of \( D_2 \)
Subderivation of \( D_1 \)
By Lemma 3 (Declarative Term Substitution)
By i.h. (B lost a quantifier)

* Case \( \Psi, \alpha : \kappa_1 \vdash A \leq^+ B \)
\[ \Psi \vdash \exists \alpha : \kappa_1. A' \leq^+ B \] \( \leq \exists \text{L} \)

\[ \begin{align*}
\Psi \vdash B \leq^+ C \\
\Psi, \alpha : \kappa_1 \vdash A' \leq^+ B \\
\Psi, \alpha : \kappa_1 \vdash A' \leq^+ B \\
\Psi, \alpha : \kappa_1 \vdash A' \leq^+ C \\
\Psi \vdash \exists \alpha : \kappa_1. A' \leq^+ C
\end{align*} \]

Given
Subderivation of \( D_1 \)
By Lemma 2 (Declarative Weakening)
By i.h. (A lost a quantifier)

• Case nonneg(B), polarity = −:

We case-analyze \( D_2 \):

– Case \( \Psi, c : \kappa_3 \vdash B \leq^+ C' \)
\[ \Psi \vdash B \leq^+ \exists c : \kappa_3. C' \] \( \leq \forall \text{R} \)

\[ \begin{align*}
\Psi, c : \kappa_3 \vdash B \leq^+ C' \\
\Psi, c : \kappa_3 \vdash A \leq^+ B \\
\Psi, c : \kappa_3 \vdash A \leq^+ C' \\
\Psi \vdash A \leq^+ \forall c : \kappa_3. C'
\end{align*} \]

Subderivation of \( D_2 \)
By Lemma 2 (Declarative Weakening)
By i.h. (C lost a quantifier)
By \( \leq \forall \text{R} \)

– Case \( \Psi \vdash B \leq^+ C \) nonneg(B) nonneg(C) \( \leq^* \)
\[ \Psi \vdash B \leq^* C \]

We case-analyze \( D_1 \):

* Case \( \Psi, \tau : \kappa_1 \)
\[ \Psi \vdash [\tau/\alpha]A' \leq^* B \] \( \leq \forall \text{L} \)

\[ \begin{align*}
\Psi \vdash B \leq^* C \\
\Psi \vdash \tau : \kappa_1 \\
\Psi \vdash [\tau/\alpha]A' \leq^* B \\
\Psi \vdash [\tau/\alpha]A' \leq^* C \\
\Psi \vdash \forall \alpha : \kappa_1. A' \leq^* C
\end{align*} \]

Given
Subderivation of \( D_1 \)
Subderivation of \( D_1 \)
By i.h. (A lost a quantifier)
By \( \leq \forall \text{L} \)
Proof of Lemma 7 (Transitivity of Declarative Subtyping).

\[ \begin{array}{c|c}
\text{* Case} & \Psi \vdash A \leq^+ B \quad \text{nonpos}(A) \quad \text{nonpos}(B) \\
\hline
\Psi \vdash A \leq^+ B & \text{Subderivation of } D_1 \\
\Psi \vdash B \leq^+ C & \text{Subderivation of } D_2 \\
\Psi \vdash A \leq^+ C & \text{By i.h.} (D_1 \text{ and } D_2 \text{ smaller}) \\
\text{nonneg}(A) & \text{Subderivation of } D_2 \\
\text{nonneg}(C) & \text{Subderivation of } D_2 \\
\Psi \vdash A \leq^- C & \text{By } \leq^+ \\
\end{array} \]

B' Substitution and Well-formedness Properties

Lemma 8 (Substitution—Well-formedness).

(i) If \( \Gamma \vdash A \text{ type} \) and \( \Gamma \vdash \tau \text{ type} \) then \( \Gamma \vdash [\tau/\alpha]A \text{ type} \).

(ii) If \( \Gamma \vdash P \text{ prop} \) and \( \Gamma \vdash \tau \text{ type} \) then \( \Gamma \vdash [\tau/\alpha]P \text{ prop} \).

Moreover, if \( p = ! \) and \( \text{FEV}(\Gamma) = \emptyset \) then \( \text{FEV}(\Gamma[\tau/\alpha]) = \emptyset \).

Proof. By induction on the derivations of \( \Gamma \vdash A \text{ type} \) and \( \Gamma \vdash P \text{ prop} \).

Lemma 9 (Uvar Preservation).

If \( \Delta \rightarrow \Omega \) then:

(i) If \( (\alpha : \kappa) \in \Omega \) then \( (\alpha : \kappa) \in [\Omega]\Delta \).

(ii) If \( (x : A) \in \Omega \) then \( (x : [\Omega]A) \in [\Omega]\Delta \).

Proof. By induction on \( \Omega \), following the definition of context application (Figure 11).

Lemma 10 (Sorting Implies Typing). If \( \Gamma \vdash t : \star \) then \( \Gamma \vdash t \text{ type} \).

Proof. By induction on the given derivation. All cases are straightforward.

Lemma 11 (Right-Hand Substitution for Sorting). If \( \Gamma \vdash t : \kappa \) then \( \Gamma \vdash [\Gamma]t : \kappa \).

Proof. By induction on \( |\Gamma \vdash t| \) (the size of \( t \) under \( \Gamma \)).

- **Cases [UnitSort]** Here \( t = 1 \), so applying \( \Gamma \) to \( t \) does not change it: \( t = [\Gamma]t \). Since \( \Gamma \vdash t : \kappa \), we have \( \Gamma \vdash [\Gamma]t : \kappa \), which was to be shown.

- **Case [VarSort]** If \( t \) is an existential variable \( \alpha \), then \( \Gamma = \Gamma_0[\alpha] \), so applying \( \Gamma \) to \( t \) does not change it, and we proceed as in the [UnitSort] case above.

If \( t \) is a universal variable \( \alpha \) and \( \Gamma \) has no equation for it, then proceed as in the [UnitSort] case.

Otherwise, \( t = \alpha \) and \( (\alpha = \tau) \in \Gamma \):

\[ \Gamma = (\Gamma_L, \alpha : \kappa, \Gamma_M, \alpha = \tau, \Gamma_R) \]

By the implicit assumption that \( \Gamma \) is well-formed, \( \Gamma_L, \alpha : \kappa, \Gamma_M \vdash \tau : \kappa \).

By Lemma [34] (Suffix Weakening), \( \Gamma \vdash \tau : \kappa \). Since \( |\Gamma \vdash \alpha| < |\Gamma \vdash \alpha| \), we can apply the i.h., giving

\[ \Gamma \vdash [\Gamma]t : \kappa \]

By the definition of substitution, \( [\Gamma]t = [\Gamma]\alpha \), so we have \( \Gamma \vdash [\Gamma]\alpha : \kappa \).
Proof of Lemma 11 (Right-Hand Substitution for Sorting). If $\Gamma \vdash \text{prop}$ then $\Gamma \vdash [\Gamma] \text{prop}$.

Proof. Use inversion (EqProp), apply Lemma 11 (Right-Hand Substitution for Sorting) to each premise, and apply EqProp again.

Lemma 13 (Right-Hand Substitution for Typing). If $\Gamma \vdash A$ a type then $\Gamma \vdash [\Gamma] A$ a type.

Proof. By induction on $|\Gamma \vdash A|$ (the size of $A$ under $\Gamma$).

Several cases correspond to cases in the proof of Lemma 11 (Right-Hand Substitution for Sorting):

- the case for UnitWF is like the case for UnitSort.
- the case for SolvedVarSort is like the cases for VarWF and SolvedVarWF.
- the case for VarSort is like the case for VarWF but in the last subcase, apply Lemma 10 (Sorting Implies Typing) to move from a sorting judgment to a typing judgment.
- the case for BinWF is like the case for BinSort.

Now, the new cases:

- Case ForallWF: In this case $A = \forall \alpha : \kappa. A_0$. By i.h., $\Gamma, \alpha : \kappa \vdash [\Gamma, \alpha : \kappa] A_0$ type. By the definition of substitution, $[\Gamma, \alpha : \kappa] A_0 = [\Gamma] A_0$, so by ForallWF $\Gamma \vdash \forall \alpha. [\Gamma] A_0$ type, which by the definition of substitution is $\Gamma \vdash [\Gamma] (\forall \alpha. A_0)$ type.
- Case ExistsWF: Similar to the ForallWF case.
- Case ImpliesWF, WithWF: Use the i.h. and Lemma 12 (Right-Hand Substitution for Propositions), then apply ImpliesWF or WithWF.

Lemma 14 (Substitution for Sorting). If $\Omega \vdash t : \kappa$ then $[\Omega] \Omega \vdash [\Omega] t : \kappa$.

Proof. By induction on $|\Omega \vdash t|$ (the size of $t$ under $\Omega$).

- Case $u : \kappa \in \Omega \vdash u : \kappa$ VarSort

We have a complete context $\Omega$, so $u$ cannot be an existential variable: it must be some universal variable $\alpha$.

If $\Omega$ lacks an equation for $\alpha$, use Lemma 9 (Uvar Preservation) and apply rule UvarSort.

Otherwise, $(\alpha = \tau \in \Omega)$, so we need to show $\Omega \vdash [\Omega] \tau : \kappa$. By the implicit assumption that $\Omega$ is well-formed, plus Lemma 34 (Suffix Weakening), $\Omega \vdash \tau : \kappa$. By Lemma 11 (Right-Hand Substitution for Sorting), $\Omega \vdash [\Omega] \tau : \kappa$.  

---

Proof of Lemma 14 (Substitution for Sorting)  lem:completion-sort
Proof of Lemma 14 (Substitution for Sorting).

\[ \text{lem:completion-sort} \]

\[ \text{43} \]

**Proof.** Only one rule derives this judgment form:

\[ \Omega \vdash t : N \]

\[ \Omega \vdash t' : N \]

\[ \Omega \vdash t = t' \text{ prop} \]

\[ \text{EqProp} \]

\[ \text{Subderivation} \]

\[ [\Omega] \Omega \vdash [\Omega] t : N \]

\[ [\Omega] \Omega \vdash [\Omega] t' : N \]

\[ [\Omega] \Omega \vdash ([\Omega] t) = ([\Omega] t') \]

\[ \text{EqProp} \]

\[ \text{Subderivation} \]

\[ \text{By def. of subst.} \]

Lemma 15 (Substitution for Prop Well-Formedness).

If \( \Omega \vdash P \text{ prop} \) then \( [\Omega] \Omega \vdash [\Omega] P \text{ prop} \).

**Proof.** Only one rule derives this judgment form:

\[ \text{Case} \]

\[ \overrightarrow{\alpha} : \kappa = \tau \in \Omega \]

\[ \Omega \vdash \overrightarrow{\alpha} : \kappa \]

\[ \text{SolvedVarSort} \]

\[ \text{Subderivation} \]

\[ \Omega = (\Omega_L, \overrightarrow{\alpha} : \kappa, \Omega_R) \]

Decomposing \( \Omega \)

\[ \Omega_L \vdash \tau : \kappa \]

By implicit assumption that \( \Omega \) is well-formed

\[ \Omega_L, \overrightarrow{\alpha} : \kappa = \tau, \Omega_R \vdash \tau : \kappa \]

By Lemma 34 (Suffix Weakening)

\[ \Omega \vdash [\Omega] \tau : \kappa \]

By Lemma 11 (Right-Hand Substitution for Sorting)

\[ = \]

\[ [\Omega] \Omega \vdash [\Omega] \alpha : \kappa \]

\[ [\Omega] \tau = [\Omega] \alpha \]

\[ \text{Case} \]

\[ \Omega \vdash 1 : \star \]

\[ \text{UnitSort} \]

Since \( 1 = [\Omega] 1 \), applying \( \text{UnitSort} \) gives the result.

\[ \text{Case} \]

\[ \Omega \vdash \tau_1 : \star \]

\[ \Omega \vdash \tau_2 : \star \]

\[ \Omega \vdash \tau_1 \oplus \tau_2 : \star \]

\[ \text{BinSort} \]

By i.h. on each premise, rule \( \text{BinSort} \), and the definition of substitution.

\[ \text{Case} \]

\[ \Omega \vdash \text{zero} : N \]

\[ \text{ZeroSort} \]

Since \( \text{zero} = [\Omega] \text{zero} \), applying \( \text{ZeroSort} \) gives the result.

\[ \text{Case} \]

\[ \Omega \vdash \text{succ} (t) : N \]

\[ \text{SuccSort} \]

By i.h., rule \( \text{SuccSort} \), and the definition of substitution.

\[ \square \]

Lemma 16 (Substitution for Type Well-Formedness).

If \( \Omega \vdash A \text{ type} \) then \( [\Omega] \Omega \vdash [\Omega] A \text{ type} \).

**Proof.** By induction on \( |\Omega| \vdash A \).

Several cases correspond to those in the proof of Lemma 14 (Substitution for Sorting):

- the \( \text{UnitWF} \) case is like the \( \text{UnitSort} \) case (using \( \text{DeclUnitWF} \) instead of \( \text{UnitSort} \)).
Proof of Lemma 16 (Substitution for Type Well-Formedness)

- the \texttt{VarWF} case is like the \texttt{VarSort} case (using \texttt{DeclUvarWF} instead of \texttt{UvarSort});
- the \texttt{SolvedVarWF} case is like the \texttt{SolvedVarSort} case.

However, uses of Lemma 11 (Right-Hand Substitution for Sorting) are replaced by uses of Lemma 13 (Right-Hand Substitution for Typing).

Now, the new cases:

- Case \( \Omega, \alpha : \kappa \vdash A \) type 
  \[
  \Omega \vdash \forall \alpha : \kappa. A \quad \text{\texttt{ForallWF}}
  \]
  Subderivation
  \[
  [\Omega, \alpha : \kappa]([\Omega] A) : \kappa' 
  \]
  By \texttt{i.h.}
  \[
  [\Omega] \Omega \vdash \forall \alpha : \kappa. [\Omega] A 
  \]
  By definition of completion
  \[
  [\Omega] \Omega \vdash [\Omega] (\forall \alpha : \kappa. A) 
  \]
  By \texttt{DeclAllWF}

- Case \texttt{ExistsWF}: Similar to the \texttt{ForallWF} case, using \texttt{DeclExistsWF} instead of \texttt{DeclAllWF}.

- Case \( \Omega \vdash A_1 \) type \( \Omega \vdash A_2 \) type
  \[
  \Omega \vdash A_1 \oplus A_2 \quad \text{\texttt{BinWF}}
  \]
  By \texttt{i.h.} on each premise, rule \texttt{DeclBinWF} and the definition of substitution.

- Case \texttt{VecWF}: Similar to the \texttt{BinWF} case.

- Case \( \Omega \vdash P \) prop \( \Omega \vdash A_0 \) type
  \[
  \Omega \vdash P \text{\texttt{\&\&}} A_0 \quad \text{\texttt{WithWF}}
  \]
  Similar to the \texttt{ImplesWF} case.

\[\]

Lemma 17 (Substitution Stability).

\( \Omega, \Omega_Z \) is well-formed and \( \Omega_Z \) is soft and \( \Omega \vdash A \) type then \( [\Omega] A = [\Omega, \Omega_Z] A \).

\textbf{Proof.} By induction on \( \Omega_Z \).

Since \( \Omega_Z \) is soft, either (1) \( \Omega_Z = \emptyset \) (and the result is immediate) or (2) \( \Omega_Z = (\Omega', \alpha : \kappa) \) or (3) \( \Omega_Z = (\Omega', \alpha : \kappa = t) \). However, according to the grammar for complete contexts such as \( \Omega_Z \), (2) is impossible. Only case (3) remains.

By \texttt{i.h.}, \( [\Omega] A = [\Omega, \Omega'] A \). Use the fact that \( \Omega \vdash A \) type implies \( \text{FV}(A) \cap \text{dom}(\Omega_Z) = \emptyset \).

Lemma 18 (Equal Domains).

If \( \Omega_1 \vdash A \) type and \( \text{dom}(\Omega_1) = \text{dom}(\Omega_2) \) then \( \Omega_2 \vdash A \) type.

\textbf{Proof.} By induction on the given derivation.
Properties of Extension

Lemma 19 (Declaration Preservation). If $\Gamma \rightarrow \Delta$ and $u$ is declared in $\Gamma$, then $u$ is declared in $\Delta$.

Proof. By induction on the derivation of $\Gamma \rightarrow \Delta$.

- **Case** $\cdot \rightarrow \cdot \rightarrow \text{id}$
  
  This case is impossible, since by hypothesis $u$ is declared in $\Gamma$.

- **Case** $\Gamma \rightarrow \Delta$ 
  $[\Delta]A = [\Delta]A'$
  $\Gamma, x : A \rightarrow \Delta, x : A' \rightarrow \text{Var}$

  - Case $u = x$: Immediate.
  - Case $u \neq x$: Since $u$ is declared in $(\Gamma, x : A)$, it is declared in $\Gamma$. By i.h., $u$ is declared in $\Delta$, and therefore declared in $(\Delta, x : A')$.

- **Case** $\Gamma \rightarrow \Delta$
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Uvar}$

  Similar to the $\rightarrow \text{Var}$ case.

- **Case** $\Gamma \rightarrow \Delta$
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Unsolved}$

  Similar to the $\rightarrow \text{Var}$ case.

- **Case** $\Gamma \rightarrow \Delta$
  $[\Delta]t = [\Delta]t'$
  $\Gamma, \alpha : \kappa = t \rightarrow \Delta, \alpha : \kappa = t' \rightarrow \text{Solved}$

  Similar to the $\rightarrow \text{Var}$ case.

- **Case** $\Gamma \rightarrow \Delta$
  $[\Delta]t = [\Delta]t'$
  $\Gamma, \alpha = t \rightarrow \Delta, \alpha = t' \rightarrow \text{Eqn}$

  It is given that $u$ is declared in $(\Gamma, \alpha = t)$. Since $\alpha = t$ is not a declaration, $u$ is declared in $\Gamma$. By i.h., $u$ is declared in $\Delta$, and therefore declared in $(\Delta, \alpha = t')$.

- **Case** $\Gamma \rightarrow \Delta$
  $\Gamma, \triangleright \alpha \rightarrow \Delta, \triangleright \alpha \rightarrow \text{Marker}$

  Similar to the $\rightarrow \text{Eqn}$ case.

- **Case** $\Gamma \rightarrow \Delta$
  $\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' = t \rightarrow \text{Solve}$

  Similar to the $\rightarrow \text{Var}$ case.
Proof of Lemma 19 (Declaration Preservation)

• Case $\Gamma \rightarrow \Delta$
  $\Gamma \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Add}$

  It is given that $u$ is declared in $\Gamma$. By i.h., $u$ is declared in $\Delta$, and therefore declared in $(\Delta, \alpha : \kappa)$.

• Case $\Gamma \rightarrow \Delta$
  $\Gamma \rightarrow \Delta, \alpha : \kappa = t \rightarrow \text{AddSolved}$

  Similar to the $\rightarrow \text{Add}$ case.

\[ \Box \]

Lemma 20 (Declaration Order Preservation). If $\Gamma \rightarrow \Delta$ and $u$ is declared to the left of $v$ in $\Gamma$, then $u$ is declared to the left of $v$ in $\Delta$.

Proof. By induction on the derivation of $\Gamma \rightarrow \Delta$.

• Case $\cdots \rightarrow \cdots \rightarrow \text{Id}$

  This case is impossible, since by hypothesis $u$ and $v$ are declared in $\Gamma$.

• Case $\Gamma \rightarrow \Delta$
  $[\Delta]A = [\Delta]A' \rightarrow \text{Var}$

  Consider whether $v = x$:
  
  – Case $v = x$:
    
    It is given that $u$ is declared to the left of $v$ in $(\Gamma, x : A)$, so $u$ is declared in $\Gamma$. By Lemma 19 (Declaration Preservation), $u$ is declared in $\Delta$. Therefore $u$ is declared to the left of $v$ in $(\Delta, x : A')$.
  
  – Case $v \neq x$:
    
    Here, $v$ is declared in $\Gamma$. By i.h., $u$ is declared to the left of $v$ in $\Delta$. Therefore $u$ is declared to the left of $v$ in $(\Delta, x : A')$.

• Case $\Gamma \rightarrow \Delta$
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Uvar}$

  Similar to the $\rightarrow \text{Var}$ case.

• Case $\Gamma \rightarrow \Delta$
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Unsolved}$

  Similar to the $\rightarrow \text{Var}$ case.

• Case $\Gamma \rightarrow \Delta$
  $[\Delta]t = [\Delta]t' \rightarrow \text{Solved}$

  Similar to the $\rightarrow \text{Var}$ case.

• Case $\Gamma \rightarrow \Delta$
  $\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' = t \rightarrow \text{Solve}$

  Similar to the $\rightarrow \text{Var}$ case.
Proof of Lemma 20 (Declaration Order Preservation). If \( \Gamma \rightarrow \Delta \) and \( u \) and \( v \) are both declared in \( \Gamma \) and \( u \) is declared to the left of \( v \) in \( \Delta \), then \( u \) is declared to the left of \( v \) in \( \Gamma \).

Proof. It is given that \( u \) and \( v \) are declared in \( \Gamma \). Either \( u \) is declared to the left of \( v \) in \( \Gamma \), or \( v \) is declared to the left of \( u \). Suppose the latter (for a contradiction). By Lemma 20 (Declaration Order Preservation), \( v \) is declared to the left of the left of \( u \) in \( \Delta \). But we know that \( u \) is declared to the left of \( v \) in \( \Delta \): contradiction. Therefore \( u \) is declared to the left of \( v \) in \( \Gamma \).

Lemma 21 (Reverse Declaration Order Preservation). If \( \Gamma \rightarrow \Delta \) and \( u \) and \( v \) are both declared in \( \Gamma \) and \( u \) is declared to the left of \( v \) in \( \sigma \), then \( u \) is declared to the left of \( v \) in \( \sigma \).

Proof. It is given that \( u \) and \( v \) are declared in \( \Gamma \). Either \( u \) is declared to the left of \( v \) in \( \Gamma \), or \( v \) is declared to the left of \( u \). Suppose the latter (for a contradiction). By Lemma 20 (Declaration Order Preservation), \( v \) is declared to the left of the left of \( u \) in \( \Delta \). But we know that \( u \) is declared to the left of \( v \) in \( \Delta \): contradiction. Therefore \( u \) is declared to the left of \( v \) in \( \Gamma \).
(v) If $\mathcal{D} :: \Gamma_0, x : A, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0, A', \text{ and } \Delta_1$
such that $\Delta = (\Delta_0, x : A', \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]A = [\Delta_0]A'$ where $\mathcal{D}' < \mathcal{D}$.

Moreover, if $\Gamma_1$ is soft, then $\Delta_1$ is soft.

Moreover, if $\text{dom}(\Gamma_0, x : A, \Gamma_1) = \text{dom}(\Delta)$ then $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$.

(vi) If $\mathcal{D} :: \Gamma_0, \hat{x} : \kappa, \Gamma_1 \rightarrow \Delta$ then either

• there exist unique $\Delta_0, \tau$, and $\Delta_1$
such that $\Delta = (\Delta_0, \hat{x} : \kappa = \tau', \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \rightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$,
or
• there exist unique $\Delta_0$ and $\Delta_1$
such that $\Delta = (\Delta_0, \hat{x} : \kappa, \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \rightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$.

Proof. In each part, we proceed by induction on the derivation of $\Gamma_0, \ldots, \Gamma_1 \rightarrow \Delta$.

Note that in each part, the $\rightarrow \text{id}$ case is impossible.

Throughout this proof, we shadow $\Delta$ so that it refers to the largest proper prefix of the $\Delta$ in the statement of the lemma. For example, in the $\rightarrow \text{Var}$ case of part (i), we really have $\Delta = (\Delta_{00}, x : A')$, but we call $\Delta_{00}$ “$\Delta$”.

(i) We have $\Gamma_0, \alpha : \kappa, \Gamma_1 \rightarrow \Delta$.

• Case $\Gamma \rightarrow \Delta$

$\Gamma, x : A \rightarrow \Delta, x : A' \rightarrow \text{Var}$

$\Gamma_0, \alpha : \kappa, \Gamma_1 \rightarrow \Delta$

Given

$\Gamma, x : A = (\Gamma_0, \alpha : \kappa, \Gamma_1)$

$\Gamma_0, \alpha : \kappa, \Gamma_1, x : A \rightarrow A$

Since the last element must be equal

$\Gamma, x : A = (\Gamma_0, \alpha : \kappa, \Gamma_0', x : A)$

By transitivity

$\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_0')$

By injectivity of syntax

$\Gamma \rightarrow \Delta$

Subderivation

$\Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta$

By equality

$\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$

By i.h.

$\Gamma_0 \rightarrow \Delta_0$

"$

\Gamma_1' \rightarrow \text{soft then } \Delta_1 \text{ soft}$

"$
\rightarrow \text{Var}$

$(\Delta, x : A') = (\Delta_0, \alpha : \kappa, \Delta_1, x : B)$

By congruence

if $\Gamma_1', x : A$ soft then $\Delta_1, x : A'$ soft Since $\Gamma_1', x : A$ is not soft

• Case $\Gamma \rightarrow \Delta$

$\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' \rightarrow \text{Var}$

$\Gamma_0, \alpha : \kappa, \Gamma_1 \rightarrow \Delta$

There are two cases:

- Case $\alpha : \kappa = \beta : \kappa'$:

  $\Gamma, \alpha : \kappa = (\Gamma_0, \alpha : \kappa, \Gamma_1)$ where $\Gamma_0 = \Gamma$ and $\Gamma_1 = \cdot$

  $\Delta, \alpha : \kappa = (\Delta_0, \alpha : \kappa, \Delta_1)$ where $\Delta_0 = \Delta$ and $\Delta_1 = \cdot$

  if $\Gamma_1$ soft then $\Delta_1$ soft since $\cdot$ is soft

Proof of Lemma 22 [Extension Inversion] lem:extension-inversion
Proof of Lemma 22 (Extension Inversion) \( \text{lem:extension-inversion} \)

- **Case** \( \alpha \neq \beta \):

  \[
  (\Gamma, \beta : \kappa') = (\Gamma_0, \alpha : \kappa, \Gamma_1) \quad \text{Given}
  
  = (\Gamma_0, \alpha : \kappa, \Gamma_1', \beta : \kappa') \quad \text{Since the last element must be equal}
  
  \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') \quad \text{By injectivity of syntax}
  
  \Gamma \rightarrow \Delta \quad \text{Subderivation}
  
  \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta \quad \text{By equality}
  
  \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \quad \text{By i.h.}
  
  \Gamma_0 \rightarrow \Delta_0 \quad \"'
  
  if \( \Gamma_1' \) soft then \( \Delta_1 \) soft \( \"' \)

  \( (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') \quad \text{By congruence} \)
  
  if \( \Gamma_1', \beta : \kappa' \) soft then \( \Delta_1, \beta : \kappa' \) soft \quad \text{Since \( \Gamma_1', \beta : \kappa' \) is not soft}

- **Case** \( \Gamma \rightarrow \Delta \)

  \[
  \begin{array}{c}
  (\Gamma, \hat{\alpha} : \kappa') = (\Gamma_0, \alpha : \kappa, \Gamma_1) \\
  \Gamma_0, \alpha : \kappa, \Gamma_1 \quad \text{Given}
  
  = (\Gamma_0, \alpha : \kappa, \Gamma_1', \hat{\alpha} : \kappa') \quad \text{Since the last element must be equal}
  
  \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') \quad \text{By injectivity of syntax}
  
  \Gamma \rightarrow \Delta \quad \text{Subderivation}
  
  \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta \quad \text{By equality}
  
  \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \quad \text{By i.h.}
  
  \Gamma_0 \rightarrow \Delta_0 \quad \"'
  
  if \( \Gamma_1' \) soft then \( \Delta_1 \) soft \( \"' \)

  \( (\Delta, \hat{\alpha} : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \hat{\alpha} : \kappa') \quad \text{By congruence} \)

  Suppose \( \Gamma_1', \hat{\alpha} : \kappa' \) soft.

  \( \Gamma_1' \) soft \quad \text{By definition of softness}
  
  \( \Delta_1 \) soft \quad \text{By induction}
  
  \( \Delta_1 \) soft \quad \text{By definition of softness}
  
  if \( \Gamma_1', \hat{\alpha} : \kappa' \) soft then \( \Delta_1, \hat{\alpha} : \kappa' \) soft \quad \text{Implication introduction}

- **Case** \( \Gamma \rightarrow \Delta \)

  \[
  [\Delta] t = [\Delta] t' \quad \quad \quad \quad \quad \text{Solved}
  
  \begin{array}{c}
  (\Gamma, \hat{\alpha} : \kappa = t) \rightarrow \Delta, \hat{\alpha} : \kappa = t' \\
  \Gamma_0, \alpha : \kappa, \Gamma_1 \quad \text{Similar to the\textcolor{red}{\quad Unsolved} case.} \end{array}
  
  \text{[\Delta] t = [\Delta] t'} \quad \text{Eqn}

  \begin{array}{c}
  (\Gamma, \beta = t) \rightarrow \Delta, \beta = t' \\
  \Gamma_0, \alpha : \kappa, \Gamma_1 \quad \text{[\Delta] t = [\Delta] t'} \end{array}
Proof of Lemma 22 (Extension Inversion)

\[(\Gamma, \beta = t) = (\Gamma_0, \alpha : \kappa, \Gamma_1)\quad \text{Given}\]

\[= (\Gamma_0, \alpha : \kappa, \Gamma_1', \beta = t)\quad \text{Since the last element must be equal}\]

\[\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1')\quad \text{By injectivity of syntax}\]

\[\Gamma \rightarrow \Delta\quad \text{Subderivation}\]

\[\Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta\quad \text{By equality}\]

\[\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)\quad \text{By i.h.}\]

\[\Rightarrow\quad \Gamma_0 \rightarrow \Delta_0\quad "\]

\[\Rightarrow\quad \text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft}\quad "\]

\[\Rightarrow\quad (\Delta, \beta = t') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta = t'\quad \text{By congruence}\]

\[\Rightarrow\quad \text{if } \Gamma_1', \beta = t \text{ soft then } \Delta_1, \beta = t' \text{ soft}\quad \text{Since } \Gamma_1', \beta = t \text{ is not soft}\]

**Case**

\[
\begin{array}{c}
\Gamma \rightarrow \Delta \\
\Gamma_0, \alpha : \kappa, \Gamma_1
\end{array}
\]

\[\Rightarrow\quad \Delta_1, \Gamma_{\alpha} \quad \text{Marker}\]

\[\Rightarrow\quad (\Gamma_{\alpha}) = (\Gamma_0, \alpha : \kappa, \Gamma_1)\quad \text{Given}\]

\[= (\Gamma_0, \alpha : \kappa, \Gamma_1', \Gamma_{\alpha})\quad \text{Since the last element must be equal}\]

\[\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1')\quad \text{By injectivity of syntax}\]

\[\Gamma \rightarrow \Delta\quad \text{Subderivation}\]

\[\Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta\quad \text{By equality}\]

\[\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)\quad \text{By i.h.}\]

\[\Rightarrow\quad \Gamma_0 \rightarrow \Delta_0\quad "\]

\[\Rightarrow\quad \text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft}\quad "\]

\[\Rightarrow\quad (\Delta, \Gamma_{\alpha}) = (\Delta_0, \alpha : \kappa, \Delta_1, \Gamma_{\alpha})\quad \text{By congruence}\]

\[\Rightarrow\quad \text{if } \Gamma_1', \Gamma_{\alpha} \text{ soft then } \Delta_1, \Gamma_{\alpha} \text{ soft}\quad \text{Since } \Gamma_1', \Gamma_{\alpha} \text{ is not soft}\]

**Case**

\[
\begin{array}{c}
\Gamma \rightarrow \Delta \\
\Gamma_0, \alpha : \kappa, \Gamma_1
\end{array}
\]

\[\Rightarrow\quad \Delta_1, \alpha : \kappa, \Gamma_{\alpha} \quad \text{Add}\]

\[\Rightarrow\quad (\Delta_1, \alpha : \kappa, \Delta_1, \Gamma_{\alpha})\quad \text{By i.h.}\]

\[\Rightarrow\quad \text{if } \Gamma_1 \text{ soft then } \Delta_1, \Gamma_{\alpha} \text{ soft}\quad "\]

\[\Rightarrow\quad (\Delta_1, \alpha : \kappa, \Gamma_{\alpha}) = (\Delta_0, \alpha : \kappa, \Delta_1, \alpha : \kappa, \Gamma_{\alpha})\quad \text{By congruence of equality}\]

Suppose \(\Gamma_1\) soft.

\[\Delta_1 \text{ soft} \quad \text{By i.h.}\]

\[\Delta_1, \alpha : \kappa' \quad \text{soft} \quad \text{By definition of softness}\]

\[\Rightarrow\quad \text{if } \Gamma_1 \text{ soft then } \Delta_1, \alpha : \kappa' \text{ soft}\quad \text{Implication introduction}\]

**Case**

\[
\begin{array}{c}
\Gamma \rightarrow \Delta \\
\Gamma_0, \alpha : \kappa, \Gamma_1
\end{array}
\]

\[\Rightarrow\quad \Delta_1, \alpha : \kappa' = t \quad \text{AddSolved}\]
Proof of Lemma 22 (Extension Inversion) lem:extension-inversion

\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]  
By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]  
"  
\[ \text{if } \Gamma_1 \text{ soft then } \Delta_1 \text{ soft} \]  
"  
\[ (\Delta, \hat{\alpha} : \kappa' = t) = (\Delta_0, \alpha : \kappa, \Delta_1, \hat{\alpha} : \kappa') \]  
By congruence of equality

Suppose \( \Gamma_1 \) soft.
\( \Delta_1 \) soft  
By i.h.
\( (\Delta_1, \hat{\alpha} : \kappa' = t) \) soft  
By definition of softness
\[ \text{if } \Gamma_1 \text{ soft then } \Delta_1, \hat{\alpha} : \kappa' = t \text{ soft} \]  
Implication introduction

- **Case**  
\[ \Gamma \rightarrow \Delta \]
\[ \Gamma', \hat{\beta} : \kappa' \rightarrow \Delta, \hat{\beta} : \kappa' = t \]
\[ \Rightarrow \text{Solve} \]
\[ (\Gamma, \hat{\beta} : \kappa') = (\Gamma_0, \alpha : \kappa, \Gamma_1) \]
Given
\[ = (\Gamma_0, \alpha : \kappa, \Gamma_1', \hat{\beta} : \kappa') \]  
Since the final elements are equal
\[ \Gamma_0 \rightarrow \Delta_0 \]  
By injectivity of context syntax
\[ \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta \]  
Subderivation
\[ \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Delta \]  
By equality
\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]  
By i.h.
\[ \Gamma_0 \rightarrow \Delta_0 \]  
"  
\[ \text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft} \]  
"  
\[ \Delta, \hat{\beta} : \kappa' = \Delta_0, \alpha : \kappa, \Delta_1, \hat{\beta} \]  
By congruence

Suppose \( \Gamma_1', \hat{\beta} : \kappa' \) soft.
\( \Gamma_1' \) soft  
By definition of softness
\( \Delta_1 \) soft  
Using i.h.
\[ \Delta_1, \hat{\beta} : \kappa' = t \text{ soft} \]  
By definition of softness
\[ \text{if } \Gamma_1', \hat{\beta} : \kappa' \text{ soft then } \Delta_1, \hat{\beta} : \kappa' = t \text{ soft} \]  
Implication intro

(ii) We have \( \Gamma_0, \tau' \rightarrow \Delta \). This part is similar to part (i) above, except for “if \( \text{dom}(\Gamma_0, \tau, \Gamma_1) = \text{dom}(\Delta) \) then \( \text{dom}(\Gamma_0) = \text{dom}(\Delta_0) \)”, which follows by i.h. in most cases. In the Marker case, either we have \( \ldots, \tau' \) where \( \tau' = u \) in which case the i.h. gives us what we need—or we have a matching \( \tau_u \). In this latter case, we have \( \Gamma_1 = \cdot \). We know that \( \text{dom}(\Gamma_0, \tau, \Gamma_1) = \text{dom}(\Delta) \) and \( \Delta = (\Delta_0, \tau_u) \). Since \( \Gamma_1 = \cdot \), we have \( \text{dom}(\Gamma_0, \tau_u) = \text{dom}(\Delta_0, \tau_u) \). Therefore \( \text{dom}(\Gamma_0) = \text{dom}(\Delta_0) \).

(iii) We have \( \Gamma_0, \alpha = \tau, \Gamma_1 \rightarrow \Delta \).

- **Case**  
\[ \Gamma \rightarrow \Delta \]
\[ \Gamma_0, \alpha = \tau, \Gamma_1 \rightarrow \Delta, \hat{\beta} : \kappa' \]
\[ \Rightarrow \text{Uvar} \]
\[ (\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \hat{\beta} : \kappa') \]
Given
\[ = (\Gamma_0, \alpha = \tau, \Gamma_1', \hat{\beta} : \kappa') \]  
Since the final elements must be equal
\[ \Gamma = (\Gamma_0, \alpha = \tau, \Gamma_1') \]  
By injectivity of context syntax
\[ \Delta = (\Delta_0, \alpha = \tau', \Delta_1) \]  
By i.h.
\[ [\Delta_0] \tau = [\Delta_0] \tau' \]  
"  
\[ \Gamma_0 \rightarrow \Delta_0 \]  
"  
\[ (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \alpha = \tau', \Delta_1, \hat{\beta}) \]  
By congruence of equality
• Case \( \Gamma \rightarrow \Delta \)
  \[ [\Delta]A = [\Delta]A' \rightarrow \text{Var} \]
  \[ \Gamma, \alpha = \tau, \Gamma_1 \]
  Similar to the \(\rightarrow \text{Uvar}\) case.

• Case \( \Gamma \rightarrow \Delta \)
  \[ \Gamma \vdash \Delta, \chi : A \rightarrow \Delta, x : A' \rightarrow \text{Marker} \]
  Similar to the \(\rightarrow \text{Uvar}\) case.

• Case \( \Gamma \rightarrow \Delta \)
  \[ \Gamma, \alpha : \kappa' \rightarrow \Delta, \alpha : \kappa' \rightarrow \text{Unsolved} \]
  Similar to the \(\rightarrow \text{Uvar}\) case.

• Case \( \Gamma \rightarrow \Delta \)
  \[ [\Delta]t = [\Delta]t' \rightarrow \text{Solved} \]
  \[ \Gamma, \alpha = \tau, \Gamma_1 \]
  Similar to the \(\rightarrow \text{Uvar}\) case.

• Case \( \Gamma \rightarrow \Delta \)
  \[ \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' \rightarrow \text{Solve} \]

• Case \( \Gamma \rightarrow \Delta \)
  \[ [\Delta]t = [\Delta]t' \rightarrow \text{Eqn} \]
  \[ \Gamma, \alpha = \tau, \Gamma_1 \]
  There are two cases:
  
  – Case \( \alpha = \beta \):
    \[ t = t \] and \( \Gamma_1 = \cdot \) and \( \Gamma_0 \)
    By injectivity of syntax
    
    \[ \Gamma_0 \rightarrow \Delta_0 \]
    Subderivation \( \Gamma_0 = \Gamma \) and let \( \Delta_0 = \Delta \)
    
    \[ (\Delta, \alpha = t') = (\Delta_0, \alpha = t', \Delta_1) \]
    where \( \Delta_1 = \cdot \)
    
    \[ [\Delta_0]t = [\Delta_0]t' \]
    By premise \( [\Delta]t = [\Delta]t' \)
    
  – Case \( \alpha \neq \beta \):
    \[ (\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \beta = t) \]
    Given
    \[ = (\Gamma_0, \alpha = \tau, \Gamma_1, \beta = t) \]
    Since the final elements must be equal
    \[ \Gamma = (\Gamma_0, \alpha = \tau, \Gamma_1) \]
    By injectivity of context syntax
    
    \[ \Delta = (\Delta_0, \alpha = \tau, \Delta_1) \]
    By i.h.
    
    \[ [\Delta_0]t = [\Delta_0]t' \]
    “”
    
    \[ \Gamma_0 \rightarrow \Delta_0 \]
    “”
    
    \[ (\Delta, \beta = t') = (\Delta_0, \alpha = \tau', \Delta_1, \beta = t) \]
    By congruence of equality
  
• Case \( \Gamma \rightarrow \Delta \)
  \[ \Gamma, \alpha : \kappa' \rightarrow \Delta, \alpha : \kappa' \rightarrow \text{Add} \]
  \[ \Gamma_0, \alpha = \tau, \Gamma_1 \]
Proof of Lemma 22 (Extension Inversion) \( \text{lem:extension-inversion} \)

\[
\Delta = (\Delta_0, \alpha = \tau', \Delta_1) \quad \text{By i.h.}
\]

\[\Delta_0 \tau = [\Delta_0] \tau' \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha = \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

- Case \( \Gamma \rightarrow \Delta \)

\[
\begin{align*}
\Gamma, \alpha : \kappa' & \rightarrow \Delta, \alpha : \kappa' = t & \rightarrow \text{AddSolved} \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0
\end{align*}
\]

\[
\begin{align*}
\Delta = (\Delta_0, \alpha = \tau', \Delta_1) & \quad \text{By i.h.} \\
\Delta_0 \tau = [\Delta_0] \tau' & \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha = \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

(iv) We have \( \Gamma_0, \alpha : \kappa = \tau, \Gamma_1 \rightarrow \Delta \).

- Case \( \Gamma \rightarrow \Delta \)

\[
\begin{align*}
\Gamma, \beta : \kappa' & \rightarrow \Delta, \beta : \kappa' & \rightarrow \text{Uvar} \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \quad \text{Given} \\
\Gamma = (\Gamma_0, \alpha : \tau, \Gamma_1') & \quad \text{Since the final elements must be equal} \\
\Delta = (\Delta_0, \alpha : \tau, \Delta_1) & \quad \text{By injectivity of context syntax} \\
\Delta_0 \tau = [\Delta_0] \tau' & \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha : \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

- Case \( \Gamma \rightarrow \Delta \)

\[
\begin{align*}
\Gamma, \beta : \kappa' & \rightarrow \Delta, \beta : \kappa' & \rightarrow \text{Var} \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \quad \text{Given} \\
\Delta = (\Delta_0, \alpha : \tau, \Delta_1) & \quad \text{By injectivity of context syntax} \\
\Delta_0 \tau = [\Delta_0] \tau' & \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha : \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

- Case \( \Gamma \rightarrow \Delta \)

\[
\begin{align*}
\Gamma, \beta : \kappa' & \rightarrow \Delta, \beta : \kappa' & \rightarrow \text{Marker} \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \quad \text{Given} \\
\Delta = (\Delta_0, \alpha : \tau, \Delta_1) & \quad \text{By injectivity of context syntax} \\
\Delta_0 \tau = [\Delta_0] \tau' & \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha : \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

- Case \( \Gamma \rightarrow \Delta \)

\[
\begin{align*}
\Gamma, \beta : \kappa' & \rightarrow \Delta, \beta : \kappa' & \rightarrow \text{Unsolved} \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \quad \text{Given} \\
\Delta = (\Delta_0, \alpha : \tau, \Delta_1) & \quad \text{By injectivity of context syntax} \\
\Delta_0 \tau = [\Delta_0] \tau' & \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha : \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

- Case \( \Gamma \rightarrow \Delta \)

\[
\begin{align*}
\Gamma, \beta : \kappa' & \rightarrow \Delta, \beta : \kappa' & \rightarrow \text{Solved} \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \\
\Gamma_0, \alpha : \tau & \rightarrow \Delta_0 \quad \text{Given} \\
\Delta = (\Delta_0, \alpha : \tau, \Delta_1) & \quad \text{By injectivity of context syntax} \\
\Delta_0 \tau = [\Delta_0] \tau' & \quad "\]

\[\Gamma_0 \rightarrow \Delta_0 \quad "\]

\[\Delta_0 \alpha : \kappa' = (\Delta_0, \alpha : \tau', \Delta_1, \alpha : \kappa') \quad \text{congruence of equality}\]

There are two cases.

- Case \( \hat{\alpha} = \hat{\beta} \):
Proof of Lemma 22 (Extension Inversion)

\[ \kappa' = \kappa \text{ and } t = \tau \text{ and } \Gamma_1 = \cdot \text{ and } \Gamma \Rightarrow \text{ injectivity of syntax} \]

- Case \( \hat{\alpha} \neq \hat{\beta} \):
  \[ (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) = (\Gamma, \hat{\beta} = t) \]
  Given
  \[ = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \hat{\beta} = t) \]
  Since the final elements must be equal
  \[ \Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') \]
  By injectivity of context syntax
  \[ \Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \]
  By i.h.

- Case \( \Gamma \rightarrow \Delta \):
  \[ [\Delta]t = [\Delta]t' \]
  From premise \([\Delta]t = [\Delta]t'\) and \(x\)

  \[ \Gamma_0 \rightarrow \Delta_0 \]
  From subderivation \( \Gamma \rightarrow \Delta \)

  \[ [\Delta_0]\tau = [\Delta_0]t' \]
  From premise \([\Delta]t = [\Delta]t'\) and \(x\)

- Case \( \Gamma \rightarrow \Delta \):
  \[ \Gamma, \hat{\beta} : \kappa' = t' \]
  By congruence of equality

\[ \Gamma_1, \hat{\beta} = t' \rightarrow \Delta, \hat{\beta} = t' \rightarrow \text{Eqn} \]

\[ \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \]

- Case \( \Gamma \rightarrow \Delta \):
  \[ [\Delta_0]\tau = [\Delta_0]t' \]
  By congruence of equality

- Case \( \Gamma \rightarrow \Delta \):
  \[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]
  By congruence of equality

\[ \Gamma, \hat{\beta} : \kappa' = t' \rightarrow \text{Add} \]

\[ \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \]

\[ [\Delta_0]t = [\Delta_0]t' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ [\Delta_0]t = [\Delta_0]t' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ [\Delta_0]t = [\Delta_0]t' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ [\Delta_0]t = [\Delta_0]t' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ [\Delta_0]t = [\Delta_0]t' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]

\[ (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \]
Proof of **Lemma 22** (Extension Inversion)  

\[(\Gamma, \beta : \kappa') = (\Gamma_0, \alpha : \kappa = \tau, \Gamma_1)\]  

Given

\[\Gamma = (\Gamma_0, \alpha : \kappa = \tau, \Gamma_1')\]

Since the last elements must be equal

By injectivity of syntax

\[\Gamma \rightarrow \Delta\]

Subderivation

\[\Gamma_0, \alpha : \kappa = \tau, \Gamma_1' \rightarrow \Delta\]

By equality

\[\Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1)\]

By i.h.

\[\equiv\]

\[|\Delta_0|_\tau = |\Delta_0|_{\tau'}\]

"

\[\equiv\]

\[\Gamma_0 \rightarrow \Delta_0\]

"

\[\equiv\]

\[(\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa')\]

By congruence of equality

(v) We have \(\Gamma_0, \alpha : A, \Gamma_1 \rightarrow \Delta\). This proof is similar to the proof of part (i), except for the domain condition, which we handle similarly to part (ii).

(vi) We have \(\Gamma_0, \alpha : \kappa, \Gamma_1 \rightarrow \Delta\).

- **Case**
  \[\Gamma \rightarrow \Delta\]
  \[\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' \quad \text{---Uvar}\]
  (\(\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma, \beta : \kappa')\)

Given

\[\tau = |\Delta_0|_{\tau'}\]

Since the final elements must be equal

\[\Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1)\]

By injectivity of context syntax

By induction, there are two possibilities:

1. \(\alpha\) is not solved:
   \[\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)\]
   By i.h.
   \[\equiv\]
   \[\Gamma_0 \rightarrow \Delta_0\]
   
2. \(\alpha\) is solved:
   \[\Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1)\]
   By i.h.
   \[\equiv\]
   \[\Gamma_0 \rightarrow \Delta_0\]
   
- **Case**
  \[\Gamma \rightarrow \Delta\]
  \[\Delta|_{A} = \Delta|_{A'} \quad \text{---Var}\]
  \[\Gamma, \alpha : A \rightarrow \Delta, \alpha : A'\]

Similar to the ---Uvar case.

- **Case**
  \[\Gamma \rightarrow \Delta\]
  \[\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' \quad \text{---Marker}\]

Similar to the ---Uvar case.

- **Case**
  \[\Gamma \rightarrow \Delta\]
  \[\Delta|_{t} = \Delta|_{t'} \quad \text{---Eqn}\]
  \[\Gamma, \beta : \kappa' = t' \rightarrow \Delta, \beta : \kappa' = t'\]

Similar to the ---Uvar case.

- **Case**
  \[\Gamma \rightarrow \Delta\]
  \[\Delta|_{t} = \Delta|_{t'} \quad \text{---Solved}\]
  \[\Gamma, \beta : \kappa' = t \rightarrow \Delta, \beta : \kappa' = t'\]

Similar to the ---Uvar case.
Proof of Lemma 22 (Extension Inversion)

Case

\[ \Gamma \rightarrow \Delta \]

\[ \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa \]

\[ \Gamma_0, \alpha : \kappa, \Gamma_1 \]

- Case \( \alpha \neq \beta \):

\[ (\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma_1, \beta : \kappa') \]

- Given

\[ = (\Gamma_0, \alpha : \kappa, \Gamma_1, \beta : \kappa') \]

Since the final elements must be equal

\[ \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') \]

By injectivity of context syntax

By induction, there are two possibilities:

* \( \alpha \) is not solved:

\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]

By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') \]

By congruence of equality

* \( \alpha \) is solved:

\[ \Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \]

By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa') \]

By congruence of equality

- Case \( \alpha = \beta \):

\[ \kappa' = \kappa \] and \( \Gamma_0 = \Gamma \) and \( \Gamma_1 = \cdot \)

By injectivity of syntax

\[ (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1) \]

where \( \Delta_0 = \Delta \) and \( \Delta_1 = \cdot \)

\[ \Gamma_0 \rightarrow \Delta_0 \]

From premise \( \Gamma \rightarrow \Delta \)

Case

\[ \Gamma \rightarrow \Delta \]

\[ \Gamma, \alpha : \kappa, \Gamma_1 \]

By induction, there are two possibilities:

- \( \alpha \) is not solved:

\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]

By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') \]

By congruence of equality

- \( \alpha \) is solved:

\[ \Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \]

By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa') \]

By congruence of equality

Case

\[ \Gamma \rightarrow \Delta \]

\[ \Gamma, \alpha : \kappa, \Gamma_1 \]

By induction, there are two possibilities:

- \( \alpha \) is not solved:

\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]

By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta, \beta : \kappa) = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') \]

By congruence of equality

- \( \alpha \) is solved:

\[ \Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \]

By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]

\[ (\Delta, \beta : \kappa) = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa') \]

By congruence of equality

Proof of Lemma 22 (Extension Inversion)
• Case \( \Gamma \rightarrow \Delta \)
  \[ \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' = t \rightarrow \text{Solve} \]
  \( \Gamma_0, \alpha : \kappa, \Gamma_1 \)

  – Case \( \alpha \neq \beta \):

  \[ (\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma, \beta : \kappa') \]
  Given

  \[ = (\Gamma_0, \alpha : \kappa, \Gamma'_1, \beta : \kappa') \]
  Since the final elements must be equal

  \[ \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma'_1) \]
  By injectivity of context syntax

  By induction, there are two possibilities:
  * \( \alpha \) is not solved:
    \[ \Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) \]
    By i.h.

    \[ \Rightarrow \Gamma_0 \rightarrow \Delta_0 \]
    "

    \[ (\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \hat{\beta} : \kappa' = t) \]
    By congruence of equality

  * \( \alpha \) is solved:
    \[ \Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \]
    By i.h.

    \[ \Rightarrow \Gamma_0 \rightarrow \Delta_0 \]
    "

    \[ (\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t) \]
    By congruence of equality

  – Case \( \alpha = \beta \):

  \[ \Gamma = \Gamma_0 \text{ and } \kappa = \kappa' \text{ and } \Gamma_1 = \cdot \]
  By injectivity of syntax

  \[ (\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \]
  where \( \Delta_0 = \Delta \) and \( \tau' = t \) and \( \Delta_1 = \cdot \)

  From premise \( \Gamma \rightarrow \Delta \) \( \blacksquare \)

**Lemma 23** (Deep Evar Introduction).  
(i) If \( \Gamma_0, \Gamma_1 \) is well-formed and \( \hat{\alpha} \) is not declared in \( \Gamma_0, \Gamma_1 \) then \( \Gamma_0, \Gamma_1 \rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \).

(ii) If \( \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \) is well-formed and \( \Gamma \vdash t : \kappa \) then \( \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \rightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \).

(iii) If \( \Gamma_0, \Gamma_1 \) is well-formed and \( \Gamma \vdash t : \kappa \) then \( \Gamma_0, \Gamma_1 \rightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \).

**Proof.**

(i) Assume that \( \Gamma_0, \Gamma_1 \) is well-formed. We proceed by induction on \( \Gamma_1 \).

• Case \( \Gamma_1 = \cdot \):
  \[ \Gamma_0 \text{ ctx} \]
  Given

  \[ \hat{\alpha} \notin \text{dom}(\Gamma_0) \]
  Given

  \[ \Gamma_0, \hat{\alpha} : \kappa \text{ ctx} \]
  By rule \( \text{VarCtx} \)

  \[ \Rightarrow \Gamma_0 \rightarrow \Gamma_0 \]
  By Lemma 32 (Extension Reflexivity)

  \[ \Rightarrow \Gamma_0 \rightarrow \Gamma_0, \hat{\alpha} : \kappa \]
  By rule \( \rightarrow \text{Add} \)

• Case \( \Gamma_1 = \Gamma'_1, x : A \):
  \[ \Gamma_0, \Gamma'_1, x : A \text{ ctx} \]
  Given

  \[ \Gamma_0, \Gamma'_1 \text{ ctx} \]
  By inversion

  \[ x \notin \text{dom}(\Gamma_0, \Gamma'_1) \]
  By inversion (1)

  \[ \Gamma_0, \Gamma'_1 \vdash A \text{ type} \]
  By inversion

  \[ \hat{\alpha} \notin \text{dom}(\Gamma_0, \Gamma'_1, x : A) \]
  Given

  \[ \hat{\alpha} \neq x \]
  By inversion (2)

  \[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1 \text{ ctx} \]
  By i.h.

  \[ \Rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1 \]
  "

  \[ \Gamma_0, \Gamma'_1 \rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1 \]
  By Lemma 36 (Extension Weakening (Sorts))

  \[ x \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1) \]
  By (1) and (2)

  \[ \Rightarrow \Gamma_0, \Gamma'_1, x : A \rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1, x : A \]
  By \( \rightarrow \text{Var} \)
Case $\Gamma_1 = \Gamma'_1, \beta : \kappa'$:

- $\Gamma_0, \Gamma'_1, \beta : \kappa'$ ctx
  - $\Gamma_0, \Gamma'_1$ ctx
    - $\beta \notin \text{dom}(\Gamma_0, \Gamma'_1)$ By inversion (1)
    - $\beta \notin \text{dom}(\Gamma_0, \Gamma'_1, \beta')$ Given
    - $\hat{\alpha} \neq \beta$ By inversion (2)
    - $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1$ ctx By i.h.
      - $\beta \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1)$ By (1) and (2)

- $\Gamma_0, \Gamma'_1, \beta : \kappa'$ ctx
  - $\Gamma_0, \Gamma'_1$ ctx
    - $\beta \notin \text{dom}(\Gamma_0, \Gamma'_1)$ By inversion (1)
    - $\beta \notin \text{dom}(\Gamma_0, \Gamma'_1, \beta')$ Given
    - $\hat{\alpha} \neq \beta$ By inversion (2)
    - $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1$ ctx By i.h.
      - $\beta \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1)$ By (1) and (2)

- $\Gamma_0, \Gamma'_1, \beta : \kappa' = t$
  - $\Gamma_0, \Gamma'_1, \hat{\beta} : \kappa' = t$ ctx
    - $\Gamma_0, \Gamma'_1$ ctx
      - $\hat{\beta} \notin \text{dom}(\Gamma_0, \Gamma'_1)$ By inversion (1)
      - $\hat{\alpha} \notin \text{dom}(\Gamma_0, \Gamma'_1, \hat{\beta} : \kappa' = t)$ Given
      - $\hat{\alpha} \neq \hat{\beta}$ By inversion (2)
      - $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1$ ctx By i.h.
        - $\Gamma_0, \Gamma'_1$ By Lemma 36 (Extension Weakening (Sorts))
          - $\hat{\beta} \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1)$ By (1) and (2)

- $\Gamma_0, \beta = t$
  - $\Gamma_0, \Gamma'_1, \beta = t$ ctx
    - $\Gamma_0, \Gamma'_1$ ctx
      - $\beta \notin \text{dom}(\Gamma_0, \Gamma'_1)$ By inversion (1)
      - $\hat{\alpha} \notin \text{dom}(\Gamma_0, \Gamma'_1, \beta = t)$ Given
      - $\hat{\alpha} \neq \beta$ By inversion (2)
      - $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1$ ctx By i.h.
        - $\Gamma_0, \beta = t$ By Lemma 36 (Extension Weakening (Sorts))
          - $\beta \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1)$ By (1) and (2)

- Case $\Gamma_1 = (\Gamma'_1, \beta)$:
Proof of Lemma 23 (Deep Evar Introduction)

(i) Assume \( \Gamma, \alpha : \kappa, \Gamma_1 \) ctx. We proceed by induction on \( \Gamma_1 \):

- Case \( \Gamma_1 = \cdot \):
  
  \[ \Gamma_0 \vdash t : \kappa \]  
  \[ \Gamma_0, \gamma_1 \Gamma_1 \text{ ctx} \]  
  By inversion
  
  \[ \Gamma_0 \text{ ctx} \]  
  \[ \Gamma_0 \rightarrow \Gamma_0 \]  
  By Lemma 32 (Extension Reflexivity)
  
  \[ \Gamma_0, \alpha : \kappa : \kappa, \Gamma_1 \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \]  
  By rule \( \rightarrow \text{Solve} \)

- Case \( \Gamma_1 = (\Gamma_1', \gamma : A) \):
  
  \[ \Gamma_0 \vdash t : \kappa \]  
  \[ \Gamma_0, \gamma_1 \Gamma_1' \text{ ctx} \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1', \gamma : A \text{ ctx} \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \text{ ctx} \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \]  
  By i.h.
  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \]  
  \[ \Gamma_0 \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, x : A \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, x : A \]  
  By rule \( \rightarrow \text{Var} \)

- Case \( \Gamma_1 = (\Gamma_1', \beta : \kappa') \):
  
  \[ \Gamma_0 \vdash t : \kappa \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1', \beta : \kappa' \text{ ctx} \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \text{ ctx} \]  
  \[ \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma_1') \]  
  By inversion (1)
  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \]  
  By i.h.
  
  \[ \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa = t, \Gamma_1') \]  
  since this is the same domain as (1)
  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1', \beta : \kappa' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, \beta : \kappa' \]  
  By rule \( \rightarrow \text{Uvar} \)

- Case \( \Gamma_1 = (\Gamma_1', \bar{\beta} : \kappa') \):
  
  \[ \Gamma_0 \vdash t : \kappa \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1', \bar{\beta} : \kappa' \text{ ctx} \]  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \text{ ctx} \]  
  \[ \bar{\beta} \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma_1') \]  
  By inversion (1)
  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \]  
  By i.h.
  
  \[ \bar{\beta} \notin \text{dom}(\Gamma_0, \alpha : \kappa = t, \Gamma_1') \]  
  since this is the same domain as (1)
  
  \[ \Gamma_0, \alpha : \kappa, \Gamma_1', \bar{\beta} : \kappa' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, \bar{\beta} : \kappa' \]  
  By rule \( \rightarrow \text{Unsolved} \)
• Case $\Gamma_1 = (\Gamma'_1, \beta : \kappa' = t')$:

  $\Gamma_0 \vdash t' : \kappa$  
  $\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1, \bar{\beta} : \kappa' = t' \text{ ctx}$  
  $\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \text{ ctx}$  
  By inversion

  $\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \vdash t' : \kappa'$  
  $\bar{\beta} \not\in \text{dom}(\Gamma_0, \bar{\alpha}, \Gamma'_1)$  
  By inversion (1)

  $\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \vdash \kappa : \kappa'$  
  $\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \vdash \kappa = t, \Gamma_1$  
  By i.h.

  $\beta \not\in \text{dom}(\Gamma_0, \bar{\alpha}, \kappa, \Gamma'_1)$  
  By i.h.

  $\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \vdash t, \Gamma_1 \vdash t' : \kappa'$  
  By Lemma 33 (Extension Transitivity)

(iii) Apply parts (i) and (ii) as lemmas, then Lemma 33 (Extension Transitivity).

Lemma 26 (Parallel Admissibility).
If $\Gamma_L \rightarrow \Delta_L$ and $\Gamma_R \rightarrow \Delta_L, \Delta_R$ then:

(i) $\Gamma_L, \bar{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa, \Delta_R$

(ii) If $\Delta_L \vdash \tau' : \kappa$ then $\Gamma_L, \bar{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau', \Delta_R$

(iii) If $\Gamma_L \vdash \tau : \kappa$ and $\Delta_L \vdash \tau' : \text{type}$ and $|\Delta_L| \tau = |\Delta_L| \tau'$, then $\Gamma_L, \bar{\alpha} : \kappa = \tau, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau', \Delta_R$

Proof. By induction on $\Delta_L$. As always, we assume that all contexts mentioned in the statement of the lemma are well-formed. Hence, $\bar{\alpha} \not\in \text{dom}(\Gamma_L) \cup \text{dom}(\Gamma_R) \cup \text{dom}(\Delta_L) \cup \text{dom}(\Delta_R)$.

(i) We proceed by cases of $\Delta_R$. Observe that in all the extension rules, the right-hand context gets smaller, so as we enter subderivations of $\Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta_R$, the context $\Delta_R$ becomes smaller.

The only tricky part of the proof is to apply the i.h., we need $\Gamma_L \rightarrow \Delta_L$. So we need to make sure that as we drop items from the right of $\Gamma_R$ and $\Delta_R$, we don't go too far and start decomposing $\Gamma_L$ or $\Delta_L$! It's easy to avoid decomposing $\Delta_L$: when $\Delta_R = \cdot$, we don't need to apply the i.h. anyway. To avoid decomposing $\Gamma_L$, we need to reason by contradiction, using Lemma 19 (Declaration Preservation).
• Case $\Delta_R = \cdot$.
  We have $\Gamma_L \rightarrow \Delta_L$. Applying $\rightarrow \text{Unsolved}$ to that derivation gives the result.

• Case $\Delta_R = (\Delta_R', \beta)$: We have $\beta \neq \alpha$ by the well-formedness assumption.
  The concluding rule of $\Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta_R, \beta$ must have been $\rightarrow \text{Unsolved}$ or $\rightarrow \text{Add}$. In both cases, the result follows by i.h. and applying $\rightarrow \text{Unsolved}$ or $\rightarrow \text{Add}$.  

  Note: In $\rightarrow \text{Add}$, the left-hand context doesn’t change, so we clearly maintain $\Gamma_L \rightarrow \Delta_L$. In $\rightarrow \text{Unsolved}$ we can correctly apply the i.h. because $\Gamma_R \neq \cdot$. Suppose, for a contradiction, that $\Gamma_R = \cdot$. Then $\Gamma_L = (\Gamma', \beta)$. It was given that $\Gamma_L \rightarrow \Delta_L$, that is, $\Gamma', \beta \rightarrow \Delta_L$. By Lemma 19 (Declaration Preservation), $\Delta_L$ has a declaration of $\beta$. But then $\Delta = (\Delta_L, \Delta_R', \beta)$ is not well-formed: contradiction. Therefore $\Gamma_R \neq \cdot$.

• Case $\Delta_R = (\Delta_R', \beta : \kappa = t)$: We have $\beta \neq \alpha$ by the well-formedness assumption.
  The concluding rule must have been $\rightarrow \text{Solved}$ or $\rightarrow \text{AddSolved}$ in each case, apply the i.h. and then the corresponding rule. (In $\rightarrow \text{Solved}$ and $\rightarrow \text{Solve}$ use Lemma 19 (Declaration Preservation) to show $\Gamma_R \neq \cdot$.)

• Case $\Delta_R = (\Delta_R', \alpha)$: The concluding rule must have been $\rightarrow \text{Var}$. The result follows by i.h. and applying $\rightarrow \text{Var}$.

• Case $\Delta_R = (\Delta_R', \alpha = \tau)$: The concluding rule must have been $\rightarrow \text{Eqn}$. The result follows by i.h. and applying $\rightarrow \text{Eqn}$.

• Case $\Delta_R = (\Delta_R', \alpha : A)$: Similar to the previous case, with rule $\rightarrow \text{Var}$.

(ii) Similar to part (i), except that when $\Delta_R = \cdot$, apply rule $\rightarrow \text{Solve}$.

(iii) Similar to part (i), except that when $\Delta_R = \cdot$, apply rule $\rightarrow \text{Solved}$ using the given equality to satisfy the second premise.

Lemma 27 (Parallel Extension Solution).
If $\Gamma_L, \alpha : \kappa, \Gamma_R \rightarrow \Delta_L, \alpha : \kappa = \tau', \Delta_R$ and $\Gamma_L \vdash \tau : \kappa$ and $|\Delta_L|\tau = |\Delta_L|\tau'$ then $\Gamma_L, \alpha : \kappa = \tau, \Gamma_R \rightarrow \Delta_L, \alpha : \kappa = \tau', \Delta_R$.

Proof. By induction on $\Delta_R$.
In the case where $\Delta_R = \cdot$, we know that rule $\rightarrow \text{Solve}$ must have concluded the derivation (we can use Lemma 19 (Declaration Preservation) to get a contradiction that rules out $\rightarrow \text{AddSolved}$); then we have a subderivation $\Gamma_L \rightarrow \Delta_L$, to which we can apply $\rightarrow \text{Solved}$.

Lemma 28 (Parallel Variable Update).
If $\Gamma_L, \alpha : \kappa, \Gamma_R \rightarrow \Delta_L, \alpha : \kappa = \tau_0, \Delta_R$ and $\Gamma_L \vdash \tau_1 : \kappa$ and $\Delta_L \vdash \tau_2 : \kappa$ and $|\Delta_L|\tau_0 = |\Delta_L|\tau_1 = |\Delta_L|\tau_2$ then $\Gamma_L, \alpha : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \alpha : \kappa = \tau_2, \Delta_R$.

Proof. By induction on $\Delta_R$. Similar to the proof of Lemma 27 (Parallel Extension Solution), but applying $\rightarrow \text{Solved}$ at the end.

Lemma 29 (Substitution Monotonicity).

(i) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash t : \kappa$ then $|\Delta| |\Gamma| t = |\Delta| t$.

(ii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash p$ prop then $|\Delta| |\Gamma| p = |\Delta| p$.

(iii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash \alpha$ type then $|\Delta| |\Gamma| \alpha = |\Delta| \alpha$.

Proof. We prove each part in turn; part (i) does not depend on parts (ii) or (iii), so we can use part (i) as a lemma in the proofs of parts (ii) and (iii).

• Proof of Part (i): By lexicographic induction on the derivation of $D :: \Gamma \rightarrow \Delta$ and $\Gamma \vdash t : \kappa$. We proceed by cases on the derivation of $\Gamma \vdash t : \kappa$. 

Proof of Lemma 29 (Substitution Monotonicity)
Proof of Lemma 29 (Substitution Monotonicity)

\[ \text{Case } \alpha : \kappa \in \Gamma \]
\[ \Gamma \vdash \alpha : \kappa \]
\[
[\Gamma] \alpha = \alpha \\
[\Delta] \alpha = [\Delta] \alpha \\
= [\Delta][\Gamma] \alpha
\]
Since \( \alpha \) is not solved in \( \Gamma \)

Consider whether or not there is a binding of the form \( (\alpha = \tau) \in \Gamma \).

* Case \( (\alpha = \tau) \in \Gamma \):

\[ \Delta = (\Delta_0, \alpha = \tau', \Delta_1) \]
\[ D' :: \quad \Gamma_0 \rightarrow \Delta_0 \]
\[ D' < D \]

(1) \[ [\Delta_0][\tau'] = [\Delta_0][\tau] \]
(2) \[ [\Delta_0][\Gamma_0][\tau] = [\Delta_0][\tau] \]

By definition

\[ [\Delta][\Gamma_0][\tau] = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0, \alpha = \tau, \Gamma_1][\alpha] \]
\[ = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0, \alpha = \tau][\alpha] \]
\[ = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0][\tau] \]
\[ = [\Delta_0][\tau'] \]
\[ = [\Delta_0, \alpha = \tau'][\alpha] \]
\[ = [\Delta_0, \alpha = \tau', \Delta_1][\alpha] \]
\[ = [\Delta][\alpha] \]

* Case \( (\alpha = \tau) \notin \Gamma \):

\[ [\Gamma][\alpha] = [\alpha] \\
[\Delta][\Gamma][\alpha] = [\Delta][\alpha] \]

Apply \( [\Delta] \) to both sides

\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} : \kappa \]

SolvedVarSort

Similar to the VarSort case.

* Case \( \_ \vdash 1 : \_ \)

UnitSort

\[ [\Delta]1 = 1 = [\Delta][\Gamma]1 \quad \text{Since } FV(1) = \emptyset \]

* Case \( \Gamma \vdash \tau_1 : \_ \\
\Gamma \vdash \tau_2 : \_ \)

BinSort

\[ [\Delta][\Gamma][\tau_1] = [\Delta][\tau_1] \quad \text{By i.h.} \]
\[ [\Delta][\Gamma][\tau_2] = [\Delta][\tau_2] \quad \text{By i.h.} \]
\[ [\Delta][\Gamma][\tau_1] + [\Delta][\Gamma][\tau_2] = [\Delta][\tau_1] + [\Delta][\tau_2] \quad \text{By congruence of equality} \]
\[ [\Delta][\Gamma](\tau_1 \oplus \tau_2) = [\Delta](\tau_1 \oplus \tau_2) \quad \text{Definition of substitution} \]

* Case \( \Gamma \vdash \text{zero} : \mathbb{N} \)

ZeroSort

\[ [\Delta]\text{zero} = \text{zero} = [\Delta][\Gamma]\text{zero} \quad \text{Since } FV(\text{zero}) = \emptyset \]
Proof of Lemma 29 (Substitution Monotonicity)

\[ \text{lem:substitution-monotonicity} \]

- Case \[ \Gamma \vdash t : N \]
  \[ \Gamma \vdash \text{succ}(t) : N \text{ SuccSort} \]
  \[ |\Delta|\Gamma| t = |\Delta| t \quad \text{By i.h.} \]
  \[ \text{succ}(|\Delta|\Gamma| t) = \text{succ}(|\Delta| t) \quad \text{By congruence of equality} \]
  \[ |\Delta|\Gamma| \text{succ}(t) = |\Delta| \text{succ}(t) \quad \text{By definition of substitution} \]

- **Proof of Part (ii):** We have a derivation of \( \Gamma \vdash P \text{ prop} \), and will use the previous part as a lemma.

- Case \[ \Gamma \vdash t : N \quad \Gamma \vdash t' : N \]
  \[ \Gamma \vdash t = t' \text{ prop} \text{ EqProp} \]
  \[ |\Delta|\Gamma| t = |\Delta| t \quad \text{By part (i)} \]
  \[ |\Delta|\Gamma| t' = |\Delta| t' \quad \text{By part (i)} \]
  \[ (|\Delta|\Gamma| t = |\Delta|\Gamma| t') = (|\Delta|r = |\Delta| r') \quad \text{By congruence of equality} \]
  \[ |\Delta|\Gamma|(t = t') = |\Delta| (t = t') \quad \text{Definition of substitution} \]

- **Proof of Part (iii):** By induction on the derivation of \( \Gamma \vdash A \text{ type} \), using the previous parts as lemmas.

- Case \[ (u : *) \in \Gamma \]
  \[ \Gamma \vdash u : * \text{ VarWF} \]
  \[ |\Delta|\Gamma| u = |\Delta| u \quad \text{By rule VarSort} \]
  \[ |\Delta|\Gamma| \text{SolvedVarWF} \]
  \[ \Gamma \vdash \alpha : * \quad \text{By rule SolvedVarSort} \]
  \[ |\Delta|\Gamma| \alpha = |\Delta| \alpha \quad \text{By part (i)} \]

- Case \[ \Gamma \vdash 1 : \text{ type} \text{ UnitWF} \]
  \[ |\Delta|\Gamma| 1 = |\Delta| 1 \quad \text{By rule UnitSort} \]
  \[ |\Delta|\Gamma| \text{ForallWF} \]
  \[ \Gamma \vdash A_1 \rightarrow A_2 : \text{ type} \text{ BinWF} \]
  \[ |\Delta|\Gamma| A_1 = |\Delta| A_1 \quad \text{By i.h.} \]
  \[ |\Delta|\Gamma| A_2 = |\Delta| A_2 \quad \text{By i.h.} \]
  \[ |\Delta|\Gamma| A_1 \rightarrow A_2 = |\Delta| A_1 \rightarrow |\Delta| A_2 \quad \text{By congruence of equality} \]
  \[ |\Delta|\Gamma| A_1 \rightarrow A_2 = |\Delta| (A_1 \rightarrow A_2) \quad \text{Definition of substitution} \]

- Case \[ \text{VecWF} \]
  Similar to the BinWF case.

- Case \[ \Gamma, \alpha : \kappa \vdash A_0 : \text{ type} \]
  \[ \Gamma \vdash \forall \alpha : \kappa. A_0 : \text{ type} \text{ ForallWF} \]
Proof of Lemma 29 (Substitution Monotonicity).

\[
\Gamma \rightarrow \Delta \quad \text{Given}
\]
\[
\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \quad \text{By rule \( \rightarrow \text{Uvar} \)}
\]
\[
[\Delta, \alpha : \kappa] \Gamma \rightarrow [\Delta, \alpha : \kappa] A_0 = [\Delta, \alpha : \kappa] A_0 
\quad \text{By i.h.}
\]
\[
[\Delta] \Gamma A_0 = [\Delta] A_0 
\quad \text{By definition of substitution}
\]
\[
\forall \alpha : \kappa. [\Delta] \Gamma A_0 = [\Delta] A_0 
\quad \text{By congruence of equality}
\]
\[
[\Delta] \Gamma (\forall \alpha : \kappa. A_0) = [\Delta] (\forall \alpha : \kappa. A_0) 
\quad \text{By definition of substitution}
\]

- Case ExistsWF: Similar to the ForallWF case.
- Case \( \Gamma \vdash \text{prop} \) \( \Gamma \vdash A_0 \) type

\[
\begin{align*}
\Gamma \vdash \text{prop} & \quad \Gamma \vdash A_0 \text{ type} \\
\hline
\end{align*}
\]

\[
[\Delta] \Gamma P = [\Delta] P 
\quad \text{By part (ii)}
\]
\[
[\Delta] \Gamma A_0 = [\Delta] A_0 
\quad \text{By i.h.}
\]
\[
[\Delta] \Gamma P \supset [\Delta] \Gamma A_0 = [\Delta] P \supset [\Delta] A_0 
\quad \text{By congruence of equality}
\]
\[
[\Delta] \Gamma (P \supset A_0) = [\Delta] (P \supset A_0) 
\quad \text{Definition of substitution}
\]

- Case \( \Gamma \vdash \text{prop} \) \( \Gamma \vdash A_0 \) type

\[
\begin{align*}
\Gamma \vdash \text{prop} & \quad \Gamma \vdash A_0 \text{ type} \\
\hline
\end{align*}
\]

Similarly to the \( \text{impliesWF} \) case.

Lemma 30 (Substitution Invariance).

(i) If \( \Gamma \rightarrow \Delta \) and \( \Gamma \vdash t : \kappa \) and \( \text{FEV}(\Gamma) = \emptyset \) then \( [\Delta] \Gamma t = [\Gamma] t \).

(ii) If \( \Gamma \rightarrow \Delta \) and \( \Gamma \vdash \text{prop} \) and \( \text{FEV}(\Gamma P) = \emptyset \) then \( [\Delta] \Gamma P = [\Gamma] P \).

(iii) If \( \Gamma \rightarrow \Delta \) and \( \Gamma \vdash \text{type} \) and \( \text{FEV}(\Gamma A) = \emptyset \) then \( [\Delta] \Gamma A = [\Gamma] A \).

Proof. Each part is a separate induction, relying on the proofs of the earlier parts. In each part, the result follows by an induction on the derivation of \( \Gamma \rightarrow \Delta \).

The main observation is that \( \Delta \) adds no equations for any variable of \( t \), \( P \), and \( A \) that \( \Gamma \) does not already contain, and as a result applying \( \Delta \) as a substitution to \( [\Gamma] t \) does nothing.

Lemma 24 (Soft Extension).
If \( \Gamma \rightarrow \Delta \) and \( \Gamma, \Theta \text{ ctx} \) and \( \Theta \) is soft, then there exists \( \Omega \) such that \( \text{dom}(\Theta) = \text{dom}(\Omega) \) and \( \Gamma, \Theta \rightarrow \Delta, \Omega \).

Proof. By induction on \( \Theta \).

- Case \( \Theta = : \) We have \( \Gamma \rightarrow \Delta \). Let \( \Omega = : \). Then \( \Gamma, \Theta \rightarrow \Delta, \Omega \).

- Case \( \Theta = (\Theta', \hat{\alpha} : \kappa = t) \):

\[
\Gamma, \Theta' \rightarrow \Gamma, \Omega' \quad \text{By i.h.}
\]

\[
\begin{align*}
\Gamma, \Theta', \hat{\alpha} : \kappa = t & \rightarrow \Delta, \Omega', \hat{\alpha} : \kappa = t \\
\hline
\end{align*}
\]

- Case \( \Theta = (\Theta', \hat{\alpha} : \kappa) \):

If \( \kappa = * \), let \( t = 1 \); if \( \kappa = \mathbb{N} \), let \( t = 0 \).

\[
\Gamma, \Theta' \rightarrow \Gamma, \Omega' \quad \text{By i.h.}
\]

\[
\begin{align*}
\Gamma, \Theta', \hat{\alpha} : \kappa & \rightarrow \Delta, \Omega', \hat{\alpha} : \kappa = t \\
\hline
\end{align*}
\]

March 16, 2018
Lemma 31 (Split Extension).
If \( \Delta \rightarrow \Omega \)
and \( \hat{\alpha} \in \text{unsolved} (\Delta) \)
and \( \Omega = \Omega_1 [\hat{\alpha} : \kappa = t_1] \)
and \( \Omega \) is canonical (Definition 3)
and \( \Omega \vdash t_2 : \kappa \)
then \( \Delta \rightarrow \Omega_1 [\hat{\alpha} : \kappa = t_2] \).

Proof. By induction on the derivation of \( \Delta \rightarrow \Omega \). Use the fact that \( \Omega_1 [\hat{\alpha} : \kappa = t_1] \) and \( \Omega_1 [\hat{\alpha} : \kappa = t_2] \) agree on all solutions except the solution for \( \hat{\alpha} \). In the \( \rightarrow \text{Solve} \) case where the existential variable is \( \hat{\alpha} \), use \( \Omega \vdash t_2 : \kappa \).

\[ \square \]

C'.1 Reflexivity and Transitivity

Lemma 32 (Extension Reflexivity).
If \( \Gamma \text{ ctx} \) then \( \Gamma \rightarrow \Gamma \).

Proof. By induction on the derivation of \( \Gamma \text{ ctx} \).

- Case \[ \text{ctx EmptyCtx} \]
  - \( \rightarrow \) \[ \text{by rule } \rightarrow \text{Id} \]

- Case \[
\begin{align*}
\Gamma \text{ ctx} & \quad x \not\in \text{dom}(\Gamma) \\
& \quad \Gamma \vdash \text{A type}
\end{align*}
\]
  - \( \rightarrow \) \[ \text{by rule } \rightarrow \text{Var} \]

- Case \[
\begin{align*}
\Gamma \text{ ctx} & \quad u : \kappa \not\in \text{dom}(\Gamma)
\end{align*}
\]
  - \( \rightarrow \) \[ \text{by rule } \rightarrow \text{Uvar} \text{ or } \rightarrow \text{Unsolved} \]

- Case \[
\begin{align*}
\Gamma \text{ ctx} & \quad \hat{\alpha} \not\in \text{dom}(\Gamma)
\end{align*}
\]
  - \( \rightarrow \) \[ \text{by rule } \rightarrow \text{SolvedCtx} \]

- Case \[
\begin{align*}
\Gamma \text{ ctx} & \quad \alpha : \kappa \in \Gamma \\
& \quad (\alpha = -) \not\in \Gamma \\
& \quad \Gamma \vdash \tau : \kappa
\end{align*}
\]
  - \( \rightarrow \) \[ \text{by rule } \rightarrow \text{EqnVarCtx} \]

Proof of Lemma 32 (Extension Reflexivity) lem:extension-reflexivity
Proof of Lemma 32 (Extension Reflexivity).

**Case** \[ \Gamma \; \mathhyphen \; \text{ctx} \; \not\in \; \Gamma \]

\[ \Gamma \mathrel{\longrightarrow} \Gamma \] By i.h.
\[ \Gamma, \llbracket u \rrbracket \rightarrow \Gamma, \llbracket u \rrbracket \] By rule \( \rightarrow \text{Marker} \)

\[ \square \]

**Lemma 33** (Extension Transitivity).

If \( D : \Gamma \rightarrow \Theta \) and \( D' : \Theta \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).

**Proof.** By induction on \( D' \).

- **Case** 
  
  \[ \Theta \mathrel{\longrightarrow} \Delta' \] 
  
  \[ [\Delta']A = [\Delta']A' \rightarrow \text{Var} \]

  \[ [\Theta']x : A \rightarrow [\Delta', x : A'] \]

  \[ \Theta \mathrel{\longrightarrow} \Delta \] 
  By inversion on \( D \)

  \[ [\Theta]A'' = [\Theta]A \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Theta' \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Delta' \] 
  By i.h.

  \[ [\Delta'][\Theta]A'' = [\Delta'][\Theta]A \] 
  By congruence of equality

  \[ [\Delta']A'' = [\Delta']A' \] 
  By Lemma 29 (Substitution Monotonicity)

  \[ = [\Delta']A' \] 
  By premise \( [\Delta']A = [\Delta']A' \)

  \[ \Gamma', x : A'' \rightarrow \Delta', x : A' \] 
  By \( \rightarrow \text{Var} \)

- **Case** 
  
  \[ \Theta' \rightarrow \Delta' \] 

  \[ [\Theta']\alpha : \kappa \rightarrow [\Delta', \alpha : \kappa] \rightarrow \text{Uvar} \]

  \[ \Theta \mathrel{\longrightarrow} \Delta \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Theta' \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Delta' \] 
  By i.h.

  \[ \Gamma', \alpha : \kappa \rightarrow \Delta', \alpha : \kappa \] 
  By \( \rightarrow \text{Uvar} \)

- **Case** 
  
  \[ \Theta' \rightarrow \Delta' \] 

  \[ [\Theta']\hat{\alpha} : \kappa \rightarrow [\Delta', \hat{\alpha} : \kappa] \rightarrow \text{Uvar} \]

  \[ \Theta \mathrel{\longrightarrow} \Delta \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Theta' \hat{\alpha} \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Delta' \hat{\alpha} \] 
  By i.h.

  \[ \Gamma', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa \] 
  By \( \rightarrow \text{Uvar} \)

- **Case** 
  
  \[ \Theta' \rightarrow \Delta' \] 

  \[ [\Theta']\phi : \lambda \rightarrow [\Delta', \phi : \lambda] \rightarrow \text{Uvar} \]

  \[ \Theta \mathrel{\longrightarrow} \Delta \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Theta' \phi \] 
  By inversion on \( D \)

  \[ \Gamma' \rightarrow \Delta' \phi \] 
  By i.h.

  \[ \Gamma', \phi : \lambda \rightarrow \Delta', \phi : \lambda \] 
  By \( \rightarrow \text{Uvar} \)

Two rules could have concluded \( D : \Gamma \rightarrow ([\Theta'], \hat{\alpha} : \kappa) \):

- **Case** 
  
  \[ \Gamma' \rightarrow \Theta' \] 

  \[ [\Gamma']\hat{\alpha} : \kappa \rightarrow [\Theta', \hat{\alpha} : \kappa] \rightarrow \text{Unsolved} \]
Proof of Lemma 33 \textbf{(Extension Transitivity)} \lem:extension-transitivity

\begin{align*}
\Gamma' \rightarrow \Delta' & \quad \text{By i.h.} \\
\Gamma', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa & \quad \text{By rule} \rightarrow \text{Add}
\end{align*}

- Case $\Gamma \rightarrow \Theta'$

\begin{align*}
\Gamma \rightarrow \Theta', \hat{\alpha} : \kappa & \rightarrow \Theta' \rightarrow \Theta', \hat{\alpha} : \kappa & \rightarrow \text{Add}
\end{align*}

$\Gamma \rightarrow \Delta'$ \quad \text{By i.h.}

$\Gamma' \rightarrow \Delta', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa \rightarrow \text{Add}$

- Case $\Theta' \rightarrow \Delta'$

$[\Delta'] t = \Theta' \rightarrow \Delta' t = \Theta' t'$

Two rules could have concluded $\Delta' : \Gamma \rightarrow (\Theta', \hat{\alpha} : \kappa = t)$:

- Case $\Gamma' \rightarrow \Theta'$

$[\Theta'] t'' = [\Theta'] t \quad \rightarrow \text{Solved}$

$\Gamma' \rightarrow \Delta'$ \quad \text{By i.h.}

$[\Theta'] t'' = [\Theta'] t \quad \text{Premise}$

$[\Delta'] [\Theta'] t'' = [\Delta'] [\Theta'] t \quad \text{By Lemma 29} \text{(Substitution Monotonicity)}$

$\Gamma', \hat{\alpha} : \kappa = t'' \rightarrow \Delta', \hat{\alpha} : \kappa = t' \quad \text{By rule} \rightarrow \text{Solved}$

- Case $\Gamma \rightarrow \Theta'$

$\Gamma \rightarrow \Theta', \hat{\alpha} : \kappa = t \rightarrow \Theta', \hat{\alpha} : \kappa = t \quad \rightarrow \text{AddSolved}$

$\Gamma \rightarrow \Delta'$ \quad \text{By i.h.}

$\Gamma \rightarrow \Delta', \hat{\alpha} : \kappa = t' \quad \text{By rule} \rightarrow \text{AddSolved}$

- Case $\Theta' \rightarrow \Delta'$

$[\Delta'] t = [\Delta'] t' \quad \rightarrow \text{Eqn}$

$\Theta \rightarrow \Delta'$ \quad \text{By i.h.}

$\Theta \rightarrow \Delta' \rightarrow \Delta' \rightarrow \text{Eqn}$

By inversion on $\Delta'$

$[\Theta'] t'' = [\Theta'] t \quad \text{By inversion on } \Delta'$

$[\Delta'] [\Theta'] t'' = [\Delta'] [\Theta'] t \quad \text{By Lemma 29} \text{(Substitution Monotonicity)}$

$\Gamma', \hat{\alpha} = t'' \rightarrow \Delta', \hat{\alpha} = t' \quad \text{By rule} \rightarrow \text{Eqn}$

- Case $\Theta \rightarrow \Delta'$

$\Theta \rightarrow \Delta' \rightarrow \Delta' \rightarrow \text{Add}$

$\Gamma \rightarrow \Delta'$ \quad \text{By i.h.}

$\Gamma \rightarrow \Delta', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa \rightarrow \text{Add}$
Proof of Lemma 33 (Extension Transitivity)

• Case \( \Theta \rightarrow \Delta' \)
  \[ \Theta \rightarrow \Delta, \pi : \kappa = t \]
  \[ \Delta \]
  \( \Gamma \rightarrow \Delta' \)
  By i.h.
  \( \Gamma \rightarrow \Delta', \pi : \kappa = t \)
  By rule AddSolved

• Case \( \Theta' \rightarrow \Delta' \)
  \[ \Theta', \pi_u \rightarrow \Delta', \pi_u \]
  \[ \Theta \]
  \( \Gamma = \Gamma', \pi_u \)
  By inversion on \( D \)
  \( \Gamma' \rightarrow \Theta' \)
  By inversion on \( D \)
  \( \Gamma' \rightarrow \Delta' \)
  By i.h.
  \( \Gamma', \pi_u \rightarrow \Delta', \pi_u \)
  By Uvar

C'.2 Weakening

Lemma 34 (Suffix Weakening). If \( \Gamma \vdash t : \kappa \) then \( \Gamma, \Theta \vdash t : \kappa \).

Proof. By induction on the given derivation. All cases are straightforward.

Lemma 35 (Suffix Weakening). If \( \Gamma \vdash A \) type then \( \Gamma, \Theta \vdash A \) type.

Proof. By induction on the given derivation. All cases are straightforward.

Lemma 36 (Extension Weakening (Sorts)). If \( \Gamma \vdash t : \kappa \) and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash t : \kappa \).

Proof. By a straightforward induction on \( \Gamma \vdash t : \kappa \).
  In the VarSort case, use Lemma 22 (Extension Inversion) (i) or (v). In the SolvedVarSort case, use Lemma 22 (Extension Inversion) (iv). In the other cases, apply the i.h. to all subderivations, then apply the rule.

Lemma 37 (Extension Weakening (Props)). If \( \Gamma \vdash P \) prop and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash P \) prop.

Proof. By inversion on rule EqProp and Lemma 36 (Extension Weakening (Sorts)) twice.

Lemma 38 (Extension Weakening (Types)). If \( \Gamma \vdash A \) type and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash A \) type.

Proof. By a straightforward induction on \( \Gamma \vdash A \) type.
  In the VarWF case, use Lemma 22 (Extension Inversion) (i) or (v). In the SolvedVarWF case, use Lemma 22 (Extension Inversion) (iv).
  In the other cases, apply the i.h. and/or (for ImpliesWF and WithWF) Lemma 37 (Extension Weakening (Props)) to all subderivations, then apply the rule.

C'.3 Principal Typing Properties

Lemma 39 (Principal Agreement).

(i) If \( \Gamma \vdash A \) type and \( \Gamma \rightarrow \Delta \) then \( \Delta|A = \Gamma|A \).

(ii) If \( \Gamma \vdash P \) prop and \( \text{FEV}(P) = \emptyset \) and \( \Gamma \rightarrow \Delta \) then \( \Delta|P = \Gamma|P \).

Proof. By induction on the derivation of \( \Gamma \rightarrow \Delta \).
  Part (i):
Proof of **Lemma 39 (Principal Agreement)**

Proof.

By cases of Part (ii):

1. Similar to part (i), using Part (ii) of **Lemma 8 (Substitution—Well-formedness)**.

   - Case 
     
     \[
     \Delta_0 \rightarrow \Delta_0' \\
     \Delta_0, \alpha = t \rightarrow \Delta_0', \alpha = t' 
     \]

     If \( \alpha \not\in FV(A) \), then:

     \[
     [\Gamma_0, \alpha = t]A = [\Gamma_0]A \\
     = [\Delta_0]A \\
     = [\Delta_0, \alpha = t']A 
     \]

     By def. of subst.

     Otherwise, \( \alpha \in FV(A) \).

     \[
     \Gamma_0 \vdash t \text{ type} \\
     \Gamma_0 \vdash [\Gamma_0]t \text{ type} 
     \]

     By Lemma 13 (Right-Hand Substitution for Typing).

     Suppose, for a contradiction, that \( \text{FEV}([\Gamma_0]t) \neq \emptyset \).

     Since \( \alpha \in FV(A) \), we also have \( \text{FEV}(\Gamma]A) \neq \emptyset \), a contradiction.

     \[
     \text{FEV}([\Gamma_0]t) \neq \emptyset \quad \text{Assumption (for contradiction)} \\
     [\Gamma_0]t = [\Gamma]\alpha \quad \text{By def. of subst.} \\
     \text{FEV}([\Gamma]\alpha) \neq \emptyset \quad \text{By above equality} \\
     \alpha \in FV(A) \quad \text{Above} \\
     \text{FEV}([\Gamma]A) \neq \emptyset \quad \text{By a property of subst.} \\
     \Gamma \vdash A \text{ ! type} \quad \text{Given} \\
     \text{FEV}([\Gamma]A) = \emptyset \quad \text{By inversion} \\
     \]

     By contradiction

     \[
     \Gamma_0 \vdash t \text{ ! type} \\
     [\Gamma_0]t = [\Delta_0]t 
     \]

     By i.h.

     \[
     \Gamma_0 \vdash [\Delta_0]t \text{ type} \quad \text{By above equality} \\
     \text{FEV}([\Delta_0]t) = \emptyset \quad \text{By above equality} \\
     \Gamma_0 \vdash [[\Delta_0]t/\alpha]A \text{ ! type} \quad \text{By Lemma 8 (Substitution—Well-formedness) (i)} \\
     [\Gamma_0][[\Delta_0]t/\alpha]A = [\Delta_0][[\Delta_0]t/\alpha]A \quad \text{By i.h. (at } [\Delta_0]t/\alpha]A) \\
     [\Gamma_0, \alpha = t]A = [\Gamma_0][[\Gamma_0]t/\alpha]A \quad \text{By def. of subst.} \\
     = [\Gamma_0][[\Delta_0]t/\alpha]A \quad \text{By above equality} \\
     = [\Delta_0][[\Delta_0]t/\alpha]A \quad \text{By above equality} \\
     = [\Delta_0][[\Delta_0]t'/\alpha]A \quad \text{By } [\Delta_0]t = [\Delta_0]t' \\
     = [\Delta]A \quad \text{By def. of subst.} 
     \]

     Similar to the \( \rightarrow \text{Eqn} \) case.

8. **Case** 

   \[
   \rightarrow \text{Eqn} \rightarrow \text{Solved} \rightarrow \text{Solve} \rightarrow \text{Add} \rightarrow \text{Solved} \rightarrow \text{Marker} 
   \]

   Straightforward, using the i.h. and the definition of substitution.

Part (ii): Similar to part (i), using part (ii) of **Lemma 8 (Substitution—Well-formedness)**.

**Lemma 40** (Right-Hand Subst. for Principal Typing). *If \( \Gamma \vdash A \text{ type} \) then \( \Gamma \vdash [\Gamma]A \text{ p type} \).*

**Proof.** By cases of p:

1. Case p = !: 

   \[
   \rightarrow \text{Solved} \rightarrow \text{Solve} \rightarrow \text{Add} \rightarrow \text{Solved} 
   \]
Proof of Lemma 40 (Right-Hand Subst. for Principal Typing)

\[ \Gamma \vdash A \text{ type} \]

By inversion

\[ \text{FEV}(\Gamma|A) = \emptyset \]

By inversion

\[ \Gamma \vdash [\Gamma]A \text{ type} \]

By Lemma 13 (Right-Hand Substitution for Typing)

\[ \Gamma \rightarrow \Gamma \]

By Lemma 32 (Extension Reflexivity)

\[ [\Gamma][\Gamma]A = [\Gamma]A \]

By Lemma 29 (Substitution Monotonicity)

\[ \text{FEV}(\Gamma)[\Gamma]A = \emptyset \]

By inversion

\[ \Gamma \vdash [\Gamma]A \text{ ! type} \]

By rule PrincipalWF

**Case p = !:**

\[ \Gamma \vdash A \text{ type} \]

By inversion

\[ \Delta \vdash A \text{ type} \]

By Lemma 13 (Right-Hand Substitution for Typing)

\[ \Delta \vdash A \text{ ! type} \]

By rule NonPrincipalWF

\[ \text{FEV}(\Delta)[\Delta]A = \emptyset \]

By inversion

\[ \Delta \vdash [\Delta]A \text{ type} \]

By Lemma 13 (Right-Hand Substitution for Typing)

\[ [\Delta]A = [\Gamma]A \]

By Lemma 30 (Substitution Invariance)

\[ \text{FEV}(\Delta)[\Delta]A = \emptyset \]

By congruence of equality

\[ \Delta \vdash [\Delta]A \text{ ! type} \]

By rule PrincipalWF

---

**Lemma 41** (Extension Weakening for Principal Typing). If \( \Gamma \vdash A \text{ p type} \) and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash A \text{ p type} \).

**Proof.** By cases of p:

- **Case p = !:**

  \[ \Gamma \vdash A \text{ type} \]

  By inversion

  \[ \Delta \vdash A \text{ type} \]

  By Lemma 38 (Extension Weakening (Types))

  \[ \Delta \vdash A \text{ ! type} \]

  By rule NonPrincipalWF

- **Case p = !:**

  \[ \Gamma \vdash A \text{ type} \]

  By inversion

  \[ \Delta \vdash A \text{ type} \]

  By Lemma 38 (Extension Weakening (Types))

  \[ \Delta \vdash A \text{ ! type} \]

  By rule NonPrincipalWF

\[ \Gamma \vdash (A \rightarrow B) \text{ p type} \text{ then } \Gamma \vdash A \text{ p type} \text{ and } \Gamma \vdash B \text{ p type} \]

\[ \Gamma \vdash (P \supset A) \text{ p type} \text{ then } \Gamma \vdash P \text{ prop} \text{ and } \Gamma \vdash A \text{ p type} \]

\[ \Gamma \vdash (A \land P) \text{ p type} \text{ then } \Gamma \vdash P \text{ prop} \text{ and } \Gamma \vdash A \text{ p type} \]

**Proof.** Proof of part 1:

We have \( \Gamma \vdash A \rightarrow B \text{ p type} \).

- **Case p = !:**

  \[ \Gamma \vdash A \rightarrow B \text{ type} \]

  By inversion

  \[ \Gamma \vdash A \text{ type} \]

  By inversion on 1

  \[ \Gamma \vdash B \text{ type} \]

  By inversion on 1

  \[ \Gamma \vdash A \text{ ! type} \]

  By rule NonPrincipalWF

  \[ \Gamma \vdash B \text{ ! type} \]

  By rule NonPrincipalWF
Proof of Lemma 42 (Inversion of Principal Typing) lem:principal-inversion

• Case $p = !$:

\[\begin{align*}
1 & \quad \Gamma \vdash A \to B \text{ type} \\
0 & \quad \text{By inversion on } \Gamma \vdash A \to B \text{! type} \\
\theta & \quad \text{By definition of substitution} \\
\text{FEV}(\Gamma)(A \to B) & \quad \text{By definition of } \text{FEV}(\vdash) \\
= & \quad \text{By properties of empty sets and unions} \\
\text{FEV}(\Gamma|A) & \quad \text{By inversion on } 1 \\
\text{FEV}(\Gamma|B) & \quad \text{By inversion on } 1 \\
\end{align*}\]

Part 2: We have $\Gamma \vdash P \supset A \text{ p type}$. Similar to Part 1.
Part 3: We have $\Gamma \vdash A \land P \text{ p type}$. Similar to Part 2.

\[\square\]

C’.4 Instantiation Extends

Lemma 43 (Instantiation Extension).
If $\Gamma \vdash \check{\alpha} := \tau : \kappa \vdash \Delta$ then $\Gamma \vdash \Delta$.

Proof. By induction on the given derivation.

• Case

\[\begin{align*}
\Gamma_l \vdash \tau : \kappa \\
\check{\alpha} & \quad \text{InstSolve} \\
\Gamma_l, \check{\alpha} : \kappa, \Gamma_R \vdash \check{\alpha} := \tau : \kappa \vdash \Gamma_l, \check{\alpha} : \kappa = \tau, \Gamma_R \\
\end{align*}\]

Follows by Lemma 23 (Deep Evar Introduction) (ii).

• Case

\[\begin{align*}
\beta & \in \text{unsolved}(\Gamma_0[\check{\alpha} : \kappa][\check{\beta} : \kappa]) \\
\Gamma_0[\check{\alpha} : \kappa][\check{\beta} : \kappa] \vdash \check{\alpha} := \beta : \kappa \vdash \Gamma_0[\check{\alpha} : \kappa][\check{\beta} = \check{\alpha}] \\
\end{align*}\]

Follows by Lemma 23 (Deep Evar Introduction) (ii).

• Case

\[\begin{align*}
\Gamma_0[\check{\alpha}_2 : \ast, \check{\alpha}_1 : \ast, \check{\alpha} : \ast = \check{\alpha}_1 \oplus \check{\alpha}_2] \vdash \check{\alpha}_1 := \tau_1 : \ast \vdash \Theta & \quad \text{Subderivation} \\
\Theta \vdash \check{\alpha}_2 := \Theta[\tau_2 : \ast \vdash \Delta] & \quad \text{Subderivation} \\
\Gamma_0[\check{\alpha} : \ast] \vdash \Delta & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\end{align*}\]

Follows by Lemma 33 (Extension Transitivity)

• Case

\[\begin{align*}
\Gamma_0[\check{\alpha} : N] \vdash \check{\alpha} := \text{zero} : N \vdash \Gamma_0[\check{\alpha} : N = \text{zero}] \\
\end{align*}\]

Follows by Lemma 23 (Deep Evar Introduction) (ii).
Proof of Lemma 43 (Instantiation Extension)  lem:instantiation-extension

- Case \( \Gamma[\alpha_1 : N, \hat{\alpha} : N = \text{succ}(\alpha_1)] \vdash \hat{\alpha} : t_1 : N \vdash \Delta \)

  \[ \Gamma[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{succ}(t_1) : N \vdash \Delta \]

  By reasoning similar to the InstBin case.

\[ \Box \]

C'.5 Equivalence Extends

Lemma 44 (Elimeq Extension).

If \( \Gamma / s \vdash t : \kappa \vdash \Delta \) then there exists \( \Theta \) such that \( \Gamma, \Theta \rightarrow \Delta \).

Proof. By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context \( \Delta \).

- Case

  \[ \Gamma / \alpha \equiv \alpha : \kappa \vdash \Gamma \]

  \[ \text{ElimeqUvarRef} \]

  Since \( \Delta = \Gamma \), applying Lemma 32 (Extension Reflexivity) suffices (let \( \Theta = \cdot \)).

- Case

  \[ \Gamma / \text{zero} \equiv \text{zero} : N \vdash \Gamma \]

  Similar to the ElimeqUvarRef case.

- Case

  \[ \Gamma / \sigma \equiv t : N \vdash \Delta \]

  \[ \text{ElimeqSucc} \]

  Follows by i.h.

- Case

  \[ \Gamma_0[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := t : \kappa \vdash \Delta \]

  \[ \text{ElimeqInstL} \]

  \[ \Gamma \vdash \hat{\alpha} := t : \kappa \vdash \Delta \]

  Subderivation

  By Lemma 43 (Instantiation Extension)

  Let \( \Theta = \cdot \).

  \( \equiv \)

  \[ \Gamma, \Theta \rightarrow \Delta \]

  By \( \Theta = \cdot \).

- Case

  \[ \alpha \notin \text{FV}([\Gamma]t) \quad (\alpha = t) \notin \Gamma \]

  \[ \Gamma / \alpha \equiv t : \kappa \vdash \Gamma, \alpha = t \]

  \[ \text{ElimeqUvarL} \]

  Let \( \Theta \) be \( (\alpha = t) \).

  \( \equiv \)

  \[ \Gamma, \alpha = t \rightarrow \Gamma, \alpha = t \]

  By Lemma 32 (Extension Reflexivity)

- Cases ElimeqInstR, ElimeqUvarR

  Similar to the respective \text{L} cases.

- Case

  \[ \sigma \not\equiv t \]

  \[ \text{ElimeqClash} \]

  The statement says that the output is a (consistent) context \( \Delta \), so this case is impossible.

\[ \Box \]
**Lemma 45** (Elimprop Extension).

*If* \( \Gamma \vdash P \rightarrow \Delta \) *then there exists* \( \Theta \) *such that* \( \Gamma, \Theta \rightarrow \Delta \).

**Proof.** By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context \( \Delta \).

- **Case** \( \Gamma / \sigma \vdash t : N \rightarrow \Delta \)
  
  \[
  \begin{align*}
  \Gamma / \sigma = t \vdash \Delta & \quad \text{ElimpropEq} \\
  \Gamma / \sigma = t \vdash \Delta & \quad \text{Subderivation} \\
  \end{align*}
  \]

  \( \Rightarrow \) \( \Gamma, \Theta \rightarrow \Delta \) By Lemma 44 (Elimeq Extension)

**Lemma 46** (Checkeq Extension).

*If* \( \Gamma \vdash A \equiv B \vdash \Delta \) *then* \( \Gamma \rightarrow \Delta \).

**Proof.** By induction on the given derivation.

- **Case** \( \Gamma \vdash u \equiv u : \kappa \rightarrow \Gamma \)
  
  \( \Rightarrow \) \( \Gamma \rightarrow \Theta \) By i.h.

- **Cases** CheckeqUnit, CheckeqZero Similar to the CheckeqVar case.

- **Case** \( \Gamma \vdash \tau_1 \equiv \tau_1' : \tau : \theta \quad \Theta \vdash [\Theta] \tau_2 \equiv [\Theta] \tau_2' : \tau : \Delta \)
  
  \[
  \begin{align*}
  \Gamma \vdash \tau_1 \odot \tau_2 \equiv \tau_1' \odot \tau_2' : \tau : \Delta & \quad \text{CheckeqBin} \\
  \end{align*}
  \]

  \( \Rightarrow \) \( \Gamma \rightarrow \Theta \) By i.h.

  \( \Theta \rightarrow \Delta \) By i.h.

  \( \Rightarrow \) \( \Gamma \rightarrow \Delta \) By Lemma 33 (Extension Transitivity)

- **Case** \( \Gamma \vdash \sigma \equiv t : N \rightarrow \Delta \)
  
  \( \Rightarrow \) \( \Gamma \rightarrow \Delta \) By i.h.

- **Case** \( \Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta \quad \hat{\alpha} \notin \text{FV}(\Gamma_0[\hat{\alpha}]) \)
  
  \[
  \begin{align*}
  \Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := t \vdash \kappa \rightarrow \Delta & \quad \text{CheckeqInstL} \\
  \end{align*}
  \]

  \( \Rightarrow \) \( \Gamma_0[\hat{\alpha}] \rightarrow \Delta \) By Lemma 43 (Instantiation Extension)

  \( \Rightarrow \) \( \Gamma_0[\hat{\alpha}] \rightarrow \Delta \) By Lemma 43 (Instantiation Extension)

  \( \Rightarrow \) \( \Gamma_0[\hat{\alpha}] \rightarrow \Delta \) By Lemma 43 (Instantiation Extension)

- **Case** CheckeqInstR Similar to the CheckeqInstL case.

**Lemma 47** (Checkprop Extension).

*If* \( \Gamma \vdash P \text{ true} \rightarrow \Delta \) *then* \( \Gamma \rightarrow \Delta \).

**Proof.** By induction on the given derivation.
C.5 Equivalence Extends

- Case \( \Gamma \vdash \sigma \equiv t : N \vdash \Delta \)
  \[ \Gamma \vdash t \triangleright \Delta \quad \text{{CheckpropEq}} \]
  \( \Gamma \vdash \sigma \equiv t : N \vdash \Delta \)
  \[ \text{Subderivation} \]
  \[ \Gamma \rightarrow \Delta \quad \text{By Lemma 46 (Checkeq Extension)} \]

Lemma 48 (Prop Equivalence Extension).
If \( \Gamma \vdash P \equiv Q \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).

Proof. By induction on the given derivation.

- Case
  \[ \Gamma \vdash \sigma_1 \equiv \tau_1 : N \vdash \Theta \]
  \[ \Theta \vdash \sigma_2 \equiv \tau_2 : N \vdash \Delta \]
  \[ \Gamma \vdash (\sigma_1 = \sigma_2) \equiv (\tau_1 = \tau_2) \vdash \Delta \quad \equiv \text{PropEq} \]
  \( \Gamma \vdash \sigma_1 \equiv \tau_1 : N \vdash \Theta \)
  \( \Theta \vdash \sigma_2 \equiv \tau_2 : N \vdash \Delta \)
  \[ \text{Subderivation} \]
  \[ \Gamma \rightarrow \Theta \quad \text{By Lemma 46 (Checkeq Extension)} \]
  \[ \Theta \rightarrow \Delta \quad \text{By Lemma 46 (Checkeq Extension)} \]
  \( \equiv \Gamma \rightarrow \Delta \quad \text{By Lemma 33 (Extension Transitivity)} \)

Lemma 49 (Equivalence Extension).
If \( \Gamma \vdash A \equiv B \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).

Proof. By induction on the given derivation.

- Case
  \( \Gamma \vdash \alpha \equiv \alpha \vdash \Gamma \) \quad \equiv \text{Var} \]
  Here \( \Delta = \Gamma \), so Lemma 32 (Extension Reflexivity) suffices.

- Case
  \( \Gamma \vdash \& \equiv \& \vdash \Gamma \) \quad \equiv \text{Exvar} \]
  Similar to the \( \equiv \text{Var} \) case.

- Case
  \( \Gamma \vdash 1 \equiv 1 \vdash \Gamma \) \quad \equiv \text{Unit} \]
  Similar to the \( \equiv \text{Var} \) case.

- Case
  \[ \Gamma \vdash A_1 \equiv B_1 \vdash \Theta \]
  \[ \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta \]
  \[ \Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \vdash \Delta \quad \equiv \oplus \]
  \( \Gamma \vdash A_1 \equiv B_1 \vdash \Theta \)
  \( \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta \)
  \[ \text{Subderivation} \]
  \[ \Gamma \rightarrow \Theta \quad \text{By i.h.} \]
  \[ \Theta \rightarrow \Delta \quad \text{By i.h.} \]
  \( \equiv \Gamma \rightarrow \Delta \quad \text{By Lemma 33 (Extension Transitivity)} \)

- Case \( \equiv \text{Vec} \)
  Similar to the \( \equiv \oplus \) case.
Proof of Lemma 49 (Equivalence Extension) lemequiv-extension

- Cases $\equiv \equiv \wedge$ Similar to the $\equiv \equiv$ case, but with Lemma 48 (Prop Equivalence Extension) on the first premise.

- Case $\Gamma, \alpha : \kappa \vdash A_0 \equiv B \vdash \Delta, \alpha : \kappa, \Delta'$
  $\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B \vdash \Delta$ $\equiv \forall$

  $\Gamma, \alpha : \kappa \vdash A_0 \equiv B \vdash \Delta, \alpha : \kappa, \Delta'$ Subderivation
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa, \Delta'$ By i.h.

  $\Gamma \rightarrow \Delta$ By Lemma 22 (Extension Inversion) (i)

- Case $\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := \tau : \ast \vdash \Delta \quad \hat{\alpha} \notin FV([\Gamma_0[\hat{\alpha}]]\tau)$ $\equiv \text{InstantiateL}$

  $\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := \tau : \ast \vdash \Delta$ Subderivation

  $\Gamma_0[\hat{\alpha}] \rightarrow \Delta$ By Lemma 43 (Instantiation Extension) (ii)

- Case $\equiv \text{InstantiateR}$ Similar to the $\equiv \text{InstantiateL}$ case.

C'.6 Subtyping Extends

Lemma 50 (Subtyping Extension). If $\Gamma \vdash A <:\tau B \vdash \Delta$ then $\Gamma \rightarrow \Delta$.

Proof. By induction on the given derivation.

- Case $\Gamma, \triangleright \hat{\alpha}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A <:\triangleright B \vdash \Delta, \triangleright \hat{\alpha}, \Theta$
  $\Gamma \vdash \forall \alpha : \kappa. A <:\triangleright B \vdash \Delta$ $\equiv \forall$

  $\Gamma, \triangleright \hat{\alpha}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A <:\triangleright B \vdash \Delta, \triangleright \hat{\alpha}, \Theta$ Subderivation
  $\Gamma, \triangleright \hat{\alpha}, \hat{\alpha} : \kappa \rightarrow \Delta, \triangleright \hat{\alpha}, \Theta$ By i.h. (i)

  $\Gamma \rightarrow \Delta$ By Lemma 22 (Extension Inversion) (ii)

- Case $\equiv \equiv \Rightarrow$ Similar to the $\equiv \forall \Rightarrow$ case.

- Case $\Gamma, \alpha : \kappa \vdash A <:\ast B \vdash \Delta, \alpha : \kappa, \Theta$
  $\Gamma \vdash A <:\ast \forall \alpha : \kappa. B \vdash \Delta$ $\equiv \forall$

  Similar to the $\equiv \forall \Rightarrow$ case, but using part (i) of Lemma 22 (Extension Inversion).

- Case $\equiv \equiv \Rightarrow$ Similar to the $\equiv \forall \Rightarrow$ case.

- Case $\Gamma \vdash A \equiv B \vdash \Delta$
  $\Gamma \vdash A <:\ast B \vdash \Delta$ $\equiv \text{Equiv}$

  $\Gamma \vdash A \equiv B \vdash \Delta$ Subderivation

  $\Gamma \rightarrow \Delta$ By Lemma 49 (Equivalence Extension)
C'.7 Typing Extends

Lemma 51 (Typing Extension).

If \( \Gamma \vdash e \triangleleft A \ p \vdash \Delta \)
or \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \)
or \( \Gamma \vdash s : A \ p \Rightarrow B \ q \vdash \Delta \)
or \( \Gamma \vdash \Pi :: A \triangleleft \ C \ p \vdash \Delta \)
then \( \Gamma \rightarrow \Delta \).

Proof. By induction on the given derivation.

- **Match judgments:**
  In rule \( \text{MatchEmpty} \), \( \Delta = \Gamma \), so the result follows by Lemma 32 (Extension Reflexivity).
  Rules \( \text{MatchBase} \), \( \text{Match} \times \) and \( \text{MatchWild} \) each have a single premise in which the contexts match the conclusion (input \( \Gamma \) and output \( \Delta \)), so the result follows by i.h. For rule \( \text{MatchSeq} \), Lemma 33 (Extension Transitivity) is also needed.
  In rule \( \text{Match}\exists \) apply the i.h., then use Lemma 22 (Extension Inversion) (i).
  \( \text{Match}\bot \): Immediate by Lemma 32 (Extension Reflexivity).
  \( \text{MatchUnify} \):
  \[
  \Gamma, P, \Theta', \Theta 
  \vdash \Theta 
  \text{By Lemma} 44 \text{ (Elimeq Extension)}
  \]
  \[
  \Theta 
  \vdash \Delta, P, \Delta' 
  \text{By i.h.}
  \]
  \[
  \Gamma, P, \Theta', \Delta, \Delta' 
  \vdash \Theta 
  \text{By Lemma} 33 \text{ (Extension Transitivity)}
  \]

- **Synthesis, checking, and spine judgments:** In rules \( \text{Var} \), \( \text{EmptySpine} \), and \( \text{\top} \), the output context \( \Delta \) is exactly \( \Gamma \), so the result follows by Lemma 32 (Extension Reflexivity).
  \- Case \( \forall \text{I} \): Use the i.h. and Lemma 33 (Extension Transitivity).
  \- Case \( \forall \text{Spine} \): By \( \rightarrow \text{Add} \), \( \Gamma \rightarrow \Gamma, \alpha : \kappa \).
  The result follows by i.h. and Lemma 33 (Extension Transitivity).
  \- Cases \( \wedge \text{Spine} \) Use Lemma 47 (Checkprop Extension), the i.h., and Lemma 33 (Extension Transitivity).
  \- Cases \( \forall \text{I} \) using reasoning found in the \( \forall \) and \( \forall \) cases.
  \- Case \( : \text{Cons} \) Using reasoning found in the \( \forall \) and \( \forall \) cases.
  \- Case \( : \text{I} \) Use the i.h.
  \- Case \( : \text{I} \)
    \[
    \Gamma, P, \Theta', \Theta 
    \vdash \Theta 
    \text{By Lemma} 45 \text{ (Elimprop Extension)}
    \]
    \[
    \Theta 
    \vdash \Delta, P, \Delta 
    \text{By i.h.}
    \]
    \[
    \Gamma, P, \Theta', \Delta, \Delta' 
    \vdash \Theta 
    \text{By Lemma} 33 \text{ (Extension Transitivity)}
    \]
  \- Cases \( \rightarrow \text{I} \) Use the i.h. and Lemma 22 (Extension Inversion).
  \- Cases \( \text{Sub} \), \( \text{Anno} \), \( \rightarrow \text{E} \), \( \rightarrow \text{Spine} \), \( \rightarrow \text{I} \), \( \times \) Use the i.h., and Lemma 33 (Extension Transitivity) as needed.
  \- Case \( \text{II}\times \) By Lemma 23 (Deep Evar Introduction) (ii).
  \- Case \( \times \text{Spine} \) By Lemma 23 (Deep Evar Introduction) (ii) twice, Lemma 23 (Deep Evar Introduction) (ii), the i.h., and Lemma 33 (Extension Transitivity).
Proof of **Lemma 51** (Typing Extension) \(\text{lem:typing-extension}\)

- **Case** \(\rightarrow \text{Id} \): Use Lemma \ref{lem:deep-evar-introduction} (Deep Evar Introduction) (i) twice, Lemma \ref{lem:deep-evar-introduction} (Deep Evar Introduction) (ii), the i.h. and Lemma \ref{lem:extension-inversion} (Extension Inversion) (v).

- **Case** \(\rightarrow \text{Var} \): Use the i.h. on the synthesis premise and the match premise, and then Lemma \ref{lem:extension-transitivity} (Extension Transitivity).

### C'.8 Unfiled

**Lemma 52** (Context Partitioning).

*If* \(\Delta, \triangleright \alpha, \Theta \rightarrow \Omega, \triangleright \alpha, \Omega_Z\) *then there is a* \(\Psi\) *such that* \([\Omega, \triangleright \alpha, \Omega_Z](\Delta, \triangleright \alpha, \Theta) = [\Omega][\Delta, \Psi]\).

**Proof.** By induction on the given derivation.

- **Case** \(\rightarrow \text{Id} \): Impossible: \(\Delta, \triangleright \alpha, \Theta \) cannot have the form \(\cdot\).

- **Case** \(\rightarrow \text{Var} \): We have \(\Omega_Z = (\Omega_Z', x : A)\) and \(\Theta = (\Theta', x : A')\). By i.h., there is \(\Psi'\) such that \([\Omega, \triangleright \alpha, \Omega_Z'](\Delta, \triangleright \alpha, \Theta') = [\Omega][\Delta, \Psi']\). Then by the definition of context application, \([\Omega, \triangleright \alpha, \Omega_Z', x : A](\Delta, \triangleright \alpha, \Theta', x : A') = [\Omega][\Delta, \Psi', x : [\Omega']A]\). Let \(\Psi = (\Psi', x : [\Omega']A)\).

- **Case** \(\rightarrow \text{Uvar} \): Similar to the \(\rightarrow \text{Var} \) case, with \(\Psi = (\Psi', \alpha : \kappa)\).

Broadly similar to the \(\rightarrow \text{Uvar} \) case, but the rightmost context element disappears in context application, so we let \(\Psi = \Psi'\).

**Lemma 54** (Completing Stability).

*If* \(\Gamma \rightarrow \Omega\) *then* \([\Omega][\Gamma] = [\Omega]\Omega\).

**Proof.** By induction on the derivation of \(\Gamma \rightarrow \Omega\).

- **Case** \(\rightarrow \text{Id} \):

  Immediate.

- **Case** \(\rightarrow \text{Var} \):

  Immediate.

- **Case** \(\rightarrow \text{Uvar} \):

  Similar to \(\rightarrow \text{Var} \).

- **Case** \(\rightarrow \text{Unsolved} \):

  Similar to \(\rightarrow \text{Var} \).
Proof of Lemma 54 (Completing Stability)

(iii) If $\Omega \rightarrow \Omega'$ and $\Omega \vdash t : \kappa$ then $[\Omega]t = [\Omega']t$.

(ii) If $\Omega \rightarrow \Omega'$ and $\Omega \vdash A$ type then $[\Omega]A = [\Omega']A$.

(iii) If $\Omega \rightarrow \Omega'$ then $[\Omega]\Omega = [\Omega']\Omega'$.

Proof.

• Part (i):

By Lemma 29 (Substitution Monotonicity) (i), $[\Omega']t = [\Omega'](\Omega)t$.

Now we need to show $[\Omega']t = [\Omega]t$. Considered as a substitution, $\Omega'$ is the identity everywhere except existential variables $\alpha$ and universal variables $\alpha$. First, since $\Omega$ is complete, $[\Omega]t$ has no free existentials. Second, universal variables free in $[\Omega]t$ have no equations in $\Omega$ (if they had, their occurrences would have been replaced). But if $\Omega$ has no equation for $\alpha$, it follows from $\Omega \rightarrow \Omega'$ and the definition of context extension in Figure 13 that $\Omega'$ also lacks an equation, so applying $\Omega \rightarrow \Omega'$ also leaves $\alpha$ alone.

Transitivity of equality gives $[\Omega']t = [\Omega]t$. 

• Case $\Gamma \rightarrow \Omega_0$

$[\Omega_0]t = [\Omega_0]t' \quad \Rightarrow \text{Solved}$

Similar to $\Rightarrow \text{Var}$

• Case $\Gamma \rightarrow \Omega_0$

$\Gamma_0, \alpha : t \rightarrow \Omega_0, \alpha : t' \quad \Rightarrow \text{Marker}$

Similar to $\Rightarrow \text{Var}$

• Case $\Gamma \rightarrow \Omega_0$

$\Gamma_0, \beta : \kappa' \rightarrow \Omega_0, \beta : \kappa' \quad \Rightarrow \text{Solve}$

Similar to $\Rightarrow \text{Var}$

• Case $\Gamma \rightarrow \Omega_0$

$\Gamma_0, \alpha = t' \rightarrow \Omega_0, \alpha = t \quad \Rightarrow \text{Eqn}$

Subderivation

$[\Omega_0]t' = [\Omega_0]t$ Subderivation

$[\Omega_0]t_0 = [\Omega_0]t_0$ By i.h.

$[[\Omega_0]t_0/\alpha](\Gamma_0) = [[\Omega_0]t_0/\alpha](\Gamma_0)$ By congruence of equality

$[\Omega_0, \alpha = t](\Gamma, \alpha = t') = [\Omega_0, \alpha = t](\Gamma, \alpha = t)$ By definition of context substitution

• Case $\Gamma \rightarrow \Omega_0$

$\Gamma \rightarrow \Omega_0, \alpha : \kappa \quad \Rightarrow \text{Add}$

Subderivation

$[\Omega_0]t = [\Omega_0]t$ By i.h.

$[\Omega_0, \alpha : \kappa](\Gamma) = [\Omega_0, \alpha : \kappa](\Omega_0, \alpha : \kappa)$ By definition of context substitution

• Case $\Gamma \rightarrow \Omega_0$

$\Gamma \rightarrow \Omega_0, \alpha : \kappa = t \quad \Rightarrow \text{AddSolved}$

Similar to the $\Rightarrow \text{Add}$ case.
Proof of Lemma 55 (Completing Completeness).

Part (ii): Similar to part (i), using Lemma 29 (Substitution Monotonicity) (iii) instead of (i).

Part (iii): By induction on the given derivation of \( \Delta \rightarrow \Omega \).

Only cases \( \rightarrow \text{Id} \rightarrow \text{Var} \rightarrow \text{UVar} \rightarrow \text{Eqn} \rightarrow \text{Solved} \rightarrow \text{AddSolved} \) and \( \rightarrow \text{Marker} \) are possible. In all of these cases, we use the i.h. and the definition of context application; in cases \( \rightarrow \text{Var} \rightarrow \text{Eqn} \) and \( \rightarrow \text{Solved} \) we also use the equality in the premise of the respective rule.

Lemma 56 (Confluence of Completeness).
If \( \Delta_1 \rightarrow \Omega \) and \( \Delta_2 \rightarrow \Omega \) then \( [\Omega] \Delta_1 = [\Omega] \Delta_2 \).

Proof.
\[
\Delta_1 \rightarrow \Omega \quad \text{Given}
\]
\[
[\Omega] \Delta_1 = [\Omega] \Omega \quad \text{By Lemma 54 (Completing Stability)}
\]
\[
\Delta_2 \rightarrow \Omega \quad \text{Given}
\]
\[
[\Omega] \Delta_2 = [\Omega] \Omega \quad \text{By Lemma 54 (Completing Stability)}
\]
\[
[\Omega] \Delta_1 = [\Omega] \Delta_2 \quad \text{By transitivity of equality}
\]

Lemma 57 (Multiple Confluence).
If \( \Delta \rightarrow \Omega \) and \( \Omega \rightarrow \Omega' \) and \( \Delta' \rightarrow \Omega' \) then \( [\Omega] \Delta = [\Omega'] \Delta' \).

Proof.
\[
\Delta \rightarrow \Omega \quad \text{Given}
\]
\[
[\Omega] \Delta = [\Omega] \Omega \quad \text{By Lemma 54 (Completing Stability)}
\]
\[
\Omega \rightarrow \Omega' \quad \text{Given}
\]
\[
[\Omega] \Omega = [\Omega'] \Omega' \quad \text{By Lemma 55 (Completing Completeness) (iii)}
\]
\[
= [\Omega'] \Delta' \quad \text{By Lemma 54 (Completing Stability) (\( \Delta' \rightarrow \Omega' \) given)}
\]

Lemma 59 (Canonical Completion).
If \( \Gamma \rightarrow \Omega \) then there exists \( \Omega_{\text{canon}} \) such that \( \Gamma \rightarrow \Omega_{\text{canon}} \) and \( \Omega_{\text{canon}} \rightarrow \Omega \) and \( \text{dom}(\Omega_{\text{canon}}) = \text{dom}(\Gamma) \) and, for all \( \hat{\kappa} : \kappa = \tau \) and \( \hat{\alpha} = \tau \) in \( \Omega_{\text{canon}} \), we have \( \text{FEV}(\tau) = \emptyset \).

Proof. By induction on \( \Omega \). In \( \Omega_{\text{canon}} \), make all solutions (for evars and uvars) canonical by applying \( \Omega \) to them, dropping declarations of existential variables that aren't in \( \text{dom}(\Gamma) \).

Lemma 60 (Split Solutions).
If \( \Delta \rightarrow \Omega \) and \( \hat{\kappa} \in \text{unsolved}(\Delta) \) then there exists \( \Omega_1 = \Omega_2' [\hat{\kappa} : \kappa = t_1] \) such that \( \Omega_1 \rightarrow \Omega \) and \( \Omega_2 = \Omega_2' [\hat{\kappa} : \kappa = t_2] \) where \( \Delta \rightarrow \Omega_2 \) and \( t_2 \neq t_1 \) and \( \Omega_2 \) is canonical.

Proof. Use Lemma 59 (Canonical Completion) to get \( \Omega_{\text{canon}} \) such that \( \Delta \rightarrow \Omega_{\text{canon}} \) and \( \Omega_{\text{canon}} \rightarrow \Omega \), where for all solutions \( t \) in \( \Omega_{\text{canon}} \) we have \( \text{FEV}(t) = \emptyset \).

We have \( \Omega_{\text{canon}} = \Omega_2' [\hat{\kappa} : \kappa = t_1] \), where \( \text{FEV}(t_1) = \emptyset \). Therefore \( \Omega_1 [\hat{\kappa} : \kappa = t_1] \rightarrow \Omega \).

Now choose \( t_2 \) as follows:

- If \( \kappa = * \), let \( t_2 = t_1 \rightarrow t_1 \).
- If \( \kappa = \mathbb{N} \), let \( t_2 = \text{suc}(t_1) \).

Thus, \( t_2 \neq t_1 \). Let \( \Omega_2 = \Omega_2' [\hat{\kappa} : \kappa = t_2] \).

Proving \( \Delta \rightarrow \Omega_2 \) by Lemma 31 (Split Extension)
Lemma 61 (Interpolating With and Exists).

(1) If $D :: \Psi \vdash \Pi :: \vec{A} \leftarrow \mathit{C_p}$ and $\Psi \vdash P_0 \text{true}$
then $D' :: \Psi \vdash \Pi :: \vec{A} \leftarrow \mathit{C \land P_0}$.

(2) If $D :: \Psi \vdash \Pi :: \vec{A} \leftarrow \lbrack \tau/\alpha \rbrack C_0 p$ and $\Psi \vdash \tau : \kappa$
then $D' :: \Psi \vdash \Pi :: \vec{A} \leftarrow \lbrack \exists \alpha : \kappa.C_0 \rbrack p$.

In both cases, the height of $D'$ is one greater than the height of $D$.
Moreover, similar properties hold for the eliminating judgment $\Psi / P \vdash \Pi :: \vec{A} \leftarrow \mathit{C_p}$.

Proof. By induction on the given match derivation.

In the DeclMatchBase case, for part (1), apply rule $\land I$. For part (2), apply rule $\exists I$.

In the DeclMatchNeg case, part (1), use Lemma 2 (Declarative Weakening) (iii). In part (2), use Lemma 2 (Declarative Weakening) (i).

Lemma 62 (Case Invertibility).

If $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow \mathit{A !}$ and $\Psi \vdash \Pi :: \vec{A} \leftarrow \mathit{C_p}$ and $\Psi \vdash \Pi$ covers $\vec{A}$
where the height of each resulting derivation is strictly less than the height of the given derivation.

Proof. By induction on the given derivation.

- Case $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A q$

  pol($B$) $\vdash \Psi \leq^* AB$

  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow B p$

  by DeclSub

  Impossible, because $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A q$ is not derivable.

- Cases Decl$\forall$ [Decl$\exists$]

  Impossible: these rules have a value restriction, but a case expression is not a value.

- Case $\Psi \vdash P \text{true}$

  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C_0 p$

  by Decl$\land$

  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C_0 \land P p$

  by i.h.

- Cases Decl$11$, Decl$\Rightarrow$, DeclRec, Decl$+$, Decl$\times$, DeclNil, DeclCons

  Impossible, because in these rules $e$ cannot have the form $\text{case}(e_0, \Pi)$.

- Case $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A !$

  $\Psi \vdash \Pi :: \vec{A} \leftarrow \mathit{C_p}$

  $\Psi \vdash \Pi \text{ covers } \vec{A}$

  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C p$

  by DeclCase

  Immediate.
E’  Miscellaneous Properties of the Algorithmic System

Lemma 63 (Well-Formed Outputs of Typing).

(Spines) If \( \Gamma \vdash s : A \triangleright C p \vdash \Delta \) or \( \Gamma \vdash A q \triangleright C [p] \vdash \Delta \)
and \( \Gamma \vdash A q \triangleright C \) p type
then \( \Delta \vdash C \) p type.

(Synthesis) If \( \Gamma \vdash e \Rightarrow A p \vdash \Delta \)
then \( A \vdash p \) type.

Proof. By induction on the given derivation.

- Case \( \text{Anno} \): Use Lemma 51 (Typing Extension) and Lemma 41 (Extension Weakening for Principal Typing).
- Case \( \forall \text{Spine} \): We have \( \Gamma \vdash (\forall \alpha : \kappa. A_0) q \triangleright C \) p type.
  By inversion, \( \Gamma, \alpha : \kappa \vdash A_0 q \triangleright C \) p type.
  By properties of substitution, \( \Gamma, ^\alpha \alpha : \kappa \vdash [^\alpha \alpha] A_0 \triangleright C \) p type.
  Now apply the i.h.
- Case \( \Rightarrow \text{Spine} \): Use Lemma 42 (Inversion of Principal Typing) (2), Lemma 47 (Checkprop Extension), and Lemma 41 (Extension Weakening for Principal Typing).
- Case \( \text{SpineRecover} \): By i.h., \( \Delta \vdash C \) p type.
  We have as premise \( \text{FEV}(C) = \emptyset \).
  Therefore \( \Delta \vdash C \) p type.
- Case \( \text{SpinePass} \): By i.h.
- Case \( \text{EmptySpine} \): Immediate.
- Case \( \rightarrow \text{Spine} \): Use Lemma 42 (Inversion of Principal Typing) (1), Lemma 51 (Typing Extension), and Lemma 41 (Extension Weakening for Principal Typing).
- Case \( ^\alpha \text{Spine} \): Show that \( ^\alpha \alpha \rightarrow ^\alpha \beta \) is well-formed, then use the i.h.

F’  Decidability of Instantiation

Lemma 64 (Left Unsolvedness Preservation).

If \( \Gamma_0, \hat{\alpha}, \Gamma_1 \vdash \hat{\alpha} := \Lambda : \kappa \vdash \Delta \) and \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \) then \( \hat{\beta} \in \text{unsolved}(\Delta) \).

Proof. By induction on the given derivation.

- Case \( \text{InstSolve} \)

\[
\begin{aligned}
\Gamma_0 \vdash \tau : \kappa & \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa & \vdash \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \\Delta
\end{aligned}
\]

Immediate, since to the left of \( \hat{\alpha} \), the contexts \( \Delta \) and \( \Gamma \) are the same.

- Case \( \text{InstReach} \)

\[
\begin{aligned}
\hat{\beta} \in \text{unsolved}(\Gamma'[^\hat{\alpha} : \kappa][\hat{\beta} : \kappa]) & \\
\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} := \hat{\beta} : \kappa & \vdash \Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}] \\Delta
\end{aligned}
\]

Immediate, since to the left of \( \hat{\alpha} \), the contexts \( \Delta \) and \( \Gamma \) are the same.
Proof of Lemma 64 (Left Unsolvedness Preservation). We have \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \). Therefore \( \hat{\beta} \in \text{unsolved}(\Gamma_0, \hat{\alpha}_2 : *) \).
Clearly, \( \hat{\alpha}_2 \in \text{unsolved}(\Gamma_0, \hat{\alpha}_2 : *) \).
We have two subderivations:

\[
\begin{align*}
\Gamma_0, \hat{\alpha}_2 : *, \hat{\alpha}_1 : *, \hat{\alpha} : * &\vdash \hat{\alpha}_1 : * + \Theta \quad \Theta \vdash \hat{\alpha}_2 : [\Theta] \tau_2 : * + \Delta \\
\Gamma_0, \hat{\alpha} : *, \Gamma_1 &\vdash \hat{\alpha} : \tau_1 + \tau_2 : * + \Delta
\end{align*}
\]

By induction on (1), \( \hat{\beta} \in \text{unsolved}(\Theta) \).
Also by induction on (1), with \( \hat{\alpha}_2 \) playing the role of \( \hat{\beta} \), we get \( \hat{\alpha}_2 \in \text{unsolved}(\Theta) \).
Since \( \hat{\beta} \in \Gamma_0 \), it is declared to the left of \( \hat{\alpha}_2 \) in \( \Gamma_0, \hat{\alpha}_2 : * \).
Hence by Lemma 20 (Declaration Order Preservation), \( \hat{\beta} \) is declared to the left of \( \hat{\alpha}_2 \) in \( \Theta \).
That is, \( \Theta = (\Theta_0, \hat{\alpha}_2 : *, \Theta_1) \), where \( \hat{\beta} \in \text{unsolved}(\Theta_0) \).
By induction on (2), \( \hat{\beta} \in \text{unsolved}(\Delta) \).

\[\Box\]

Lemma 65 (Left Free Variable Preservation). If \( \Gamma, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} : t : \kappa \rightarrow \Delta \) and \( \Gamma \vdash s : \kappa' \) and \( \hat{\alpha} \notin \text{FV}(\Gamma) \) and \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \) and \( \hat{\beta} \notin \text{FV}(\Gamma) \), then \( \hat{\beta} \notin \text{FV}(\Delta) \).

Proof. By induction on the given instantiation derivation.

\[\Box\]

Proof of Lemma 65 (Left Free Variable Preservation).
We have \( \Gamma \vdash \sigma \) type and \( \hat{\alpha} \not\in FV(\Gamma|\sigma) \) and \( \hat{\beta} \not\in FV(\Gamma|\sigma) \).
By weakening, we get \( \Gamma' \vdash \sigma : \kappa' \); since \( \hat{\alpha} \not\in FV(\Gamma|\sigma) \) and \( \Gamma' \) only adds a solution for \( \hat{\alpha} \), it follows that \( [\Gamma']|\sigma = [\Gamma]|\sigma \).
Therefore \( \hat{\alpha}_1 \notin FV([\Gamma']|\sigma) \) and \( \hat{\alpha}_2 \notin FV([\Gamma']|\sigma) \) and \( \hat{\beta} \notin FV([\Gamma']|\sigma) \).
Since we have \( \hat{\beta} \in \Gamma_0 \), we also have \( \hat{\beta} \in (\Gamma_0, \hat{\alpha}_2 : \ast) \).
By induction on the first premise, \( \hat{\beta} \notin FV((\Theta|\sigma)) \).
Also by induction on the first premise, with \( \hat{\alpha}_2 \) playing the role of \( \hat{\beta} \), we have \( \hat{\alpha}_2 \notin FV((\Theta|\sigma)) \).
Note that \( \hat{\alpha}_2 \) is unsolved(\(\Gamma_0, \hat{\alpha}_2 : \ast\)).
By Lemma 64 (Left Unsolvedness Preservation), \( \hat{\alpha}_2 \in \text{unsolved}(\Theta) \).
Therefore \( \Theta \) has the form (\(\Theta_0, \hat{\alpha}_2 : \ast, \Theta_1\)).
Since \( \hat{\beta} \neq \hat{\alpha}_2 \), we know that \( \hat{\beta} \) is declared to the left of \( \hat{\alpha}_2 \) in (\(\Gamma_0, \hat{\alpha}_2 : \ast\)), so by Lemma 20 (Declaration Order Preservation), \( \hat{\beta} \) is declared to the left of \( \hat{\alpha}_2 \) in \( \Theta \). Hence \( \hat{\beta} \in \Theta_0 \).
Furthermore, by Lemma 43 (Instantiation Extension), we have \( \Gamma' \rightarrow \Theta \).
Then by Lemma 36 (Extension Weakening (Sorts)), we have \( \Delta \vdash \sigma : \kappa' \).
Using induction on the second premise, \( \hat{\beta} \notin FV([\Delta]|\sigma) \).

**Case**

\[
\frac{\Gamma' [\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \text{zero} : \mathbb{N} \rightarrow \Gamma' [\hat{\alpha} : \mathbb{N} = \text{zero}] }{\Delta \vdash \hat{\alpha} : \text{zero} \rightarrow \Delta} \quad \text{InstZero}
\]

We have \( \hat{\alpha} \notin FV([\Gamma]|\sigma) \). Since \( \Delta \) differs from \( \Gamma \) only in \( \hat{\alpha} \), it must be the case that \( [\Gamma]|\sigma = [\Delta]|\sigma \). It is given that \( \hat{\beta} \notin FV([\Gamma]|\sigma) \), so \( \hat{\beta} \notin FV([\Delta]|\sigma) \).

**Case**

\[
\frac{\Gamma [\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \text{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : \mathbb{N} \rightarrow \Delta }{\Gamma' [\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \text{succ}(t_1) : \mathbb{N} \rightarrow \Delta} \quad \text{InstSucc}
\]

\[
\begin{align*}
\Gamma & \vdash \sigma : \kappa' & \text{Given} \\
\Theta & \vdash \sigma : \kappa' & \text{By weakening} \\
\hat{\alpha} & \not\in FV([\Gamma]|\sigma) & \text{Given} \\
\hat{\alpha} & \not\in FV((\Theta|\sigma)) & \text{\(\hat{\alpha} \not\in FV([\Gamma]|\sigma) \) and \(\Theta\) only solves \(\hat{\alpha}\)} \\
\Theta & = (\Gamma_0, \hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \text{succ}(\hat{\alpha}_1), \Gamma_1) & \text{Given} \\
\hat{\beta} & \not\in \text{unsolved}(\Gamma_0) & \text{Given} \\
\hat{\beta} & \not\in \text{unsolved}(\Gamma_0, \hat{\alpha}_1 : \mathbb{N}) & \text{\(\hat{\alpha}_1\) fresh} \\
\hat{\beta} & \not\in FV([\Gamma]|\sigma) & \text{Given} \\
\hat{\beta} & \not\in FV((\Theta|\sigma)) & \text{\(\hat{\alpha}_1\) fresh} \\
\hat{\beta} & \not\in FV([\Delta]|\sigma) & \text{By i.h.} \\
\end{align*}
\]

**Lemma 66** (Instantiation Size Preservation). If \( \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Delta \) and \( \Gamma \vdash \sigma : \kappa' \) and \( \hat{\alpha} \notin FV([\Gamma]|s) \), then \( ||[\Gamma]|s|| = ||[\Delta]|s|| \), where \( |C| \) is the plain size of the term \( C \).

**Proof.** By induction on the given derivation.

**Case**

\[
\frac{\Gamma \vdash \tau : \kappa }{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Delta, \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1} \quad \text{InstSolve}
\]
Since $\Delta$ differs from $\Gamma$ only in solving $\check{\alpha}$, and we know $\check{\alpha} \notin FV([\Gamma]\sigma)$, we have $|[\Delta]\sigma = |[\Gamma]\sigma$; therefore $|[\Delta]\sigma = |[\Gamma]\sigma$.

- Case $\Gamma'\vdash \check{\alpha} : N \iff \check{\alpha} := \text{zero} : N \rightarrow \Gamma''[\check{\alpha} : N = \text{zero}]$ $\Delta$ InstZero

Similar to the InstSolve case.

- Case $\Gamma'\vdash \check{\beta} \in \text{unsolved}(\Gamma'[\check{\alpha} : \kappa][\check{\beta} : \kappa])$

$\Gamma'[\check{\alpha} : \kappa][\check{\beta} : \kappa] \vdash \check{\alpha} := \check{\beta} : \kappa \rightarrow \Gamma''[\check{\alpha} : \kappa][\check{\beta} : \kappa = \check{\alpha}]$ $\Delta$ InstReach

Here, $\Delta$ differs from $\Gamma$ only in solving $\check{\beta}$ to $\check{\alpha}$. However, $\check{\alpha}$ has the same size as $\check{\beta}$, so even if $\check{\beta} \in FV([\Gamma]\sigma)$, we have $|[\Delta]\sigma = |[\Gamma]\sigma$.

- Case $\Gamma'\vdash \check{\alpha}_2 : *, \check{\alpha}_1 : *, \check{\alpha} : \check{\alpha}_1 + \check{\alpha}_2 \vdash \check{\alpha}_1 := \tau_1 : * \rightarrow \Theta \vdash \check{\alpha}_2 := [\Theta]_{\tau_2} : * \rightarrow \Delta$

$\Gamma[\check{\alpha} : \ast] \vdash \check{\alpha} := \tau_1 \oplus \tau_2 : * \rightarrow \Delta$ InstBin

We have $\Gamma \vdash \sigma : \kappa'$ and $\check{\alpha} \notin FV([\Gamma]\sigma)$.

Since $\check{\alpha}_1, \check{\alpha}_2 \notin \text{dom}(\Gamma)$, we have $\check{\alpha}, \check{\alpha}_1, \check{\alpha}_2 \notin FV([\Gamma]\sigma)$.

By Lemma 23 (Deep Evar Introduction), $\Gamma[\check{\alpha} : \ast] \rightarrow \Gamma'$.

By Lemma 36 (Extension Weakening (Sorts)), $\Gamma' \vdash \sigma : \kappa'$.

Since $\check{\alpha} \notin FV(\sigma)$, it follows that $|[\Gamma']\sigma = |[\Gamma]\sigma$, and so $|[\Gamma']\sigma = |[\Gamma]\sigma$.

By induction on the first premise, $|[\Gamma']\sigma = |[\Theta]\sigma$.

By Lemma 20 (Declaration Order Preservation), since $\check{\alpha}_2$ is declared to the left of $\check{\alpha}_1$ in $\Gamma'$, we have that $\check{\alpha}_2$ is declared to the left of $\check{\alpha}_1$ in $\Theta$.

By Lemma 64 (Left Unsolvedness Preservation), since $\check{\alpha}_2 \in \text{unsolved}(\Gamma')$, it is unsolved in $\Theta$: that is, $\Theta = (\Theta_1, \check{\alpha}_2 : \ast, \Theta_1)$.

By Lemma 43 (Instantiation Extension), we have $\Gamma' \rightarrow \Theta$.

By Lemma 36 (Extension Weakening (Sorts)), $\Theta \vdash \sigma : \kappa'$.

Since $\check{\alpha}_2 \notin FV([\Gamma']\sigma)$, Lemma 65 (Left Free Variable Preservation) gives $\check{\alpha}_2 \notin FV([\Theta]\sigma)$.

By induction on the second premise, $|[\Theta]\sigma = |[\Delta]\sigma|$, and by transitivity of equality, $|[\Gamma']\sigma = |[\Delta]\sigma|$.

- Case $\Gamma'\vdash \check{\alpha}_1 : N, \check{\alpha} : N = \text{succ}(\check{\alpha}_1) \vdash \check{\alpha}_1 := \text{t}_1 : N \rightarrow \Delta$

$\Gamma[\check{\alpha} : \ast] \vdash \check{\alpha} := \text{succ}(\text{t}_1) : N \rightarrow \Delta$ InstSucc

Given $\check{\alpha} \notin [\Gamma[\check{\alpha} : \ast]]\sigma$.

By Lemma 23 (Deep Evar Introduction), $\Gamma' \vdash \sigma : \kappa'$.

By Lemma 36 (Extension Weakening (Sorts)), $|[\Gamma']\sigma = |[\Gamma[\check{\alpha} : \ast]]\sigma$.

By congruence of equality, $\check{\alpha}_1 \notin [\Gamma']\sigma$.

By i.h., $|[\Gamma']\sigma = |[\Theta]\sigma|$, and $\check{\alpha}_1 \notin \text{dom}(\Gamma[\check{\alpha} : \ast])$.

By transitivity of equality.

\[\blacksquare\]

**Lemma 67** (Decidability of Instantiation). If $\Gamma = \Gamma_0[\check{\alpha} : \kappa']$ and $\Gamma \vdash t : \kappa$ such that $|[\Gamma]t = t$ and $\check{\alpha} \notin FV(t)$, then:
(1) Either there exists $\Delta$ such that $\Gamma_0[\hat{\alpha} : \kappa'] \vdash \hat{\alpha} := t : \kappa \vdash \Delta$, or not.

Proof. By induction on the derivation of $\Gamma \vdash t : \kappa$.

- **Case** $(u : \kappa) \in \Gamma$
  
  $\begin{array}{c}
  \Gamma_L, \hat{\alpha} : \kappa', \Gamma_R \vdash u : \kappa \\
  \end{array}$

  **VarSort**

  If $\kappa \neq \kappa'$, no rule matches and no derivation exists.
  Otherwise:
  - If $(u : \kappa) \in \Gamma_L$, we can apply rule **InstSolve**.
  - If $u$ is some unsolved existential variable $\hat{\beta}$ and $(\hat{\beta} : \kappa) \in \Gamma_R$, then we can apply rule **InstReach**.
  - Otherwise, $u$ is declared in $\Gamma_R$ and is a universal variable; no rule matches and no derivation exists.

- **Case** $(\hat{\beta} : \kappa = \tau) \in \Gamma$
  
  $\begin{array}{c}
  \Gamma \vdash \hat{\beta} : \kappa \\
  \end{array}$

  **SolvedVarSort**

  By inversion, $(\hat{\beta} : \kappa = \tau) \in \Gamma$, but $[\Gamma]\hat{\beta} = \hat{\beta}$ is given, so this case is impossible.

- **Case** **UnitSort**
  
  If $\kappa' = \star$, then apply rule **InstSolve**. Otherwise, no rule matches and no derivation exists.

- **Case** $\Gamma \vdash \tau_1 : \star$, $\Gamma \vdash \tau_2 : \star$
  
  $\begin{array}{c}
  \Gamma_L, \hat{\alpha} : \kappa', \Gamma_R \vdash \tau_1 \oplus \tau_2 : \star \\
  \end{array}$

  **BinSort**

  If $\kappa' \neq \star$, then no rule matches and no derivation exists. Otherwise:
  Given, $[\Gamma][\tau_1 \oplus \tau_2] = \tau_1 \oplus \tau_2$ and $\hat{\alpha} \notin FV([\Gamma][\tau_1 \oplus \tau_2])$.
  If $\Gamma_L \vdash \tau_1 \oplus \tau_2 : \star$, then we have a derivation by **InstSolve**.
  If not, the only other rule whose conclusion matches $\tau_1 \oplus \tau_2$ is **InstBin**.
  First, consider whether $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_R \vdash \hat{\alpha} := t : \star \vdash \Delta$ is decidable.
  
  By definition of substitution, $[\Gamma][\tau_1 \oplus \tau_2] = ([\Gamma][\tau_1] \oplus ([\Gamma][\tau_2])$. Since $[\Gamma][\tau_1 \oplus \tau_2] = \tau_1 \oplus \tau_2$, we have $[\Gamma][\tau_1] = \tau_1$ and $[\Gamma][\tau_2] = \tau_2$.
  
  By weakening, $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_R \vdash \tau_1 \oplus \tau_2 : \star$.
  
  Since $\Gamma \vdash \tau_1 : \star$ and $\Gamma \vdash \tau_2 : \star$, we have $\hat{\alpha}_1, \hat{\alpha}_2 \notin FV(\tau_1) \cup FV(\tau_2)$.
  
  Since $\hat{\alpha} \notin FV(t) \supseteq FV(\tau_1)$, it follows that $[\Gamma'][\tau_1] = \tau_1$.
  
  By i.h., either there exists $\Theta$ s.t. $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_R \vdash \hat{\alpha} := \tau_1 : \star \vdash \Theta$, or not.
  
  If not, then no derivation by **InstBin** exists.
  Otherwise, there exists such a $\Theta$. By Lemma 64 (Left Unsolvedness Preservation), we have $\hat{\alpha}_2 \in$ unsolved$(\Theta)$.
  
  By Lemma 65 (Left Free Variable Preservation), we know that $\hat{\alpha}_2 \notin FV([\Theta][\tau_2])$.
  
  Substitution is idempotent, so $[\Theta][\tau_2] = [\Theta][\tau_2]$.
  
  By i.h., either there exists $\Delta$ such that $\Theta \vdash \hat{\alpha}_2 := [\Theta][\tau_2] : \kappa \vdash \Delta$, or not.
  
  If not, no derivation by **InstBin** exists.
  Otherwise, there exists such a $\Delta$. By rule **InstBin** we have $\Gamma \vdash \hat{\alpha} := t : \kappa \vdash \Delta$.

- **Case** $\Gamma \vdash \mathbf{zero} : \mathbb{N}$
  
  **ZeroSort**

  If $\kappa' \neq \mathbb{N}$, then no rule matches and no derivation exists. Otherwise, apply rule **InstSolve**.
Proof.

By induction on

If \( \kappa' \neq \mathbb{N} \), then no rule matches and no derivation exists. Otherwise:
If \( \Gamma \vdash \text{succ}(t_0) : \mathbb{N} \), then we have a derivation by \text{InstSucc}.
If not, the only other rule whose conclusion matches succ\((t_0)\) is \text{InstSucc}.
The remainder of this case is similar to the \text{BinSort} case, but shorter.

\[ \square \]

**G' Separation**

**Lemma 68** (Transitivity of Separation).

If \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\kappa} (\Theta_L \ast \Theta_R) \) and \( (\Theta_L \ast \Theta_R) \xrightarrow{\kappa'} (\Delta_L \ast \Delta_R) \)
then \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\kappa'} (\Delta_L \ast \Delta_R) \).

**Proof.**

\[
\begin{align*}
(\Gamma_L \ast \Gamma_R) & \xrightarrow{\kappa} (\Theta_L \ast \Theta_R) \quad \text{Given} \\
(\Gamma_L, \Gamma_R) & \xrightarrow{\kappa} (\Theta_L, \Theta_R) \quad \text{By Definition 5} \\
\Gamma_L & \subseteq \Theta_L \text{ and } \Gamma_R \subseteq \Theta_R \\
(\Theta_L \ast \Theta_R) & \xrightarrow{\kappa'} (\Delta_L \ast \Delta_R) \quad \text{Given} \\
(\Theta_L, \Theta_R) & \xrightarrow{\kappa'} (\Delta_L, \Delta_R) \quad \text{By Definition 5} \\
\Theta_L & \subseteq \Delta_L \text{ and } \Theta_R \subseteq \Delta_R \\
(\Gamma_L, \Gamma_R) & \xrightarrow{\kappa} (\Delta_L, \Delta_R) \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma_L & \subseteq \Delta_L \text{ and } \Gamma_R \subseteq \Delta_R \\
\text{\#} & (\Gamma_L \ast \Gamma_R) \xrightarrow{\kappa'} (\Delta_L \ast \Delta_R) \quad \text{By Definition 5}
\end{align*}
\]

**Lemma 69** (Separation Truncation).

If \( H \) has the form \( \alpha : \kappa \text{ or } \mathbf{\uparrow} \mathbf{\alpha} \text{ or } \mathbf{\uparrow} \mathbf{p} \text{ or } \mathbf{x} : \mathbf{\Lambda} \mathbf{p} \)
and \( (\Gamma_L \ast (\Gamma_R, H)) \xrightarrow{\kappa} (\Delta_L \ast \Delta_R) \)
then \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\kappa} (\Delta_L \ast \Delta_R) \) where \( \Delta_R = (\Delta_0, H, \varnothing) \).

**Proof.** By induction on \( \Delta_R \).

If \( \Delta_R = (\ldots, H) \), we have \( (\Gamma_L \ast \Gamma_R, H) \xrightarrow{\kappa} (\Delta_L \ast (\Delta, H)) \), and inversion on \( \mathbf{\uparrow} \mathbf{\uparrow} \mathbf{u} \mathbf{v} \mathbf{a} \mathbf{r} \) if \( H \) is \( (\alpha : \kappa) \), or the corresponding rule for other forms) gives the result (with \( \varnothing = \varnothing \)).

Otherwise, proceed into the subderivation of \( (\Gamma_L, \Gamma_R, \alpha : \kappa) \xrightarrow{\kappa'} (\Delta_L, \Delta_R) \), with \( \Delta_R = (\Delta', \Delta') \) where \( \Delta' \) is a single declaration. Use the i.h. on \( \Delta'_R \), producing some \( \Theta' \). Finally, let \( \Theta = (\Theta', \Delta') \).

**Lemma 70** (Separation for Auxiliary Judgments).

(i) If \( \Gamma_L \ast \Gamma_R \vdash \sigma \models \tau : \kappa \vdash \Delta \)
and \( \text{FEV}(\sigma) \cup \text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\kappa} (\Delta_L \ast \Delta_R) \).

(ii) If \( \Gamma_L \ast \Gamma_R \vdash \mathbf{p} \text{ true } \vdash \Delta \)
and \( \text{FEV}(\mathbf{p}) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\kappa} (\Delta_L \ast \Delta_R) \).

(iii) If \( \Gamma_L \ast \Gamma_R \vdash \sigma \models \tau : \kappa \vdash \Delta \)
and \( \text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset \)
then \( \Delta = (\Delta_L \ast (\Delta_R, \varnothing)) \) and \( (\Gamma_L \ast (\Gamma_R, \varnothing)) \xrightarrow{\kappa} (\Delta_L \ast \Delta_R) \).

(iv) If \( \Gamma_L \ast \Gamma_R \vdash \mathbf{p} \vdash \Delta \)
and \( \text{FEV}(\mathbf{p}) = \emptyset \)
then \( \Delta = (\Delta_L \ast (\Delta_R, \varnothing)) \) and \( (\Gamma_L \ast (\Gamma_R, \varnothing)) \xrightarrow{\kappa} (\Delta_L \ast \Delta_R) \).
Lemma 71 (Separation for Subtyping). If $\Gamma_L * \Gamma_R \vdash A \triangleleft : \pm B \vdash \Delta$
and $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$
and $\text{FEV}(B) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L * \Gamma_R) \not\vdash (\Delta_L \ast \Delta_R)$.

Proof. Part (i): By induction on the derivation of the given checkeq judgment. Cases $\text{CheckeqVar}$ and $\text{CheckeqUnit}$ are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). For case $\text{CheckeqSucc}$ apply the i.h. For cases $\text{CheckeqInstL}$ and $\text{CheckeqInstR}$ use the i.h. (v). For case $\text{CheckeqBin}$ use reasoning similar to that in the $\not\vdash$ case of Lemma 72 (Separation—Main) (transitivity of separation, and applying $\Theta$ in the second premise).

Part (ii), checkprop: Use the i.h. (i).

Part (iii), elimeq: Cases $\text{ElimeqUvarRef}$, $\text{ElimeqUnit}$, and $\text{CheckeqZero}$ are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). For case $\text{ElimeqSucc}$ apply the i.h. For cases $\text{ElimeqInstL}$ and $\text{ElimeqInstR}$ use the i.h. (vi). For case $\text{ElimeqBin}$ is similar to the case $\text{CheckeqBin}$ in part (i).

Part (iv), elimprop: Use the i.h. (iii).

Part (v), instjudg:

- **Case** $\text{InstSolve}$: Here, $\Gamma = (\Gamma_0, \hat{x} : \kappa, \Gamma_1)$ and $\Delta = (\Gamma_0, \hat{x} : \kappa, \Gamma_1)$. We have $\hat{x} \in \text{dom}(\Gamma_R)$, so the declaration $\hat{x} : \kappa$ is in $\Gamma_R$. Since $\text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$, the context $\Delta$ maintains the separation.

- **Case** $\text{InstReach}$: Here, $\Gamma = \Gamma_0[\hat{x} : \kappa][\hat{\beta} : \kappa]$ and $\Delta = \Gamma_0[\hat{x} : \kappa][\hat{\beta} : \kappa]$. We have $\hat{x} \in \text{dom}(\Gamma_R)$, so the declaration $\hat{x} : \kappa$ is in $\Gamma_R$. Since $\hat{\beta}$ is declared to the right of $\hat{x}$, it too must be in $\Gamma_R$, which can also be shown from $\text{FEV}(\beta) \subseteq \text{dom}(\Gamma_R)$. Both declarations are in $\Gamma_R$, so the context $\Delta$ maintains the separation.

- **Case** $\text{InstZero}$: In this rule, $\Delta$ is the same as $\Gamma$ except for a solution zero, which doesn’t violate separation.

- **Case** $\text{InstSucc}$: The result follows by i.h., taking care to keep the declaration $\hat{x}_1 : \mathbb{N}$ on the right when applying the i.h., even if $\hat{x} : \mathbb{N}$ is the leftmost declaration in $\Gamma_R$, ensuring that $\text{succ}(\hat{x}_1)$ does not violate separation.

- **Case** $\text{InstBin}$: As in the $\text{InstSucc}$ case, the new declarations should be kept on the right-hand side of the separator. Otherwise the case is straightforward (using the i.h. twice and the transitivity).

Part (vi), propequivjudg: Similar to the $\text{CheckeqBin}$ case of part (i), using the i.h. (i).

Part (vii), equivjudg:

- **Cases** $\equiv \equiv \equiv \equiv$ : Immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$).

- **Case** $\equiv \equiv$ : Similar to the case $\text{CheckeqBin}$ in part (i).

- **Case** $\equiv \equiv$ : Similar to the case $\text{CheckeqBin}$ in part (i).

- **Case** $\equiv \equiv$ : Similar to the case $\text{CheckeqBin}$ in part (i).

- **Case** $\equiv \equiv \equiv \equiv$ : Similar to the case $\text{CheckeqBin}$ in part (i), using the i.h. (vi).

- **Case** $\equiv \equiv \equiv \equiv$ : Use the i.h. (v).

Lemma 71 (Separation for Subtyping). If $\Gamma_L * \Gamma_R \vdash A \triangleleft : \pm B \vdash \Delta$
and $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$
and $\text{FEV}(B) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L * \Gamma_R) \not\vdash (\Delta_L \ast \Delta_R)$. □
Proof of Lemma 71 (Separation for Subtyping). By induction on the given derivation. In the \(\llcorner:\text{Equiv}\) case, use Lemma 70 (Separation for Auxiliary Judgments) (vii). Otherwise, the reasoning needed follows that used in the proof of Lemma 72 (Separation—Main).

Lemma 72 (Separation—Main).

- **(Spines)** If \(\Gamma \ast \Gamma_R \vdash s : \mathcal{A} \vartriangleright \mathcal{C} \vdash \Delta\)
or \(\Gamma \ast \Gamma_R \vdash s : \mathcal{A} \vartriangleright \mathcal{C} [q] \vdash \Delta\)
  and \(\Gamma \ast \Gamma_R \vdash \mathcal{A} \vdash \mathcal{R}\)
  and \(\text{FEV}(\mathcal{A}) \subseteq \text{dom}(\Gamma_R)\)
  then \(\Delta = (\Delta_L \ast \Delta_R)\) and \((\Gamma_L \ast \Gamma_R) \xrightarrow{\mathcal{A}} (\Delta_L \ast \Delta_R)\) and \(\text{FEV}(\mathcal{C}) \subseteq \text{dom}(\Delta_R)\).

- **(Checking)** If \(\Gamma \ast \Gamma_R \vdash e \leftrightarrow \mathcal{C} \vdash \Delta\)
  and \(\Gamma \ast \Gamma_R \vdash \mathcal{C} \vdash \mathcal{R}\)
  and \(\text{FEV}(\mathcal{C}) \subseteq \text{dom}(\Gamma_R)\)
  then \(\Delta = (\Delta_L \ast \Delta_R)\) and \((\Gamma_L \ast \Gamma_R) \xrightarrow{\mathcal{C}} (\Delta_L \ast \Delta_R)\).

- **(Synthesis)** If \(\Gamma \ast \Gamma_R \vdash e \Rightarrow \mathcal{A} \vdash \Delta\)
  then \(\Delta = (\Delta_L \ast \Delta_R)\) and \((\Gamma_L \ast \Gamma_R) \xrightarrow{\mathcal{A}} (\Delta_L \ast \Delta_R)\).

- **(Match)** If \(\Gamma \ast \Gamma_R \vdash \Pi :: \ddot{\mathcal{A}} \leftrightarrow \mathcal{C} \vdash \Delta\)
  and \(\text{FEV}(\ddot{\mathcal{A}}) = \emptyset\)
  and \(\text{FEV}(\mathcal{C}) \subseteq \text{dom}(\Gamma_R)\)
  then \(\Delta = (\Delta_L \ast \Delta_R)\) and \((\Gamma_L \ast \Gamma_R) \xrightarrow{\mathcal{C}} (\Delta_L \ast \Delta_R)\).

- **(Match Elim.)** If \(\Gamma \ast \Gamma_R \vdash P \vdash \Pi :: \ddot{\mathcal{A}} \leftrightarrow \mathcal{C} \vdash \Delta\)
  and \(\text{FEV}(P) = \emptyset\)
  and \(\text{FEV}(\ddot{\mathcal{A}}) = \emptyset\)
  and \(\text{FEV}(\mathcal{C}) \subseteq \text{dom}(\Gamma_R)\)
  then \(\Delta = (\Delta_L \ast \Delta_R)\) and \((\Gamma_L \ast \Gamma_R) \xrightarrow{\mathcal{C}} (\Delta_L \ast \Delta_R)\).

Proof. By induction on the given derivation.

First, the (Match) judgment part, giving only the cases that motivate the side conditions:

- **Case** [MatchBase] Here we use the i.h. (Checking), for which we need \(\text{FEV}(\mathcal{C}) \subseteq \text{dom}(\Gamma_R)\).

- **Case** [Match/\] Here we use the i.h. (Match Elim.), which requires that \(\text{FEV}(P) = \emptyset\), which motivates \(\text{FEV}(\ddot{\mathcal{A}}) = \emptyset\).

- **Case** [MatchNeg] In its premise, this rule appends a type \(\mathcal{A} \in \ddot{\mathcal{A}}\) to \(\Gamma_R\) and claims it is principal (\(\llcorner: \text{Principal} \) \(\llcorner\) ), which motivates \(\text{FEV}(\ddot{\mathcal{A}} = \emptyset)\).

Similarly, (Match Elim.):

- **Case** [MatchUnify] Here we use Lemma 70 (Separation for Auxiliary Judgments) (iii), for which we need \(\text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset\), which motivates \(\text{FEV}(P) = \emptyset\).

Now, we show the cases for the (Spine), (Checking), and (Synthesis) parts.

- **Cases** [Var [II] \(\llcorner: \Gamma \llcorner\)] In all of these rules, the output context is the same as the input context, so just let \(\Delta_L = \Gamma_L\) and \(\Delta_R = \Gamma_R\).

- **Case** \(\Gamma_L \ast \Gamma_R \vdash \mathcal{A} \vartriangleright \mathcal{C} \vdash \Delta_L\) \(\llcorner: \Gamma \llcorner\)

  \(\llcorner: \Gamma \llcorner\) \(\llcorner: \Gamma \llcorner\)

  Let \(\Delta_L = \Gamma_L\) and \(\Delta_R = \Gamma_R\).
  We have \(\text{FEV}(\mathcal{A}) \subseteq \text{dom}(\Gamma_R)\). Since \(\Delta_R = \Gamma_R\) and \(\mathcal{C} = \mathcal{A}\), it is immediate that \(\text{FEV}(\mathcal{C}) \subseteq \text{dom}(\Delta_R)\).
Proof of Lemma 72

Case \( \Gamma \vdash \alpha \text{ type} \)
\[ \Gamma \vdash e \iff [\Gamma] \alpha \vdash \Delta \]

By i.h., \( \Theta = (\Theta_L \times \Theta_R) \) and \( (\Gamma_L \times \Gamma_R) \vdash_\Delta^\ast (\Theta_L \times \Theta_R) \).

By Lemma 68 (Transitivity of Separation), \( (\Gamma_L \times \Gamma_R) \vdash_\Delta^\ast (\Delta_L \times \Delta_R) \).

By i.h.; since \( FEV(A) = \emptyset \), the condition on the (Checking) part is trivial.

Adding a solution with a ground type cannot destroy separation.

Case \( \nu \chk-I \)
\[ \Gamma_L, \Gamma_R, \alpha : \kappa \vdash \nu \iff A_0 p \vdash \Delta, \alpha : \kappa, \Theta \]

By Definition 5

\[ \Delta = (\Delta_L \times \Delta_R) \]

By above equation
\[ \alpha \text{ not multiply declared} \]

Case \( \Gamma_L, \Gamma_R, \hat{\alpha} : \kappa \vdash e \ s : [\hat{\alpha}/\alpha]A_0 \gg C q \vdash \Delta \)

By Definition 5
\[ \Delta = (\Delta_L \times \Delta_R) \]

By above equation
\[ \Delta = (\Delta_L \times \Delta_R) \]

By Definition 5

\[ \Delta = (\Delta_L \times \Delta_R) \]

By Lemma 51 (Typing Extension)
Proof of Lemma 72 (Separation—Main)

\[ \Gamma_L \ast \Gamma_R \vdash (A_0 \land P) \text{ p type} \]  
\[ \Gamma_L \ast \Gamma_R \vdash P \text{ prop} \]  
\[ \Gamma_L \ast \Gamma_R \vdash A_0 \text{ p type} \]  
\[ \text{FEV}(A_0 \land P) \subseteq \text{dom}(\Gamma_R) \]  
\[ \text{FEV}(P) \subseteq \text{dom}(\Gamma_R) \]  
\[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \]  
\[ \Theta = (\Theta_L \ast \Theta_R) \]  
\[ (\Gamma_L \ast \Gamma_R) \not\vdash (\Theta_L \ast \Theta_R) \]  
\[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \]  
\[ \text{dom}(\Gamma_R) \subseteq \text{dom}(\Theta_R) \]  
\[ \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R) \]  
\[ \text{FEV}(P) \subseteq \text{dom}(\Theta_R) \]  
\[ \Theta = (\Theta_L \ast \Theta_R) \]  
\[ (\Gamma_L \ast \Gamma_R) \not\vdash (\Theta_L \ast \Theta_R) \]  

\[ \text{Case Nil} \]  
Similar to a section of the \( \forall \) case.

\[ \text{Case Cons} \]  
Similar to the \( \exists \) case, with an extra use of the i.h. for the additional second premise.

\[ \text{Case } v \text{ chk-I} \]  
\[ \Gamma_L \ast (\Gamma_R,\triangleright) / P \vdash \Theta \]  
\[ \Theta \vdash v \leftrightarrow [\Theta]A_0 ! \vdash \Delta, \triangleright, \Delta' \]  
\[ \Gamma_L \ast \Gamma_R \vdash v \leftrightarrow P \supset A_0 ! \vdash \Delta \]  

\[ \Gamma_L \ast \Gamma_R \vdash (P \supset A_0) ! \text{ type} \]  
\[ \text{Given} \]  
\[ \Gamma_L \ast \Gamma_R \vdash P \supset A_0 \text{ prop} \]  
\[ \text{By inversion} \]  
\[ \text{FEV}(P) = \emptyset \]  
\[ \text{By def. of FEV} \]  
\[ \Gamma_L \ast (\Gamma_R,\triangleright) / P \vdash \Theta \]  
\[ \Theta = (\Theta_L \ast (\Theta_R, \Theta_Z)) \]  
\[ (\Gamma_L \ast (\Gamma_R,\triangleright,\Theta_Z)) \not\vdash (\Theta_L \ast (\Theta_R, \Theta_Z)) \]  

\[ \Gamma_L \ast \Gamma_R \vdash (P \supset A_0) ! \text{ type} \]  
\[ \text{Given} \]  
\[ \Gamma_L, \Gamma_R \vdash A_0 ! \text{ type} \]  
\[ \text{By Lemma 42 (Inversion of Principal Typing) (2)} \]  
\[ \Gamma_L, \Gamma_R, \triangleright, \Theta_Z \vdash A_0 ! \text{ type} \]  
\[ \Theta \vdash [\Theta]A_0 ! \text{ type} \]  
\[ \text{By Lemma 35 (Suffix Weakening)} \]  
\[ \text{FEV}(A_0) = \emptyset \]  
\[ \text{Above and def. of FEV} \]  
\[ \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R, \Theta_Z) \]  
\[ (\Delta, \triangleright, \Delta') = (\Delta_L \ast \Delta_R') \]  
\[ (\Theta_L \ast (\Theta_R, \Theta_Z)) \not\vdash (\Delta_L \ast \Delta_R') \]  

\[ (\Gamma_L \ast (\Gamma_R,\triangleright)) \not\vdash (\Delta_L \ast \Delta_R') \]  
\[ \text{By Lemma 68 (Transitivity of Separation)} \]  
\[ (\Gamma_L \ast \Gamma_R) \not\vdash (\Delta_L \ast \Delta_R') \]  
\[ \text{By Lemma 69 (Separation Truncation)} \]  
\[ \Delta = (\Delta_L, \Delta_R) \]  
\[ \text{Similar to the } \forall \text{ case} \]
Proof of Lemma 72\textit{ (Separation—Main)} \lemseparation-main

\begin{itemize}
  \item Case $Γ_L ∗ Γ_R ⊨ \text{true} \vdash Θ \quad Θ ⊨ e s : [Θ]A_0 \quad p \gg C q ⊨ Δ$

  \begin{align*}
    Γ_L ∗ Γ_R ⊨ e s : P ⊃ A_0 \quad p \gg C q ⊨ Δ
  \end{align*}
  
  \begin{itemize}
    \item Given
    \item By inversion
    \item Subderivation
    \item By Lemma 70 \textit{(Separation for Auxiliary Judgments)} (i)
  \end{itemize}

  \begin{itemize}
    \item $Γ_L ∗ Γ_R ⊢ (P ⊃ A_0) \ p \text{ type}$
    \item $Γ_L ∗ Γ_R ⊢ P \ prop$
    \item $Γ_L, Γ_R ⊢ P \ true \vdash Θ$
    \item $Θ = (Θ_L ∗ Θ_R)$
    \item $Γ_L ∗ Γ_R \overset{Γ}{→} (Θ_L ∗ Θ_R)$
  \end{itemize}

  \begin{itemize}
    \item $Θ ⊨ e s : [Θ]A_0 \ p \gg C q ⊨ Δ$
    \item $Δ, x : C p, Θ = (Δ_L, Δ_R)$
    \item $Γ_L ∗ Γ_R \overset{Γ}{→} (Δ_L ∗ Δ_R)$
    \item By Lemma 68 \textit{(Transitivity of Separation)}
  \end{itemize}

  \begin{itemize}
    \item $Γ_L, Γ_R, x : C p \vdash v \leftrightarrow C p ⊨ Δ, x : C p, Θ$
    \item $Γ_L, Γ_R \vdash \text{rec} x.v \leftrightarrow C p ⊨ Δ$
    \item $Γ_L ∗ Γ_R \vdash C p \ → Δ$
  \end{itemize}

  \begin{itemize}
    \item $Γ_L ∗ Γ_R \vdash C p \ → Δ$
    \item $Δ = (Δ_L, Δ_R)$
  \end{itemize}

  \begin{itemize}
    \item $Γ_L ∗ Γ_R \vdash λ x.e \leftrightarrow A \ → B p ⊨ Δ$
    \item $Γ_L, Γ_R \vdash λ x.e \leftrightarrow A \ → B p ⊨ Δ$
  \end{itemize}

  \begin{itemize}
    \item $Γ_L ∗ Γ_R \vdash (A → B) \ p \text{ type}$
    \item $Γ_L ∗ Γ_R \vdash B p \text{ type}$
    \item $FEV(A → B) \subseteq \text{dom}(Γ_R)$
    \item $Γ_L ∗ Γ_R \vdash B p \text{ type}$
    \item By def. of FEV
    \item $Γ_L ∗ Γ_R \vdash A \ p \text{ type}$
    \item $Γ_L ∗ Γ_R \vdash A \ p \text{ type}$
    \item By weakening and Definition \ref{def:separation}
    \item $Δ, x : A p, Θ = (Δ_L, Δ_R)$
    \item $Γ_L ∗ Γ_R \overset{Γ}{→} (Δ_L ∗ Δ_R)$
    \item By i.h.
    \item $Δ = (Δ_L, Δ_R)$
  \end{itemize}

  \begin{itemize}
    \item $Γ_L, Γ_R, x : A p \vdash e \leftrightarrow B p ⊨ Δ, x : A p, Θ$
    \item $Γ_L, Γ_R, x : A p \vdash e \leftrightarrow B p ⊨ Δ, x : A p, Θ$
    \item $Γ_L ∗ Γ_R \vdash (A → B) \ p \text{ type}$
    \item $Γ_L ∗ Γ_R \vdash B p \text{ type}$
    \item $Γ_L, Γ_R, x : A p \vdash e \leftrightarrow B p ⊨ Δ, x : A p, Θ$
    \item $Γ_L, Γ_R, x : A p, Θ = (Δ_L, Δ_R)$
    \item $Γ_L ∗ Γ_R \overset{Γ}{→} (Δ_L ∗ Δ_R)$
    \item By i.h.
    \item $Δ = (Δ_L, Δ_R)$
  \end{itemize}

  \begin{itemize}
    \item $Γ_L[Δ_1 : *, Δ_2 : *, Δ : ] \vdash Δ_1 \rightarrow Δ_2, x : Δ_1 \vdash e_0 \leftrightarrow Δ_2 \ → Δ, x : Δ_1, Δ'$
    \item $Γ_L[Δ_1 : *] \vdash λ x. e_0 \leftrightarrow Δ \ → Δ$
    \item $Γ_L ∗ Γ_R \vdash λ x. e_0 \leftrightarrow Δ \ → Δ$
  \end{itemize}

  \begin{itemize}
    \item \text{We have } (Γ_L ∗ Γ_R) = Γ_0[Δ : *]. \text{ We also have } FEV(Δ) \subseteq \text{dom}(Γ_R). \text{ Therefore } Δ ∈ \text{dom}(Γ_R) \text{ and }$
    \item $Γ_0[Δ : *] = Γ_L, Γ_2, Δ : *, Γ_3$
  \end{itemize}

\end{itemize}
Proof of Lemma 72 (Separation—Main)

\[ \Gamma_R = (\Gamma_2, \ast : \ast, \Gamma_3) \]

Then the input context in the premise has the following form:

\[ \Gamma_0[\Gamma_1; \ast, \ast, \ast \mapsto \Gamma_2, x : \ast, \ast, \ast \mapsto \Gamma_1 \rightarrow \Gamma_2, \Gamma_3, x : \ast_1] \]

Let us separate this context at the same point as \( \Gamma_0[\ast] \), that is, after \( \Gamma_1 \) and before \( \Gamma_2 \), and call the resulting right-hand context \( \Gamma_R \). That is,

\[ \Gamma_0[\Gamma_1; \ast, \ast, \ast \mapsto \Gamma_2, x : \ast, \ast, \ast \mapsto \Gamma_1 \rightarrow \Gamma_2, \Gamma_3, x : \ast_1] = \Gamma_1 \ast \left( \Gamma_2, \ast, \ast \mapsto \Gamma_1 \rightarrow \Gamma_2, \Gamma_3, x : \ast_1 \right) \]

\[ \text{FEV}(\ast) \subseteq \text{dom}(\Gamma_R) \]

\[ \Gamma_1 \ast \Gamma_R \vdash e_0 \Leftarrow \Delta, x : \ast, \Delta' \] (Subderivation)

\[ \text{FEV}(\ast) \subseteq \text{dom}(\Gamma_R) \]

\[ \Gamma_1 \ast \Gamma_R \vdash \ast \] (Type)

\[ \text{FEV}(\ast) \subseteq \text{dom}(\Gamma_R) \]

\[ \Delta = (\Delta_1, \Delta_2) \] (By i.h.)

\[ \text{FEV}(\ast) \subseteq \text{dom}(\Gamma_R) \]

\[ \text{FEV}(\ast) \subseteq \text{dom}(\Gamma_R) \]

\[ \text{FEV}(\ast) \subseteq \text{dom}(\Gamma_R) \]

Case 1. \( \Gamma \vdash e \Rightarrow A \mid p - \Theta \quad \Theta \vdash s : [\Theta] A \mid p \gg C \mid q - \Delta \)

Use the i.h. and Lemma 68 (Transitivity of Separation), with Lemma 89 (Well-formedness of Algorithmic Typing) and Lemma 13 (Right-Hand Substitution for Typing).

Case 2. \( \Gamma \vdash s : A \mid ! \gg C \mid i - \Delta \)

Use the i.h.

Case 3. \( \Gamma \vdash e \Leftarrow A_1 \mid p - \Theta \quad \Theta \vdash s : [\Theta] A_2 \mid p \gg C \mid q - \Delta \)

Use the i.h.

- Case 1. \( \Gamma \vdash e \Rightarrow A \mid p - \Theta \quad \Theta \vdash s : [\Theta] A_2 \mid p \gg C \mid q - \Delta \)

- Case 2. \( \Gamma \vdash s : A \mid ! \gg C \mid i - \Delta \)

- Case 3. \( \Gamma \vdash s : A \mid p \gg C \mid q - \Delta \)
Proof of Lemma 72 (Separation—Main)

\[ \Gamma \vdash e \iff A_k \vdash \Delta \]
\[ \Gamma \vdash \text{inj}_k e \iff A_1 + A_2 \vdash \Delta \]

Use the i.h. (inverting \( \Gamma \vdash (A_1 + A_2) \vdash \Delta \)).

\[ \text{Case } \Gamma \vdash e_1 \iff A_1 \vdash \Theta \quad \Theta \vdash e_2 \iff [\Theta]A_2 \vdash \Delta \]

\[ \Gamma \vdash \langle e_1, e_2 \rangle \iff A_1 \times A_2 \vdash \Delta \]

\[ \Gamma \vdash (A_1 \times A_2) \vdash \Delta \quad \text{Given} \]
\[ \Gamma \vdash A_1 \vdash \Theta \quad \text{By inversion} \]
\[ \Gamma \vdash e_1 \iff A_1 \vdash \Theta \quad \text{Subderivation} \]
\[ \Theta = (\Theta_L, \Theta_R) \quad \text{By i.h.} \]
\[ (\Gamma_L \star \Gamma_R) \rightsquigarrow (\Theta_L \star \Theta_R) \quad " \]

\[ \Gamma \vdash A_2 \vdash \Theta \quad \text{By Lemma 31 (Typing Extension)} \]
\[ \Theta \vdash A_2 \vdash \Theta \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \]
\[ \Theta \vdash \Theta A_2 \vdash \Delta \quad \text{By Lemma 13 (Right-Hand Substitution for Typing)} \]

\[ \Theta \vdash e_2 \iff [\Theta]A_2 \vdash \Delta \quad \text{Subderivation} \]

\[ \Delta = (\Delta_L, \Delta_R) \quad \text{By i.h.} \]
\[ (\Theta_L \star \Theta_R) \rightsquigarrow (\Delta_L \star \Delta_R) \quad " \]
\[ \text{By Lemma 68 (Transitivity of Separation)} \]
Proof of Lemma 72 (Separation—Main) lem:separation-main

• Case $\Gamma[\alpha_2;*, \alpha_1;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2] \vdash e_1 \iff \hat{\alpha}_1 \vdash \Theta \quad \Theta \vdash e_2 \iff [\Theta] \hat{\alpha}_2 \vdash \Delta$

  $\Gamma[\hat{\alpha} ; \ast ] \vdash \langle e_1, e_2 \rangle \iff \hat{\alpha} \vdash \Delta \times \lnot \hat{\alpha}$

  We have $(\Gamma_L \ast \Gamma_R) = \Gamma_0[\hat{\alpha} : \ast ]$. We also have $\text{FEV}(\hat{\alpha}) \subseteq \text{dom}(\Gamma_R)$. Therefore $\hat{\alpha} \in \text{dom}(\Gamma_R)$ and

  $\Gamma_0[\hat{\alpha} : \ast ] = \Gamma_L, \Gamma_2, \hat{\alpha} : \ast, \Gamma_3$

  where $\Gamma_R = (\Gamma_2, \hat{\alpha} : \ast, \Gamma_3)$.

  Then the input context in the premise has the following form:

  $\Gamma_0[\hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2] = (\Gamma_L, \Gamma_2, \hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3)$

  Let us separate this context at the same point as $\Gamma_0[\hat{\alpha} : \ast ]$, that is, after $\Gamma_L$ and before $\Gamma_2$, and call the resulting right-hand context $\Gamma'_R$:

  $\Gamma_0[\hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2] = \Gamma_L \ast (\Gamma_2, \hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3)$

  $\Gamma'_R$

  $\text{FEV}(\hat{\alpha}) \subseteq \text{dom}(\Gamma_R)$

  $\Gamma_L \ast \Gamma'_R \vdash e_1 \iff \hat{\alpha}_1 \vdash \Theta$

  $\text{FEV}(\hat{\alpha}_2) \subseteq \text{dom}(\Gamma'_R)$

  $\Theta \vdash e_2 \iff (\Theta) \hat{\alpha}_2 \vdash \Delta$

  $\text{dom}(\Gamma'_R) \subseteq \text{dom}(\Theta)$

  $\text{FEV}(\hat{\alpha}_2) \subseteq \text{dom}(\Theta)$

  $\text{FEV}(\hat{\alpha}_2) \subseteq \text{dom}(\Theta)$

  By above $\subseteq$

  $\text{FEV}(\hat{\alpha}_2) \subseteq \text{dom}(\Theta)$

  By Definition 5

  $\text{FEV}(\hat{\alpha}_2) \subseteq \text{dom}(\Theta)$

  By Definition 4

  $\Delta = (\Delta_L, \Delta_R)$

  By i.h.

  $\Delta = (\Delta_L, \Delta_R)$

  By i.h.

  $\Gamma_R = (\Gamma_2, \hat{\alpha} : \ast, \Gamma_3)$

  Above

  $\Gamma'_R = (\Gamma_2, \hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \hat{\alpha} : \ast = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3)$

  Above

  By Lemma 23 (Deep Evar Introduction) (i), (ii) and the definition of separation, we can show

  $(\Gamma_L \ast (\Gamma_2, \hat{\alpha} : \ast, \Gamma_3)) \vdash (\Gamma_L \ast (\Gamma_2, \hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3))$

  $(\Gamma_L \ast \Gamma_R) \vdash (\Gamma_L \ast \Gamma'_R)$

  By above equalities

  $(\Gamma_L \ast \Gamma_R) \vdash (\Delta_L, \Delta_R)$

  By Lemma 68 (Transitivity of Separation) twice

  • Case $\Gamma[\hat{\alpha}_1;\ast, \hat{\alpha}_2;\ast, \alpha;\ast = \hat{\alpha}_1 \times \hat{\alpha}_2] \vdash e \iff \hat{\alpha}_1 \vdash \Delta$

    $\Gamma[\hat{\alpha} ; \ast ] \vdash \text{inj}_k e \iff \hat{\alpha} \vdash \Delta \times \lnot \hat{\alpha}$

    Similar to the $\times \lnot \hat{\alpha}$ case, but simpler.

  • Case $\Gamma[\hat{\alpha}_2;\ast, \hat{\alpha}_1;\ast, \alpha;\ast = \hat{\alpha}_1 \rightarrow \hat{\alpha}_2] \vdash e \quad s_0 : (\hat{\alpha}_1 \rightarrow \hat{\alpha}_2) \rightarrow C \vdash \Delta$

    $\Gamma[\hat{\alpha} ; \ast ] \vdash e \ast s_0 : \alpha \rightarrow C \vdash \Delta$

    Similar to the $\times \lnot \hat{\alpha}$ and $\lnot \hat{\alpha}$ cases, except that (because we're in the spine part of the lemma) we have to show that $\text{FEV}(C) \subseteq \text{dom}(\Delta_R)$. But we have the same C in the premise and conclusion, so we get that by applying the i.h.

  • Case $\Gamma \vdash e \Rightarrow \Lambda ! \vdash \Theta \quad \Theta \vdash \Pi :: \Lambda \iff [\Theta] C \vdash \Delta \quad \Delta \vdash \Pi \text{ covers } [\Delta] \Lambda$

    $\Gamma \vdash \text{case}(e, \Pi) \iff C \vdash \Delta$

    Use the i.h. and Lemma 68 (Transitivity of Separation).
Decidability of Algorithmic Subtyping

Lemma 73 (Substitution Isn’t Large).
For all contexts Θ, we have \( \#(\Theta \Delta) = \#(\Delta) \).

Proof. By induction on \( \Delta \), following the definition of substitution.

Lemma 74 (Instantiation Solves).
If \( \Gamma \vdash \check{\alpha} : \tau \vdash \kappa \vdash \Delta \) and \( \Gamma \vdash \tau = \tau \) and \( \check{\alpha} \not\in FV(\Gamma) \) then \( |\text{unsolved}(\Gamma)| = |\text{unsolved}(\Delta)| + 1 \).

Proof. By induction on the given derivation.

- **Case**
  \[ \Gamma_L \vdash \tau : \kappa \]
  \[ \Gamma_L, \check{\alpha} : \kappa, \Gamma_R \vdash \check{\alpha} : \tau : \kappa \]
  \[ \Gamma_L, \check{\alpha} : \kappa, \Gamma_R \vdash \check{\alpha} : \tau : \Gamma_L, \check{\alpha} : \kappa = \tau, \Gamma_R \]

  It is evident that \( |\text{unsolved}(\Gamma_L, \check{\alpha} : \kappa, \Gamma_R)| = |\text{unsolved}(\Gamma_L, \check{\alpha} = \tau, \Gamma_R)| + 1 \).

- **Case**
  \[ \check{\beta} \in \text{unsolved}(\Gamma[\check{\alpha} : \kappa][\check{\beta} : \kappa]) \]
  \[ \Gamma[\check{\alpha} : \kappa][\check{\beta} : \kappa] \vdash \check{\alpha} := \check{\beta} : \kappa \vdash \Gamma[\check{\alpha} : \kappa][\check{\beta} : \kappa] = \check{\alpha} \]

  Similar to the previous case.

- **Case**
  \[ \Gamma_0[\check{\alpha}_2 : \ast, \check{\alpha}_1 : \ast, \check{\alpha} : \ast = \check{\alpha}_1 : \check{\alpha}_2] 
  \vdash \check{\alpha}_1 := \check{\alpha}_2 : \ast \vdash \check{\alpha}_1 : \ast \vdash \Theta \vdash \check{\alpha}_2 := \Theta[\ast]_2 : \ast \vdash \Delta \]

  \[ \Gamma_0[\check{\alpha} : \ast] \vdash \check{\alpha} := \check{\alpha}_1 : \ast \vdash \Gamma_0[\check{\alpha} : \ast] \]

  \[ \text{unsolved}(\Gamma_0[\check{\alpha}_2 : \ast, \check{\alpha}_1 : \ast, \check{\alpha} = \check{\alpha}_1 : \check{\alpha}_2]) = \text{unsolved}(\Gamma_0[\check{\alpha}]) + 1 \quad \text{Immediate} \]
  \[ \text{unsolved}(\Gamma_0[\check{\alpha}_2 : \ast, \check{\alpha}_1 : \ast, \check{\alpha} = \check{\alpha}_1 : \check{\alpha}_2]) = \text{unsolved}(\Theta) + 1 \quad \text{By i.h.} \]

  \[ \text{unsolved}(\Gamma_0) = \text{unsolved}(\Theta) \quad \text{Subtracting 1} \]

  \[ |\text{unsolved}(\Gamma_0)| = |\text{unsolved}(\Theta)| \quad \text{By i.h.} \]

- **Case**
  \[ \Gamma[\check{\alpha} : N] \vdash \check{\alpha} := \text{zero} : N \vdash \Gamma[\check{\alpha} : N = \text{zero}] \]

  Similar to the \( \text{InstSolve} \) case.

- **Case**
  \[ \Gamma_0[\check{\alpha}_1 : N, \check{\alpha} : N = \text{succ}(\check{\alpha}_1)] 
  \vdash \check{\alpha}_1 := t_1 : N \vdash \Delta \]

  \[ \Gamma_0[\check{\alpha} : N] \vdash \check{\alpha} := \text{succ}(t_1) : N \vdash \Delta \]

  \[ |\text{unsolved}(\Delta)| + 1 = |\text{unsolved}(\Gamma_0[\check{\alpha}_1 : N, \check{\alpha} : N = \text{succ}(\check{\alpha}_1)])| \quad \text{By i.h.} \]

  \[ |\text{unsolved}(\Gamma_0[\check{\alpha} : N])| = |\text{unsolved}(\Gamma_0[\check{\alpha}_1 : N, \check{\alpha} : N = \text{succ}(\check{\alpha}_1)])| \quad \text{By definition of unsolved(−)} \]

Lemma 75 (Checkeq Solving). If \( \Gamma \vdash s = t : \kappa \vdash \Delta \) then either \( \Delta = \Gamma \) or \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \).

Proof. By induction on the given derivation.

- **Case**
  \[ \Gamma \vdash u = u : \kappa \vdash \Delta \]

  Here \( \Delta = \Gamma \).
If $\Gamma \vdash \sigma \equiv t : N \rightarrow \Delta$

$\Gamma \vdash \text{succ}(\sigma) \equiv \text{succ}(t) : N \rightarrow \Delta$

Follows by i.h.

• Case

$\Gamma_{0}[\bar{\alpha}] \vdash \bar{\alpha} := t : \kappa \rightarrow \Delta$

$\bar{\alpha} \notin \text{FV}(t)$

$\Gamma_{0}[\bar{\alpha}] \vdash \bar{\alpha} \equiv t : \kappa \rightarrow \Delta$

$\Gamma_{0}[\bar{\alpha}] \vdash \bar{\alpha} := t : \kappa \rightarrow \Delta$

$\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| - 1$

$\vdash$

$\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$

$\Theta$

By Lemma 74 (Instantiation Solves)

• Case

$\Gamma[\bar{\alpha} : \kappa] \vdash \bar{\alpha} := t : \kappa \rightarrow \Delta$

$\bar{\alpha} \notin \text{FV}(t)$

$\Gamma[\bar{\alpha} : \kappa] \vdash t \equiv \bar{\alpha} : \kappa \rightarrow \Delta$

$\vdash$

Similar to the CheckeqInstL case.

• Case

$\Gamma \vdash \sigma_{1} \equiv \tau_{1} : * \rightarrow \Theta$

$\Theta \vdash [\Theta] \sigma_{2} \equiv [\Theta] \tau_{2} : * \rightarrow \Delta$

$\Gamma \vdash \sigma_{1} \oplus \sigma_{2} \equiv \tau_{1} \oplus \tau_{2} : * \rightarrow \Delta$

$\vdash$

$\Theta = \Gamma$ or $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$

By i.h.

$\Theta = \Gamma$:

$\Theta \vdash [\Theta] \sigma_{2} \equiv [\Theta] \tau_{2} : * \rightarrow \Delta$

$\Gamma \vdash [\Gamma] \sigma_{2} \equiv [\Gamma] \tau_{2} : * \rightarrow \Delta$

$\vdash$

$\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| + 1$

By i.h.

• Case $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$:

$\Theta \vdash [\Theta] \sigma_{2} \equiv [\Theta] \tau_{2} : * \rightarrow \Delta$

$\vdash$

$\Delta = \Theta$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$

By i.h.

If $\Delta = \Theta$ then substituting $\Delta$ for $\Theta$ in $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$ gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

If $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$ then transitivity of $<$ gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

Lemma 76 (Prop Equiv Solving).

If $\Gamma \vdash P \equiv Q \rightarrow \Delta$ then either $\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

Proof. Only one rule can derive the judgment:

• Case

$\Gamma \vdash \sigma_{1} \equiv t_{1} : N \rightarrow \Theta$

$\Theta \vdash [\Theta] \sigma_{2} \equiv [\Theta] t_{2} : N \rightarrow \Delta$

$\Gamma \vdash (\sigma_{1} = \sigma_{2}) \equiv (t_{1} = t_{2}) \rightarrow \Delta$

By Lemma 75 (Checkeq Solving) on the first premise, either $\Theta = \Gamma$ or $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$.

In the former case, the result follows from Lemma 75 (Checkeq Solving) on the second premise.
In the latter case, applying Lemma 75 (CheckEq Solving) to the second premise either gives $\Delta = \Theta$, and therefore
\[
|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|
\]
or gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$, which also leads to $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$. \qed

**Lemma 77** (Equiv Solving).

*If* $\Gamma \vdash A \equiv B \vdash \Delta$ *then either* $\Delta = \Gamma$ *or* $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

**Proof.** By induction on the given derivation.

- **Case**
  \[
  \Gamma \vdash \alpha \equiv \alpha \vdash \Gamma \equiv \text{Var}
  \]
  Here $\Delta = \Gamma$.

- **Cases**
  - $\equiv \text{Exvar}$
  - $\equiv \text{Unit}$
  Similar to the $\equiv \text{Var}$ case.

- **Case**
  \[
  \Gamma \vdash A_1 \equiv B_1 \vdash \Theta \quad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta
  \]
  \[
  \Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \vdash \Delta
  \equiv \oplus
  \]
  By i.h., either $\Theta = \Gamma$ or $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$.
  In the former case, apply the i.h. to the second premise. Now either $\Delta = \Theta$—and therefore $\Delta = \Gamma$—or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$. Since $\Theta = \Gamma$, we have $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.
  In the latter case, we have $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$. By i.h. on the second premise, either $\Delta = \Theta$, and substituting $\Delta$ for $\Theta$ gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$—or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$, which combined with $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$ gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

- **Case**
  \[
  \Gamma, \alpha : k \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : k, \Delta'
  \]
  \[
  \Gamma \vdash \forall \alpha : k. A_0 \equiv \forall \alpha : k. B_0 \vdash \Delta
  \equiv \forall
  \]
  By i.h., either $(\Delta, \alpha : k, \Delta') = (\Gamma, \alpha : k)$, or $|\text{unsolved}(\Delta, \alpha : k, \Delta')| < |\text{unsolved}(\Gamma, \alpha : k)|$.
  In the former case, Lemma 22 (Extension Inversion) (i) tells us that $\Delta' = \cdot$. Thus, $(\Delta, \alpha : k) = (\Gamma, \alpha : k)$, and so $\Delta = \Gamma$.
  In the latter case, we have $|\text{unsolved}(\Delta, \alpha : k, \Delta')| < |\text{unsolved}(\Gamma, \alpha : k)|$, that is:
  \[
  |\text{unsolved}(\Delta)| + 0 + |\text{unsolved}(\Delta')| < |\text{unsolved}(\Gamma)| + 0
  \]
  Since $|\text{unsolved}(\Delta')|$ cannot be negative, we have $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

- **Case**
  \[
  \Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta
  \]
  \[
  \Gamma \vdash P \cup A_0 \equiv Q \cup B_0 \vdash \Delta
  \equiv \cup
  \]
  Similar to the $\equiv \cup$ case, but using Lemma 76 (Prop Equiv Solving) on the first premise instead of the i.h.

- **Case**
  \[
  \Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta
  \]
  \[
  \Gamma \vdash A_0 \land P \equiv B_0 \land Q \vdash \Delta
  \equiv \wedge
  \]
  Similar to the $\equiv \wedge$ case.
Decidability of Propositional Judgments

Proof of Lemma 77 (Equiv Solving) \lem:equiv-solving

- Case \( \Gamma_0[\alpha] \vdash \textit{\alpha} := \tau : * \vdash \Delta \quad \text{\(\alpha\) \notin FV(\tau)} \)
  
  \[ \Gamma_0[\alpha] \vdash \textit{\alpha} \equiv \tau \vdash \Delta \]

  By Lemma 74 (Instantiation Solves), \(|\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| - 1\).

- Case \( \Gamma_0[\alpha] \vdash \textit{\alpha} := \tau : * \vdash \Delta \quad \text{\(\alpha\) \notin FV(\tau)} \)
  
  \[ \Gamma_0[\alpha] \vdash \tau \equiv \textit{\alpha} \vdash \Delta \]

  Similar to the \(\text{\textit{\alpha} \equiv \tau \vdash \Delta \equiv \textit{\alpha} \vdash \Delta \})\) case.

\end{proof}

Lemma 78 (Decidability of Propositional Judgments).

The following judgments are decidable, with \(\Delta\) as output in (1)–(3), and \(\Delta^\perp\) as output in (4) and (5).

We assume \(\sigma = [\Gamma]\sigma\) and \(t = [\Gamma]t\) in (1) and (4). Similarly, in the other parts we assume \(P = [\Gamma]P\) and (in part (3)) \(Q = [\Gamma]Q\).

1. Decidability of \(\Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta\): By induction on the sizes of \(\sigma\) and \(t\).

   - Cases \(\text{CheckeqVar, CheckeqUnit, CheckeqZero}\) No premises.
   - Case \(\text{CheckeqSucc}\) Both \(\sigma\) and \(t\) get smaller in the premise.
   - Cases \(\text{CheckeqInstL, CheckeqInstR}\) Follows from Lemma 67 (Decidability of Instantiation).

2. Decidability of \(\Gamma \vdash P \true \vdash \Delta\): By induction on \(\sigma\) and \(t\). But we have only one rule deriving this judgment form, \(\text{CheckpropEq}\), which has the judgment in (1) as a premise, so decidability follows from part (1).

3. Decidability of \(\Gamma \vdash P \equiv Q \vdash \Delta\): By induction on \(P\) and \(Q\). But we have only one rule deriving this judgment form, \(\text{\equiv PropEq}\), which has two premises of the form (1), so decidability follows from part (1).

4. Decidability of \(\Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta^\perp\): By lexicographic induction, first on the number of unsolved variables (both universal and existential) in \(\Gamma\), then on \(\sigma\) and \(t\). We also show that the number of unsolved variables is nonincreasing in the output context (if it exists).

   - Cases \(\text{ElimeqUvarRefI, ElimeqZero}\) No premises, and the output is the same as the input.
   - Case \(\text{ElimeqClash}\) The only premise is the clash judgment, which is clearly decidable. There is no output.
   - Case \(\text{ElimeqBin}\) In the first premise, we have the same \(\Gamma\) but both \(\sigma\) and \(t\) are smaller. By i.h., the first premise is decidable; moreover, either some variables in \(\Theta\) were solved, or no additional variables were solved.

   If some variables in \(\Theta\) were solved, the second premise is smaller than the conclusion according to our lexicographic measure, so by i.h., the second premise is decidable.

   If no additional variables were solved, then \(\Theta = \Gamma\). Therefore \(\Theta|\tau_2 = [\Gamma]|\tau_2\). It is given that \(\sigma = [\Gamma]|\sigma\) and \(t = [\Gamma]|t\), so \([\Gamma]|\tau_2 = \tau_2\). Likewise, \([\Theta]|\tau_2 = [\Gamma]|\tau_2' = \tau_2'\), so we are making a recursive call on a strictly smaller subterm.

   Regardless, \(\Delta^\perp\) is either \(\perp\), or is a \(\Delta\) which has no more unsolved variables than \(\Theta\), which in turn has no more unsolved variables than \(\Gamma\).
Proof of Lemma 78 (Decidability of Propositional Judgments) lem:prop-decidable

- Case ElimqBinBot: The premise is invoked on subterms, and does not yield an output context.

- Case ElimqSucc: Both $\sigma$ and $t$ get smaller. By i.h., the output context has fewer unsolved variables, if it exists.

- Cases ElimqInstL, ElimqInstR: Follows from Lemma 76 (Decidability of Instantiation). Furthermore, by Lemma 80 (Instantiation Solves), instantiation solves a variable in the output.

- Cases ElimqUvarL, ElimqUvarR: These rules have no nontrivial premises, and $\alpha$ is solved in the output context.

- Cases ElimqUvarL⊥, ElimqUvarR⊥: These rules have no nontrivial premises, and produce the output context $\bot$.

(5) Decidability of $\Gamma / P \vdash A \vdash B \vdash \Delta$: By induction on $P$. But we have only one rule deriving this judgment form, ElimpropEq, for which decidability follows from part (4).

Lemma 79 (Decidability of Equivalence).

Given a context $\Gamma$ and types $A$, $B$ such that $\vdash \Gamma \vdash A$ type and $\vdash \Gamma \vdash B$ type and $[\Gamma]A = A$ and $[\Gamma]B = B$, it is decidable whether there exists $\Delta$ such that $\vdash \Gamma \vdash A \equiv B \vdash \Delta$.

Proof. Let the judgment $\vdash \Gamma \vdash A \equiv B \vdash \Delta$ be measured lexicographically by

(E1) $\#(\text{large}(A)) + \#(\text{large}(B))$;

(E2) $|\text{unsolved}(\Gamma)|$, the number of unsolved existential variables in $\Gamma$;

(E3) $|A| + |B|$.

- Cases $\equiv \text{Var} \equiv \text{Exvar} \equiv \text{Unit}$: No premises.

- Case $\Gamma \vdash A_1 \equiv B_1 \equiv \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \equiv \Delta$

  $\Gamma \vdash A_1 \equiv A_2 \equiv B_1 \equiv B_2 \equiv \Delta$

  In the first premise, part (E1) either gets smaller (if $A_2$ or $B_2$ have large connectives) or stays the same. Since the first premise has the same input context, part (E2) remains the same. However, part (E3) gets smaller.

  In the second premise, part (E1) either gets smaller (if $A_1$ or $B_1$ have large connectives) or stays the same.

- Case $\equiv \text{Vec}$: Similar to a special case of $\equiv \text{Var}$, where two of the types are monotypes.

  $\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \equiv \Delta, \alpha : \kappa, \Delta'$

  $\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \equiv \Delta$

  Since $\#(\text{large}(A_0)) + \#(\text{large}(B_0)) = \#(\text{large}(A)) + \#(\text{large}(B)) - 2$, the first part of the measure gets smaller.

- Case $\Gamma \vdash P \equiv Q \equiv \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \equiv \Delta$

  $\Gamma \vdash P \equiv A_0 \equiv Q \equiv B_0 \equiv \Delta$

  The first premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (3).

  For the second premise, by Lemma 73 (Substitution Isn't Large), $\#(\text{large}([\Theta]A_0)) = \#(\text{large}(A_0))$ and $\#(\text{large}([\Theta]B_0)) = \#(\text{large}(B_0))$. Since $\#(\text{large}(A)) = \#(\text{large}(A_0)) + 1$ and $\#(\text{large}(B)) = \#(\text{large}(B_0)) + 1$, we have

  $\#(\text{large}([\Theta]A_0)) + \#(\text{large}([\Theta]B_0)) < \#(\text{large}(A)) + \#(\text{large}(B))$

  which makes the first part of the measure smaller.
Proof of Lemma 79 (Decidability of Equivalence)

• Case

\[ \Gamma \vdash P \equiv Q \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta \]

\[ \Gamma \vdash A_0 \land P \equiv B_0 \land Q \vdash \Delta \]

Similar to the \(\equiv\) case.

• Case

\[ \Gamma[\delta] \vdash \delta := \tau \vdash \Delta \quad \delta \notin FV(\tau) \]

\[ \Gamma[\delta] \vdash \delta \equiv \tau \vdash \Delta \]

\(\equiv\text{Instantiatel}\)

Follows from Lemma 67 (Decidability of Instantiation).

• Case \(\equiv\text{Instantiatel}\) Similar to the \(\equiv\text{Instantiatel}\) case.

\[ \square \]

H'.2 Decidability of Subtyping

Theorem 1 (Decidability of Subtyping).

Given a context \(\Gamma\) and types \(A, B\) such that \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type and \([\Gamma]A = A\) and \([\Gamma]B = B\), it is decidable whether there exists \(\Delta\) such that \(\Gamma \vdash A <: \pm B \vdash \Delta\).

Proof. Let the judgments be measured lexicographically by \#large\(A\) + \#large\(B\).

For each subtyping rule, we show that every premise is smaller than the conclusion, or already known to be decidable. The condition that \([\Gamma]A = A\) and \([\Gamma]B = B\) is easily satisfied at each inductive step, using the definition of substitution.

Now, we consider the rules deriving \(\Gamma \vdash A <: \pm B \vdash \Delta\).

• Case

\[ A \text{ not headed by } \forall/\exists \]

\[ B \text{ not headed by } \forall/\exists \]

\[ \Gamma \vdash A \equiv B \vdash \Delta \]

\[ \Gamma \vdash A <: \pm B \vdash \Delta \]

\(\equiv\text{Equiv}\)

In this case, we appeal to Lemma 79 (Decidability of Equivalence).

• Case

\[ B \text{ not headed by } \forall \]

\[ \Gamma, \xi, \delta : \kappa \vdash [\delta/\alpha]A <: \pm B \vdash \Delta, \xi, \Theta \]

\[ \Gamma \vdash \forall \alpha : \kappa. A <: \pm B \vdash \Delta \]

\(\equiv\text{VL}\)

The premise has one fewer quantifier.

• Case

\[ \Gamma, \beta : \kappa \vdash A <: \pm B \vdash \Delta, \beta : \kappa, \Theta \]

\[ \Gamma \vdash A <: \pm \forall \beta : \kappa. B \vdash \Delta \]

\(\equiv\text{VR}\)

The premise has one fewer quantifier.

• Case

\[ \Gamma, \alpha : \kappa \vdash A <: \pm B \vdash \Delta, \alpha : \kappa, \Theta \]

\[ \Gamma \vdash \exists \alpha : \kappa. A <: \pm B \vdash \Delta \]

\(\equiv\text{EL}\)

The premise has one fewer quantifier.

• Case

\[ A \text{ not headed by } \exists \]

\[ \Gamma, \xi, \beta : \kappa \vdash A <: \pm [\beta/\beta]B \vdash \Delta, \xi, \Theta \]

\[ \Gamma \vdash A <: \pm \exists \beta : \kappa. B \vdash \Delta \]

\(\equiv\text{ER}\)

The premise has one fewer quantifier.
Proof of Theorem 1 (Decidability of Subtyping) \(\text{thm:subtyping-decidable}\)

- Case
  \[
  \frac{\Gamma \vdash A <: - B \vdash \Delta \quad \text{neg}(A) \quad \text{nonpos}(B)}{\Gamma \vdash A <: + B \vdash \Delta} \quad \leq_{L}
  \]

Consider whether \(B\) is negative.

- Case \(\text{neg}(B)\):
  \[
  B = \forall \beta : \kappa . B' \\
  \Gamma, \beta : \kappa \vdash A <: - B' \vdash \Delta, \beta : \kappa, \Theta \quad \text{Inversion on the premise}
  \]
  There is one fewer quantifier in the subderivation.

- Case \(\text{nonneg}(B)\):
  In this case, \(B\) is not headed by a \(\forall\).
  \[
  A = \forall \alpha : \kappa . A' \\
  \Gamma, \vdash_{\alpha, \Delta} : \kappa \vdash [\alpha/\alpha] A' <: - \vdash \Delta, \vdash_{\alpha}, \Theta \quad \text{Inversion on the premise}
  \]
  There is one fewer quantifier in the subderivation.

- Case
  \[
  \frac{\Gamma \vdash A <: - B \vdash \Delta \quad \text{nonpos}(A) \quad \text{neg}(B)}{\Gamma \vdash A <: + B \vdash \Delta} \quad \leq_{L}
  \]

  \[
  B = \forall \beta : \kappa . B' \\
  \Gamma, \beta : \kappa \vdash A <: - B' \vdash \Delta, \beta : \kappa, \Theta \quad \text{Inversion on the premise}
  \]
  There is one fewer quantifier in the subderivation.

- Case
  \[
  \frac{\Gamma \vdash A <: + B \vdash \Delta \quad \text{pos}(A) \quad \text{nonneg}(B)}{\Gamma \vdash A <: - B \vdash \Delta} \quad \leq_{L}
  \]

  This case is similar to the \(\leq_{R}\) case.

- Case
  \[
  \frac{\Gamma \vdash A <: + B \vdash \Delta \quad \text{nonneg}(A) \quad \text{pos}(B)}{\Gamma \vdash A <: - B \vdash \Delta} \quad \leq_{R}
  \]

  This case is similar to the \(\leq_{L}\) case.

\[\square\]

H’.3 Decidability of Matching and Coverage

Lemma 80 (Decidability of Expansion Judgments).
Given branches \(\Pi\), it is decidable whether:

1. there exists \(\Pi'\) such that \(\Pi \preceq \Pi'\);
2. there exist \(\Pi_L\) and \(\Pi_R\) such that \(\Pi \dashv \Pi_L \parallel \Pi_R\);
3. there exists \(\Pi'\) such that \(\Pi \overset{\text{var}}{\sim} \Pi'\);
4. there exists \(\Pi'\) such that \(\Pi \overset{\bot}{\sim} \Pi'\).
Proof. In each part, by induction on $\Pi$: Every rule either has no premises, or breaks down $\Pi$ in its nontrivial premise.

Theorem 2 (Decidability of Coverage). Given a context $\Gamma$, branches $\Pi$ and types $\vec{\Lambda}$, it is decidable whether $\Gamma \vdash \Pi$ covers $\vec{\Lambda}$ is derivable.

Proof. By induction on, lexicographically, (1) the number of $\land$ connectives appearing in $\vec{\Lambda}$, and then (2) the size of $\vec{\Lambda}$, considered to be the sum of the sizes $|\Lambda|$ of each type $\Lambda$ in $\vec{\Lambda}$.

(For $\text{CoversVar}$, $\text{Covers\times}$ and $\text{Covers\mp}$ we also use the appropriate part of Lemma 80 (Decidability of Expansion Judgments).)

- Case $\text{CoversEmpty}$: No premises.
- Case $\text{CoversVar}$: The number of $\land$ connectives does not grow, and $\vec{\Lambda}$ gets smaller.
- Case $\text{Covers1}$: The number of $\land$ connectives does not grow, and $\vec{\Lambda}$ gets smaller.
- Case $\text{Covers\times}$: The number of $\land$ connectives does not grow, and $\vec{\Lambda}$ gets smaller, since $|\Lambda_1| + |\Lambda_2| < |\Lambda_1 \times \Lambda_2|$.
- Case $\text{Covers\mp}$: Here we have $\vec{\Lambda} = (\Lambda_1 + \Lambda_2, \vec{\Lambda})$. In the first premise, we have $(\Lambda_1, \vec{\Lambda})$, which is smaller than $\vec{\Lambda}$, and in the second premise we have $(\Lambda_2, \vec{\Lambda})$, which is likewise smaller. (In both premises, the number of $\land$ connectives does not grow.)
- Case $\text{Covers\exists}$: The number of $\land$ connectives does not grow, and $\vec{\Lambda}$ gets smaller.
- Case $\text{CoversEq}$: The first premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (4). The number of $\land$ connectives in $\vec{\Lambda}$ gets smaller (note that applying $\Delta$ as a substitution cannot add $\land$ connectives).
- Case $\text{CoversEqBot}$: Decidable by Lemma 78 (Decidability of Propositional Judgments) (4).

H'.4 Decidability of Typing

Theorem 3 (Decidability of Typing).

(i) Synthesis: Given a context $\Gamma$, a principality $p$, and a term $e$,
    it is decidable whether there exist a type $\Lambda$ and a context $\Delta$ such that
    $\Gamma \vdash e \Rightarrow A \leftarrow \leftarrow \leftarrow \Delta$.

(ii) Spines: Given a context $\Gamma$, a spine $s$, a principality $p$, and a type $\Lambda$ such that $\Gamma \vdash A$ type,
    it is decidable whether there exist a type $B$, a principality $q$ and a context $\Delta$ such that
    $\Gamma \vdash s : A p \Rightarrow B q \leftarrow \Delta$.

(iii) Checking: Given a context $\Gamma$, a principality $p$, a term $e$, and a type $B$ such that $\Gamma \vdash B$ type,
    it is decidable whether there is a context $\Delta$ such that
    $\Gamma \vdash e \Leftarrow B p \leftarrow \Delta$.

(iv) Matching: Given a context $\Gamma$, branches $\Pi$, a list of types $\vec{\Lambda}$, a type $C$, and a principality $p$, it is decidable
    whether there exists $\Delta$ such that $\Gamma \vdash \Pi : \vec{\Lambda} \Leftarrow C p \leftarrow \Delta$.
    Also, if given a proposition $P$ as well, it is decidable whether there exists $\Delta$ such that $\Gamma / P \vdash \Pi : \vec{\Lambda} \Leftarrow C p \leftarrow \Delta$.

Proof. For rules deriving judgments of the form

\[
\begin{align*}
\Gamma \vdash e & \Rightarrow - - - - \\
\Gamma \vdash e & \Leftarrow B p \leftarrow - \\
\Gamma \vdash s : B p & \Rightarrow - - - - \\
\Gamma \vdash \Pi : \vec{\Lambda} & \Leftarrow C p \leftarrow -
\end{align*}
\]
(where we write “−” for parts of the judgments that are outputs), the following induction measure on such judgments is adequate to prove decidability:

\[
\langle e/s/\Pi, \Rightarrow / \gg, \#\text{large}(B), B \rangle \ 	ext{Match, } \vec{A}, \text{match judgment form}
\]

where \(<\ldots>\) denotes lexicographic order, and where (when comparing two judgments typing terms of the same size) the synthesis judgment (top line) is considered smaller than the checking judgment (second line). That is,

\[
\Rightarrow \prec \gg / \gg / \text{Match}
\]

Two match judgments are compared according to, first, the list of branches \(\Pi\) (which is a subterm of the containing case expression, allowing us to invoke the i.h. for the \texttt{Case} rule), then the size of the list of types \(\vec{A}\) (considered to be the sum of the sizes \(\|A\|\) of each type \(A\) in \(\vec{A}\)), and then, finally, whether the judgment is \(\Gamma/P \vdash \ldots\) or \(\Gamma \vdash \ldots\), considering the former judgment (\(\Gamma/P \vdash \ldots\)) to be larger.

Note that this measure only uses the input parts of the judgments, leading to a straightforward decidability argument.

We will show that in each rule deriving a synthesis, checking, spine or match judgment, every premise is smaller than the conclusion.

- **Case\texttt{EmptySpine}**: No premises.
- **Case\texttt{→Spine}**: In each premise, the expression/spine gets smaller (we have \(e/s\) in the conclusion, \(e\) in the first premise, and \(s\) in the second premise).
- **Case\texttt{Var}**: No nontrivial premises.
- **Case\texttt{Sub}**: The first premise has the same subject term \(e\) as the conclusion, but the judgment is smaller because our measure considers synthesis to be smaller than checking.

The second premise is a subtyping judgment, which by Theorem 1 is decidable.

- **Case\texttt{Anno}**: It is easy to show that the judgment \(\Gamma \vdash A!\) type is decidable. The second premise types \(e\), but the conclusion types \((e : A)\), so the first part of the measure gets smaller.

- **Cases\texttt{1I, 1^α}**: No premises.
- **Case\texttt{∀I}**: Both the premise and conclusion type \(e\), and both are checking; however, \(#\text{large}(A_0) < \#\text{large}(\forall \alpha : \kappa. A_0)\), so the premise is smaller.
- **Case\texttt{∀Spine}**: Both the premise and conclusion type \(e/s\), and both are spine judgments; however, \(#\text{large}(−)\) decreases.
- **Case\texttt{∧I}**: By Lemma 78 (Decidability of Propositional Judgments) (2), the first premise is decidable. For the second premise, \(#\text{large}(\Theta A_0) = \#\text{large}(\forall \alpha : \kappa. A_0) < \#\text{large}(\forall \alpha : \kappa. A_0 \land p)\).
- **Case\texttt{⊃I}**: For the first premise, use Lemma 78 (Decidability of Propositional Judgments) (5). In the second premise, \(#\text{large}(−)\) gets smaller (similar to the\texttt{∧I} case).
- **Case\texttt{⊃I⊥}**: The premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (5).
- **Case\texttt{⊃Spine}**: Similar to the\texttt{∧I} case.
- **Cases\texttt{→I, →I^α}**: In the premise, the term is smaller.
- **Cases\texttt{→E, →E-}**: In all premises, the term is smaller.
- **Cases\texttt{+Iκ, +I^ακ, ×I, ×Iκ}**: In all premises, the term is smaller.
- **Case\texttt{Case}**: In the first premise, the term is smaller. In the second premise, we have a list of branches that is a proper subterm of the case expression. The third premise is decidable by Theorem 2.
We now consider the match rules:

- **Case** `MatchEmpty` No premises.
- **Case** `MatchSeq` In each premise, the list of branches is properly contained in \( \Pi \), making each premise smaller by the first part (“e/s/\Pi”) of the measure.
- **Case** `MatchBase` The term \( e \) in the premise is properly contained in \( \Pi \).
- **Cases** \( \{ \text{Match} \times, \text{Match}+, \text{MatchNeg}, \text{MatchWild} \} \) Smaller by part (2) of the measure.
- **Case** `Match\⊥` The premise has a smaller \( \bar{A} \), so it is smaller by the \( \bar{A} \) part of the measure. (The premise is the other judgment form, so it is *larger* by the “match judgment form” part, but \( \bar{A} \) lexicographically dominates.)
- **Case** `MatchUnify` For the premise, use Lemma \[78\] (Decidability of Propositional Judgments) (4).

\[ \text{Lemma 81 (Determinacy of Auxiliary Judgments)} \]

(1) Elimeq: Given \( \Gamma, \sigma, t, \kappa \) such that \( \text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset \) and \( D_1 :: \Gamma / \sigma \Downarrow t : \kappa \vdash \Delta_1 \) and \( D_2 :: \Gamma / \sigma \Downarrow t : \kappa \vdash \Delta_2 \),

it is the case that \( \Delta_1 = \Delta_2 \).

(2) Instantiation: Given \( \Gamma, \hat{\alpha}, t, \kappa \) such that \( \hat{\alpha} \in \text{unsolved}(\Gamma) \) and \( \Gamma \vdash t : \kappa \) and \( \hat{\alpha} \notin \text{FV}(t) \)

and \( D_1 :: \Gamma \vdash \hat{\alpha} := t : \kappa \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash \hat{\alpha} := t : \kappa \vdash \Delta_2 \),

it is the case that \( \Delta_1 = \Delta_2 \).

(3) Symmetric instantiation:

Given \( \Gamma, \hat{\alpha}, \hat{\beta}, \kappa \) such that \( \hat{\alpha}, \hat{\beta} \in \text{unsolved}(\Gamma) \) and \( \hat{\alpha} \neq \hat{\beta} \)

and \( D_1 :: \Gamma \vdash \hat{\alpha} := \hat{\beta} : \kappa \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash \hat{\beta} := \hat{\alpha} : \kappa \vdash \Delta_2 \)

it is the case that \( \Delta_1 = \Delta_2 \).

(4) Checkeq: Given \( \Gamma, \sigma, t, \kappa \) such that \( D_1 :: \Gamma \vdash \sigma \Downarrow t : \kappa \vdash \Delta_1 \)

and \( D_2 :: \Gamma \vdash \sigma \Downarrow t : \kappa \vdash \Delta_2 \),

it is the case that \( \Delta_1 = \Delta_2 \).

(5) Elimprop: Given \( \Gamma, P \) such that \( D_1 :: \Gamma / P \Downarrow \Delta_1 \) and \( D_2 :: \Gamma / P \Downarrow \Delta_2 \)

it is the case that \( \Delta_1 = \Delta_2 \).

(6) Checkprop: Given \( \Gamma, P \) such that \( D_1 :: \Gamma \vdash P \text{ true } \Downarrow \Delta_1 \) and \( D_2 :: \Gamma \vdash P \text{ true } \Downarrow \Delta_2 \),

it is the case that \( \Delta_1 = \Delta_2 \).

**Proof.**

**Proof of Part (1) (Elimeq).**

Rule **ElimeqZero** applies if and only if \( \sigma = t = \text{zero} \).

Rule **ElimeqSucc** applies if and only if \( \sigma \) and \( t \) are headed by **succ**.

Now suppose \( \sigma = \alpha \).

- Rule **ElimeqUvarRef** applies if and only if \( t = \alpha \). (Rule **ElimeqClash** cannot apply; rules **ElimeqUvarL** and **ElimeqUvarR** have a free variable condition; rules **ElimeqUvarLL** and **ElimeqUvarRL** have a condition that \( \sigma \neq t \).)

In the remainder, assume \( t \neq \alpha \).
Proof of Lemma 81 (Determinacy of Auxiliary Judgments)

- If $\alpha \in \text{FV}(t)$, then rule $\text{ElimeqUvarL}$ applies, and no other rule applies (including $\text{ElimeqUvarR}$ and $\text{ElimeqClash}$).
  In the remainder, assume $\alpha \notin \text{FV}(t)$.
- Consider whether $\text{ElimeqUvarR}$ applies. The conclusion matches if we have $t = \beta$ for some $\beta \neq \alpha$ (that is, $\sigma = \alpha$ and $t = \beta$). But $\text{ElimeqUvarR}$ has a condition that $\beta \in \text{FV}(\sigma)$, and $\sigma = \alpha$, so the condition is not satisfied.

Proof of Part (5) (Elimprop).
There is only one rule deriving this judgment; the result follows by part (1).

Proof of Part (6) (Checkprop).
There is only one rule deriving this judgment; the result follows by part (4).

Lemma 82 (Determinacy of Equivalence).

1. Propositional equivalence: Given $\Gamma$, $P$, $Q$ such that $D_1 : \Gamma \vdash P \equiv Q \vdash \Delta_1$ and $D_2 : \Gamma \vdash P \equiv Q \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.
(2) Type equivalence: Given $\Gamma, A, B$ such that $D_1 :: \Gamma \vdash A \equiv B \vdash \Delta_1$ and $D_2 :: \Gamma \vdash A \equiv B \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

Proof.

Proof of Part (1) (propositional equivalence). Only one rule derives judgments of this form; the result follows from Lemma 81 (Determinacy of Auxiliary Judgments) (4).

Proof of Part (2) (type equivalence). If neither $A$ nor $B$ is an existential variable, they must have the same head connectives, and the same rule must conclude both derivations.

If $A$ and $B$ are the same existential variable, then only $\equiv \text{Exvar}$ applies (due to the free variable conditions in $\equiv \text{InstantiateL}$ and $\equiv \text{InstantiateR}$).

If $A$ and $B$ are different unsolved existential variables, the judgment matches the conclusion of both $\equiv \text{InstantiateL}$ and $\equiv \text{InstantiateR}$ but by part (3) of Lemma 81 (Determinacy of Auxiliary Judgments), we get the same output context regardless of which rule we choose.

Theorem 4 (Determinacy of Subtyping).

(1) Subtyping: Given $\Gamma, e, A, B$ such that $D_1 :: \Gamma \vdash A \ll B \vdash \Delta_1$ and $D_2 :: \Gamma \vdash A \ll B \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

Proof. First, we consider whether we are looking at positive or negative subtyping, and then consider the outermost connective of $A$ and $B$:

- If $\Gamma \vdash A \ll B \vdash \Delta_1$ and $\Gamma \vdash A \ll B \vdash \Delta_2$, then we know the last rule ending the derivation of $D_1$ and $D_2$ must be:

  \[
  \begin{array}{ccc}
  \forall & \exists & \text{other} \\
  \forall & <:\forall R & <:\forall L \\
  \exists & <:\exists L & <:\exists R \\
  \text{other} & <:\exists L & <:\exists R \\
  & <:\forall L & <:\forall R \\
  & <:\exists R & <:\exists L \\
  & \equiv & \equiv \\
  \end{array}
  \]

  The only case in which there are two possible final rules is in the $\forall/\forall$ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A \ll B \vdash \Delta_1$ and $\Gamma \vdash A \ll B \vdash \Delta_2$.

- If $\Gamma \vdash A \ll B \vdash \Delta_1$ and $\Gamma \vdash A \ll B \vdash \Delta_2$, then we know the last rule ending the derivation of $D_1$ and $D_2$ must be:

  \[
  \begin{array}{ccc}
  \forall & \exists & \text{other} \\
  \forall & <:\forall R & <:\forall L \\
  \exists & <:\exists L & <:\exists R \\
  \text{other} & <:\exists L & <:\exists R \\
  & <:\forall L & <:\forall R \\
  & <:\exists R & <:\exists L \\
  & \equiv & \equiv \\
  \end{array}
  \]

  The only case in which there are two possible final rules is in the $\forall/\forall$ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A \ll B \vdash \Delta_1$ and $\Gamma \vdash A \ll B \vdash \Delta_2$.

As a result, the result follows by a routine induction.

Theorem 5 (Determinacy of Typing).

(1) Checking: Given $\Gamma, e, A, p$ such that $D_1 :: \Gamma \vdash e \iff A \Rightarrow p \vdash \Delta_1$ and $D_2 :: \Gamma \vdash e \iff A \Rightarrow p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$. 

March 16, 2018


(2) Synthesis: Given $\Gamma, e \text{ such that } D_1 \vdash e \Rightarrow B_1 \ p_1 \vdash \Delta_1 \ \text{and} \ D_2 \vdash e \Rightarrow B_2 \ p_2 \vdash \Delta_2$, it is the case that $B_1 = B_2$ and $p_1 = p_2$ and $\Delta_1 = \Delta_2$.

(3) Spine judgments:

Given $\Gamma, e, A, p$ such that $D_1 \vdash e : A \Rightarrow C_1 \ q_1 \vdash \Delta_1$ and $D_2 \vdash e : A \Rightarrow C_2 \ q_2 \vdash \Delta_2$, it is the case that $C_1 = C_2$ and $q_1 = q_2$ and $\Delta_1 = \Delta_2$.

The same applies for derivations of the principality-recovering judgments $\Gamma \vdash e : A \Rightarrow C_k \ [q_k] \vdash \Delta_k$.

(4) Match judgments:

Given $\Gamma, \Pi, \tilde{A}, p, C$ such that $D_1 \vdash \Pi : \tilde{A} \Leftarrow C \ p \vdash \Delta_1$ and $D_2 \vdash \Pi : \tilde{A} \Leftarrow C \ p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

Given $\Gamma, \Pi, \tilde{A}, p, C$

such that $D_1 \vdash \Gamma / \Pi : \tilde{A} \Leftarrow C \ p \vdash \Delta_1$ and $D_2 \vdash \Gamma / \Pi : \tilde{A} \Leftarrow C \ p \vdash \Delta_2$,

it is the case that $\Delta_1 = \Delta_2$.

**Proof.**

**Proof of Part (1) (checking).**

The rules with a checking judgment in the conclusion are: $\forall I \exists I \land I \lor I \rightarrow I \times I 1 \vec{\alpha} \vec{x} \alpha \text{Nil Cons}$

The table below shows which rules apply for given $e$ and $A$. The extra “chk-I” column highlights the role of the “chk-I” (“check-intro”) category of syntactic forms: we restrict the introduction rules for $\forall$ and $\rightarrow$ to type only these forms. For example, given $e = x$ and $A = (\forall \alpha : x. A_0)$, we need not choose between $\text{Sub}$ and $\forall I$, the latter is ruled out by its chk-I premise.

<table>
<thead>
<tr>
<th>$e$</th>
<th>$A$</th>
<th>$\land$</th>
<th>$\lor$</th>
<th>$\rightarrow$</th>
<th>$\times$</th>
<th>$\vec{\alpha}$</th>
<th>$\vec{x}$</th>
<th>$\text{Nil}$</th>
<th>$\text{Cons}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda x. e_0$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\text{rec } x. v$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\text{inj}_k e_0$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\langle e_1, e_2 \rangle$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$[]$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$e_1 :: e_2$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$()$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\text{case}(e_0, \Pi)$</td>
<td>$\text{chk-I}$</td>
<td>$\forall I$</td>
<td>$\exists I$</td>
<td>$\land I$</td>
<td>$\rightarrow I$</td>
<td>$\times I$</td>
<td>$\vec{\alpha}$</td>
<td>$\vec{x}$</td>
<td>$\text{Nil}$</td>
</tr>
<tr>
<td>$\forall x. A$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
</tr>
<tr>
<td>$e_1 s$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
<td>$\forall I$</td>
</tr>
</tbody>
</table>

**Notes:**

- **Note 1:** The choice between $\forall I$ and $\exists I$ is resolved by Lemma 81 (Determinacy of Auxiliary Judgments) (5).

- **Note 2:** Fixed points are a checking form, but not an introduction form. So if $e$ is $\text{rec } x. v$, we need not choose between an introduction rule for a large connective and the $\text{rec}$ rule: only the $\text{rec}$ rule is viable. Large connectives must, therefore, be introduced inside the typing of the body $v$.

- **Note 3:** Case expressions are a checking form, but not an introduction form. So if $e$ is a case expression, we need not choose between an introduction rule for a large connective and the $\text{case}$ rule: only the $\text{case}$ rule is viable. Large connectives must, therefore, be introduced inside the branches.
Proof of Part (2) (synthesis). Only four rules have a synthesis judgment in the conclusion: \( \text{Var} \), \( \text{Anno} \), \( \rightarrow \text{E} \), and \( \rightarrow \text{E} \). Rule \( \text{Var} \) applies if and only if \( e \) has the form \( x \). Rule \( \text{Anno} \) applies if and only if \( e \) has the form \( (e_0 : A) \).

Otherwise, the judgment can be derived only if \( e \) has the form \( e_1 \, e_2 \), by \( \rightarrow \text{E} \) or \( \rightarrow \text{E} \). If \( D_1 \) and \( D_2 \) both end in \( \rightarrow \text{E} \) or \( \rightarrow \text{E} \), we are done. Suppose \( D_1 \) ends in \( \rightarrow \text{E} \) and \( D_2 \) ends in \( \rightarrow \text{E} \). By i.h., the \( p \) in the first subderivation of \( \rightarrow \text{E} \) must be equal to the one in the first subderivation of \( \rightarrow \text{E} \); that is, \( p = 1 \). Thus the inputs to the respective second subderivations match, so by i.h. their outputs match; in particular, \( q = f \).

However, from the condition in \( \rightarrow \text{E} \) it must be the case that \( \text{FEV}(\Delta | C) \neq \emptyset \), which contradicts the condition \( \text{FEV}(\Delta | C) = \emptyset \) in \( \rightarrow \text{E} \).

Proof of Part (3) (spine judgments). For the ordinary spine judgment, rule \( \text{EmptySpine} \) applies if and only if the given spine is empty. However, the choice of rule is determined by the head constructor of the input type: \( \rightarrow \text{Spine} \lor \text{Spine} \lor \text{Spine} \lor \text{Spine} \).

For the principality-recovering spine judgment: If \( p = 1 \) and \( q = 1 \), only rule \( \text{SpinePass} \) applies. If \( p = 1 \) and \( q = ! \), only rule \( \text{SpinePass} \) applies. If \( p = ! \) and \( q = ! \), then the rule is determined by \( \text{FEV}(C) \): if \( \text{FEV}(C) = 0 \) then only \( \text{SpineRecover} \) applies; otherwise, \( \text{FEV}(C) \neq 0 \) and only \( \text{SpinePass} \) applies.

Proof of Part (4) (matching). First, the elimination judgment form \( \Gamma / \rho \vdash \ldots \) cannot be the case that both \( \Gamma / \sigma \vdash t : \kappa \vdash \perp \) and \( \Gamma / \sigma \vdash t : \kappa \vdash \Theta \), so either \( \text{Match} \vdash \perp \) concludes both \( D_1 \) and \( D_2 \) (and the result follows), or \( \text{Match} \vdash \perp \) concludes both \( D_1 \) and \( D_2 \) (in which case, apply the i.h.).

Now the main judgment form, without “/ \( \rho \)”: either \( \Pi \) is empty, or has length one, or has length greater than one. \( \text{MatchEmpty} \) applies if and only if \( \Pi \) is empty, and \( \text{MatchSeq} \) applies if and only if \( \Pi \) has length greater than one. In this part of the proof, we assume \( \Pi \) has length one.

Moreover, \( \text{MatchBase} \) applies if and only if \( A \) has length zero. So in the rest of this part, we assume the length of \( A \) is at least one.

Let \( A \) be the first type in \( A \). Inspection of the rules shows that given particular \( A \) and \( \rho \), where \( \rho \) is the first pattern, only a single rule can apply, or no rule (“\( \emptyset \)” can apply, as shown in the following table:

<table>
<thead>
<tr>
<th>( \exists )</th>
<th>( \land )</th>
<th>( + )</th>
<th>( \times )</th>
<th>Vec</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>( \text{Match} \vdash +, )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{Match} \vdash )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>( \text{Match} \vdash \perp )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( J' \)

Soundness

\( J'.1 \) Instantiation

Lemma 83 (Soundness of Instantiation).

If \( \Gamma \vdash \hat{\alpha} : \tau : \kappa \vdash \Delta \) and \( \hat{\alpha} \notin \text{FV}(\Gamma | \tau) \) and \( \Gamma | \tau = \tau \) and \( \Delta \to \Omega \) then \( \Omega | \hat{\alpha} = \Omega | \tau \).

Proof. By induction on the derivation of \( \Gamma \vdash \hat{\alpha} : \tau : \kappa \vdash \Delta \).

- Case
  \[
  \Gamma_0 \vdash \tau : \kappa, \hat{\alpha} \vdash \hat{\alpha} : \tau : \kappa \vdash \Delta \to \Omega
  \]

  \[
  \Delta \vdash \hat{\alpha} = \Delta | \tau \quad \text{By definition}
  \]
  \[
  \Omega | \hat{\alpha} = \Omega | \tau \quad \text{By Lemma 29 (Substitution Monotonicity) to each side}
  \]

Proof of Lemma 83 (Soundness of Instantiation) lem:instantiation-soundness
• Case \( \hat{\beta} \in \text{unsolved}(\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa]) \)
  \[
  \Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} := \hat{\beta} : \kappa \vdash \Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \]  

  \[
  \Delta \vdash \Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] = \text{unsolved} \]

  By definition

  \[
  [\Delta]\hat{\beta} = [\Delta]\hat{\alpha} \]  

  Applying \( \Omega \) to each side

  \[
  [\Omega][\Delta]\hat{\beta} = [\Omega][\Delta]\hat{\alpha} \]

  By Lemma 29 (Substitution Monotonicity) to each side

  \[
  \frac{[\Delta]\hat{\beta} = [\Delta]\hat{\alpha}}{[\Omega][\Delta]\hat{\beta} = [\Omega][\Delta]\hat{\alpha}} \]

• Case
  \[
  \Gamma'[\hat{\alpha}_1 : \ast, \hat{\alpha}_2 : \ast, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 := \tau_1 : \ast \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \ast \vdash \Delta \]

  \[
  \frac{\Gamma' \vdash \hat{\alpha}_1 := \tau_1 : \ast \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \ast \vdash \Delta}{\Gamma[\hat{\alpha} : \ast] \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \ast \vdash \Delta} \]

  By definition of substitution

  \[
  \Delta \rightarrow \Omega \]  

  Given

  \[
  \Gamma' \vdash \hat{\alpha}_1 := \tau_1 : \ast \vdash \Theta \]

  Subderivation

  \[
  \Theta \rightarrow \Delta \]

  By Lemma 43 (Instantiation Extension)

  \[
  \Theta \rightarrow \Omega \]

  By Lemma 33 (Extension Transitivity)

  \[
  [\Omega]\hat{\alpha}_1 = [\Omega][\Theta]\tau_1 \]

  By i.h.

  \[
  \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \ast \vdash \Delta \]

  Subderivation

  \[
  [\Omega]\hat{\alpha}_2 = [\Omega][\Theta]\tau_2 \]

  By i.h.

  \[
  = [\Omega][\Theta]\tau_2 \]

  By Lemma 29 (Substitution Monotonicity)

  \[
  (\Omega[\Theta]\tau_1) \oplus (\Omega[\Theta]\tau_2) = (\Omega[\Theta](\hat{\alpha}_1 \oplus \hat{\alpha}_2)) \]

  By above equalities

  \[
  = [\Omega](\hat{\alpha}_1 \oplus \hat{\alpha}_2) \]

  By definition of substitution

  \[
  = [\Omega](\Gamma'[\hat{\alpha}]) \]

  By definition of substitution

  \[
  = [\Omega]\hat{\alpha} \]

  By Lemma 29 (Substitution Monotonicity)

  \[
  [\Omega][\Theta](\tau_1 \oplus \tau_2) = [\Omega][\Theta](\hat{\alpha}_1 \oplus \hat{\alpha}_2) \]

  By definition of substitution

  \[
  = [\Omega][\Theta]\hat{\alpha} \]

  By definition of substitution

  \[
  [\Omega][\Theta]\, (\tau_1 \oplus \tau_2) = [\Omega][\Theta]\hat{\alpha} \]

  By definition of substitution

• Case
  \[
  \Gamma_0[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \text{zero} : \mathbb{N} \rightarrow \Gamma_0[\hat{\alpha} : \mathbb{N} = \text{zero}] \]

  Similar to the \text{InstSolve} case.

• Case
  \[
  \Gamma_0[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \text{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : \mathbb{N} \rightarrow \Delta \]

  \[
  \Gamma_0[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \text{succ}(t_1) : \mathbb{N} \rightarrow \Delta \]

  Similar to the \text{InstBin} case, but simpler.

\textbf{Lemma 84} (Soundness of Checkeq).

*If \( \Gamma \vdash \sigma : \kappa \rightarrow \Delta \) where \( \Delta \rightarrow \Omega \) then \[\Omega][\sigma] = [\Omega]t.\]*

\textit{Proof.} By induction on the given derivation.

• Case
  \[
  \Gamma \vdash u \equiv u : \kappa \rightarrow \Gamma \]

  \[
  [\Omega]u = [\Omega]u \]

  By reflexivity of equality
Proof of Lemma 84 (Soundness of Checkeq)

- **Cases** `CheckeqUnit`, `CheckeqZero` Similar to the `CheckeqVar` case.

- **Case** \( \Gamma \vdash \sigma_0 \equiv t_0 : N \vdash \Delta \) \( \text{CheckeqSucc} \)
  
  \[
  \Gamma \vdash \operatorname{succ}(\sigma_0) \equiv \operatorname{succ}(t_0) : N \vdash \Delta
  \]

  \[\Gamma \vdash \sigma_0 \equiv t_0 : N \vdash \Delta \quad \text{Subderivation}\]
  \[\Theta \vdash \Theta \sigma_1 \equiv \Theta t_1 : * \vdash \Delta \quad \text{Subderivation}\]
  \[\Delta \rightarrow \Omega \quad \text{Given}\]
  \[\Theta \rightarrow \Delta \quad \text{By Lemma 46 (Checkeq Extension)}\]
  \[\Theta \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)}\]
  \[\Theta \rightarrow \Delta \quad \text{By i.h. on first subderivation}\]
  \[\Theta \rightarrow \Omega \quad \text{By i.h. on second subderivation}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By Lemma 29 (Substitution Monotonicity)}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By Lemma 29 (Substitution Monotonicity)}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By transitivity of equality}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By congruence of equality}\]

- **Case** \( \Gamma \vdash \sigma_0 \equiv t_0 : * \vdash \Omega \) \( \Theta \vdash \Theta \sigma_1 \equiv \Theta t_1 : * \vdash \Delta \) \( \text{CheckeqBin} \)
  
  \[
  \Gamma \vdash \sigma_0 + \sigma_1 \equiv t_0 + t_1 : * \vdash \Delta
  \]

  \[\Gamma \vdash \sigma_0 \equiv t_0 : N \vdash \Delta \quad \text{Subderivation}\]
  \[\Theta \vdash \Theta \sigma_1 \equiv \Theta t_1 : * \vdash \Delta \quad \text{Subderivation}\]
  \[\Delta \rightarrow \Omega \quad \text{Given}\]
  \[\Theta \rightarrow \Delta \quad \text{By Lemma 46 (Checkeq Extension)}\]
  \[\Theta \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)}\]
  \[\Theta \rightarrow \Delta \quad \text{By i.h. on first subderivation}\]
  \[\Theta \rightarrow \Omega \quad \text{By i.h. on second subderivation}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By Lemma 29 (Substitution Monotonicity)}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By Lemma 29 (Substitution Monotonicity)}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By transitivity of equality}\]
  \[\Theta \sigma_1 = \Theta t_1 \quad \text{By congruence of equality}\]

- **Case** \( \Gamma[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \vdash \Delta \) \( \hat{\alpha} \notin \text{FV}(t) \) \( \text{CheckeqInsl} \)
  
  \[
  \Gamma[\hat{\alpha}] \vdash \hat{\alpha} \equiv t : \kappa \vdash \Delta
  \]

  \[\Gamma[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \vdash \Delta \quad \text{Subderivation}\]
  \[\hat{\alpha} \notin \text{FV}(t) \quad \text{Premise}\]
  \[\Omega[\hat{\alpha}] = \Omega t \quad \text{By Lemma 83 (Soundness of Instantiation)}\]

- **Case** \( \Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \sigma : \kappa \vdash \Delta \) \( \hat{\alpha} \notin \text{FV}(t) \) \( \text{CheckeqInsr} \)
  
  \[
  \Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} \equiv \sigma : \kappa \vdash \Delta
  \]

  Similar to the `CheckeqInsl` case.

**Lemma 85** (Soundness of Propositional Equivalence).
If \( \Gamma \vdash P \equiv Q \vdash \Delta \) where \( \Delta \rightarrow \Omega \) then \( [\Omega]^P = [\Omega]^Q \).

**Proof.** By induction on the given derivation.

- **Case** \( \Gamma \vdash \sigma_1 \equiv t_1 : N \vdash \Theta \) \( \Theta \vdash \Theta \sigma_2 \equiv \Theta t_2 : N \vdash \Delta \) \( \equiv \text{PropEq} \)
  
  \[
  \Gamma[\sigma_1 \equiv \sigma_2] \equiv (t_1 = t_2) : N \vdash \Delta
  \]
Proof of Lemma 85 (Soundness of Propositional Equivalence).


Proof. By induction on the given derivation.

- Case

  \[
  Γ ⊢ α ≡ α → Γ \]

  \[≡\text{Var}\]

  \[\sqsubseteq [Ω]α = [Ω]α \text{ By reflexivity of equality}\]

- Cases $≡\text{Exvar}$, $≡\text{Unit}$ Similar to the $≡\text{Var}$ case.

- Case

  \[
  Γ ⊢ A_1 ≡ B_1 ⊢ Θ \quad Θ ⊢ [Θ]A_2 ≡ [Θ]B_2 ⊢ Δ
  \]

  \[≡ \]

  \[Γ ⊢ A_1 ⊕ A_2 ≡ B_1 ⊕ B_2 ⊢ Δ \]

  \[
  Δ → Ω \quad \text{Given}
  \]

  \[
  Θ ⊢ [Θ]A_2 ≡ [Θ]B_2 ⊢ Δ \quad \text{Subderivation}
  \]

  \[
  Θ → Δ \quad \text{By Lemma 49 (Equivalence Extension)}
  \]

  \[
  Θ → Ω \quad \text{By Lemma 33 (Extension Transitivity)}
  \]

  \[
  Γ ⊢ A_1 ≡ B_1 ⊢ Θ \quad \text{Subderivation}
  \]

  \[
  [Ω]A_1 = [Ω]B_1 \quad \text{By i.h.}
  \]

  \[
  Δ → Ω \quad \text{Given}
  \]

  \[
  [Ω][Θ]A_2 = [Ω][Θ]B_2 \quad \text{By i.h.}
  \]

  \[
  [Ω]A_2 = [Ω]B_2 \quad \text{By Lemma 29 (Substitution Monotonicity)}
  \]

  \[\sqsubseteq ([Ω]A_1) ⊕ ([Ω]A_2) = ([Ω]B_1) ⊕ ([Ω]B_2) \quad \text{By above equations}\]

- Case

  \[
  Γ, α : κ ⊢ A_0 ≡ B_0 ⊢ Δ, α : κ, Δ' \]

  \[≡ V\]

  \[Γ ⊢ ∀α : κ. A_0 ≡ ∀α : κ. B_0 ⊢ Δ \]

  \[≡ ν\]

  \[
  Γ, α : κ ⊢ A_0 ≡ B_0 ⊢ Δ, α : κ, Δ' \quad \text{Subderivation}
  \]

  \[
  Δ → Ω \quad \text{Given}
  \]

  \[
  Γ, α : κ, \cdot → Δ, α : κ, Δ' \quad \text{By Lemma 49 (Equivalence Extension)}
  \]

Lemma 86 (Soundness of Algorithmic Equivalence).

Proof of Lemma 86 (Soundness of Algorithmic Equivalence)

\[ \Delta' \text{ soft} \]
\[ \Delta, \alpha : \kappa, \Delta' \rightarrow \Omega, \alpha : \kappa, \Omega Z \]
\[ \Gamma, \alpha : \kappa \vdash A_0 \text{ type} \]
\[ \Gamma, \alpha : \kappa \vdash B_0 \text{ type} \]
\[ FV(A_0) \subseteq \text{dom}(\Gamma, \alpha : \kappa) \]
\[ FV(B_0) \subseteq \text{dom}(\Gamma, \alpha : \kappa) \]
\[ \Gamma, \alpha : \kappa \rightarrow \Omega, \alpha : \kappa \]
\[ FV(A_0) \subseteq \text{dom}(\Omega, \alpha : \kappa) \]
\[ FV(B_0) \subseteq \text{dom}(\Omega, \alpha : \kappa) \]
\[ [\Omega, \alpha : \kappa, \Omega Z]A_0 = [\Omega, \alpha : \kappa]A_0 \]
\[ [\Omega, \alpha : \kappa, \Omega Z]B_0 = [\Omega, \alpha : \kappa]B_0 \]
\[ [\Omega, \alpha : \kappa]A_0 = [\Omega, \alpha : \kappa]B_0 \]
\[ [\Omega]A_0 = [\Omega]B_0 \]
\[ \forall \alpha : \kappa. [\Omega]A_0 = \forall \alpha : \kappa. [\Omega]B_0 \]
\[ [\Omega]([\forall \alpha : \kappa]A_0) = [\Omega]([\forall \alpha : \kappa]B_0) \]

• Case \( \Gamma \vdash P \equiv Q \rightarrow \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \)

\[
\begin{align*}
\Gamma \vdash A_0 &\equiv [\Theta]A_0 \\
\Gamma \vdash \Theta \equiv [\Theta]B_0 \\
\Gamma \vdash Q &\equiv [\Theta]B_0 \\
\Delta &\rightarrow \Omega \\
\Theta &\vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \\
\Theta &\vdash \Delta \\
\Gamma &\vdash \Delta \\
[\Omega]P &\equiv [\Omega]Q \\
\Theta &\vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \\
[\Omega][\Theta]A_0 &\equiv [\Omega][\Theta]B_0 \\
[\Omega]A_0 &\equiv [\Omega]B_0 \\
\end{align*}
\]

• Case \( \Gamma \vdash P \equiv Q \rightarrow \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \)

\[
\begin{align*}
\Gamma &\vdash A_0 \land P \equiv B_0 \land Q \rightarrow \Delta \\
\end{align*}
\]

Similar to the \( \equiv \rightarrow \) case.

• Case \( \Gamma[\delta] \vdash \delta : \tau \rightarrow \Delta \quad \delta \notin FV(\tau) \)

\[
\begin{align*}
\Gamma[\delta] &\vdash \delta \equiv \tau \rightarrow \Delta \\
\end{align*}
\]

\( \equiv \) Instantiation

\[
\begin{align*}
\Gamma[\delta] &\vdash \delta : \tau \rightarrow \Delta \\
[\Omega][\delta] &\equiv [\Omega][\tau] \\
\end{align*}
\]

\( \approx \) Instantiation

• Case Similar to the \( \equiv \) Instantiation case.

\( \approx \) Instantiation

J'.2 Soundness of Checkprop

Lemma 87 (Soundness of Checkprop).

If \( \Gamma \vdash P \text{ true } \rightarrow \Delta \) and \( \Delta \rightarrow \Omega \) then \( \Psi \vdash [\Omega]P \text{ true } \).

Proof. By induction on the derivation of \( \Gamma \vdash P \text{ true } \rightarrow \Delta \).
• Case \[ \Gamma \vdash \sigma = t : N \vdash \Delta \] \[ \Gamma \vdash \sigma = \text{true} \vdash \Delta \] \[ \text{CheckpropEq} \]

\[ \Gamma \vdash \sigma = t : N \vdash \Delta \] Subderivation
\[ [\Omega] \sigma = [\Omega] t \] By Lemma 84 (Soundness of Checkprop)
\[ \Psi \vdash [\Omega] \sigma = [\Omega] t \text{ true} \] By DeclCheckprop
\[ \Psi \vdash [\Omega] (\sigma = t) \text{ true} \] By def. of subst.
\[ \Psi \vdash [\Omega] P \text{ true} \] By \( P = (\sigma = t) \)

\[ \square \]

J’3 Soundness of Eliminations (Equality and Proposition)

Lemma 88 (Soundness of Equality Elimination).
If \( \Gamma \sigma = \sigma \) and \( [\Gamma] t = t \) and \( \Gamma \vdash \sigma : \kappa \) and \( \Gamma \vdash t : \kappa \) and \( \text{FEV} (\sigma) \cup \text{FEV} (t) = \emptyset \), then:

(1) If \( \Gamma / \sigma \triangleq t : \kappa \vdash \Delta \)
then \( \Delta = (\Gamma, \Theta) \) where \( \Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n) \) and
for all \( \Omega \) such that \( \Gamma \lambda \Omega \rangle \Omega \)
and all \( \tau^* \) such that \( \Omega \vdash \tau^* : \kappa^* \),
it is the case that \( [\Omega, \Theta] \tau^* = [\emptyset] [\Omega] \tau^* \), where \( \Theta = \text{mgu} (\sigma, t) \).

(2) If \( \Gamma / \sigma \triangleq t : \kappa \vdash \bot \) then \( \text{mgu} (\sigma, t) = \bot \) (that is, no most general unifier exists).

Proof. First, we need to recall a few properties of term unification.

(i) If \( \sigma \) is a term, then \( \text{mgu} (\sigma, \sigma) = \text{id} \).

(ii) If \( f \) is a unary constructor, then \( \text{mgu} (f(\sigma), f(t)) = \text{mgu} (\sigma, t) \), supposing that \( \text{mgu} (\sigma, t) \) exists.

(iii) If \( f \) is a binary constructor, and \( \sigma = \text{mgu} (f(\sigma_1, \sigma_2), f(t_1, t_2)) \) and \( \sigma_1 = \text{mgu} (\sigma_1, t_1) \)
and \( \sigma_2 = \text{mgu} ([\sigma_1] \sigma_2, [\sigma_1] t_2) \), then \( \sigma = \sigma_2 \circ \sigma_1 = \sigma_1 \circ \sigma_2 \).

(iv) If \( \alpha \notin \text{FV} (t) \), then \( \text{mgu} (\alpha, t) = (\alpha = t) \).

(v) If \( f \) is an \( n \)-ary constructor, and \( \sigma_i \) and \( t_i \) (for \( i \leq n \)) have no unifier, then \( f(\sigma_1, \ldots, \sigma_n) \) and \( f(t_1, \ldots, t_n) \)
have no unifier.

We proceed by induction on the derivation of \( \Gamma / \sigma \triangleq t : \kappa \vdash \Delta^\perp \), proving both parts with a single induction.

• Case \[ \Gamma / \alpha \triangleq \alpha : \kappa \vdash \Gamma \] \[ \text{ElimeqUvarRefl} \]

Here we have \( \Delta = \Gamma \), so we are in part (1).
Let \( \Theta = \text{id} \) (which is \( \text{mgu} (\sigma, \sigma) \)).
We can easily show \( [\text{id}] [\Omega] \alpha = [\Omega, \alpha] = [\Omega, \cdot] \alpha \).

• Case \[ \Gamma / \text{zero} \triangleq \text{zero} : N \vdash \Gamma \] \[ \text{ElimeqZero} \]

Similar to the \[ \text{ElimeqUvarRefl} \] case.

• Case \[ \Gamma / t_1 \triangleq t_2 : N \vdash \Delta^\perp \] \[ \Gamma / \text{succ}(t_1) \triangleq \text{succ}(t_2) : N \vdash \Delta^\perp \] \[ \text{ElimeqSucc} \]

We distinguish two subcases:
Proof of Lemma 88 (Soundness of Equality Elimination) lem:elimeq-soundness

- Case $\Delta^\perp = \Delta$:
  Since we have the same output context in the conclusion and premise, the “for all $t'$...” part follows immediately from the i.h. (1).
  The i.h. also gives us $\theta_0 = \text{mgu}(t_1, t_2)$.
  Let $\theta = \theta_0$. By property (ii), $\text{mgu}(t_1, t_2) = \text{mgu}((\text{succ}(t_1), \text{succ}(t_2))) = \theta$.

- Case $\Delta^\perp = \bot$:
  $\Gamma / t_1 \equiv t_2 : \bot \vdash \bot$ Subderivation
  $\text{mgu}(t_1, t_2) = \bot$ By i.h. (2) (\text{mgu}(t_1, t_2) = \bot)
  $\text{mgu}((\text{succ}(t_1), \text{succ}(t_2))) = \bot$ By contrapositive of property (ii)

- Case $\alpha \not\in FV(t)$ $(\alpha = -) \not\in \Gamma$ : 
  $\Gamma / \alpha \equiv t_0 : \kappa \vdash \Gamma, \alpha = t$ ElimeqUvarL

Here $\Delta \neq \bot$, so we are in part (1).

\[ [\Omega, \alpha = t]t' = \Theta \text{ by a property of substitution} \]

\[ = [\Omega][t/\alpha]\Theta t' \text{ by a property of substitution} \]

\[ = [\Omega]\Theta t' \text{ by } \text{mgu}(\alpha, t) = (\alpha/t) \]

$\therefore$ \[ = [\Omega]\Theta t' \text{ by a property of substitution (0 creates no evars)} \]

- Case $\alpha \not\in FV(t)$ $(\alpha = -) \not\in \Gamma$ : 
  $\Gamma / t \equiv \alpha : \kappa \vdash \Gamma, \alpha = t$ ElimeqUvarR

Similar to the ElimeqUvarL case.

- Case $\Gamma / 1 \equiv 1 : \ast \vdash \Gamma$ ElimeqUnit

Similar to the ElimeqUvarRef case.

- Case $\Gamma / \tau_1 \equiv \tau_1' : \ast \vdash \Theta \vdash [\Theta]\tau_1 \equiv [\Theta]\tau_2' : \ast \vdash \Delta^\perp$ ElimeqBin

Either $\Delta^\perp$ is some $\Delta$, or it is $\bot$.

- Case $\Delta^\perp = \Delta$:
  $\Gamma / \tau_1 \equiv \tau_1' : \ast \vdash \Theta \vdash [\Theta]\tau_1 \equiv [\Theta]\tau_2' : \ast \vdash \Delta^\perp$ Subderivation
  $\Theta = ([\Gamma, \Delta_1]$ By i.h. (1)
  \[(\text{IH-1st}) \quad [\Omega, \Delta_1]u_1 = [\theta_1][\Theta]u_1 \]
  \[\theta_1 = \text{mgu}(\tau_1, \tau_1') \]
  $\Theta / [\Theta]\tau_1 \equiv [\Theta]\tau_2' : \ast \vdash \Delta$ Subderivation
  $\Delta = ([\Theta, \Delta_2]$ By i.h. (1)
  \[(\text{IH-2nd}) \quad [\Omega, \Delta_1, \Delta_2]u_2 = [\theta_2][\Theta]u_2 \]
  \[\theta_2 = \text{mgu}(\tau_2, \tau_2') \]

Suppose $\Omega \vdash u : \kappa'$.

$[\Omega, \Delta_1, \Delta_2]u = [\theta_2][\Omega, \Delta_1]u$ By (IH-2nd), with $u_2 = u$
$= [\theta_2][\theta_1][\Theta]u$ By (IH-1st), with $u_1 = u$
$\therefore$ $= [\theta_2 \circ \theta_1]u$ By a property of substitution
$\therefore \theta_2 \circ \theta_1 = \text{mgu}((\tau_1 \oplus \tau_2), (\tau_1' \oplus \tau_2'))$ By property (iii) of substitution
Proof of [Lemma 88](Soundness of Equality Elimination) (lem:elimeq-soundness)

- Case $\Delta = \bot$:
  
  Use the i.h. (2) on the second premise to show $\text{mgu}(\tau_2, \tau'_2) = \bot$, then use property (v) of unification to show $\text{mgu}((\tau_1 \oplus \tau_2), (\tau'_1 \oplus \tau'_2)) = \bot$.

  - Case $\Gamma \vdash \tau_1 : \star \vdash \bot$:
    
    $\Gamma \vdash \tau_1 \oplus \tau_2 : \star \vdash \bot$ [ElimeqBinBot]
    
    Similar to the $\bot$ subcase for ElimeqSucc but using property (v) instead of property (ii).

  - Case $\sigma \# t$
    
    $\Gamma \vdash \sigma \equiv t : \kappa \vdash \bot$ [ElimeqClash]
    
    Since $\sigma \# t$, we know $\sigma$ and $t$ have different head constructors, and thus no unifier.  

March 16, 2018
Theorem 6 (Soundness of Algorithmic Subtyping).
If $[\Gamma]A = A$ and $[\Gamma]B = B$ and $\Gamma \vdash A$ type and $\Gamma \vdash B$ type and $\Delta \rightarrow \Omega$ and $\Gamma \vdash A \prec B \vdash \Delta$ then $[\Omega]A \vdash [\Omega]A \leq^\pm [\Omega]B$.

Proof. By induction on the given derivation.

- Case $B$ not headed by $\forall$.

Let $\Omega' = (\Omega, [[\alpha] \Theta])$.

\[
\begin{align*}
\Gamma, \alpha, \Theta &\vdash \Delta, \alpha, \Theta \\
\Gamma &\vdash \forall \alpha : \kappa. A_\alpha \prec B \vdash \Delta, \alpha, \Theta
\end{align*}
\]

Let $\Omega' = (\Omega, [[\alpha] \Theta])$.

\[
\begin{align*}
\Gamma, \alpha, \Theta &\vdash \Delta, \alpha, \Theta \\
\Gamma &\vdash \forall \alpha : \kappa. A_\alpha \prec B \vdash \Delta, \alpha, \Theta
\end{align*}
\]

Subderivation

- Case $\llhd : \llhd$ Similar to the $\llhd : \llhd$ case.

- Case $\llhd : \llhd$ Similar to the $\llhd : \llhd$ case.

\[
\begin{align*}
\Gamma, \beta : \kappa &\vdash A \prec B \vdash \Delta, \beta : \kappa, \Theta \\
\Gamma &\vdash A \llhd \forall \beta : \kappa. B_\beta \llhd \Delta
\end{align*}
\]
Proof of Theorem 6 (Soundness of Algorithmic Subtyping)

Case \(\vdash A <:\aleph B \vdash \Delta, \beta : \kappa, \Theta\)

- Subderivation

\(\Gamma, \beta : \kappa \vdash A <:\aleph B_0 \vdash \Delta, \beta : \kappa, \Theta\)

- Given

\(\Gamma \vdash A \text{ type}\)

- By Lemma 25 (Filling Completes)

\(\Gamma, \beta : \kappa \vdash A \text{ type}\)

- By Lemma 35 (Suffix Weakening)

\(\Gamma \vdash \forall \beta : \kappa. B_0 \text{ type}\)

- Given

\(\Gamma, \beta : \kappa \vdash B_0 \text{ type}\)

- By inversion (ForallWF)

\([\Omega'](\Delta, \beta : \kappa, \Theta) \vdash [\Omega']A \leq [\Omega']B_0\)

- By i.h.

\(\Gamma, \beta : \kappa \vdash \Delta, \beta : \kappa, \Theta\)

- By Lemma 54 (Subtyping Extension)

\(\Theta \text{ is soft}\)

- By Lemma 22 (Extension Inversion) (i)

\([\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega, \beta : \kappa]A \leq [\Omega, \beta : \kappa]B_0\)

- By Lemma 17 (Substitution Stability)

\([\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega]A \leq [\Omega]B_0\)

- By def. of substitution

\([\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega]A \leq [\Omega](\forall \beta : \kappa.B_0)\)

- By \(\forall R\)

- By def. of substitution

- Case \(\leq\aleph\) Similar to the \(\leq\forall R\) case.

- Case \(\Delta \rightarrow \Omega\)

\(\Gamma \vdash A \equiv B \vdash \Delta\)

- Given

\(\Gamma \vdash A \equiv B \vdash \Delta\)

- By Lemma 86 (Soundness of Algorithmic Equivalence)

\(\Delta \rightarrow \Omega\)

- By Lemma 49 (Equivalence Extension)

\(\Gamma \vdash A \text{ type}\)

- By Lemma 16 (Substitution for Type Well-Formedness)

\([\Omega]A \vdash [\Omega]A \equiv\equiv [\Omega]B\)

- By Lemma 54 (Completing Stability)

\(-\equiv\)

\([\Omega]A \vdash [\Omega]A \leq_\pm [\Omega]B\)

- By \(\equiv\text{Ref}\) 

\(\neg(A)\)

\(\negpos(A)\)

\(\negpos(B)\)

\(\negpos(A)\)

\(\negpos(B)\)

- Since \(\neg(A)\)

\(\negpos(A)\)

\(\negpos(B)\)

- By induction

\(\negpos(A)\)

\(\negpos(B)\)

- By \(\equiv\aleph\)

- Case \(\negpos(A)\)

\(\neg(A)\)

\(\negpos(B)\)

- By inversion

\(\negpos(B)\)

- By inversion

\(\negpos(B)\)

- By inversion

\(\negpos(A)\)

- Similar to the \(\negpos\) case.
• Case
\[
\begin{align*}
\Gamma \vdash A &<: B \vdash \Delta \quad \text{pos}(A) \\
\Gamma \vdash A &<: B \vdash \Delta \quad \text{nonneg}(B)
\end{align*}
\]
\[
\Gamma \vdash A <: B \vdash \Delta
\]
\(<: L L>

Similar to the \(<: L L>\) case.

• Case
\[
\begin{align*}
\Gamma \vdash A &<: B \vdash \Delta \quad \text{nonneg}(A) \\
\Gamma \vdash A &<: B \vdash \Delta \quad \text{pos}(B)
\end{align*}
\]
\[
\Gamma \vdash A <: B \vdash \Delta
\]
\(<: R R>

Similar to the \(<: L L>\) case.

\[\]

J'.4 Soundness of Typing

Theorem 7 (Soundness of Match Coverage).

1. If \(\Gamma \vdash \Pi \) covers \(\bar{A}\) and \(\Gamma \rightarrow \Omega\) and \(\Gamma \vdash \bar{A}\) ! types and \([\Gamma]\bar{A} = \bar{A}\) then \([\Omega]\Gamma \vdash \Pi \) covers \(\bar{A}\).

2. If \(\Gamma / P \vdash \Pi \) covers \(\bar{A}\) and \(\Gamma \rightarrow \Omega\) and \(\Gamma \vdash \bar{A}\) ! types and \([\Gamma]\bar{A} = \bar{A}\) and \([\Gamma]P = P\) then \([\Omega]\Gamma / P \vdash \Pi \) covers \(\bar{A}\).

Proof. By mutual induction on the given algorithmic coverage derivation.

1. • Case
\[
\Gamma \vdash \cdot \Rightarrow e_1 \ldots \text{covers } \cdot
\]
\([\Omega]\Gamma \vdash \cdot \Rightarrow e_1 \ldots \text{covers } \cdot \quad \text{By DeclCoversEmpty}

• Cases
\[
\text{CoversVar} \quad \text{Covers1} \quad \text{Covers×} \quad \text{Covers}+ \quad \text{Covers=} \quad \text{Covers∧} \quad \text{CoversVec}
\]
Use the i.h. and apply the corresponding declarative rule.

2. • Case
\[
\begin{align*}
\Gamma / [\Gamma]t_1 = \bar{A} &; \kappa \vdash \Delta \\
\Delta / [\Delta]\Pi \text{ covers } [\Delta]\bar{A}
\end{align*}
\]
\[
\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \bar{A}
\]
\(\text{CoversEq}\)

\[
\begin{align*}
\Gamma / [\Gamma]t_1 = \bar{A} &; \kappa \vdash \Delta \\
\Delta / [\Delta]\Pi \text{ covers } [\Delta]\bar{A}
\end{align*}
\]
Subderivation

\[
\begin{align*}
\Delta / [\Delta]\Pi \text{ covers } [\Delta]\bar{A}
\end{align*}
\]
Subderivation

\[
[\Omega]\Delta / [\Delta]\Pi \text{ covers } [\Delta]A_0, [\Delta]\bar{A}
\]
By i.h.

\[
\Delta = (\Gamma, \Theta) \quad \text{By Lemma} \quad \text{Soundness of Equality Elimination} \quad (1)
\]
\[
mgu(t_1, t_2) = \emptyset
\]

\[
\begin{align*}
[\Omega]\Delta & = [\emptyset][\Omega]\Gamma \\
[\Delta]\Pi & = [\emptyset]\Pi \\
([\Delta]\bar{A}) & = ([\emptyset]A_0, [\emptyset]\bar{A})
\end{align*}
\]
By Lemma \(93\) (Substitution Upgrade) (iii)

\[
\begin{align*}
[\emptyset][\Omega]\Pi & \vdash [\emptyset]\Pi \text{ covers } [\emptyset]\bar{A}
\end{align*}
\]
By Lemma \(93\) (Substitution Upgrade) (iv)

\[
\begin{align*}
[\emptyset][\Omega]\Gamma & \vdash [\emptyset]\Pi \text{ covers } [\emptyset]\bar{A}
\end{align*}
\]
By above equalities

\[
[\Omega]\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \bar{A}
\]
By DeclCoversEq
Proof of Theorem 7 (Soundness of Match Coverage) thm:coverage-soundness

119

- Case \( \Gamma / [\Gamma]t_1 \equiv [\Gamma]t_2 : \kappa \vdash \perp \)

  \( \Gamma / t_1 = t_2 \vdash \Pi \text{ covers } A \) \[\text{CoversEqBot} \]

\( \Gamma / [\Gamma]t_1 \equiv [\Gamma]t_2 : \kappa \vdash \perp \) Subderivation

\( \text{mgu}(\Gamma[t_1], [\Gamma]t_2) = \perp \) By Lemma 88 (Soundness of Equality Elimination) \[\text{(2)}\]

\( \text{mgu}(t_1, t_2) = \perp \) By given equality

\( [\Omega] \Gamma / t_1 = t_2 \vdash \Pi \text{ covers } A \) By DeclCoversEqBot

Lemma 89 (Well-formedness of Algorithmic Typing).

*Given* \( \Gamma \text{ ctx} \):

(i) If \( \Gamma \vdash e \Rightarrow A p \vdash \Delta \) then \( \Delta \vdash A p \text{ type} \).

(ii) If \( \Gamma \vdash s : A p \gg B q \vdash \Delta \) and \( \Gamma \vdash A p \text{ type} \) then \( \Delta \vdash B q \text{ type} \).

Proof. 1. Suppose \( \Gamma \vdash e \Rightarrow A p \vdash \Delta \):

- Case \( (x : A p) \in \Gamma \)

  \( \Gamma \vdash x \Rightarrow [\Gamma]A p \vdash \Gamma \) \[\text{Var} \]

  \( \Gamma = (\Gamma_0, x : A p, \Gamma_1) \) \( (x : A p) \in \Gamma \)

  \( \Gamma \vdash A p \text{ type} \) Follows from \( \Gamma \text{ ctx} \)

- Case \( \Gamma \vdash A ! \text{ type} \)

  \( \Gamma \vdash e \Leftarrow [\Gamma]A ! \vdash \Delta \) \[\text{Anno} \]

  By inversion

  \( \Gamma \Rightarrow \Delta \) By Lemma 51 (Typing Extension)

  \( \Delta \vdash A ! \text{ type} \) By Lemma 41 (Extension Weakening for Principal Typing)

  \( \Rightarrow \Delta \vdash [\Delta]A ! \text{ type} \) By Lemma 39 (Principal Agreement) \[\text{(i)}\]

- Case \( \Gamma \vdash e \Rightarrow A p \vdash \Theta \)

  \( \Theta \vdash s : [\Theta]A p \gg C q \vdash \Delta \)

  \( p = \perp \text{ or } q = ! \)

  \( \text{or } \text{FEV}(\Delta[C]) \neq \emptyset \) \[\text{→E} \]

  \( \Gamma \vdash e s \Rightarrow C q \vdash \Delta \) By inversion

  \( \Theta \vdash A p \text{ type} \) By induction

  \( \Theta \vdash [\Theta]A p \text{ type} \) By Lemma 40 (Right-Hand Subst. for Principal Typing)

  \( \Theta \text{ ctx} \) By implicit assumption

  \( \Theta \vdash s : [\Theta]A p \gg C q \vdash \Delta \) By inversion

  \( \Rightarrow \Delta \vdash C q \text{ type} \) By mutual induction

- Case \( \Gamma \vdash e \Rightarrow A ! \vdash \Theta \)

  \( \Theta \vdash s : [\Theta]A ! \gg C \vdash \Delta \)

  \( \text{FEV}(\Delta[C]) = \emptyset \) \[\text{→E} \]

  \( \Gamma \vdash e s \Rightarrow C ! \vdash \Delta \) By inversion

  \( \Theta \vdash A p \text{ type} \) By induction

  \( \Theta \vdash [\Theta]A p \text{ type} \) By Lemma 40 (Right-Hand Subst. for Principal Typing)

  \( \Theta \text{ ctx} \) By implicit assumption

  \( \Theta \vdash s : [\Theta]A p \gg C \vdash \Delta \) By inversion

  \( \Rightarrow \Delta \vdash C ! \text{ type} \) By mutual induction

  \( \text{FEV}(\Delta[C]) = \emptyset \) By inversion

  \( \Rightarrow \Delta \vdash C ! \text{ type} \) By PrincipalWF
2. Suppose \( \Gamma \vdash s : A \triangleright B \triangleright \Delta \) and \( \Gamma \vdash A p \) type:

- **Case**
  \[
  \begin{array}{c}
  \Gamma \vdash : A \triangleright A p \rightarrow \Gamma \\
  \end{array}
  \]
  \( \text{EmptySpine} \)

- **Case**
  \[
  \begin{array}{c}
  \Gamma \vdash e \leftrightarrow A p \rightarrow \Theta \quad \Theta \vdash s : [\Theta] B p \rightarrow C q \rightarrow \Delta \\
  \end{array}
  \]
  \( \rightarrow \text{Spine} \)

  \( \Gamma \vdash A \rightarrow B p \) type \( \text{Given} \)

  \( \Gamma \vdash B p \) type \( \text{By Lemma 42 (Inversion of Principal Typing)} \)

  \( \Theta \vdash B p \) type \( \text{By Lemma 41 (Extension Weakening for Principal Typing)} \)

  \( \Theta \vdash [\Theta] B p \) type \( \text{By Lemma 40 (Right-Hand Subst. for Principal Typing)} \)

  \( \Delta \vdash C q \) type \( \text{By induction} \)

- **Case**
  \[
  \begin{array}{c}
  \Gamma \vdash \alpha : \kappa \vdash e s : [\alpha/\alpha] A \rightarrow C q \rightarrow \Delta \\
  \end{array}
  \]
  \( \rightarrow \text{Spine} \)

  \( \Gamma \vdash \forall \alpha : \kappa . A p \rightarrow C q \rightarrow \Delta \) \( \text{Given} \)

  \( \Gamma \vdash \forall \alpha : \kappa . A p \rightarrow \text{By inversion} \)

  \( \Gamma, \alpha : \kappa \vdash A p \rightarrow \text{By inversion} \)

  \( \Gamma, \alpha : \kappa, \alpha : \kappa \vdash A p \rightarrow \text{By weakening} \)

  \( \Gamma, \alpha : \kappa \vdash [\alpha/\alpha] A p \rightarrow \text{By substitution} \)

  \( \Rightarrow \text{Spine} \)

  \( \Delta \vdash C q \) type \( \text{By induction} \)

- **Case**
  \[
  \begin{array}{c}
  \Gamma \vdash P \rightarrow \Theta \quad \Theta \vdash e s : [\Theta] A p \rightarrow C q \rightarrow \Delta \\
  \end{array}
  \]
  \( \rightarrow \text{Spine} \)

  \( \Gamma \vdash P \rightarrow A p \) type \( \text{Given} \)

  \( \Gamma \vdash P \rightarrow \text{prop} \) \( \text{By Lemma 42 (Inversion of Principal Typing)} \)

  \( \Gamma \vdash A p \) type \( \text{"} \)

  \( \Gamma \rightarrow \Theta \) \( \text{By Lemma 47 (Checkprop Extension)} \)

  \( \Theta \vdash A p \) type \( \text{By Lemma 41 (Extension Weakening for Principal Typing)} \)

  \( \Theta \vdash [\Theta] A p \) type \( \text{By Lemma 40 (Right-Hand Subst. for Principal Typing)} \)

  \( \Rightarrow \text{Spine} \)

  \( \Delta \vdash C q \) type \( \text{By induction} \)

  \( \Rightarrow \text{Spine} \)

  \( \Theta \vdash \alpha_1 \rightarrow \alpha_2 \) type \( \text{By rules} \)

  \( \Rightarrow \text{Spine} \)

  \( \Delta \vdash C q \) type \( \text{By induction} \)

\[\square\]

**Theorem 8** (Soundness of Algorithmic Typing).

*Given* \( \Delta \rightarrow \Omega : *

(i) If \( \Gamma \vdash e \leftrightarrow A p \rightarrow \Delta \) and \( \Gamma \vdash A p \) type then \( [\Omega] \Delta \vdash [\Omega] e \leftrightarrow [\Omega] A p \).

(ii) If \( \Gamma \vdash e \rightarrow A p \rightarrow \Delta \) then \( [\Omega] \Delta \vdash [\Omega] e \rightarrow [\Omega] A p \).
(iii) If $\Gamma \vdash s : A \triangleright B \ q \vdash \Delta$ and $\Gamma \vdash A \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] s : [\Omega] A \ p \triangleright [\Omega] B \ q$.

(iv) If $\Gamma \vdash s : A \ p \triangleright B \ [q] \vdash \Delta$ and $\Gamma \vdash A \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] s : [\Omega] A \ p \triangleright [\Omega] B \ [q]$.

(v) If $\Gamma \vdash \Pi :: A \leftarrow C \ p \ types$ and $[\Gamma] \widetilde{A} = \widetilde{A}$ and $\Gamma \vdash C \ p \ type$ then $[\Omega] \Delta \vdash [\Omega] \Pi :: [\Omega] \widetilde{A} \leftarrow [\Omega] C \ p$.

(vi) If $\Gamma / P \vdash \Pi :: \widetilde{A} \leftarrow C \ p \ types$ and $\Gamma \vdash \widetilde{A} ! \ types$ and $\Gamma \vdash C \ p \ type$ then $[\Omega] \Delta / [\Omega] P \vdash [\Omega] \Pi :: [\Omega] \widetilde{A} \leftarrow [\Omega] C \ p$.

Proof. By induction, using the measure in Definition 7.
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\(\Gamma \rightarrow \Delta\) By Lemma 51 (Typing Extension)
\(\Delta \rightarrow \Omega\) Given
\(\Gamma \rightarrow \Omega\)
\(\Omega \vdash A_0\) type
\([\Omega] \Omega \vdash [\Omega] A_0\) type
\([\Omega] \Omega = [\Omega] \Delta\) By Lemma 54 (Completing Stability)
\([\Omega] \Delta \vdash [\Omega] A_0\) type

\([\Omega][\Gamma] A_0 = [\Omega] A_0\)
\([\Omega] \Delta \vdash [\Omega] e_0 \Leftarrow [\Omega] A_0!\)
\([\Omega] \Delta \vdash [\Omega] (e_0 : A_0) \Rightarrow [\Omega] A_0!\) By above equality

\(\Gamma \vdash () \Leftarrow 1 \p \dashv \Gamma\) By Lemma 51 (Typing Extension)
\([\Omega] \Delta \vdash () \Leftarrow 1 \p\) By definition of substitution
\(\Omega \vdash () \Leftarrow 1 \p\) By above equality

\(\Gamma_0[\hat{\alpha} : \star] \vdash () \Leftarrow \hat{\alpha} \f \Gamma_0[\hat{\alpha} : \star = 1]\)
\(\Gamma_0[\hat{\alpha} : \star = 1] \rightarrow \Omega\) Given
\([\Omega] \hat{\alpha} = [\Omega]\hat{\alpha} = [\Omega]1\)
\(= 1\)
\([\Omega] \Delta \vdash () \Leftarrow 1 \f\) By Lemma 29 (Substitution Monotonicity) (i)
\(\Omega \vdash () \Leftarrow 1 \p\) By above equality

\(\nu \chk I\) \(\Gamma_0 : \k \vdash \nu \Leftarrow A_0\ p \dashv \Delta, \alpha : \k, \Theta\)
\(\Gamma \vdash \nu \Leftarrow \forall \alpha : \k, A_0\ p \dashv \Delta\)
\(\Delta \rightarrow \Omega\) Given
\(\Delta, \alpha \rightarrow \Omega, \alpha\) By \(--\Uvar\)
\(\Gamma, \alpha \rightarrow \Delta, \alpha, \Theta\) By Lemma 51 (Typing Extension)
\(\Theta\) soft By Lemma 22 (Extension Inversion) (i) (with \(\Gamma_R = \cdot\), which is soft)
\(\Delta, \alpha, \Theta \rightarrow \Omega, \Delta, \Theta\) By Lemma 25 (Filling Completes)
\(\Gamma, \alpha \vdash \nu \Leftarrow A_0\ p \dashv \Delta\) Subderivation
\([\Omega][\Gamma] A_0 = [\Omega] A_0\)
\([\Omega][\Gamma] A_0 = [\Omega][\Gamma] A_0\)
\([\Omega][\Gamma] A_0 = [\Omega][\Gamma] A_0\)
\([\Omega][\Gamma] A_0 = [\Omega][\Gamma] A_0\) By above equality
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \Delta, \alpha, \Theta \rightarrow \Omega, \alpha, [\Theta] \]

Above

\[ \Theta \text{ is soft} \]

By Lemma 53 (Softness Goes Away)

\[ [\Omega'] \Delta' = ([\Omega] \Delta, \alpha) \]

By above equality

\[ [\Omega] \Delta \vdash [\Omega] \nu \leftrightarrow [\Omega] A_0 p \]

By above equality

\[ [\Omega] \Delta \vdash [\Omega] \nu \leftrightarrow [\Omega] (\forall \alpha. A_0) p \]

By definition of substitution

- Case \( \Gamma, \hat{\alpha} : \kappa \vdash e s_0 : [\hat{\alpha}/\alpha] A_0 \gg C q \vdash \Delta \)

Subderivation

\[ \Gamma \vdash e s_0 : \forall \alpha : \kappa. A_0 p \gg C q \vdash \Delta \]

By Definition

\[ \Gamma, \hat{\alpha} : \kappa \rightarrow \Delta \]

By Lemma 51 (Typing Extension)

\[ \Delta \vdash \hat{\alpha} : \kappa \]

By Lemma 36 (Extension Weakening (Sorts))

\[ \Delta \rightarrow \Omega \]

By Lemma 58 (Bundled Substitution for Sorting)

\[ [\Omega] \Delta \vdash [\Omega] (e s_0) : \forall \alpha : \kappa. A_0 p \gg [\Omega] C q \]

By definition of substitution

\[ [\Omega] \Delta \vdash [\Omega] (e s_0) : [\Omega] (\forall \alpha : \kappa. A_0) p \gg [\Omega] C q \]

By definition of substitution
Proof of Theorem 8 (Soundness of Algorithmic Typing) thm:typing-soundness

• Case \[ e \text{ chk-I} \quad \Gamma \vdash P \text{ true } \vdash \Theta \quad \Theta \vdash e \not\iff [\Theta]\cdot A_0 \vdash p \vdash \Delta \]

\[
\Gamma \vdash P \text{ true } \vdash \Theta \quad \Delta \rightarrow \Omega \\
\Theta \rightarrow \Delta \quad \text{By Lemma 51 (Typing Extension)} \\
\Theta \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)} \\
[\Omega]\Theta \vdash [\Omega]P \text{ true} \quad \text{By Lemma 87 (Soundness of Checkprop)} \\
[\Omega]A_0 \vdash [\Omega]P \text{ true} \quad \text{By Lemma 56 (Confluence of Completeness)}
\]

Subderivation

\[
\Theta \vdash e \not\iff [\Theta]\cdot A_0 \vdash p \vdash \Delta \\
[\Omega]\Theta \vdash [\Omega]P \text{ true} \quad \text{By i.h.} \\
[\Omega]A_0 = [\Omega]A_0 \quad \text{By Lemma 29 (Substitution Monotonicity)} (iii) \\
[\Omega]A_0 \vdash [\Omega]A_0 \quad \text{By above equality}
\]

\[
\equiv [\Omega]A_0 \vdash [\Omega]P \text{ true} \quad \text{By def. of substitution}
\]

• Case \[ e \Gamma \vdash \text{ true } \vdash \Delta \\
\Gamma \vdash \text{ Nil } \vdash (\text{ Vec } t A) p \vdash \Delta \]

\[
\Gamma \vdash e \not\iff [\Theta]\cdot A_0 \vdash p \vdash \Delta \\
\Delta \rightarrow \Omega \quad \text{Given} \\
[\Omega]A_0 \vdash [\Omega]t = \text{ zero } \vdash [\Theta] \quad \text{By Lemma 87 (Soundness of Checkprop)} \\
[\Omega]A_0 \vdash [\Theta]t = \text{ zero } \quad \text{By def. of substitution} \\
\equiv [\Omega]A_0 \vdash (\text{ Vec } [\Theta]t [\Theta]A) p \quad \text{By DeclNil}
\]

• Case \[ \Gamma, \rightarrow_{\Delta}, \rightarrow : N \vdash t = \text{ succ } (\rightarrow) \text{ true } \vdash \Gamma' \]

\[
\Gamma' \vdash e_1 \not\iff \Gamma' A_0 \vdash \Theta \\
\Theta' \vdash e_2 \not\iff [\Theta'] (\text{ Vec } \rightarrow A_0 ) \vdash \Delta, \rightarrow_{\Delta'}, \Delta' \\
\Gamma \vdash e_1 : e_2 \not\iff (\text{ Vec } t A_0 ) p \vdash \Delta \quad \text{Cons}
\]

\[
\Gamma' \vdash e_1 \not\iff [\Gamma']A_0 \vdash \Theta \\
\Theta' \vdash e_2 \not\iff [\Theta'] (\text{ Vec } \rightarrow A_0 ) \vdash \Delta, \rightarrow_{\Delta'}, \Delta' \\
\Gamma \vdash e_1 : e_2 \not\iff (\text{ Vec } t A_0 ) p \vdash \Delta \quad \text{Cons}
\]

\[
\Gamma' \vdash e_1 \not\iff [\Gamma']A_0 \vdash \Theta \\
\Theta' \vdash e_2 \not\iff [\Theta'] (\text{ Vec } \rightarrow A_0 ) \vdash \Delta, \rightarrow_{\Delta'}, \Delta' \\
\Gamma \vdash e_1 : e_2 \not\iff (\text{ Vec } t A_0 ) p \vdash \Delta \quad \text{Cons}
\]

\[
\Gamma' \vdash e_1 \not\iff [\Gamma']A_0 \vdash \Theta \\
\Theta' \vdash e_2 \not\iff [\Theta'] (\text{ Vec } \rightarrow A_0 ) \vdash \Delta, \rightarrow_{\Delta'}, \Delta' \\
\Gamma \vdash e_1 : e_2 \not\iff (\text{ Vec } t A_0 ) p \vdash \Delta \quad \text{Cons}
\]

\[
\Gamma' \vdash e_1 \not\iff [\Gamma']A_0 \vdash \Theta \\
\Theta' \vdash e_2 \not\iff [\Theta'] (\text{ Vec } \rightarrow A_0 ) \vdash \Delta, \rightarrow_{\Delta'}, \Delta' \\
\Gamma \vdash e_1 : e_2 \not\iff (\text{ Vec } t A_0 ) p \vdash \Delta \quad \text{Cons}
\]
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \Gamma, \triangledown, \Delta : \forall \alpha : \mathbb{N} \vdash t = \text{succ}(\alpha) \text{ true} \rightarrow \Gamma' \]

\[
\begin{align*}
\Delta & \rightarrow \Omega \\
\Gamma' & \rightarrow \Theta \\
\Theta & \rightarrow \Delta, \triangledown, \Delta'
\end{align*}
\]

\[ \Delta, \triangledown, \Delta' \rightarrow \Omega' \]

\[ [\Omega'](\Delta, \triangledown, \Delta') \vdash [\Omega'](t = \text{succ}(\alpha)) \text{ true} \]

\[ [\Omega'](\Delta, \triangledown, \Delta') \vdash [\Omega](t = \text{succ}(\alpha)) \text{ true} \]

\[ [\Omega] \Delta \vdash [\Omega](t = \text{succ}(\alpha)) \text{ true} \]

1. \[ \Gamma' \vdash e_1 \iff [\Gamma']A_0 \rightarrow \Theta \]

2. \[ \Theta \vdash e_2 \iff [\Theta](\text{Vec } \alpha A_0) \rightarrow \Delta, \triangledown, \Delta' \]

3. \[ [\Omega] \Delta \vdash [\Omega]e_2 \iff [\Omega](\text{Vec } \alpha A_0) \rightarrow \]

By i.h.

By def. of substitution

By def. of substitution

By DeclCons (premises: 1, 2, 3)
Proof of Theorem 8 (Soundness of Algorithmic Typing)

Case \( v \) \( \text{chk-I} \)

\[
\begin{array}{c}
\Gamma, \rho \vdash P \Rightarrow (\Theta^+) \\
\Theta^+ \vdash v \iff [\Theta^+] A_0 ! \vdash \Delta, \rho, \Delta'
\end{array}
\]

\[
\Gamma \vdash \Delta ! \ 	ext{type}
\]

\[
\text{FEV}([\Gamma] A) = \emptyset
\]

\[
\text{FEV}([\Gamma] P) = \emptyset
\]

\[
\Gamma, \rho \vdash P \Rightarrow \Theta^+
\]

\[
\Gamma, \rho \vdash \sigma \triangleq t : \kappa \Rightarrow \Theta^+
\]

\[
\text{FEV}([\Gamma] \sigma) \cup \text{FEV}([\Gamma] t) = \emptyset
\]

\[
\Theta^+ = (\Gamma, \rho, \Theta)
\]

\[
[\Omega', \Theta] t' = [\Theta][\Gamma, \rho] t'
\]

\[
\theta = \text{mgu}(\sigma, t)
\]

\[
\Delta 
\rightarrow \Omega
\]

\[
\Theta^+ 
\rightarrow \Delta, \rho, \Delta'
\]

\[
\Gamma, \rho, \Theta 
\rightarrow \Delta, \rho, \Delta'
\]

\[
\text{By repeated } \rightarrow\text{Eqn}
\]

\[
\text{By Lemma 33 (Extension Transitivity)}
\]

\[
[\Omega', \Theta] B = [\Theta][\Gamma, \rho] B
\]

\[
\text{By Lemma 93 (Substitution Upgrade) (i)}
\]

\[
\text{For all } \Omega' \text{ extending } (\Gamma, \rho) \text{ and } t' \text{ s.t. } \Omega' \vdash t' : \kappa'
\]

\[
\text{By i.h.}
\]

\[
\Gamma, \rho, \Theta 
\rightarrow \Omega, \rho, \Delta'
\]

\[
\text{By Lemma 33 (Extension Transitivity)}
\]

\[
\Gamma 
\rightarrow \Omega
\]

\[
[\Omega^+][\Theta^+] A_0 = [\Omega^+] A_0
\]

\[
= [\Theta][\Omega^+] A_0
\]

\[
= [\Theta][\Omega] A_0
\]

\[
\text{Above, with } (\Omega, \rho) \text{ as } \Omega' \text{ and } A_0 \text{ as } B
\]

\[
\text{By def. of substitution}
\]

\[
[\Omega, \rho, \Theta][\Delta, \rho, \Delta'] = [\Theta][\Omega] \Delta
\]

\[
\text{By Lemma 93 (Substitution Upgrade) (iii)}
\]

\[
[\Theta][\Omega] A \vdash [\Theta][\Omega] \Delta
\]

\[
\text{By above equalities}
\]

\[
[\Omega^+][\Delta, \rho, \Delta'] / (\sigma = t) \vdash [\Theta] v \iff [\Omega] A_0 !
\]

\[
[\Omega^+][\Delta, \rho, \Delta'] / (\sigma = t) \vdash [\Theta] v \iff [\Omega] A_0 !
\]

\[
[\Omega] A \vdash [\Theta] v \iff [\Omega] \Delta
\]

\[
[\Omega] \Delta / (\sigma = t) \vdash [\Theta] v \iff [\Omega] A_0 !
\]

\[
[\Omega] A \vdash [\Theta] v \iff (\sigma = t) \cap [\Omega] A_0 !
\]

\[
[\Omega] \Delta / [\Theta] v \iff (\sigma = t) \cap [\Omega] A_0 !
\]

\[
\text{By Decl-I}
\]

\[
\text{By FEV condition above}
\]

Case \( v \) \( \text{chk-I} \)

\[
\Gamma, \rho \vdash P \Rightarrow P
\]

\[
\Gamma \vdash v \iff P \supset A_0 ! \vdash \Gamma
\]

\[
\Gamma, \rho \vdash P \Rightarrow P
\]

\[
\Gamma, \rho \vdash \sigma \triangleq t : \kappa \Rightarrow P
\]

\[
\text{By inversion}
\]

\[
\text{FEV}([\Gamma] \sigma) \cup \text{FEV}([\Gamma] t) = \emptyset
\]

\[
\text{As in (i) case (above)}
\]

\[
\text{By Lemma 88 (Soundness of Equality Elimination)}
\]
Proof of Theorem 8 (Soundness of Algorithmic Typing) \( \text{thm:typing-soundness} \)

\[ [\Omega] \Delta \vdash (\sigma = t) \iff [\Omega] A \Theta \]
\[ [\Omega] \Delta \vdash [\Omega] v \iff [\sigma = t] \vdash [\Omega] A \Theta \]
\[ [\Omega] \Delta \vdash [\Omega] v \iff ([\Delta] = t) \vdash [\Omega] A \Theta \]

\[ [\Omega] \Delta \vdash [\Omega] v \iff [\Delta] (P \vdash A) \Theta \]

Let \( \Omega' = \Omega \).
\[ \Omega \rightarrow \Omega' \]
\[ \Delta \rightarrow \Omega' \]

By definition of subst.

\[ \Theta \vdash e \equiv: \Theta \vdash e \equiv: \Theta \vdash e \equiv: \Theta \]

By Lemma 32 (Extension Reflexivity)

Given

• Case \( \Gamma \vdash P \text{ true } \rightarrow \Theta \)

\[ \Theta \vdash e_0 : [\Theta] A \Theta \vdash C \iff \Delta \]

\[ \Gamma \vdash e_0 : P \vdash A \Theta \vdash C \iff \Delta \]

Subderivation

\[ \Theta \rightarrow \Delta \]
\[ \Delta \rightarrow \Omega \]
\[ \Theta \rightarrow \Omega \]

By Lemma 51 (Typing Extension)

Given

By Lemma 33 (Extension Transitivity)

\[ [\Omega] \Theta \vdash [\Omega] A \Theta \vdash C \iff \Delta \]

By i.h.

\[ [\Delta] \Theta\vdash C \iff \Theta \]

By Lemma 29 (Substitution Monotonicity) (iii)

By above equality

\[ \Gamma \vdash P \text{ true } \rightarrow \Theta \]

\[ [\Omega] \Theta \vdash [\Omega] P \text{ true } \]
\[ [\Omega] \Theta = [\Omega] \Delta \]
\[ [\Omega] \Delta \vdash [\Omega] P \text{ true } \]

\[ [\Omega] \Delta \vdash [\Omega] A \Theta \vdash C \iff \Delta \]

By Decl \& Spine

By i.h.

By Lemma 95 (Completeness of Checkprop)

By Lemma 56 (Confluence of Completeness)

By above equality

\[ \Gamma \vdash e_0 \equiv: \Gamma \vdash e_0 \equiv: \Gamma \vdash e_0 \equiv: \Gamma \]

By def. of subst.

• Case \( \Gamma, x : A_1 P \vdash e_0 \equiv: \Gamma, x : A_1 P, \Theta \)

\[ \Gamma \vdash \lambda x. e_0 \equiv: \Gamma, x : A_1 P \iff \Delta' \]

\[ \Delta \rightarrow \Omega \]
\[ \Delta, x : A_1 P \rightarrow \Omega, x : [\Delta] A_1 P \]
\[ \Gamma, x : A_1 P \rightarrow \Delta, x : A_1 P, \Theta \]
\[ \Theta \text{ soft} \]

By Var

By Lemma 51 (Typing Extension)

By Lemma 22 (Extension Inversion) (v)

(with \( \Gamma_K = \cdot \), which is soft)

By Lemma 25 (Filling Completes)

\[ \Gamma, x : A_1 P \vdash e_0 \equiv: \Gamma, x : A_1 P \iff \Delta' \]

Subderivation

\[ [\Omega' \Delta'] \equiv: [\Delta'] e_0 \equiv: [\Omega'] A_2 P \]
\[ [\Omega'] A_2 = [\Omega] A_2 \]
\[ [\Omega'] \Delta' \equiv: [\Omega] e_0 \equiv: [\Omega] A_2 P \]

By i.h.

By Lemma 17 (Substitution Stability)

By above equality

\[ \Delta, x : A_1 P, \Theta \rightarrow \Omega, x : [\Delta] A_1 P, \Theta \iff \Delta' \]

Above

\[ \Theta \text{ soft} \]
\[ [\Delta'] \Delta' = ([\Delta], x : [\Delta] A_1 P) \]
\[ [\Omega] A_2, x : [\Omega] A_1 P \vdash e_0 \equiv: [\Omega] A_2 P \]

By above equality

By Lemma 53 (Softness Goes Away)
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \Gamma \vdash \lambda x. \Delta \] \[ \Delta \vdash \lambda x. \Delta \] (By \text{Decl--I})

\[ \Delta \vdash \lambda x. \Delta \] (By definition of substitution)

- **Case** \( \nu \text{chk-I} \)
  \[
  \Gamma, x : A, p \vdash \nu \leftarrow A \vdash \Delta, x : A, p, \Theta
  \]
  \[ \Gamma \vdash \text{Rec} \ x, \nu \leftarrow A \vdash \Delta \] (Rec)

Similar to the \( \text{Decl} \) case, applying \text{DeclRec} instead of \text{Decl--I}

- **Case**
  \[
  \Gamma[\Delta_1 : \ast, \Delta_2 : \ast, \ast : \ast = \Delta_1 \rightarrow \Delta_2], x : \Delta_1, j \vdash e_0 \leftarrow \Delta_2 \vdash \Delta, x : \Delta_1, j, \Theta
  \]
  \[ \Gamma[\Delta_1 : \ast, \Delta_2 : \ast, \ast : \ast = \Delta_1 \rightarrow \Delta_2] \vdash \lambda x. e_0 \leftarrow \Delta \] (\text{Subderivation})

  \[ \Delta \vdash \Theta \] (soft)

  \[ \Gamma[\Delta_1 : \ast, \Delta_2 : \ast, \ast : \ast = \Delta_1 \rightarrow \Delta_2] \vdash \Delta 
  \]

  \[ \Delta \vdash \Omega \]

  \[ \Delta, x : \Delta_1, j \vdash \Omega, x : [\Delta] \]

  \[ \Delta, x : \Delta_1, j, \Theta \]

  \[ \Delta' \]

  \[ \Omega' \]

  \[ \Gamma[\Delta_1 : \ast, \Delta_2 : \ast, \ast : \ast = \Delta_1 \rightarrow \Delta_2], x : \Delta_1, j \vdash e_0 \leftarrow \Delta_2 \vdash \Delta, x : \Delta_1, j, \Theta \]

  Above and Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash [\Omega] \varepsilon = [\Omega] \] \[ \Delta \vdash \lambda x. \Delta \]

\[ \Delta = [\Omega] \] \[ \Delta = [\Omega] \] \[ \Delta = [\Omega] \]

By i.h.

By Lemma 17 (Substitution Stability)

By definition of substitution

By definition of context substitution

By above equalities

\[ [\Omega] \Delta \vdash [\Omega] \varepsilon = [\Omega] \] \[ \Delta \vdash [\Omega] \] \[ \Delta = [\Omega] \] \[ \Delta = [\Omega] \] \[ \Delta = [\Omega] \]

By Lemma 29 (Substitution Monotonicity) (i)

By definition of substitution

By definition of substitution

By Lemma 29 (Substitution Monotonicity) (i)

By above equality

- **Case**
  \[ \Gamma \vdash e_0 \Rightarrow A \ q \vdash \Theta \]
  \[ \Theta \vdash s_0 : A \ q \gg C \ [p] \vdash \Delta \]

  \[ \Gamma \vdash e_0 \ s_0 : C \ p \vdash \Delta \] (\text{\text{-E})}

Proof of Theorem 8 (Soundness of Algorithmic Typing) thm:typing-soundness 128
Proof of Theorem 8 (Soundness of Algorithmic Typing) thm:typing-soundness

\[ \Gamma \vdash e_0 \Rightarrow A \triangleright \Theta \]

Subderivation

\[ \Theta \vdash s_0 : A \triangleright C \triangleright p \vdash \Delta \]

Subderivation

\[ \Gamma \rightarrow \Theta \text{ and } \Theta \rightarrow \Delta \]

By Lemma 51 (Typing Extension)

\[ \Delta \rightarrow \Omega \]

Given

\[ \Theta \rightarrow \Omega \]

By Lemma 33 (Extension Transitivity)

\[ \Gamma \rightarrow \Omega \]

By Lemma 33 (Extension Transitivity)

\[ [\Omega] \Gamma = [\Omega] \Theta = [\Omega] \Delta \]

By Lemma 56 (Confluence of Completeness)

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \Rightarrow [\Omega] A \triangleright \]

By i.h.

\[ [\Omega] \Delta \vdash [\Omega] e_0 \Rightarrow [\Omega] A \triangleright \]

By above equality

\[ [\Omega] \Theta \vdash [\Omega] s_0 : [\Omega] A \triangleright \Delta \]

By i.h.

\[ [\Omega] \Delta \vdash [\Omega] \left( e_0 s_0 \right) \Rightarrow [\Omega] C \triangleright \]

By rule Decl-E

\[ [\Omega] \Delta \vdash [\Omega] \left( e_0 s_0 \right) \Rightarrow [\Omega] C \triangleright \]

\[ \Gamma \vdash s : A \Rightarrow C \triangleright \Delta \]

FEV(C) = \emptyset

\[ \Gamma \vdash s : A \Rightarrow C \triangleright \Delta \]

SpineRecover

\[ \Gamma \vdash s : A \Rightarrow C \triangleright \Delta \]

Subderivation

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A \Rightarrow [\Omega] C \triangleright \]

By i.h.

We show the quantified premise of DeclSpineRecover, namely,

for all \( C' \).

if \( [\Omega] \Theta \vdash s : [\Omega] A \Rightarrow C' \triangleright \) then \( C' = [\Omega] C \).

Suppose we have \( C' \) such that \( [\Omega] \Gamma \vdash s : [\Omega] A \Rightarrow C' \triangleright \). To apply DeclSpineRecover, we need to show \( C' = [\Omega] C \).

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A \Rightarrow C' \triangleright \]

Assumption

\[ \Omega_{\text{canon}} \rightarrow \Omega \]

By Lemma 59 (Canonical Completion)

\[ \text{dom}(\Omega_{\text{canon}}) = \text{dom}(\Gamma) \]

""

\[ \Gamma \rightarrow \Omega_{\text{canon}} \]

""

\[ [\Omega] \Gamma = [\Omega_{\text{canon}}] \Gamma \]

By Lemma 57 (Multiple Confluence)

\[ [\Omega] A = [\Omega_{\text{canon}}] A \]

By Lemma 55 (Completing Completeness) (ii)

\[ [\Omega_{\text{canon}}] \Gamma \vdash [\Omega] s : [\Omega_{\text{canon}}] A \Rightarrow C' \triangleright \]

By above equalities

\[ \Gamma \vdash s : [\Gamma] A \Rightarrow C'' \triangleright \Delta'' \]

By Theorem 11

\[ \Omega_{\text{canon}} \rightarrow \Omega'' \]

""

\[ \Delta'' \rightarrow \Omega'' \]

""

\[ C'' = [\Omega''] C'' \]

""

\[ C' = [\Omega''] C' \]

Above

\[ = [\Omega''] C \]

By above equality

\[ = [\Omega_{\text{canon}}] C \]

By Lemma 55 (Completing Completeness) (ii)

\[ = [\Omega] C \]

By Lemma 55 (Completing Completeness) (ii)

We have thus shown the above “for all \( C' \ldots \)" statement.

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A \Rightarrow [\Omega] C \triangleright \]

By DeclSpineRecover

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A \Rightarrow [\Omega] C \triangleright \]

\[ \Gamma \vdash s : A \Rightarrow C \triangleright \Delta \]

(FEV(C) \neq \emptyset)

\[ \Gamma \vdash s : A \Rightarrow C \triangleright \Delta \]

SpinePass
Proof of Theorem 8 (Soundness of Algorithmic Typing)

• Case

\[ \Gamma \vdash \vdash : A \ p \ \triangleright \ A \ p \ \triangleright \Delta \]

Subderivation

\[ [\Omega] \Gamma \vdash [\Omega] \vdash : [\Omega] A \ p \ \triangleright \ [\Omega] C \ q \quad \text{By i.h.} \]

\[ [\Omega] \Gamma \vdash [\Omega] \vdash : [\Omega] A \ p \ \triangleright \ [\Omega] C \ [q] \quad \text{By DeclSpinePass} \]

• Case

\[ \Gamma \vdash \vdash : A \ p \ \triangleright \ A \ p \ \triangleright \Gamma \]

EmptySpine

\[ [\Omega] \Gamma \vdash [\Omega] A \ p \ \triangleright \ [\Omega] A \ p \quad \text{By DeclEmptySpine} \]

• Case

\[ \Gamma \vdash e_0 \leftarrow A_1 \ p \vdash \Theta \quad \Theta \vdash s_0 : [\Theta] A_2 \ p \ \triangleright \ [\Theta] C \ q \ \triangleright \Delta \]

\[ \Gamma \vdash e_0, s_0 : A_1 \rightarrow A_2 \ p \ \triangleright \ [\Omega] C \ q \ \triangleright \Delta \quad \text{→Spine} \]

\[ \Delta \rightarrow \Omega \]

Given

\[ \Theta \rightarrow \Delta \quad \text{By Lemma 51 (Typing Extension)} \]

\[ \Theta \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)} \]

\[ \Gamma \vdash e_0 \leftarrow A_1 \ p \vdash \Theta \quad \text{Subderivation} \]

\[ [\Omega] \Theta = [\Omega] \Delta \quad \text{By Lemma 56 (Confluence of Completeness)} \]

\[ [\Omega] \Delta + [\Omega] e_0 \leftarrow [\Omega] A_1 \ p \quad \text{By above equality} \]

\[ \Theta \vdash s_0 : [\Theta] A_2 \ p \ \triangleright \ [\Theta] C \ q \ \triangleright \Delta \]

Subderivation

\[ [\Omega] \Delta \vdash [\Omega] s_0 : [\Omega] A_2 \ p \ \triangleright \ [\Omega] C \ q \quad \text{By above equality} \]

\[ [\Omega] \Delta \vdash [\Omega] e_0 s_0 : [\Omega] (A_1 \rightarrow A_2) \ p \ \triangleright \ [\Omega] C \ q \quad \text{By Decl→Spine} \]

• Case

\[ \Gamma \vdash e_0 \leftarrow A_k \ p \vdash \Delta \]

\[ \Gamma \vdash \text{inj}_k e_0 \leftarrow A_1 + A_2 \ p \vdash \Delta + \text{inj}_k \]

\[ [\Omega] \Delta \vdash [\Omega] e_0 \leftarrow [\Omega] A_k \ p \quad \text{By above equality} \]

\[ [\Omega] \Delta \vdash \text{inj}_k [\Omega] e_0 \leftarrow ([\Omega] A_1) + ([\Omega] A_2) \ p \quad \text{By Decl+inj}_k \]

• Case

\[ [\hat{\alpha}_1 : * \vdash \hat{\alpha}_2 : * \vdash : \hat{\alpha}_1 \hat{\alpha}_2] \]

\[ [\Gamma] [\hat{\alpha} : * \vdash \text{inj}_k e_0 \leftarrow \hat{\alpha} \vdash \hat{\alpha} \ p \vdash \Delta] + \text{inj}_k \]

\[ [\hat{\alpha}_1 : * \vdash \hat{\alpha}_2 : * \vdash : \hat{\alpha}_1 + \hat{\alpha}_2] \]

\[ \Gamma \vdash e_0 \leftarrow \hat{\alpha}_1 + \hat{\alpha}_2 \ p \vdash \hat{\alpha}_k \ p \vdash \Delta \]

Subderivation

\[ [\Omega] \Delta \vdash [\Omega] e_0 \leftarrow [\Omega] \hat{\alpha}_k \ p \quad \text{By i.h.} \]

\[ [\Omega] \Delta \vdash \text{inj}_k [\Omega] e_0 \leftarrow ([\Omega] \hat{\alpha}_1) + ([\Omega] \hat{\alpha}_2) \ p \quad \text{By Decl+inj}_k \]

\[ [\Omega] \Delta \vdash [\Omega] (\text{inj}_k e_0) \leftarrow [\Omega] (\hat{\alpha}_1 + \hat{\alpha}_2) \ p \quad \text{By def. of substitution} \]

• Case

\[ \Gamma \vdash e_1 \leftarrow A_1 \ p \vdash \Theta \quad \Theta \vdash e_2 : [\Theta] A_2 \ p \vdash \Delta \]

\[ \Gamma \vdash e_1, e_2 \leftarrow A_1 \times A_2 \ p \vdash \Delta \quad \text{x1} \]
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \varnothing \vdash e_2 \iff [\varnothing]A_2 \vdash \Delta \]

\[ \Theta \rightarrow \Delta \]

\[ \Theta \rightarrow \Omega \]

\[ \Gamma \vdash e_1 \iff A_1 \vdash \Theta \]

\[ [\Omega]\varnothing \vdash [\Omega]e_1 \iff [\Omega]A_1 \vdash \varnothing \]

\[ [\Omega]\Delta \vdash [\Omega]e_1 \iff [\Omega]A_1 \vdash \varnothing \]

Subderivation

By Lemma 51 (Typing Extension)

By Lemma 33 (Extension Transitivity)

By i.h.

By Confluence of Completeness

\[ [\Omega]\Delta \vdash [\Omega]e_2 \iff [\Omega]A_2 \vdash \varnothing \]

Subderivation

By i.h.

By Lemma 56 (Confluence of Completeness)

Given

By Substitution Monotonicity

\[ [\Omega]\Delta \vdash ([\Omega]e_1, [\Omega]e_2) \iff ([\Omega]A_1 \times [\Omega]A_2) \vdash \varnothing \]

By Decl\times I

By def. of substitution

\[ [\Omega]\Delta \vdash [\Omega](e_1, e_2) \iff [\Omega](A_1 \times A_2) \vdash \varnothing \]

Subderivation

By i.h.

By Lemma 56 (Confluence of Completeness)

By above equality

\[ [\Omega]\Delta \vdash [\Omega](e_1, e_2) \iff [\Omega](\alpha) \vdash \varnothing \]

Similar to the \( \alpha \) case (above)

By above equality

\[ \Gamma \vdash e_2 \iff [\Theta]A_2 \vdash \Delta \]

\[ \Theta \rightarrow \Delta \]

\[ \Theta \rightarrow \Omega \]

\[ \Gamma \vdash e_1 \iff A_1 \vdash \Theta \]

\[ [\Omega]\Delta \vdash (\alpha_1 \times \alpha_2) \vdash e_1 \iff (\alpha_1 \times \alpha_2) \vdash e_1 \]

Subderivation

By Lemma 51 (Typing Extension)

By Lemma 33 (Extension Transitivity)

By i.h.

By Confluence of Completeness

\[ [\Omega]\Delta \vdash [\Omega]e_1 \iff [\Omega]\alpha_1 \vdash \varnothing \]

Subderivation

By i.h.

By Substitution Monotonicity

By above equality

\[ [\Omega]\Delta \vdash [\Omega]e_2 \iff [\Omega]\alpha_2 \vdash \varnothing \]

Subderivation

By above equality

\[ [\Omega]\Delta \vdash ([\Omega]e_1, [\Omega]e_2) \iff ([\Omega]\alpha_1) \times [\Omega]\alpha_2 \vdash \varnothing \]

Similar to the \( \alpha \) case (above)

By above equality

\[ \Gamma \vdash (\alpha_2 : \star, \alpha_1 : \star, \alpha : \star = \alpha_1 \times \alpha_2) \vdash e_1 \iff (\alpha_1 \times \alpha_2) \vdash e_1 \]

\[ \Gamma \vdash (\alpha_2 : \star, \alpha_1 : \star, \alpha : \star = \alpha_1 \times \alpha_2) \vdash e_0 s_0 : (\alpha_1 \vdash \alpha_2) \vdash C \vdash \Delta \]

\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff C \vdash \Delta \]

Subderivation

By i.h.

By above equality

By above equality

\[ [\Omega]\Delta \vdash (\alpha_1 \vdash \alpha_2) \vdash \alpha \vdash \varnothing \]

\[ [\Omega]\Delta \vdash (\alpha_1 \vdash \alpha_2) \vdash C \vdash \Delta \]

\[ [\Omega]\Delta \vdash (\alpha_0 s_0) : ([\Omega](\alpha_0) \vdash \alpha \vdash \varnothing) \vdash [\Omega]C \vdash \Delta \]

\[ [\Omega]\Delta \vdash (\alpha_0 s_0) : ([\Omega](\alpha_0) \vdash \alpha \vdash \varnothing) \vdash (\alpha_1 \vdash \alpha_2) \vdash C \vdash \Delta \]

Subderivation

By i.h.

Similar to the \( \alpha \) case

By above equality

\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff C \vdash \Delta \]

\[ \Gamma \vdash e_0 \Rightarrow B ! \vdash \Theta \]

\[ \Theta \vdash \Pi : [\Theta]B \iff [\Theta]C \vdash \Delta \]

\[ \Delta \vdash \Pi \text{ covers } [\Delta]B \]

\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff C \vdash \Delta \]
\( \Gamma \vdash e_0 \Rightarrow B \ ! \ + \Theta \)  
\( \Theta \rightarrow \Delta \)  
Subderivation  
By Lemma 51 (Typing Extension)
\( \Theta \rightarrow \Omega \)  
Subderivation  
By Lemma 33 (Extension Transitivity)
\([\Omega] \Theta \vdash [\Omega] e_0 \Rightarrow [\Omega] B \ ! \)  
By i.h.
\([\Omega] \Delta \vdash [\Omega] e_0 \Rightarrow [\Omega] B \ ! \)  
By Lemma 56 (Confluence of Completeness)

\( \Theta \vdash \Pi \iff [\Theta] C \ p \rightarrow \Delta \)  
Subderivation  
By Lemma 63 (Well-Formed Outputs of Typing) (Synthesis)
\( \Gamma \vdash C \ p \rightarrow \Theta \)  
Given  
By Lemma 51 (Typing Extension)
\( \Theta \vdash C \ p \rightarrow \Delta \)  
By Lemma 41 (Extension Transitivity)  
By Lemma 40 (Right-Hand Subst. for Principal Typing)
\([\Omega] \Delta \vdash [\Omega] \Pi \iff [\Omega] B \iff [\Omega] C \ p \)  
By above equalities
\( \Delta \vdash \Pi \text{ covers } [\Delta] B \)  
Subderivation  
By idempotence of substitution  
By Lemma 63 (Well-Formed Outputs of Typing)
\( \Theta \vdash B \ ! \ + \type \)  
By Lemma 41 (Extension Weakening for Principal Typing)  
By Lemma 40 (Right-Hand Subst. for Principal Typing)
\( \Delta \vdash [\Delta] B \ ! \ + \type \)  
By Lemma 40 (Right-Hand Subst. for Principal Typing)
\([\Omega] \Delta \vdash [\Omega] \Pi \iff [\Omega] B \iff [\Omega] C \ p \)  
By above equalities

**Part (v):**

- **Case** MatchEmpty  Apply rule DeclMatchEmpty
- **Case**  
  \( \Gamma \vdash e \iff C \ p \rightarrow \Delta \)  
  \( \Gamma \vdash \{ \Rightarrow e \} \equiv \cdot \iff C \ p \rightarrow \Delta \)  
  MatchBase
  
  Apply the i.h. and DeclMatchBase

- **Case** MatchUnit  Apply the i.h. and DeclMatchUnit

- **Case**  
  \( \Gamma \vdash \pi : \vec{A} \iff C \ p \rightarrow \Theta \)  
  \( \Theta \vdash \Pi' : \vec{A} \iff C \ p \rightarrow \Delta \)  
  \( \Gamma \vdash \pi \vdash \Pi' : \vec{A} \iff C \ p \rightarrow \Delta \)  
  MatchSeq
  
  Apply the i.h. to each premise, using lemmas for well-formedness under \( \Theta \); then apply DeclMatchSeq

- **Cases**  
  Match\-\|  MatchWild  MatchNil  MatchCons
  
  Apply the i.h. and the corresponding declarative match rule.

- **Cases**  
  Match\-\*  Match\+\k
  
  We have \( \Gamma \vdash \vec{A} \ ! \ + \types \), so the first type in \( \vec{A} \) has no free existential variables.
  
  Apply the i.h. and the corresponding declarative match rule.
• Case

\[ \Gamma \vdash z : \tau \Rightarrow e : \phi \quad \frac{\text{MatchNeg}}{\Gamma \vdash z : \tau \Rightarrow e : \phi} \]

Construct \( \Omega' \) and show \( \Delta, z : \tau \Rightarrow \Delta' \) as in the \( \to \) case.

Use the i.h., then apply rule \( \text{DeclMatchNeg} \).

Part (vi):

• Case

\[ \frac{\Gamma / \sigma \vdash \tau : \kappa \to \bot}{\Gamma / \sigma \vdash \tau : \kappa \to \bot} \quad \frac{\text{Match\bot}}{\Gamma / \sigma \vdash \tau : \kappa \to \bot} \]

Subderivation

\[ \frac{\Gamma (\sigma = \tau) = (\sigma = \tau)}{\text{Given}} \]

\[ (\sigma = \tau) = [\Gamma] (\sigma = \tau) \quad \frac{\text{By Lemma 29 (Substitution Monotonicity) (i)}}{\text{By above equality}} \]

Substitution

\[ \text{mgu}(\sigma, \tau) = \bot \quad \frac{(\sigma = \tau) = [\Gamma] (\sigma = \tau)}{\text{By above equality}} \]

Substitution

\[ \Theta = (\alpha_1 : t_1, \ldots, \alpha_n : t_n) \quad \frac{\text{By Lemma 88 (Soundness of Equality Elimination)}}{} \]

Substitution

\[ \theta = \text{mgu}(\sigma, \tau) \quad \frac{\text{By above equality}}{} \]

Substitution

\[ \begin{align*}
\omega &= [\Omega \Gamma] (\sigma = \tau) + [\Omega] (\sigma = \tau) :: [\Omega] A &= [\Omega] C p \quad \text{By DeclMatchUnify}
\end{align*} \]

\[ \begin{align*}
\text{Case } 
\delta / \sigma \vdash \tau : \kappa \to \tau' & \quad \frac{\Gamma / \sigma \vdash \tau : \kappa \to \tau' & \Gamma / \sigma \vdash \tau : \kappa \to \tau' \Rightarrow e : \phi \quad \frac{\text{MatchUnify}}{\Gamma / \sigma \vdash \tau : \kappa \to \tau' \Rightarrow e : \phi} \]

Subderivation

\[ \frac{\Gamma (\sigma = \tau) = [\Gamma] (\sigma = \tau)}{\text{Given}} \]

\[ (\sigma = \tau) = [\Gamma] (\sigma = \tau) 
\]
Proof. By induction on \( \tau \).

We have \( [\Omega] \Gamma \vdash [\Omega] \hat{\alpha} \leq^* [\Omega] \hat{A} \). We now case-analyze the shape of \( \tau \).

- **Case** \( \tau = \hat{\beta} \):

  \( \hat{\alpha} \notin \text{FV}(\hat{\beta}) \) \hspace{1cm} \text{Given}
  \( \hat{\alpha} \neq \hat{\beta} \) \hspace{1cm} \text{From definition of FV(\(\_\) )}
  \( \hat{\beta} \in \text{unsolved}(\Gamma) \) \hspace{1cm} \text{From } [\Gamma] \hat{\beta} = \hat{\beta}

  Let \( \Omega' = \Omega \).

  \( \Rightarrow \quad \Omega \rightarrow \Omega' \) \hspace{1cm} \text{By Lemma 32 [Extension Reflexivity]}

Now consider whether \( \hat{\alpha} \) is declared to the left of \( \hat{\beta} \), or vice versa.

- **Case** \( \Gamma = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \):

  Let \( \Delta = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}] \).

  \( \Gamma \vdash \hat{\alpha} := \hat{\beta} : \kappa \rightarrow \Delta \) \hspace{1cm} \text{By InstReach}
  \( [\Omega] \hat{\alpha} = [\Omega] \hat{\beta} \) \hspace{1cm} \text{Given}
  \( \Gamma \rightarrow \Omega \) \hspace{1cm} \text{Given}

  \( \Rightarrow \quad \Delta \rightarrow \Omega \) \hspace{1cm} \text{By Lemma 27 [Parallel Extension Solution]}

  \( \Rightarrow \quad \text{dom}(\Delta) = \text{dom}(\Omega') \) \hspace{1cm} \text{dom}(\Delta) = \text{dom}(\Gamma) \text{ and } \text{dom}(\Omega') = \text{dom}(\Omega) \)

- **Case** \( \Gamma = \Gamma_0[\hat{\beta} : \kappa][\hat{\alpha} : \kappa] \):

  Similar, but using InstSolve instead of InstReach.

- **Case** \( \tau = \alpha \):

  We have \( [\Omega] \hat{\alpha} = \alpha \), so (since \( \Omega \) is well-formed), \( \alpha \) is declared to the left of \( \hat{\alpha} \) in \( \Omega \).

  We have \( \Gamma \rightarrow \Omega \).

  By Lemma 21 [Reverse Declaration Order Preservation], we know that \( \alpha \) is declared to the left of \( \hat{\alpha} \) in \( \Gamma \); that is, \( \Gamma = \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa] \).

  Let \( \Delta = \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa = \hat{\alpha}] \) and \( \Omega' = \Omega \).

  By InstSolve \( \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \alpha : \kappa \rightarrow \Delta \).

  By Lemma 27 [Parallel Extension Solution], \( \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa = \hat{\alpha}] \rightarrow \Omega \).

  We have \( \text{dom}(\Delta) = \text{dom}(\Gamma) \text{ and } \text{dom}(\Omega') = \text{dom}(\Omega) \); therefore, \( \text{dom}(\Delta) = \text{dom}(\Omega') \).

- **Case** \( \tau = 1 \):

  Similar to the \( \tau = \alpha \) case, but without having to reason about where \( \alpha \) is declared.

- **Case** \( \tau = \text{zero} \):

  Similar to the \( \tau = 1 \) case.

- **Case** \( \tau = \tau_1 \oplus \tau_2 \):
Proof of Lemma 90 (Completeness of Instantiation)

Given $([\Omega]_2)\hat{x} = ([\Omega](\tau_1 \oplus \tau_2))$

$\tau_1 \oplus \tau_2 = [\Gamma](\tau_1 \oplus \tau_2)$

By definition of substitution

By i.h., we have

By definition of substitution and congruence

Similarly

By i.h., there are

From definition of $FV(-)$

Similarly

By Lemma 55 (Completing Completeness) (i), we know that

By Lemma 55 (Completing Completeness) (iii), we know that

By Lemma 64 (Left Unsolvedness Preservation), we know that

Next, note that

Since $\hat{x} \notin FV(\tau)$, it follows that $[\Gamma_1]_2 \tau = [\Gamma]_2 \tau = \tau$.

Therefore $\hat{x}_1 \notin FV(\tau_1)$ and $\hat{x}_1 \notin FV(\tau_2)$.

By Lemma 55 (Completing Completeness) (i) and (iii), $[\Omega]_2 \Gamma_1 = [\Omega]_2 \Gamma$ and $[\Omega]_2 \hat{x}_1 = \tau_1$.

By i.h., there are $\Delta_2$ and $\Omega_2$ such that $\Gamma_1 \vdash \hat{x}_1 := \tau : \kappa \vdash \Delta_2$ and $\Delta_2 \vdash \Omega_2$ and $\Omega_1 \vdash \Omega_2$.

Next, note that $[\Delta_2][\Delta_2]_2 \tau = [\Delta_2]_2 \tau_2$.

By Lemma 64 (Left Unsolvedness Preservation), we know that $\hat{x}_2 \in \text{unsolved}(\Delta_2)$.

By Lemma 65 (Left Free Variable Preservation), we know that $\hat{x}_2 \notin FV([\Delta_2]_2 \tau_2)$.

By Lemma 33 (Extension Transitivity), $\Omega \vdash \Omega_2$.

We know $[\Omega_2]_2 \Delta_2 = [\Omega_2]_2 \Gamma$ because:

By Lemma 55 (Completing Completeness) (i), we know that $[\Omega_2]_2 \hat{x}_2 = [\Omega_2]_2 \hat{x}_2 = \tau_2$.

By Lemma 55 (Completing Completeness) (i), we know that $[\Omega_2]_2 \tau_2 = [\Omega_2]_2 \tau_2$.

Hence we know that $[\Omega_2]_2 \Delta_2 \vdash [\Omega_2]_2 \hat{x}_2 \leq [\Omega_2]_2 \tau_2$.

By i.h., we have $\Delta$ and $\Omega'$ such that $\Delta_2 \vdash \hat{x}_2 := [\Delta_2]_2 \tau_2 : \kappa \vdash \Delta$ and $\Omega_2 \vdash \Omega'$ and $\Delta \rightarrow \Omega'$.

By rule $\text{InstBin}$, $\Gamma \vdash \hat{x} := \tau : \kappa \vdash \Delta$.

By Lemma 33 (Extension Transitivity), $\Omega \rightarrow \Omega'$.
• **Case** $\tau = \text{succ}(\tau_0)$:
  Similar to the $\tau = \tau_1 \oplus \tau_2$ case, but simpler.

**Lemma 91** (Completeness of Checkeq).

Given $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash \sigma : \kappa$ and $\Gamma \vdash \tau : \kappa$ and $[\Omega] \sigma = [\Omega] \tau$ then $\Gamma \vdash [\Gamma] \sigma \equiv [\Gamma] \tau : \kappa \vdash \Delta$ where $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$.

**Proof.** By mutual induction on the sizes of $[\Gamma] \sigma$ and $[\Gamma] \tau$.

We distinguish cases of $[\Gamma] \sigma$ and $[\Gamma] \tau$.

• **Case** $[\Gamma] \sigma = [\Gamma] \tau = 1$:

  $\triangleleft$
  $\Gamma \vdash 1 \equiv 1 : * \vdash \Delta$
  
  By CheckeqUnit

  Let $\Omega' = \Omega$.

  $\Gamma \rightarrow \Omega$ Given

  $\Delta \rightarrow \Omega'$ $\Delta = \Gamma$ and $\Omega' = \Omega$

  $\text{dom}(\Gamma) = \text{dom}(\Omega)$ Given

  $\Omega \rightarrow \Omega'$ By Lemma 32 (Extension Reflexivity)

• **Case** $[\Gamma] \sigma = [\Gamma] t = \text{zero}$:

  Similar to the case for 1, applying CheckeqZero instead of CheckeqUnit

• **Case** $[\Gamma] \sigma = [\Gamma] t = \alpha$:

  Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit

• **Case** $[\Gamma] \sigma = \hat{\alpha}$ and $[\Gamma] t = \hat{\beta}$:

  - If $\hat{\alpha} = \hat{\beta}$: Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit
  - If $\hat{\alpha} \neq \hat{\beta}$:

    $\Gamma \rightarrow \Omega$

    $\hat{\alpha} \notin \text{FV}(\hat{\beta})$

    By definition of FV(-)

    $[\Omega] \sigma = [\Omega] t$

    $[\Omega] [\Gamma] \sigma = [\Omega] [\Gamma] t$

    By Lemma 29 (Substitution Monotonicity) (i) twice

    $[\Omega] \alpha = [\Omega] [\Gamma] t$

    $\text{dom}(\Gamma) = \text{dom}(\Omega)$

    Given

    $\Gamma \vdash \hat{\alpha} := [\Gamma] t : \kappa \vdash \Delta$

    By Lemma 90 (Completeness of Instantiation)

    $\Omega \rightarrow \Omega'$

    "$

    \Delta \rightarrow \Omega$

    "$

    \text{dom}(\Delta) = \text{dom}(\Omega')$

    "$

    \Gamma \vdash \hat{\alpha} \equiv [\Gamma] t : \kappa \vdash \Delta$

    By CheckeqInstL

• **Case** $[\Gamma] \sigma = \hat{\alpha}$ and $[\Gamma] t = 1$ or zero or $\alpha$:

  Similar to the previous case, except:

  $\hat{\alpha} \notin \text{FV}(\hat{1})$

  By definition of FV(-)

  and similarly for 1 and $\alpha$. 

---

Proof of **Lemma 91** (Completeness of Checkeq) 
1em:checkeq-completeness
Proof of Lemma 91 (Completeness of Checkeq)

- Case $[\Gamma]t = \hat{\alpha}$ and $[\Gamma] \sigma = 1$ or zero or $\alpha$: Symmetric to the previous case.

- Case $[\Gamma] \sigma = \hat{\alpha}$ and $[\Gamma] t = \text{succ}(\Gamma t_0)$:
  If $\hat{\alpha} \notin \text{FV}(\Gamma t_0)$, then $\hat{\alpha} \notin \text{FV}([\Gamma]t)$. Proceed as in the previous several cases.

  The other case, $\hat{\alpha} \in \text{FV}((\Gamma) t_0)$, is impossible:
  - We have $\hat{\alpha} \notin [\Gamma]t_0$.
  - Therefore $\hat{\alpha} \prec \text{succ}([\Gamma]t_0)$, that is, $\hat{\alpha} \prec [\Gamma]t$.
  - By a property of substitutions, $[\Omega]\hat{\alpha} \prec [\Omega][\Gamma]t$.
  - Since $\Gamma \rightarrow \Omega$, by Lemma 29 (Substitution Monotonicity) (i), $[\Omega] [\Gamma] t = [\Omega] t$, so $[\Omega] \hat{\alpha} \prec [\Omega]t$. But it is given that $[\Omega] \hat{\alpha} = [\Omega] t$, a contradiction.

- Case $[\Gamma] t = \hat{\alpha}$ and $[\Gamma] \sigma = \text{succ}(\Gamma \sigma_0)$: Symmetric to the previous case.

- Case $[\Gamma] \sigma = [\Gamma] \sigma_1 \oplus [\Gamma] \sigma_2$ and $[\Gamma] t = [\Gamma] t_1 \oplus [\Gamma] t_2$:
  - Given $\Gamma \rightarrow \Omega$,
  - $\Gamma \vdash [\Gamma] \sigma_1 \equiv [\Gamma] t_1 : * - \Theta$ By i.h.
  - $\Theta \rightarrow \Omega_0$ "
  - $\Omega \rightarrow \Omega_0$ "
  - $\text{dom(}\Theta\text{)} = \text{dom(}\Omega_0\text{)}$ "
  - $\Theta \vdash [\Theta][\Gamma] \sigma_2 \equiv [\Theta][\Gamma] t_2 : * - \Delta$ By i.h.
  - $\Delta \rightarrow \Omega'$ "
  - $\Omega_0 \rightarrow \Omega'$ "
  - $\text{dom(}\Delta\text{)} = \text{dom(}\Omega'\text{)}$ "
  - $\Omega \rightarrow \Omega'$ By Lemma 33 (Extension Transitivity)
  - $\Gamma \vdash [\Gamma] \sigma_1 \oplus [\Gamma] \sigma_2 \equiv [\Gamma] t_1 \oplus [\Gamma] t_2 : * - \Delta$ By CheckeqBin

- Case $[\Gamma] \sigma = \hat{\alpha}$ and $[\Gamma] t = t_1 \oplus t_2$: Similar to the $\hat{\alpha}/\text{succ}(\_)$ case, showing the impossibility of $\hat{\alpha} \in \text{FV}(\Gamma t_k)$ for $k = 1$ and $k = 2$.

- Case $[\Gamma] t = \hat{\alpha}$ and $[\Gamma] \sigma = \sigma_1 \oplus \sigma_2$: Symmetric to the previous case.

Lemma 92 (Completeness of Elimeq).
If $[\Gamma] \sigma = \sigma$ and $[\Gamma] t = t$ and $\Gamma \vdash \sigma : \kappa$ and $\Gamma \vdash t : \kappa$ and $\text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset$ then:

1. If $\text{mguy}(\sigma, t) = \emptyset$
   then $\Gamma / \sigma \equiv t : \kappa - (\Gamma, \Delta)$
   where $\Delta$ has the form $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
   and for all $u$ such that $\Gamma \vdash u : \kappa$, it is the case that $[\Gamma, \Delta] u = 0([\Gamma] u)$.

2. If $\text{mguy}(\sigma, t) = \bot$ (that is, no most general unifier exists) then $\Gamma / \sigma \equiv t : \kappa - \bot$.

Proof: By induction on the structure of $[\Gamma] \sigma$ and $[\Gamma] t$.

- Case $[\Omega] \sigma = t = \emptyset$:
  By properties of unification,
  - $\text{mguy}(\text{zero}, \text{zero}) = \cdot$
  - $\Gamma / \text{zero} \equiv \text{zero} : \text{N} - \Gamma$
  - $\Gamma / \text{zero} \equiv \text{zero} : \text{N} - \Gamma, \Delta$
  - $\text{Suppose } \Gamma \vdash u : \kappa'$.
  - $[\Gamma, \Delta] u = [\Gamma] u$
  - where $\Delta = \cdot$
  - $\theta$ is the identity

- Case $\sigma = \text{succ}(\sigma')$ and $t = \text{succ}(t')$:
Proof of Lemma 92 (Completeness of Elimeq)

Case First we establish some properties of the subterms: σ

- Case \( \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \emptyset \):

  \[
  \text{mgu}(\sigma', t') = \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \emptyset \quad \text{By properties of unification}
  \]

  \[
  \text{succ}(\sigma') = [\Gamma] \text{succ}(\sigma') \\
  = \text{succ}([\Gamma] \sigma') \\
  \sigma' = [\Gamma] \sigma' \\
  \text{succ}(t') = [\Gamma] \text{succ}(t') \\
  = \text{succ}([\Gamma] t') \\
  t' = [\Gamma] t' \\
  \]

  \[
  \Gamma / \sigma' \triangleright t' : N \rightarrow \Gamma, \Delta \\
  \]

  \[
  \text{Given} \\
  \text{By definition of substitution} \\
  \text{By injectivity of successor} \\
  \text{By definition of substitution} \\
  \text{By injectivity of successor} \\
  \text{By i.h.}
  \]

- Case \( \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \perp \):

  \[
  \text{mgu}(\sigma', t') = \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \perp \quad \text{By properties of unification}
  \]

  \[
  \text{succ}(\sigma') = [\Gamma] \text{succ}(\sigma') \\
  = \text{succ}([\Gamma] \sigma') \\
  \sigma' = [\Gamma] \sigma' \\
  \text{succ}(t') = [\Gamma] \text{succ}(t') \\
  = \text{succ}([\Gamma] t') \\
  t' = [\Gamma] t' \\
  \]

  \[
  \Gamma / \sigma' \triangleright t' : N \rightarrow \perp \\
  \]

  \[
  \text{Given} \\
  \text{By definition of substitution} \\
  \text{By injectivity of successor} \\
  \text{By i.h.}
  \]

- Case \( \sigma = \sigma_1 \oplus \sigma_2 \) and \( t = t_1 \oplus t_2 \):

  First we establish some properties of the subterms:

  \[
  \sigma_1 \oplus \sigma_2 = [\Gamma](\sigma_1 \oplus \sigma_2) \\
  = [\Gamma] \sigma_1 \oplus [\Gamma] \sigma_2 \\
  \]

  \[
  [\Gamma] \sigma_1 = \sigma_1 \\
  [\Gamma] \sigma_2 = \sigma_2 \\
  [\Gamma] t_1 = t_1 \\
  [\Gamma] t_2 = t_2 \\
  \]

  \[
  \text{Given} \\
  \text{By injectivity of \( \oplus \)} \\
  \text{By injectivity of \( \oplus \)} \\
  \text{By injectivity of \( \oplus \)} \\
  \text{By injectivity of \( \oplus \)}
  \]

- Subcase \( \text{mgu}(\sigma, t) = \perp \):

  * Subcase \( \text{mgu}(\sigma_1, t_1) = \perp \):

    \[
    \Gamma / \sigma_1 \triangleright t_1 : \kappa \rightarrow \perp \\
    \Gamma / \sigma_1 \oplus \sigma_2 \triangleright t_1 \oplus t_2 : \kappa \rightarrow \perp \\
    \]

    \[
    \text{By i.h.} \\
    \text{By rule ElimeqBinBot}
    \]

  * Subcase \( \text{mgu}(\sigma_1, t_1) = \emptyset \) and \( \text{mgu}(\emptyset_1(\sigma_2), \emptyset_1(t_2)) = \perp \):

    \[
    \Gamma / \sigma_1 \triangleright t_1 : \kappa \rightarrow \Gamma, \Delta_1 \\
    \]

    \[
    [\Gamma, \Delta_1] u = \emptyset_1([\Gamma] u) \quad \text{for all } u \text{ such that } \ldots \quad " \\
    \]

    \[
    [\Gamma, \Delta_1] \sigma_2 = \emptyset_1([\Gamma] \sigma_2) \\
    = \emptyset_1(\sigma_2) \\
    [\Gamma] \sigma_2 = \sigma_2 \\
    [\Gamma, \Delta_1] t_2 = \emptyset_1([\Gamma] t_2) \\
    = \emptyset_1(t_2) \\
    \]

    \[
    \text{Above line with } \sigma_2 \text{ as } u \\
    \text{Since } [\Gamma] \sigma_2 = \sigma_2 \\
    \text{Above line with } t_2 \text{ as } u \\
    \text{By transitivity of equality}
    \]

    \[
    [\Gamma, \Delta_1] u = \emptyset_1([\Gamma] u) \quad \text{for all } u \text{ such that } \ldots \quad " \\
    \]

    \[
    [\Gamma, \Delta_1] \sigma_2 = [\Gamma, \Delta_1] \sigma_2 \\
    [\Gamma, \Delta_1] t_2 = [\Gamma, \Delta_1] t_2 \\
    \]

    \[
    \text{By Lemma 29 (Substitution Monotonicity)} \\
    \text{By Lemma 29 (Substitution Monotonicity)}
    \]
\[ \Gamma, \Delta_1 \vdash [\Gamma, \Delta_1]\sigma_2 \triangleq [\Gamma, \Delta_1]t_2 : \kappa \vdash \perp \] 

By i.h. 

\[ \Gamma / \sigma_1 \vdash \sigma_2 \triangleq t_1 \oplus t_2 : \kappa \vdash \perp \] 

By rule ElimeqBin

** Subcase mgu(\sigma, t) = \emptyset:

\[ mgu(\sigma_1 \oplus \sigma_2, t_1 \oplus t_2) = \emptyset = \emptyset \circ \emptyset \] 

By properties of unifiers

\[ mgu(\sigma_1, t_1) = \emptyset \] 

By i.h.

\[ mgu(\sigma_2, t_2) = \emptyset \] 

By i.h.

\[ \Gamma / \sigma_1 \vdash t_1 : \kappa \vdash \Gamma, \Delta_1 \] 

By i.h.

\[ \Gamma, \Delta_1 \vdash [\Gamma, \Delta_1]\sigma_2 = \emptyset ([\Gamma, \Delta_1]\sigma_2) \] 

Above line with \sigma_2 as u

\[ = \emptyset_1(\sigma_2) \] 

\[ = [\Gamma]\sigma_2 = \sigma_2 \] 

\[ [\Gamma, \Delta_1]t_2 = \emptyset_1([\Gamma]t_2) \] 

Above line with t_2 as u

\[ = \emptyset_1(t_2) \] 

\[ = [\Gamma]\sigma_2 = \sigma_2 \] 

\[ mgu([\Gamma, \Delta_1]\sigma_2, [\Gamma, \Delta_1]t_2) = \emptyset_2 \] 

By transitivity of equality

\[ [\Gamma, \Delta_1][\Gamma, \Delta_1]\sigma_2 = [\Gamma, \Delta_1]\sigma_2 \] 

By Lemma 29 (Substitution Monotonicity)

\[ [\Gamma, \Delta_1][\Gamma, \Delta_1]t_2 = [\Gamma, \Delta_1]t_2 \] 

By Lemma 29 (Substitution Monotonicity)

\[ \Gamma, \Delta_1 / [\Gamma, \Delta_1]\sigma_2 \triangleq [\Gamma, \Delta_1]t_2 : \kappa \vdash \Gamma, \Delta_1, \Delta_2 \] 

By i.h.

\[ [\Gamma, \Delta_1, \Delta_2]u' = \emptyset_2([\Gamma, \Delta_1]u') \text{ for all } u' \text{ such that } \ldots \] 

\[ mgu([\Gamma, \Delta_1]u'), [\Gamma, \Delta_1]u' = \emptyset_2([\Gamma, \Delta_1]u') \] 

Above line with \sigma_2 as u

\[ = \emptyset_2_{\emptyset_1}(\emptyset_1([\Gamma]u)) \] 

By *

\[ = \emptyset_1([\Gamma]u) \] 

By *

\[ \emptyset = \emptyset_2 \circ \emptyset_1 \]

** Suppose \( \Gamma \vdash u : \kappa' \):

\[ [\Gamma, \Delta_1, \Delta_2]u = \emptyset_2([\Gamma, \Delta_1]u) \] 

By **

\[ = \emptyset_2_{\emptyset_1}(\emptyset_1([\Gamma]u)) \] 

By *

\[ = \emptyset_1([\Gamma]u) \] 

\[ \emptyset = \emptyset_2 \circ \emptyset_1 \]

*** Case \( \sigma = \alpha \): Similar to previous case.

\[ \emptyset \]

Lemma 93 (Substitution Upgrade).
If \( \Delta \) has the form \( \alpha_1 = t_1, \ldots, \alpha_n = t_n \) and, for all \( u \) such that \( \Gamma \vdash u : \kappa \), it is the case that \( [\Gamma, \Delta]u = \emptyset([\Gamma]u) \), then:

(i) If \( \Gamma \vdash A \) type then \( [\Gamma, \Delta]A = \emptyset([\Gamma]A) \).

(ii) If \( \Gamma \rightarrow \Omega \) then \( [\Omega]\Gamma = \emptyset([\Omega]\Gamma) \).
(iii) If $\Gamma \rightarrow \Omega$ then $[\Omega, \Delta|\Gamma, \Delta] = \theta([\Omega]|\Gamma)$.

(iv) If $\Gamma \rightarrow \Omega$ then $[\Omega, \Delta]e = \theta([\Omega]|e)$.

Proof. Part (i): By induction on the given derivation, using the given “for all” at the leaves.

Part (ii): By induction on the given derivation, using part (i) in the $\text{Var}$ case.

Part (iii): By induction on $\Delta$. In the base case ($\Delta = \epsilon$), use part (ii). Otherwise, use the i.h. and the definition of context substitution.

Part (iv): By induction on $e$, using part (i) in the $e = (e_0 : A)$ case. $\Box$

Lemma 94 (Completeness of Propequiv).

Given $\Gamma \rightarrow \Omega$ and $\Gamma \vdash P \text{ prop and } \Gamma \vdash Q \text{ prop}$ and $[\Omega]P = [\Omega]Q$

then $\Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \vdash \Delta$

where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$.

Proof. By induction on the given derivations. There is only one possible case:

- Case $\Gamma \vdash \sigma_1 : N \quad \Gamma \vdash \sigma_2 : N \quad \text{EqProp}$

\[\begin{aligned}
\Gamma &\vdash \sigma_1 = \sigma_2 \text{ prop} \\
[\Omega]|(\sigma_1 = \sigma_2) &= [\Omega]|(\tau_1 = \tau_2) \\
[\Omega]|\sigma_1 &= [\Omega]|\tau_1 \\
[\Omega]|\sigma_2 &= [\Omega]|\tau_2 \\
\Gamma &\vdash \sigma_1 : N \\
\Gamma &\vdash \tau_1 : N \\
\Gamma &\vdash [\Gamma]|\sigma_1 \equiv [\Gamma]|\sigma_2 \vdash \Theta \\
\Theta &\rightarrow \Omega_0 \\
\Omega &\rightarrow \Omega_0 \\
\Gamma &\vdash \sigma_2 : N \\
\Theta &\vdash \sigma_2 : N \\
\Theta &\vdash \tau_2 : N \\
\Theta &\vdash [\Theta]|\tau_1 \equiv [\Theta]|\tau_2 \vdash \Delta \\
\Delta &\rightarrow \Omega_0 \\
\Omega_0 &\rightarrow \Omega' \\
[\Theta]|\tau_1 &= [\Theta]|\Gamma|\tau_1 \\
[\Theta]|\tau_2 &= [\Theta]|\Gamma|\tau_2 \\
\Theta &\vdash [\Theta]|\Gamma|\tau_1 \equiv [\Theta]|\Gamma|\tau_2 \vdash \Delta \\
\Theta &\rightarrow \Omega \\
\Omega &\rightarrow \Omega' \\
\Gamma &\vdash ([\Gamma]|\sigma_1 = [\Gamma]|\sigma_2) \equiv ([\Gamma]|\tau_1 = [\Gamma]|\tau_2) \vdash \Gamma \\
\Gamma &\vdash [\Gamma]|\sigma_1 = [\Gamma]|\sigma_2 \equiv ([\Gamma]|\tau_1 = [\Gamma]|\tau_2) \vdash \Gamma \\
\Gamma &\vdash [\Gamma]|\sigma_2 \equiv ([\Gamma]|\tau_1 = [\Gamma]|\tau_2) \vdash \Gamma \\
\end{aligned}\]

By above equalities

By Lemma 91 (Completeness of Checkeq)

By above equalities

By Lemma 91 (Completeness of Checkeq)

By Lemma 36 (Extension Weakening (Sorts))

Similarly

By Lemma 29 (Substitution Monotonicity) (i)

By Lemma 33 (Extension Transitivity)

By PropEq

By above equalities

\[\Box\]

Lemma 95 (Completeness of Checkprop).

If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$

and $\Gamma \vdash P \text{ prop}$

and $[\Gamma]P = P$

and $[\Omega]|\Gamma \vdash [\Omega]P \text{ true}$

then $\Gamma \vdash P \text{ true } \vdash \Delta$

where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$.
Proof of Lemma 95 (Completeness of Checkprop) lem:checkprop-completeness

Proof. Only one rule, \texttt{DeclCheckpropEq}, can derive \([\Omega]\Gamma \vdash [\Omega]P \text{ true}\), so by inversion, \(P\) has the form \([t_1 = t_2]\) where \([\Omega]t_1 = [\Omega]t_2\).

By inversion on \(\Gamma \vdash (t_1 = t_2) \text{ prop}\), we have \(\Gamma \vdash t_1 : \text{N}\) and \(\Gamma \vdash t_2 : \text{N}\).

Then by Lemma 91 (Completeness of Checkeq), \(\Gamma \vdash [\Gamma]t_1 \equiv [\Gamma]t_2 : \text{N} \rightarrow \Delta\) where \(\Delta \rightarrow \Omega'\) and \(\Omega \rightarrow \Omega'\).

By \texttt{CheckpropEq}, \(\Gamma \vdash (t_1 = t_2) \text{ true} \rightarrow \Delta\).

\[\]

K.2 Completeness of Equivalence and Subtyping

Lemma 96 (Completeness of Equiv).

If \(\Gamma \rightarrow \Omega\) and \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type

and \([\Omega]A = [\Omega]B\)

then there exist \(\Delta\) and \(\Omega'\) such that \(\Delta \rightarrow \Omega'\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash [\Gamma]A \equiv [\Gamma]B \rightarrow \Delta\).

Proof. By induction on the derivations of \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type.

We distinguish cases of the rule concluding the first derivation. In the first four cases, it follows from \([\Omega]A = [\Omega]B\) and the syntactic invariant that \(\Omega\) substitutes terms \(t\) (rather than types \(A\)) that the second derivation is concluded by the same rule. Moreover, if none of these three rules concluded the first derivation, the rule concluding the second derivation must not be \texttt{ImpliesWF}, \texttt{ForallWF}, \texttt{ExistsWF}, or \texttt{WithWF} either.

Because \(\Omega\) is predicative, the head connective of \([\Gamma]A\) must be the same as the head connective of \([\Omega]A\).

We distinguish cases that are \textit{imposs.} (impossible), \textit{fully written out}, and \textit{similar to fully-written-out cases}. For the lower-right case, where both \([\Gamma]A\) and \([\Gamma]B\) have a binary connective \(\oplus\), it must be the same connective.

The \textit{Vec} type is omitted from the table, but can be treated similarly to \(\lor\) and \(\land\).

\[
\begin{array}{cccccccc}
\text{\(\lor\)} & \land & \forall \beta. B' & \exists \beta. B' & 1 & \alpha & \beta & B_1 \oplus B_2 \\
\hline
\text{Implies} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\end{array}
\]

- Case \(\Gamma \vdash P \text{ prop}\) \(\Gamma \vdash A_0\) type

\[
\Gamma \vdash P \rightarrow A_0\) type \hspace{1cm} \texttt{ImpliesWF}
\]

This case of the rule concluding the first derivation coincides with the \texttt{Implies} entry in the table.

We have \([\Omega]A = [\Omega]B\), that is, \([\Omega](P \rightarrow A_0) = [\Omega]B\).

Because \(\Omega\) is predicative, \(B\) must have the form \(Q \supset B_0\), where \([\Omega]P = [\Omega]Q\) and \([\Omega]A_0 = [\Omega]B_0\).
Proof of Lemma 96 (Completeness of Equiv) lem:equiv-completeness

\( \Gamma \vdash P \) prop
\( \Gamma \vdash A_0 \) type
\( \Gamma \vdash Q \triangleright B_0 \) type
\( \Gamma \vdash Q \) prop
\( \Gamma \vdash B_0 \) type
\( \Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \vdash \Theta \) By Lemma 94 (Completeness of Propequiv)
\( \Theta \rightarrow \Omega_0 \)
\( \Omega \rightarrow \Omega_0 \)

\( \Gamma \rightarrow \Theta \) By Lemma 48 (Prop Equivalence Extension)
\( \Gamma \vdash A_0 \) type
\( \Gamma \vdash B_0 \) type
\( [\Omega]A_0 = [\Omega]B_0 \) Above
\( [\Omega_0]A_0 = [\Omega_0]B_0 \) Above
\( \Gamma \vdash [\Gamma]A_0 \equiv [\Gamma]B_0 \vdash \Delta \) By i.h.

\[ \Delta \rightarrow \Omega' \]
\[ \Omega_0 \rightarrow \Omega' \]

\[ \Omega \rightarrow \Omega' \]

\[ \Gamma \vdash ([\Gamma]P \triangleright [\Gamma]A_0) \equiv ([\Gamma]Q \triangleright [\Gamma]B_0) \vdash \Delta \]

\[ \Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \vdash [\Gamma]B_0 \]

\[ \Gamma \vdash [\Gamma]Q \triangleright [\Gamma]B_0 \]

\[ \Gamma \vdash \forall \alpha : \kappa. [\Gamma]A_0 \equiv [\Gamma]B_0 \rightarrow \Delta_0 \]

\[ \Delta_0 \rightarrow \Omega \]
\( \Omega, \alpha : \kappa \rightarrow \Omega \)

\[ \Omega \rightarrow \Omega' \] By Lemma 33 (Extension Transitivity)

Case WithWF: Similar to the ImpliesWF case, coinciding with the With entry in the table.

Case \( \Gamma, \alpha : \kappa \vdash A_0 \) type
\( \Gamma \vdash \forall \alpha : \kappa. A_0 \) type

This case coincides with the Forall entry in the table.

\[ \Gamma = \forall \alpha : \kappa. B_0 \]
\( [\Omega]A_0 = [\Omega]B_0 \)

\[ \Omega \rightarrow \Omega' \] By Lemma 22 (Extension Inversion) (i)

\[ \Delta_0 = (\Delta, \alpha : \kappa, \Delta') \]

\[ \Gamma \vdash \forall \alpha : \kappa. [\Gamma]A_0 \equiv [\Gamma]B_0 \rightarrow \Delta \]

\[ \Gamma \vdash [\Gamma](\forall \alpha : \kappa. A_0) \equiv [\Gamma](\forall \alpha : \kappa. B_0) \vdash \Delta \]

Case ExistsWF: Similar to the ForallWF case. (This is the Exists entry in the table.)

Case BinWF: If BinWF also concluded the second derivation, then the proof is similar to the ImpliesWF case, but on the first premise, using the i.h. instead of Lemma 94 (Completeness of Propequiv). This is the 2.Bins entry in the lower right corner of the table.

If BinWF did not conclude the second derivation, we are in the 2.AEx.Bin or 2.BEx.Bin entries; see below.
In the remainder, we cover the $4 \times 4$ region in the lower right corner, starting from 2.Units. We already handled the 2.Bins entry in the extreme lower right corner. At this point, we split on the forms of $[\Gamma]A$ and $[\Gamma]B$ instead; in the remaining cases, one or both types is atomic (e.g. 2.Uvars, 2.AEx.Bin) and we will not need to use the induction hypothesis.

**Case 2.Units:** $[\Gamma]A = [\Gamma]B = 1$

- $\Gamma \vdash 1 \equiv 1 \rightarrow \Gamma$  \hspace{1cm} By $\equiv$Unit
- $\Gamma \rightarrow \Omega$  \hspace{1cm} Given
- Let $\Omega' = \Omega$.
- $\Delta \rightarrow \Omega$  \hspace{1cm} $\Delta = \Gamma$
- $\Omega \rightarrow \Omega'$  \hspace{1cm} By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

**Case 2.Uvars:** $[\Gamma]A = [\Gamma]B = \alpha$

- $\Gamma \rightarrow \Omega$  \hspace{1cm} Given
- Let $\Omega' = \Omega$.
- $\Gamma \vdash \alpha \equiv \alpha \rightarrow \Gamma$  \hspace{1cm} By $\equiv$Var
- $\Delta \rightarrow \Omega$  \hspace{1cm} $\Delta = \Gamma$
- $\Omega \rightarrow \Omega'$  \hspace{1cm} By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

**Case 2.AExUnit:** $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = 1$

- $\Gamma \rightarrow \Omega$  \hspace{1cm} Given
- $1 = [\Omega]1$  \hspace{1cm} By definition of substitution
- $\hat{\alpha} \notin FV(1)$  \hspace{1cm} By definition of $FV(-)$
- $[\Omega]1 \vdash [\Omega]\hat{\alpha} \leq [\Omega]1$  \hspace{1cm} Given
- $\Gamma \vdash \hat{\alpha} := 1 : \ast \rightarrow \Delta$  \hspace{1cm} By Lemma 90 (Completeness of Instantiation) (1)
- $\Omega \rightarrow \Omega'$  \hspace{1cm} “
- $\Delta \rightarrow \Omega'$  \hspace{1cm} “
- $1 = [\Gamma]1$  \hspace{1cm} By definition of substitution
- $\hat{\alpha} \notin FV(1)$  \hspace{1cm} By definition of $FV(-)$
- $\Gamma \vdash \hat{\alpha} \equiv 1 \rightarrow \Delta$  \hspace{1cm} By $\equiv$InstantiateL

**Case 2.BExUnit:** $[\Gamma]A = 1$ and $[\Gamma]B = \hat{\alpha}$

Symmetric to the 2.AExUnit case.

**Case 2.AEx.Uvar:** $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = \alpha$

Similar to the 2.AEx.Unit case, using $\beta = [\Omega]\beta = [\Gamma]\beta$ and $\hat{\alpha} \notin FV(\beta)$.

**Case 2.BExUvar:** $[\Gamma]A = 1$ and $[\Gamma]B = \hat{\alpha}$

Symmetric to the 2.AExUvar case.

**Case 2.AEx.SameEx:** $[\Gamma]A = \hat{\alpha} = \hat{\beta} = [\Gamma]B$

- $\Gamma \vdash \hat{\alpha} \equiv \hat{\beta} \equiv \hat{\alpha} \rightarrow \Gamma$  \hspace{1cm} By $\equiv$Exvar ($\hat{\alpha} = \hat{\beta}$)
- $[\Gamma]\hat{\alpha} = \hat{\beta}$  \hspace{1cm} $\hat{\alpha}$ unsolved in $\Gamma$
- $\Gamma \vdash [\Gamma]\hat{\alpha} \equiv [\Gamma]\hat{\beta} \rightarrow \Gamma$  \hspace{1cm} By above equality + $\hat{\alpha} = \hat{\beta}$
- $\Gamma \rightarrow \Omega$  \hspace{1cm} Given
- $\Delta \rightarrow \Omega$  \hspace{1cm} $\Delta = \Gamma$
- Let $\Omega' = \Omega$.
- $\Omega \rightarrow \Omega'$  \hspace{1cm} By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$
• **Case 2.AEx.OtherEx**: $[\Gamma]A = \check{\alpha}$ and $[\Gamma]B = \check{\beta}$ and $\check{\alpha} \neq \check{\beta}$
  
  Either $\check{\alpha} \in \text{FV}([\Gamma]\check{\beta})$, or $\check{\alpha} \notin \text{FV}([\Gamma]\check{\beta})$.
  
  - $\check{\alpha} \in \text{FV}([\Gamma]\check{\beta})$:
    
    We have $\check{\alpha} \preceq [\Gamma]\check{\beta}$.
    
    Therefore $\check{\alpha} = [\Gamma]\check{\beta}$, or $\check{\alpha} \prec [\Gamma]\check{\beta}$.
    
    But we are in Case 2.AEx.OtherEx, so the former is impossible.
    
    Therefore, $\check{\alpha} \prec [\Gamma]\check{\beta}$.
    
    By a property of substitutions, $[\Omega]\check{\alpha} \prec [\Omega][\Gamma]\check{\beta}$.
    
    Since $\Gamma \rightarrow \Omega$, by Lemma 29 (Substitution Monotonicity) (iii), $[\Omega][\Gamma]\check{\alpha} = [\Omega]\check{\beta}$, so $[\Omega]\check{\alpha} \prec [\Omega]\check{\beta}$.
    
    But it is given that $[\Omega]\check{\alpha} = [\Omega]\check{\beta}$, a contradiction.
  
  - $\check{\alpha} \notin \text{FV}([\Gamma]\check{\beta})$:
    
    $\Gamma \vdash \check{\alpha} := [\Gamma]\check{\beta} : * \rightarrow \Delta$  
    
    By Lemma 90 (Completeness of Instantiation)
    
    $\Gamma \vdash \check{\alpha} = [\Gamma]\check{\beta} : * \rightarrow \Delta$  
    
    By $\equiv\text{Instantiate}_L$
    
    $\Delta \rightarrow \Omega'$  
    
    " "
    
    $\Omega \rightarrow \Omega'$  
    
    " "

• **Case 2.AEx.Bin**: $[\Gamma]A = \check{\alpha}$ and $[\Gamma]B = B_1 \oplus B_2$
  
  Since $[\Gamma]B$ is an arrow, it cannot be exactly $\check{\alpha}$. By the same reasoning as in the previous case (2.AEx.OtherEx), $\check{\alpha} \notin \text{FV}([\Gamma]\check{\beta})$.
  
  $\Gamma \vdash \check{\alpha} := [\Gamma]\check{\beta} : * \rightarrow \Delta$  
  
  By Lemma 90 (Completeness of Instantiation)
  
  $\Delta \rightarrow \Omega'$  
  
  " "
  
  $\Omega \rightarrow \Omega'$  
  
  " "
  
  $\Gamma \vdash [\Gamma]A = [\Gamma]B : \Delta$  
  
  By $\equiv\text{Instantiate}_L$

• **Case 2.BEx.Bin**: $[\Gamma]A = A_1 \oplus A_2$ and $[\Gamma]B = \check{\beta}$
  
  Symmetric to the 2.AEx.Bin case, applying $\equiv\text{Instantiate}_R$ instead of $\equiv\text{Instantiate}_L$. □

**Theorem 9** (Completeness of Subtyping).

If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash A$ type and $\Gamma \vdash B$ type

and $[\Omega][\Gamma][A] \leq^\pm [\Omega][B]$

then there exist $\Delta$ and $\Omega'$ such that $\Delta \rightarrow \Omega'$

and $\text{dom}(\Delta) = \text{dom}(\Omega')$

and $\Omega \rightarrow \Omega'$

and $\Gamma \vdash [\Gamma]A <^\pm [\Gamma]B \rightarrow \Delta$.

**Proof.** By induction on the number of $\forall/\exists$ quantifiers in $[\Omega]A$ and $[\Omega]B$.

It is straightforward to show $\text{dom}(\Delta) = \text{dom}(\Omega')$; for examples of the necessary reasoning, see the proof of Theorem $[\Gamma]$.

We have $[\Omega][\Gamma] \vdash [\Omega]A \leq^\pm [\Omega]B$.

• **Case** $[\Omega][\Gamma] \vdash [\Omega]A$ type
  
  nonpos($[\Omega]A$)
  
  $[\Omega][\Gamma] \vdash [\Omega]A \leq^\pm [\Omega]A$  
  
  $[\Omega][\Gamma] \vdash [\Omega]A \leq^\pm [\Omega][B]$

  $\leq\text{Refl}$

First, we observe that, since applying $\Omega$ as a substitution leaves quantifiers alone, the quantifiers that head $A$ must also head $B$. For convenience, we alpha-vary $B$ to quantify over the same variables as $A$.

- If $A$ is headed by $\forall$, then $[\Omega]A = (\forall \alpha : \kappa. [\Omega]A_0) = (\forall \alpha : \kappa. [\Omega]B_0) = [\Omega]B$.
  
  Let $\Gamma_0 = (\Gamma, \alpha : \kappa, \vartriangleright_\alpha, \check{\alpha} : \kappa)$.
  
  Let $\Omega_0 = (\Omega, \alpha : \kappa, \vartriangleright_\alpha, \check{\alpha} : \kappa = \alpha)$.
Case

• We begin by considering whether or not \( \alpha : [\beta <:] \Omega \).

\[
\begin{align*}
\Delta & \rightarrow \Omega' \\
\Omega & \rightarrow \Omega' \\
\Gamma & \rightarrow [\Gamma]A <: [\Gamma]B & \text{By Lemma 96 (Completeness of Subtyping)}
\end{align*}
\]

* If \( \text{pol}(A_0) \in \{-, 0\} \), then:

\[
\begin{align*}
[\Omega_0]_\Gamma & \Gamma \vdash [\Omega]A_0 \leq [\Omega]A_0 & \text{By } \leq_{\text{Refl}} \\
[\Omega_0]_\Gamma & \Gamma \vdash [\Omega][\beta/\alpha]A_0 \leq A_0 & \text{By def. of subst.} \\
A_0 & \rightarrow \Omega' & \text{By i.h. (fewer quantifiers)} \\
\Omega_0 & \rightarrow \Omega' & \\
\Gamma_0 & \vdash [\Gamma]_\Omega[\beta/\alpha]A_0 <: [\Gamma]_\Omega B_0 \vdash \Delta_0 & \\
\Gamma_0 & \vdash [\beta/\alpha]_\Gamma A_0 <: [\Gamma]_\Omega B_0 \vdash \Delta_0 & \text{\( \alpha \) unsolved in } \Gamma_0 \\
\Gamma_0 & \vdash [\beta/\alpha]_\Gamma A_0 <: [\Gamma]_\Omega B_0 \vdash \Delta_0 & \Gamma_0 \text{ substitutes as } \Gamma \\
\Gamma, \alpha : \kappa \vdash \forall \alpha : \kappa. [\Gamma]_\Omega A_0 <: [\Gamma]_\Omega B_0 \vdash \Delta, \alpha : \kappa, \Theta & \text{By } \leq_{\forall L} \\
\Gamma & \vdash \forall \alpha : \kappa. [\Gamma]_\Omega A_0 <: [\Gamma]_\Omega B_0 \vdash \Delta & \text{By def. of subst.} \\
\Delta & \rightarrow \Omega & \text{By lemma} \\
\Omega & \rightarrow \Omega' & \text{By lemma} \\
\Gamma & \vdash [\Gamma]_\Omega A <: [\Gamma]_\Omega B & \vdash \Delta & \text{By } \leq_{\text{Equiv}} \\
\end{align*}
\]

* If \( \text{pol}(A_0) = + \), then proceed as above, but apply \( \leq_{\text{Refl+}} \) instead of \( \leq_{\text{Refl}} \) and apply \( \leq_{+:L} \) after applying the i.h. (Rule \( \leq_{+:R} \) also works.)

- If \( A \) is not headed by \( \forall \):

We have \( \text{nonneg}(\Omega[A] \vdash \tau) \). Therefore \( \text{nonneg}(\Omega[A]) \), and thus \( A \) is not headed by \( \exists \). Since the same quantifiers must also head \( B \), the conditions in rule \( \leq_{\text{Equiv}} \) are satisfied.

\[
\begin{align*}
\Gamma & \rightarrow \Omega & \text{Given} \\
\Gamma & \vdash [\Gamma]A \equiv [\Gamma]B \vdash \Delta & \text{By Lemma 96 (Completeness of Subtyping)} \\
\Delta & \rightarrow \Omega' & \\
\Omega & \rightarrow \Omega' & \\
\Gamma & \vdash [\Gamma]A <: [\Gamma]B \vdash \Delta & \text{By } \leq_{\text{Equiv}}
\end{align*}
\]

* Case \( \leq_{\text{Refl+}} \) Symmetric to the \( \leq_{\text{Refl}} \) case, using \( \leq_{+:L} \) (or \( \leq_{+:R} \)), and \( \leq_{+:R} \leq_{+:L} \) instead of \( \leq_{+:L} \leq_{+:R} \).

* Case \[
[\Omega]_\Gamma \vdash \tau : \kappa \\
\begin{array}{c}
[\Omega]_\Gamma \vdash [\tau/\alpha]_\Gamma A_0 \leq [\Omega]_\Gamma B \\
[\Omega]_\Gamma \vdash \forall \alpha : \kappa. [\Omega]_\Gamma A_0 \leq [\Omega]_\Gamma B \\
[\Omega]_\Gamma \vdash [\Omega]_\Gamma A \leq [\Omega]_\Gamma B & \text{By } \leq_{\forall L}
\end{array}
\]

We begin by considering whether or not \( [\Omega]_\Gamma B \) is headed by a universal quantifier.

- \( [\Omega]_\Gamma B = (\forall \beta : \kappa. \beta') \):

\[
[\Omega]_\Gamma \beta : \kappa' \vdash [\Omega]_\Gamma A \leq B' & \text{By Lemma 5 (Subtyping Inversion)}
\]

The remaining steps are similar to the \( \leq_{\forall R} \) case.

- \( [\Omega]_\Gamma B \) not headed by \( \forall \):

\[
\begin{align*}
\begin{array}{c}
\begin{array}{c}
[\Omega]_\Gamma \vdash \tau : \kappa \\
\Gamma & \rightarrow \Omega & \text{Subderivation} \\
\Gamma, [\beta/\alpha]_\Gamma & \rightarrow [\Omega, [\beta/\alpha]_\Gamma] & \text{By } \rightarrow_{\text{Marker}} \\
\Gamma, [\beta/\alpha]_\Gamma & \vdash \Delta, [\beta/\alpha]_\Gamma & \rightarrow [\kappa = \tau] & \text{By } \rightarrow_{\text{Solve}} \\
[\Omega]_\Gamma & = [\Omega_0]_\Gamma [\Gamma, [\beta/\alpha]_\Gamma] & \text{By definition of context application (lines 16, 13)}
\end{array}
\end{array}
\end{align*}
\]
Proof of **Theorem 9** (Completeness of Subtyping) thm:subtyping-completeness

\[ [\Omega]|\Gamma \vdash [\tau/\alpha][\Omega]A_0 \leq \cdot [\Omega]B \quad \text{Subderivation} \]
\[ [\Omega_0]|\Gamma,\alpha,\beta : \kappa \vdash [\tau/\alpha][\Omega]A_0 \leq \cdot [\Omega]B \quad \text{By above equality} \]
\[ [\Omega_0]|\Gamma,\alpha,\beta : \kappa \vdash [\Omega_0][\alpha/\alpha][\Omega]A_0 \leq \cdot [\Omega]B \quad \text{By definition of substitution} \]
\[ [\Omega_0]|\Gamma,\alpha,\beta : \kappa \vdash [\Omega_0][\alpha/\alpha][\Omega]A_0 \leq \cdot [\Omega_0]B \quad \text{By definition of substitution} \]
\[ [\Omega_0]|\Gamma,\alpha,\beta : \kappa \vdash [\Omega_0][\alpha/\alpha][\Omega]A_0 \leq \cdot [\Omega_0]B \quad \text{By distributivity of substitution} \]

\[ \Gamma,\alpha,\beta : \kappa \vdash [\Gamma][\alpha/\alpha][\Omega]A_0 \leq \cdot [\Gamma]B \rightarrow \Delta_0 \quad \text{By definition of substitution} \]
\[ \Delta_0 \rightarrow \Omega'' \]
\[ \Omega_0 \rightarrow \Omega'' \]

\[ \Gamma,\alpha,\beta : \kappa \rightarrow \Delta_0 \quad \text{By Lemma 50} \]
\[ \Delta_0 = (\Delta,\alpha,\Theta) \quad \text{By Lemma 22 Extension Inversion (ii)} \]
\[ \Gamma \rightarrow \Delta \]
\[ \Omega'' = (\Omega',\alpha,\Theta,\Omega_Z) \quad \text{By Lemma 22 Extension Inversion (ii)} \]
\[ \Delta \rightarrow \Omega' \quad \text{Above} \]
\[ \Omega_0 \rightarrow \Omega' \quad \text{By Lemma 22 Extension Inversion (ii)} \]

\[ \Gamma,\alpha,\beta : \kappa \rightarrow \Delta_0 \quad \text{By above equality} \]
\[ \Delta_0 = (\Delta,\alpha,\Theta) \quad \text{By def. subst. ([\Gamma][\alpha/\alpha] \rightarrow \Delta,\alpha,\Theta)} \]
\[ \Gamma \rightarrow \Delta \quad \text{From the case assumption} \]
\[ \Gamma \vdash \forall \alpha : \kappa. [\Gamma]A_0 \leq \cdot [\Gamma]B \rightarrow \Delta \quad \text{By \textless \textless \forall L} \]
\[ \Gamma \vdash [\Gamma](\forall \alpha : \kappa. A_0) \leq \cdot [\Gamma]B \rightarrow \Delta \quad \text{By definition of substitution} \]

- **Case**

\[ [\Omega],\beta : \kappa \vdash [\Omega]A \leq \cdot [\Omega]B_0 \]
\[ [\Omega]|\Gamma,\beta : \kappa \vdash [\Omega]A \leq \cdot [\Omega]B_0 \quad \text{\textless \forall R} \]

\[ B = \forall \beta : \kappa. B_0 \quad \Omega \text{ predicative} \]
\[ [\Omega]|\Gamma \vdash [\Omega]A \leq \cdot [\Omega]B \quad \text{Given} \]
\[ [\Omega]|\Gamma \vdash [\Omega]A \leq \cdot \forall \beta. [\Omega]B_0 \quad \text{By above equality} \]
\[ [\Omega]|\Gamma,\beta : \kappa \vdash [\Omega]A \leq \cdot [\Omega]B_0 \quad \text{Subderivation} \]
\[ [\Omega,\beta : \kappa](\Gamma, \beta : \kappa) \vdash [\Omega,\beta : \kappa]A \leq \cdot [\Omega,\beta : \kappa]B_0 \quad \text{By definitions of substitution} \]
\[ \Gamma,\beta : \kappa \vdash [\Gamma,\beta:k]A \leq \cdot [\Gamma,\beta:k]B_0 \rightarrow \Delta' \quad \text{By i.h. (B lost a quantifier)} \]
\[ \Delta' \rightarrow \Omega_0'' \quad \text{Above} \]
\[ \Omega,\beta : \kappa \rightarrow \Omega_0'' \quad \text{By definition of substitution} \]
\[ \Gamma,\beta : \kappa \vdash [\Gamma]A \leq \cdot [\Gamma]B_0 \rightarrow \Delta' \quad \text{By definition of substitution} \]
\[ \Delta' = (\Delta,\beta : \kappa,\Theta) \quad \text{By Lemma 43 Instantiation Extension} \]
\[ \Gamma \rightarrow \Delta \quad \text{By Lemma 22 Extension Inversion (i)} \]
\[ \Delta,\beta : \kappa,\Theta \rightarrow \Omega_0' \quad \text{By \Delta' \rightarrow \Omega_0' \ and \ above \ equality} \]
\[ \Omega_0' = (\Omega',\beta : \kappa,\Omega_R) \quad \text{By Lemma 22 Extension Inversion (i)} \]
\[ \Delta \rightarrow \Omega' \quad \text{"} \]

\[ \text{Proof of **Theorem 9** (Completeness of Subtyping) thm:subtyping-completeness} \]
Proof of Theorem 9 (Completeness of Subtyping)

\[\Gamma, \beta : \kappa \vdash [\Gamma]A < : [\Gamma]B_0 \vdash \Delta, \beta : \kappa, \Theta\]

By above equality

\[\Omega, \beta : \kappa \rightarrow \Omega', \beta : \kappa, \Omega_R\]

By above equality

\[\Omega \rightarrow \Omega'\]

By Lemma 33 (Extension Transitivity)

\[\Gamma \vdash [\Gamma]A < : \forall \beta : \kappa. [\Gamma]B_0 \rightarrow \Delta\]

By \(\because \forall R\)

\[\Gamma \vdash [\Gamma]A < : [\Gamma](\forall \beta : \kappa. B_0) \rightarrow \Delta\]

By definition of substitution

- Case \([\Omega] [\Gamma, \alpha : \kappa \vdash [\Omega]A_0 \leq^+ [\Omega]B\]

\[\Omega^L\]

\[\Omega \vdash \exists \alpha : \kappa, [\Omega]A_0 \leq^+ [\Omega]B\]

\[\Omega \vdash (\alpha : \kappa, \alpha : \kappa, \Omega)\]

By above equality

\[\Omega \vdash A \leq [\Omega]B\]

\[\Omega \vdash \exists \alpha : \kappa, [\Omega]A_0 \leq^+ [\Omega]B\]

\[\Omega \vdash [\Omega]A_0 \leq^+ [\Omega]B\]

\[\Omega \vdash \exists \alpha : \kappa, [\Omega]A_0 \leq^+ [\Omega]B\]

By above equality

\[\Omega, \alpha : \kappa \vdash [\Omega]A_0 < : [\Omega]B_0 \vdash \Delta'\]

By definition of substitution

\[\Gamma,\alpha : \kappa \vdash [\Gamma]A < : [\Gamma]B_0 \vdash \Delta'\]

By Lemma 43 (Instantiation Extension)

\[\Delta' = (\Delta, \alpha : \kappa, \Theta)\]

By Lemma 22 (Extension Inversion) (i)

\[\Gamma \rightarrow \Delta'\]

By definition of substitution

\[\Delta_0' = (\Omega_0', \alpha : \kappa, \Omega_R)\]

By Lemma 22 (Extension Inversion) (i)

\[\Delta \rightarrow \Omega'\]

By definition of substitution

\[\Gamma, \alpha : \kappa \vdash [\Gamma]A_0 < : [\Gamma]B_0 \vdash \Delta, \alpha : \kappa, \Theta\]

By above equality

\[\Omega, \alpha : \kappa \rightarrow \Omega', \alpha : \kappa, \Omega_R\]

By above equality

\[\Omega \rightarrow \Omega'\]

By Lemma 33 (Extension Transitivity)

\[\Gamma \vdash \exists \alpha : \kappa, [\Gamma]A_0 < : [\Gamma]B_0 \rightarrow \Delta\]

By \(\because \forall R\)

\[\Gamma \vdash [\Gamma][\exists \alpha : \kappa, A_0] < : [\Gamma]B_0 \rightarrow \Delta\]

By definition of substitution

- Case

\[\Psi \vdash \tau : \kappa \quad \Psi \vdash [\Omega]A \leq^+ [\tau/\beta]B_0\]

\[\Psi \vdash [\Omega]A \leq^+ [\exists \beta : \kappa, B_0]\]

\[\Psi \vdash [\exists \beta : \kappa, B_0] \leq R\]

We consider whether \([\Omega]A\) is headed by an existential.

If \([\Omega]A = \exists \alpha : \kappa', A'\):

\[\Omega [\Gamma, \alpha : \kappa \vdash A' \leq^+ [\Omega]B\]

By Lemma 5 (Subtyping Inversion)

The remaining steps are similar to the \(\leq^+ \Omega\) case.

If \([\Omega]A\) not headed by \(\exists:\)

\[\Omega [\Gamma] \vdash \tau : \kappa\]

Subderivation

\[\Gamma \rightarrow \Omega\]

Given

\[\Gamma, \beta : \kappa \rightarrow \Omega, \beta : \kappa\]

By \(\rightarrow \) Marker

\[\Gamma, \beta : \kappa, \beta : \kappa \rightarrow \Omega, \beta : \kappa, \beta : \kappa\]

By \(\rightarrow \) Solve

\[\Omega_0 \rightarrow \Omega_0\]

By definition of context application (lines 16, 13)
Proof of Theorem 9 (Completeness of Subtyping)

\[ [\Omega] \Gamma \vdash [\Omega] A \leq^+ [\tau/\beta][\Omega] B_0 \]

By definition of substitution

\[ \Omega_0 \vdash [\Omega] A \leq^+ [\tau/\beta][\Omega] B_0 \]

By above equality

\[ \Omega_0 \vdash [\Omega_0] A \leq^+ [\Omega_0][\tau/\beta] [\Omega_0] B_0 \]

By distributivity of substitution

\[ \Omega_0 \vdash [\Omega_0] A \leq^+ [\Omega_0][\tau/\beta] B_0 \]

By definition of substitution

\[ \Gamma, \vec{\alpha} : \kappa \vdash [\Gamma, \vec{\alpha}, \vec{\alpha} : \kappa] A <^+ [\Gamma, \vec{\alpha}, \vec{\alpha} : \kappa][\vec{\alpha}/\beta] B_0 \rightarrow \Delta_0 \]

By i.h. (B lost a quantifier)

\[ \Delta_0 \rightarrow \Omega'' \]

\[ \Omega_0 \rightarrow \Omega'' \]

\[ \Gamma, \vec{\alpha} : \kappa \vdash [\Gamma, \vec{\alpha}, \vec{\alpha} : \kappa] \rightarrow \Delta_0 \]

By definition of substitution

\[ \Gamma, \vec{\alpha} : \kappa \rightarrow \Delta_0 \]

By Lemma 50 (Subtyping Extension)

\[ \Gamma \rightarrow \Delta \]

By Definition of Substitution

\[ \Omega'' = (\Omega', \vec{\alpha}, \Omega_Z) \]

By Lemma 22 (Extension Inversion) (ii)

\[ \Delta \rightarrow \Omega' \]

By Definition of Substitution

\[ \Omega_0 \rightarrow \Omega'' \]

By above equalities

\[ \Omega, \vec{\alpha} : \kappa \rightarrow \Omega', \vec{\alpha}, \Omega_Z \]

By Definition of Substitution

\[ \Omega, \vec{\alpha} : \kappa \rightarrow \Omega' \]

By Lemma 22 (Extension Inversion) (ii)

\[ \Gamma, \vec{\alpha}, \vec{\alpha} : \kappa \vdash [\Gamma] A <^+ [\Gamma][\vec{\alpha}/\beta] B_0 \rightarrow \Delta, \vec{\alpha}, \Theta \]

By above equality \( \Delta_0 = (\Delta, \vec{\alpha}, \Theta) \)

\[ \Gamma, \vec{\alpha}, \vec{\alpha} : \kappa \vdash [\Gamma] A <^+ [\vec{\alpha}/\beta][\Gamma] B_0 \rightarrow \Delta, \vec{\alpha}, \Theta \]

By def. of subst. \((\Gamma|\vec{\alpha} = \vec{\alpha} \text{ and } \Gamma|\beta = \beta)\)

\[ [\Gamma] A \text{ not headed by } \exists \]

From the case hypothesis

\[ \Gamma \vdash [\Gamma] A <^+ [\Gamma](\exists \beta : \kappa. B_0) \rightarrow \Delta \]

By \( <^+ \exists \mathcal{R} \)

\[ \Gamma \vdash [\Gamma] A <^+ [\Gamma](\exists \beta : \kappa. B_0) \rightarrow \Delta \]

By Definition of Substitution

K'.3 Completeness of Typing

Theorem 10 (Completeness of Match Coverage).

1. If \([\Omega] \Gamma \vdash [\Omega] \Pi \text{ covers } [\Omega] \vec{A} \text{ and } \Gamma \rightarrow \Omega \text{ and } \Gamma \vdash \vec{A} \) 
   types and \([\tau/\beta][\Gamma] \vec{A} = \vec{A} \) 
   then \( \Gamma \vdash \Pi \text{ covers } \vec{A} \).

2. If \([\Omega] \Gamma / [\Omega] \Pi \vdash [\Omega] \Pi \text{ covers } [\Omega] \vec{A} \) and \( \Gamma \rightarrow \Omega \) 
   and \( \Gamma \vdash \vec{A} \) 
   types and \([\tau/\beta][\Gamma] \vec{A} = \vec{A} \) and \( [\Gamma] \Pi = \Pi \) 
   then \( \Gamma \vdash [\Pi] \Pi \vdash \Pi \text{ covers } \vec{A} \).

Proof. By mutual induction on the derivation of the given coverage rule.

1. Case \( [\Omega] \Gamma \vdash \cdots \Rightarrow e_1 \mid \ldots \) covers \( \cdots \)

   Apply CoversEmpty

   \[ \text{Cases} \text{ } \text{DeclCoversEmpty, DeclCoversVar, DeclCovers1, DeclCovers} \times, \text{DeclCovers} +, \text{DeclCovers} \exists, \text{DeclCovers} \wedge, \text{DeclCovers} \text{Vec:} \]

   Use the i.h. and apply the corresponding algorithmic coverage rule.

2. Case \( \theta = \mgu(t_1, t_2) \)

   \[ [\theta][\Omega] \Gamma \vdash [\theta][\Omega] \Pi \text{ covers } [\theta] \vec{A} \]

   \[ [\Omega] \Gamma / [\Omega] (t_1 = t_2) \vdash [\Omega] \Pi \text{ covers } [\theta] \vec{A} \]

   \[ \mgu(t_1, t_2) = \theta \]

   Premise

   \[ \Gamma \vdash t_1 \equiv t_2 : \kappa \vdash \Gamma, \Theta \]

   By Lemma 92 (Completeness of Elim1)

   \[ \Gamma \vdash [\Gamma] t_1 \equiv [\Gamma] t_2 : \kappa \vdash \Gamma, \Theta \]

   Follows from given assumption
Proof of Theorem 10 (Completeness of Match Coverage) 

Thm: coverage-completeness

By induction, using the measure in Definition 7.

Γ

(iii) If

Γ

(vi) If

Γ

Γ

(ii) If

Γ

Γ

then there exist

∆

Γ

γ

∆

Γ

then there exist

∆

Γ

then there exist

∆

Γ

By Lemma 92 (Completeness of Elimeq) (2)

Follows from given assumption

Γ

/ t_1 = t_2 \vdash \Pi \text{ covers } \vec{A}

By CoversEqBot

Case

mg(u(t_1, t_2) = \bot

\Omega, \Theta)(\Gamma) \vdash \Omega, \Theta \Pi \text{ covers } \{\Gamma, \Theta\vec{A}

By i.h.

\Rightarrow

\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \vec{A}

By CoversEq

• Case

mg(u(t_1, t_2) = \bot

\Omega, \Theta)(\Gamma) \vdash \Omega \Pi \text{ covers } \vec{A}

Premise

\Gamma / t_1 \equiv t_2 : \kappa \vdash \bot

By Lemma 93 (Substitution Upgrade) (iii)

\Gamma / \{\Gamma\} t_1 \equiv \{\Gamma\} t_2 : \kappa \vdash \bot

By above equalities

\Rightarrow

\Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \vec{A}

By CoversEqBot

Theorem 11 (Completeness of Algorithmic Typing). Given \Gamma \rightarrow \Omega such that dom(\Gamma) = dom(\Omega):

(i) If \Gamma \vdash A p type and \Omega, \Theta)(\Gamma) \vdash \Omega e \leftrightarrow \Omega A p and p' \subseteq p

then there exist \Delta and \Omega'

such that \Delta \rightarrow \Omega' and dom(\Delta) = dom(\Omega') and \Omega \rightarrow \Omega'

and \Gamma \vdash e \leftrightarrow \Gamma A p' \vdash \Delta.

(ii) If \Gamma \vdash A p type and \Omega, \Theta)(\Gamma) \vdash \Omega e \Rightarrow A p

then there exist \Delta, \Omega', A', and p' \subseteq p

such that \Delta \rightarrow \Omega' and dom(\Delta) = dom(\Omega') and \Omega \rightarrow \Omega'

and \Gamma \vdash e \Rightarrow A' p' \vdash \Delta and A' = [\Delta]A' and A = [\Omega']A'.

(iii) If \Gamma \vdash A p type and \Omega, \Theta)(\Gamma) \vdash \Omega s : \Omega A p \gg B q and p' \subseteq p

then there exist \Delta, \Omega', B', and q' \subseteq q

such that \Delta \rightarrow \Omega' and dom(\Delta) = dom(\Omega') and \Omega \rightarrow \Omega'

and \Gamma \vdash s : \{\Gamma\} A p' \gg B' q' \vdash \Delta and B' = [\Delta]B' and B = [\Omega']B'.

(iv) If \Gamma \vdash A p type and \Omega, \Theta)(\Gamma) \vdash \Omega s : \Omega A p \gg B [q] and p' \subseteq p

then there exist \Delta, \Omega', B', and q' \subseteq q

such that \Delta \rightarrow \Omega' and dom(\Delta) = dom(\Omega') and \Omega \rightarrow \Omega'

and \Gamma \vdash s : \{\Gamma\} A p' \gg B' [q'] \vdash \Delta and B' = [\Delta]B' and B = [\Omega']B'.

(v) If \Gamma \vdash \vec{A} ! p types and \Gamma \vdash C p type and \Omega, \Theta)(\Gamma) \vdash \Omega \vec{A} \leftrightarrow \Omega C p and p' \subseteq p

then there exist \Delta, \Omega', and C

such that \Delta \rightarrow \Omega' and dom(\Delta) = dom(\Omega') and \Omega \rightarrow \Omega'

and \Gamma \vdash \Pi : \{\Gamma\} \vec{A} \leftrightarrow \{\Gamma\} C p' \vdash \Delta.

(vi) If \Gamma \vdash \vec{A} ! p types and \Gamma \vdash P prop and FEV(P) = \emptyset and \Gamma \vdash C p type

and \Omega, \Theta)(\Gamma) \vdash \Omega \vec{A} \leftrightarrow \Omega C p

and p' \subseteq p

then there exist \Delta, \Omega', and C

such that \Delta \rightarrow \Omega' and dom(\Delta) = dom(\Omega') and \Omega \rightarrow \Omega'

and \Gamma / \{\Gamma\} P \vdash \Pi : \{\Gamma\} \vec{A} \leftrightarrow \{\Gamma\} C p' \vdash \Delta.

Proof. By induction, using the measure in Definition 7
**Proof of** [Theorem 11](#thm:typing-completeness) *(Completeness of Algorithmic Typing)*

- **Case** $(x : A, p) \in [\Omega] \Gamma$
  
  $[\Omega] \Gamma \vdash x \Rightarrow A, p$

  $\text{DeclVar}$

  $(x : A, p) \in [\Omega] \Gamma$

  Premise

  $\Gamma \rightarrow \Omega$

  Given

  $(x : A', p) \in \Gamma$ where $[\Omega] A' = A$

  From definition of context application

  Let $\Delta = \Gamma$.
  Let $\Omega' = \Omega$.

  $\text{Var}$

  $\Gamma \rightarrow \Omega$

  Given

  $\Omega' \rightarrow \Omega$

  By Lemma 32 *(Extension Reflexivity)*

  $\text{Var}$

  $\Gamma \vdash x \Rightarrow [\Gamma]' A', p \vdash \Gamma$

  By idempotence of substitution

  $\text{Var}$

  Let $\Delta = \Gamma$.

  $\Delta \rightarrow \Omega$

  Given

  $\Omega' \rightarrow \Omega$

  Given

  $[\Omega] [\Gamma]' A' = [\Omega] A'$

  By Lemma 32 *(Substitution Monotonicity)* (iii)

  $\text{Var}$

  $\Omega \vdash B \leq \pm [\Omega] A$

  By above equality

- **Case**

  $[\Omega] \Gamma \vdash [\Omega] e \Rightarrow B, q$

  $[\Omega] \Gamma \vdash B \leq \pm [\Omega] A$

  $[\Omega] \Gamma \vdash [\Omega] e \Rightarrow B, q \iff [\Omega] A$

  $\text{DeclSub}$

  $\text{DeclSub}$

  $\text{Var}$

  $\Gamma \vdash e \Rightarrow B', q \vdash \Theta$

  By i.h.

  $B = [\Omega] B'$

  $\text{Subderivation}$

  $\Theta \rightarrow \Omega_0$

  $\text{Subderivation}$

  $\Omega \rightarrow \Omega_0$

  $\text{Subderivation}$

  $\text{dom(}\Theta) = \text{dom}(\Omega_0)$

  $\text{Subderivation}$

  $\Gamma \rightarrow \Omega$

  Given

  $\Gamma \rightarrow \Omega_0$

  By Lemma 33 *(Extension Transitivity)*

  $[\Omega] \Gamma \vdash B \leq \pm [\Omega] A$

  Subderivation

  $[\Omega] \Gamma = [\Omega] \Theta$

  By Lemma 56 *(Confluence of Completeness)*

  $[\Omega] \Theta \vdash B \leq \pm [\Omega] A$

  By above equalities

  $\Theta \rightarrow \Omega_0$

  Above

  $\Theta \vdash B' < \pm A \vdash \Delta$

  By Theorem 9

  $\Omega_0 \rightarrow \Omega'$

  $\text{Subderivation}$

  $\text{dom(}\Delta) = \text{dom}(\Omega')$

  $\text{Subderivation}$

  $\Delta \rightarrow \Omega'$

  By Lemma 33 *(Extension Transitivity)*

  $\Delta \rightarrow \Omega'$

  By Lemma 33 *(Extension Transitivity)*

  $\Omega \rightarrow \Omega'$

  By Lemma 33 *(Extension Transitivity)*

  $[\Omega] \Gamma \vdash e \iff A, p \vdash \Delta$

  By **Sub**

- **Case**

  $[\Omega] \Gamma \vdash [\Omega] A$

  $[\Omega] \Gamma \vdash [\Omega] e_0 \iff [\Omega] A$

  $[\Omega] \Gamma \vdash [\Omega] (e_0 : A) \Rightarrow A$

  $\text{DeclAnno}$

  $\text{DeclAnno}$
Proof of **Theorem 11** (Completeness of Algorithmic Typing)

In the latter case, since $\text{dom}(\Delta) = \text{dom}(\Delta')$ $\Delta \rightarrow \Omega'$

$\Gamma \vdash A ! \text{ type}$

By Lemma 33 (Extension Transitivity)

$\Gamma \vdash (e_0 : A) \Rightarrow |\Delta|A \rightarrow \Delta$

By Anno

$[\Delta]A = [\Delta]|\Delta|A$

By idempotence of substitution

$A = [\Omega]A$

By Lemma 55 (Completing Completeness)  (ii)

$= [\Omega']A$

By Lemma 29 (Substitution Monotonicity)

$\Rightarrow e_0 = [\Gamma]A \rightarrow \Delta$

By i.h.

$\Rightarrow \Delta \rightarrow \Omega$

Subderivation

By Lemma 29 (Substitution Monotonicity)

$\Rightarrow |\Omega|\Gamma \vdash \emptyset \leftarrow p$

DeclI

We have $[\Omega]A = 1$. Either $|\Gamma|A = 1$, or $|\Gamma|A = \hat{\alpha}$ where $\hat{\alpha} \in \text{unsolved}(\Gamma)$.

In the former case:

Let $\Delta = \Gamma$.

Let $\Omega' = \Omega$.

$\Rightarrow \Gamma \rightarrow \Omega$

Given

$\Rightarrow \Omega \rightarrow \Omega'$

By Lemma 32 (Extension Reflexivity)

$\Rightarrow \text{dom}(\Gamma) = \text{dom}(\Omega)$

By Lemma 11 (Parallel Extension Solution)

$\Rightarrow \Gamma \vdash (p \rightarrow \Gamma)$

By def. of subst.

$\Rightarrow \Gamma \vdash (\text{Uvar})$

By Lemma 32 (Extension Reflexivity)

For case $\nu \text{chk-I}$

$\Gamma \vdash (\nu \alpha : \kappa)\rightarrow (\nu \alpha : \kappa)\vdash A_0 p$

By def. of subst. and predicativity of $\Omega$

$\Rightarrow \nu \alpha : \kappa \vdash A_0 p$

By def. of subst. and predicativity of $\Omega$

$\Rightarrow \nu \alpha : \kappa \vdash A_0 p$

By def. of subst. and predicativity of $\Omega$

Subderivation and above equality

$\Rightarrow \Gamma \rightarrow \Omega$

By Uvar

$\Rightarrow \Gamma \vdash (\nu \alpha : \kappa)\rightarrow (\nu \alpha : \kappa)$

By definition of context substitution

$\Rightarrow \nu \alpha : \kappa \vdash A_0 p$

By above equality

$\Rightarrow \nu \alpha : \kappa \vdash A_0 p$

By definition of substitution
Proof of Theorem 11 (Completeness of Algorithmic Typing)

Case

\[ \Gamma, \alpha : \kappa \vdash v \leftarrow [\Gamma, \alpha : \kappa]A' \vdash \Delta' \] By i.h.

\[ \Delta' \rightarrow \Omega_0' \]

\[ \Omega, \alpha : \kappa \rightarrow \Omega_0' \]

\[ \text{dom} (\Delta') = \text{dom} (\Omega_0') \]

\[ \Gamma, \alpha : \kappa \rightarrow \Delta' \] By Lemma 51 (Typing Extension)

\[ \Delta' = (\Delta, \alpha : \kappa, \Theta) \] By Lemma 22 (Extension Inversion) (i)

\[ \Delta, \alpha : \kappa, \Theta \rightarrow \Omega_0' \]

\[ \Omega_0' = (\Omega', \alpha : \kappa, \Omega_2) \] By Lemma 22 (Extension Inversion) (i)

\[ \Delta \rightarrow \Omega' \]

\[ \text{dom} (\Delta) = \text{dom} (\Omega') \] By Lemma 22 (Extension Inversion) on \( \Omega, \alpha : \kappa \rightarrow \Omega_0' \)

Case

\[[\Omega]\Gamma \vdash \tau : \kappa ([\Omega]e \text{ s}0 : [\tau/\alpha][\Omega]A_0 \gg B \quad \text{Decl/Solve}]\]

\[[\Omega]\Gamma \vdash [\Omega]e \text{ s}0 : [\tau/\alpha][\Omega]A_0 \gg B \quad \text{Subderivation} \]

\[\Gamma \rightarrow \Omega\]

\[\Gamma, \hat{\alpha} : \kappa \rightarrow \Omega, \hat{\alpha} : \kappa = \tau\]

\[[\Omega]\Gamma \vdash [\Omega][e \text{ s}0] : [\tau/\alpha][\Omega]A_0 \gg B \quad \text{Subderivation} \]

\[\tau = [\Omega]\tau\]

\[[\tau/\alpha][\Omega]A_0 = [\tau/\alpha][\Omega, \hat{\alpha} : \kappa = \tau]A_0\]

\[= [\Omega][\tau/\alpha][\Omega, \hat{\alpha} : \kappa = \tau]A_0\]

\[= [\Omega, \hat{\alpha} : \kappa = \tau] \hat{\alpha}/\alpha A_0\]

\[= [\Omega, \hat{\alpha} : \kappa = \tau]((\Gamma, \hat{\alpha} : \kappa)\text{ by definition of context application}) \]

\[[\Omega, \hat{\alpha} : \kappa = \tau]((\Gamma, \hat{\alpha} : \kappa)\text{ by above equalities}) \]

\[\Gamma, \hat{\alpha} : \kappa \vdash e \text{ s}0 : [\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 \gg B \quad \text{By above equalities} \]

\[\Gamma, \hat{\alpha} : \kappa \vdash B : \Delta' \quad \text{By i.h.} \]

\[B = [\Omega, \hat{\alpha} : \kappa = \tau]B' \]

\[\Delta \rightarrow \Omega' \]

\[\text{dom} (\Delta) = \text{dom} (\Omega') \]

\[\Omega \rightarrow \Omega' \]

\[B' \rightarrow \Omega' \]

\[B \rightarrow \Omega' B' \]

\[[\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 \rightarrow [\Gamma][\hat{\alpha}/\alpha]A_0 \quad \text{By def. of context application} \]

\[[\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 \rightarrow \hat{\alpha}/\alpha A_0 \quad \text{By i.h.} \]

\[\Gamma, \hat{\alpha} : \kappa \vdash B : \Delta' \quad \text{By def. of context application} \]

\[\Gamma, \hat{\alpha} : \kappa \vdash e \text{ s}0 : [\hat{\alpha}/\alpha][\Gamma]A_0 \gg B' \quad \text{By above equalities} \]

\[\Gamma \vdash \Gamma e \text{ s}0 : [\forall \alpha : \kappa. \Gamma]A_0 \gg B' \quad \text{By decl/spine} \]

\[\Gamma \vdash \Gamma e \text{ s}0 : [\forall \alpha : \kappa. \Gamma]A_0 \gg B' \quad \text{By def. of subst.} \]

\[\nu \text{ chk-I} \quad [\Omega]\Gamma \vdash [\Omega]P \vdash [\Omega]v \leftarrow [\Omega]A_0 \quad \text{Decl/L} \]

\[[\Omega]\Gamma \vdash [\Omega]v \leftarrow ([\Omega]P) \gg [\Omega]A_0 \quad \text{Decl/L} \]

\[[\Omega]\Gamma \vdash [\Omega]v \leftarrow [\Omega]A_0 \quad \text{Subderivation} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

The concluding rule in this subderivation must be \texttt{DeclCheck} or \texttt{DeclCheckUnify}. In either case, \([\Omega]\!P\) has the form \(\sigma' = \tau'\) where \(\sigma' = [\Omega]!\sigma\) and \(\tau' = [\Omega]!\tau\).

\begin{itemize}
  \item Case
    \begin{align*}
    \text{mgu}([\Omega]\!\sigma, [\Omega]\!\tau) &= \bot \\
    [\Omega]\!\Gamma / [\Omega]\!([\sigma = \tau]) &\vdash [\Omega]\!\nu \leftarrow [\Omega]\!A_{\Theta} ! \texttt{DeclCheck}
    \end{align*}

    We have \(\text{mgu}([\Omega]\!\sigma, [\Omega]\!\tau) = \bot\). To apply Lemma 92 (Completeness of ElimEq) (2), we need to show conditions 1–5.

    ***
    \[
    \begin{array}{ll}
    \Gamma \vdash (\sigma = \tau) \supset A_{\Theta} ! \text{type} & \text{Given} \\
    [\Omega](\sigma = \tau) \supset A_{\Theta} & \text{By Lemma 39 (Principal Agreement) (i)} \\
    [\Omega]\!\sigma = [\Gamma]\!\sigma & \text{By a property of subst.} \\
    [\Omega]\!\tau = [\Gamma]\!\tau & \text{Similar} \\
    \Gamma \vdash \sigma : \kappa & \text{By inversion} \\
    \Gamma \vdash [\Gamma]\!\sigma : \kappa & \text{By Lemma 11 (Right-Hand Substitution for Sorting)} \\
    \Gamma \vdash [\Gamma]\!\tau : \kappa & \text{Similar}
    \end{array}
    \]

    Given

    \begin{align*}
    \text{mgu}([\Omega]\!\sigma, [\Omega]\!\tau) &= \bot \\
    \text{mgu}([\Gamma]\!\sigma, [\Gamma]\!\tau) &= \bot & \text{By above equalities} \\
    \text{FEV}(\sigma) \cup \text{FEV}(\tau) &= \emptyset & \text{By inversion on ***} \\
    \text{FEV}(\sigma) \cup \text{FEV}(\tau) &= \emptyset & \text{By a property of complete contexts} \\
    \text{FEV}(\sigma) \cup \text{FEV}(\tau) &= \emptyset & \text{By above equalities} \\
    \text{FEV}(\sigma) \cup \text{FEV}(\tau) &= \emptyset & \text{By above equalities}
    \end{align*}

    \begin{align*}
    \Gamma / [\Gamma]\!\sigma &\equiv [\Gamma]\!\tau : \kappa \setminus \bot & \text{By Lemma 92 (Completeness of ElimEq) (2)} \\
    \Gamma, p &\vdash [\Gamma]\!\tau : \kappa \setminus \bot & \text{By \texttt{ElimpropEq}} \\
    \Gamma \vdash \nu \leftarrow ([\Gamma]\!\sigma = [\Gamma]\!\tau) \supset [\Gamma]\!A_{\Theta} ! \vdash \Gamma & \text{By \texttt{DeclCheck}} \\
    \Gamma \vdash \nu \leftarrow ([\Gamma]\!\sigma = [\Gamma]\!\tau) \supset [\Gamma]\!A_{\Theta} ! \vdash \Gamma & \text{By def. of subst.}
    \end{align*}

    Given

    \begin{align*}
    \Gamma \vdash \Omega & & \text{By Lemma 32 (Extension Reflexivity)} \\
    \Gamma \vdash \Omega & & \text{By \texttt{DeclCheckUnify}} \\
    \end{align*}

    \begin{align*}
    \text{dom}(\Gamma) &= \text{dom}(\Omega) & \text{By \texttt{DeclCheck}} \\
    \end{align*}

    \begin{itemize}
    \item Case
      \begin{align*}
      \text{mgu}([\Omega]\!\sigma, [\Omega]\!\tau) &= \theta \\
      \theta([\Omega]\!\Gamma) &\vdash \theta([\Omega]\!\nu) \leftarrow \theta([\Omega]\!A_{\Theta} ! \\
      [\Omega]\!\Gamma / ([\Omega]\!\sigma) &= [\Omega]\!\tau & \text{By \texttt{DeclCheckUnify}}
      \end{align*}

      We have \(\text{mgu}([\Omega]\!\sigma, [\Omega]\!\tau) = \theta\), and will need to apply Lemma 92 (Completeness of ElimEq) (1). That lemma has five side conditions, which can be shown exactly as in the \texttt{DeclCheck} case above.

      \begin{align*}
      \text{mgu}(\sigma, \tau) &= \theta & \text{Premise} \\
      \text{Let } \Omega_{\Theta} &= \{\Gamma, p\}. & \text{Given} \\
      \Gamma &\vdash \Omega & \text{Given} \\
      \Gamma, p &\vdash \Omega_{\Theta} & \text{By \texttt{DeclCheckUnify}} \\
      \text{dom}(\Gamma) &= \text{dom}(\Omega) & \text{Given} \\
      \text{dom}(\Gamma, p) &= \text{dom}(\Omega_{\Theta}) & \text{By def. of dom(\_)}
      \end{align*}

      \begin{align*}
      \Gamma, p &\vdash [\Gamma]\!\sigma \equiv [\Gamma]\!\tau : \kappa \vdash \Gamma, p, \Theta & \text{By Lemma 92 (Completeness of ElimEq) (1)} \\
      \Gamma, p &\vdash [\Gamma]\!\nu \leftarrow ([\Gamma]\!\sigma = [\Gamma]\!\tau) \vdash \Gamma, p, \Theta & \text{By \texttt{ElimpropEq}} \\
      \end{align*}

    EQO for all \(\Gamma, p \vdash u : \kappa\). [\Gamma, p, \Theta]u = \theta([\Gamma, p]u)
  \end{itemize}
Proof of Theorem 11: Completeness of Algorithmic Typing

\[ \Gamma \vdash P \gg A_0 \quad \text{type} \]
\[ \Gamma \vdash A_0 \quad \text{type} \]
\[ \Gamma \longrightarrow \Omega \quad \text{Given} \]

\[ \Omega \vdash \Theta \]

\[ \text{Let } \Omega_1 = (\Omega_1, \triangleright p, \Theta) \]
\[ \theta(\Omega_1) \vdash \theta(e) \iff \theta([\Omega]A_0) ! \quad \text{Subderivation} \]

\[ \Gamma, \triangleright p, \Theta \longrightarrow \Omega_1 \quad \text{By induction on } \Theta \]

\[ \theta([\Omega]A_0) = \theta([\Gamma]A_0) \quad \text{By above equality EQa} \]

\[ = [\Gamma, \triangleright p, \Theta]A_0 \quad \text{By Lemma 93 (Substitution Upgrade) (i) (with EQ0)} \]

\[ = [\Omega_1]A_0 \quad \text{By Lemma 39 (Principal Agreement) (i)} \]

\[ [\Omega_1]A_0 \]

\[ \theta([\Omega]e) = [\Omega]e \quad \text{By Lemma 93 (Substitution Upgrade) (iv)} \]

\[ [\Omega_1](\Gamma, \triangleright p, \Theta) \vdash [\Omega_1]e \iff [\Omega_1][\Gamma, \triangleright p, \Theta]A_0 ! \quad \text{By above equalities} \]

\[ \text{dom}(\Gamma, \triangleright p, \Theta) = \text{dom}(\Omega_1) \]
\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \]

\[ \Gamma, \triangleright p, \Theta \vdash e \iff [\Gamma, \triangleright p, \Theta]A_0 ! \vdash \Delta' \quad \text{By i.h.} \]
\[ \Delta' \longrightarrow \Omega' \quad " \]
\[ \Omega_1 \longrightarrow \Omega_2 \quad " \]

\[ \text{dom}(\Delta') = \text{dom}(\Omega_1) \quad " \]
\[ \Delta' = (\Delta, \triangleright p, \Delta'') \quad \text{By Lemma 22 (Extension Inversion) (ii)} \]
\[ \Omega'' = (\Omega', \triangleright p, \Omega_Z) \quad \text{By Lemma 22 (Extension Inversion) (ii)} \]

\[ \Delta \longrightarrow \Omega' \quad " \]

\[ \Omega_0 \longrightarrow \Omega_2 \quad \text{By Lemma 33 (Extension Transitivity)} \]
\[ \Omega_0, \triangleright p \longrightarrow \Omega', \triangleright p, \Omega_Z \quad \text{By above equalities} \]
\[ \text{dom}(\Delta) = \text{dom}(\Omega') \quad " \]

\[ \Gamma, \triangleright p, \Theta \vdash e \iff [\Gamma, \triangleright p, \Theta]A_0 ! \vdash \Delta, \triangleright p, \Delta'' \quad \text{By above equality} \]
\[ \Gamma \vdash e \iff ([\Gamma]\sigma = [\Gamma]\tau) \supset [\Gamma]A_0 ! \vdash \Delta \quad \text{By } \square \]
\[ \Gamma \vdash e \iff [\Gamma](P \supset A_0) ! \vdash \Delta \quad \text{By def. of subst.} \]

- **Case**
  \[ [\Omega]\Gamma \vdash [\Omega]P \text{ true} \]
  \[ [\Omega]\Gamma \vdash [\Omega](e \ s_0) : [\Omega]A_0 \gg B \quad \text{Decl} \supset \text{Spine} \]

\[ [\Omega]\Gamma \vdash [\Omega](e \ s_0) : ([\Omega]P) \supset [\Omega]A_0 \gg B \quad \text{Decl} \supset \text{Spine} \]

\[ [\Omega]\Gamma \vdash [\Omega]P \text{ true} \quad \text{Subderivation} \]
\[ [\Omega]\Gamma \vdash [\Omega]P \text{ true} \quad \text{By Lemma 29 (Substitution Monotonicity) (ii)} \]
\[ [\Gamma]P \text{ true } \vdash \Theta \quad \text{By Lemma 95 (Completeness of Checkprop)} \]
\[ \Theta \longrightarrow \Omega_1 \quad " \]
\[ \Omega \longrightarrow \Omega_1 \quad " \]

\[ \text{dom}(\Theta) = \text{dom}(\Omega_1) \quad " \]

\[ \Gamma \longrightarrow \Omega \quad \text{Given} \]
\[ [\Omega]\Gamma = [\Omega_1]\Theta \quad \text{By Lemma 57 (Multiple Confluence)} \]
\[ [\Omega]A_0 = [\Omega_1]A_0 \quad \text{By Lemma 55 (Completing Completeness) (ii)} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ [\Omega] [\Gamma] \vdash [\Omega] (e \ s_0) : [\Omega] A_0 \ p \gg B \ q \]  
Subderivation

\[ [\Omega_1] [\Theta] \vdash [\Omega_1] (e \ s_0) : [\Omega_1] A_0 \ p \gg B \ q \]  
By above equalities

\[ \Theta \vdash e \ s_0 : (\Theta) A_0 \ p \gg B' \ q \ + \Delta \]  
By i.h.

\[ B' = |\Delta|B' \]  
"

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]  
"

\[ B = |\Omega'|B' \]  
"

\[ \Delta \longrightarrow \Omega' \]  
"

\[ \Omega \longrightarrow \Omega' \]  
By Lemma 29 (Substitution Monotonicity) (iii)

\[ (\Theta) A_0 = (\Theta) [\Gamma] A_0 \]  
By Lemma 33 (Extension Transitivity)

\[ \Theta \vdash e \ s_0 : (\Theta) [\Gamma] A_0 \ p \gg B' \ q \ + \Delta \]  
By above equality

\[ \Gamma \vdash e \ s_0 : ([\Gamma] P) \supset (\Gamma) A_0 \ p \gg B' \ q \ + \Delta \]  
By Spine

\[ \Gamma \vdash e \ s_0 : ([\Gamma] (P \supset A_0)) \ p \gg B' \ q \ + \Delta \]  
By def. of subst.

Case

\[ [\Omega] [\Gamma] \vdash [\Omega] \text{e} \ s_0 \leq A_{k'} \ p \]  

\[ [\Omega] [\Gamma] \vdash \text{inj}_k [\Omega] \text{e} \ s_0 \leq A_{k'} + A_{k'} \ p \]  
Decl + 1

Either \([\Gamma] A = A_1 + A_2\) (where \([\Omega] A_k = A_{k'}\)) or \([\Gamma] A = \alpha \in \text{unsolved}(\Gamma)\).

In the former case:

\[ [\Omega] [\Gamma] \vdash [\Omega] \text{e} \ s_0 \leq A_{k'} \ p \]  
Subderivation

\[ [\Omega] [\Gamma] \vdash [\Omega] \text{e} \ s_0 \leq [\Omega] A_k \ p \]  
\([\Omega] A_k = A_{k'}\)

\[ \Gamma \vdash e \ s_0 \leq [\Gamma] A_k \ p \ + \Delta \]  
By i.h.

\[ \Delta \longrightarrow \Omega' \]  
"

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]  
"

\[ \Omega \longrightarrow \Omega' \]  
"

\[ \Gamma \vdash \text{inj}_k e \ s_0 \leq ([\Gamma] A_1) + ([\Gamma] A_2) \ p \ + \Delta \]  
By + 1

\[ \Gamma \vdash \text{inj}_k e \ s_0 \leq [\Gamma] (A_1 + A_2) \ p \ + \Delta \]  
By def. of subst.

In the latter case, \(A = \alpha\) and \([\Omega] A = [\Omega] \alpha = A_{k'} + A_{k'} = \tau_{k'} + \tau_{k'}\).

By inversion on \(\Gamma \vdash \alpha \ p \text{ type}\), it must be the case that \(p = f\).

\(\Gamma \longrightarrow \Omega\)  
Given

\(\Gamma = \Gamma_0[\alpha : \star]\)  
\(\alpha \in \text{unsolved}(\Gamma)\)

\(\Omega = \Omega_0[\\\alpha : * = \tau_{0}]\)  
By Lemma 22 (Extension Inversion) (vi)

Let \(\Omega_2 = \Omega_0[\\\alpha_{1} : * = \tau_{1}, \ \alpha_{1} : * = \tau_{2}, \ \alpha_{1 : * = \alpha_{1}} + \alpha_{2}]\).

Let \(\Gamma_2 = \Gamma_0[\\\alpha_{1} : * = \tau_{1}, \ \alpha_{2} : * = \alpha_{1} + \alpha_{2}]\).

\(\Gamma \longrightarrow \Gamma_2\)  
By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (ii)

\(\Omega \longrightarrow \Omega_2\)  
By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (iii)

\(\Gamma_2 \longrightarrow \Omega_2\)  
By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)
Proof of Theorem 11 (Completeness of Algorithmic Typing)

In the former case:

\[\dom(\Delta) = \dom(\Omega')\]

By Lemma 33 (Extension Transitivity)

\[\Omega \rightarrow \Omega'\]

By Lemma 33 (Extension Transitivity)

\[\Gamma \vdash \text{inj}_k e_0 \Rightarrow \hat{\alpha} \vdash \Delta\]

By i.h.

\[\Gamma \vdash \text{inj}_k e \Rightarrow [\Gamma]\hat{\alpha} \vdash \Delta\]

\[\hat{\alpha} \in \text{unsolved}(\Gamma)\]

\[\text{Case } [\Omega]\Gamma, x : A_1' \vdash [\Omega]e_0 \Leftarrow A_2' p\]

We have \([\Omega]\Lambda = A_1' \rightarrow A_2'.\) Either \([\Gamma] \Lambda = A_1 \rightarrow A_2\) where \(A_1' = [\Omega]A_1\) and \(A_2' = [\Omega]A_2\) — or \([\Gamma] \Lambda = \hat{\alpha}\) and \([\Omega] \hat{\alpha} = A_1' \rightarrow A_2'.\)

In the former case:

\[\Omega, x : A_1' \vdash [\Omega]e_0 \Leftarrow A_2' p\]

Subderivation

\[A_1' = [\Omega]A_1\]

Known in this subcase

\[A_2' = [\Omega]\Omega\Gamma A_1\]

By Lemma 30 (Substitution Invariance)

\[A_2' = [\Omega]\Omega\Gamma A_1\]

Applying \(\Omega\) on both sides

\[A_2' = [\Omega]\Omega\Gamma A_1\]

By idempotence of substitution

\[\Omega, x : A_1' \vdash [\Gamma]A_1 p \vdash [\Omega]e_0 \Leftarrow A_2' p\]

By definition of context application

\[\Omega, x : A_1' \vdash [\Gamma]A_1 p \vdash [\Omega]e_0 \Leftarrow A_2' p\]

By above equality

\[\Gamma \vdash \Omega\]

By \(\Gamma \vdash \Omega\)

\[\Omega, x : A_1' \vdash \Omega, x : A_1' p\]

Given

\[\text{dom}(\Gamma) = \text{dom}(\Omega)\]

By def. of \(\text{dom}(\cdot)\)

\[\Gamma, x : [\Gamma]A_1 p + e_0 \Leftarrow A_2 p \vdash \Delta'\]

By i.h.

\[\Delta' \vdash \Omega_0'\]

By \(\Delta' \vdash \Omega_0'\)

\[\Omega_0' = (\Omega', x : A_1' p, \Omega_Z)\]

By Lemma 22 (Extension Inversion) (v)

\[\Omega \rightarrow \Omega'\]

By \(\Omega \rightarrow \Omega'\)

\[\Gamma, x : [\Gamma]A_1 p \vdash \Delta'\]

By Lemma 51 (Typing Extension)

\[\Delta' = (\Delta, x : \cdots, \Theta)\]

By Lemma 22 (Extension Inversion) (v)

\[\Delta, x : \cdots, \Theta \rightarrow \Omega', x : A_1' p, \Omega_Z\]

By above equalities

\[\Delta \rightarrow \Omega'\]

By Lemma 22 (Extension Inversion) (v)

\[\text{dom}(\Delta) = \text{dom}(\Omega')\]

By \(\text{dom}(\Delta) = \text{dom}(\Omega')\)

\[\Gamma, x : [\Gamma]A_1 p \vdash e_0 \Leftarrow [\Gamma]'Z p + \Delta, x : \cdots, p, \Theta\]

By above equality

\[\Gamma \vdash \lambda x. e_0 \Leftarrow ([\Gamma]A_1) \rightarrow ([\Gamma]A_2) p \vdash \Delta\]

By \(\Gamma \vdash \lambda x. e_0 \Leftarrow ([\Gamma]A_1) \rightarrow ([\Gamma]A_2) p \vdash \Delta\)

By definition of substitution

In the latter case \(([\Gamma] \Lambda = \hat{\alpha} \in \text{unsolved}(\Gamma)\) and \([\Omega] \hat{\alpha} = A_1' \rightarrow A_2' = \tau_1' \rightarrow \tau_2'):\]
By inversion on $\Gamma \vdash \& p$ type, it must be the case that $p = \lambda x. e$.

Since $\& \in \text{unsolved}(\Gamma)$, the context $\Gamma$ must have the form $\Gamma_0[\& : *]$.

Let $\Gamma_2 = \Gamma_0[\&_1 : *, \&_2 : *, \& : * = \&_1 \rightarrow \&_2]$.  

$\Gamma \rightarrow \Gamma_2$  

By Lemma 23 (Deep Evar Introduction) (iii) twice  
and Lemma 26 (Parallel Admissibility) (ii)

$[\Omega]\& = \tau'_1 \rightarrow \tau'_2$  
Known in this subcase

$\Gamma \rightarrow \Omega$  

Given

$\Omega = \Omega_0[\& : * = \tau_0]$  
By Lemma 22 (Extension Inversion) (vi)

Let $\Omega_2 = \Omega_0[\&_1 : * = \tau'_1, \&_2 : * = \tau'_2, \& : * = \&_1 \rightarrow \&_2]$.  

$\Gamma \rightarrow \Gamma_2$  

By Lemma 23 (Deep Evar Introduction) (iii) twice  
and Lemma 26 (Parallel Admissibility) (ii)

$\Omega \rightarrow \Omega_2$  

By Lemma 23 (Deep Evar Introduction) (iii) twice  
and Lemma 26 (Parallel Admissibility) (iii)

$\Gamma_2 \rightarrow \Omega_2$  

By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)

$[\Omega]\Gamma, x : \tau'_1 \vdash [\Omega]e_0 \leftarrow \tau'_2 \gamma$  
Subderivation

$[\Omega]\Gamma = [\Omega_2]\Gamma_2$  
By Lemma 57 (Multiple Confluence)

$\tau'_2 = [\Omega]\&_2$  
From above equality

$= [\Omega_2][\&_2]$  
By Lemma 55 (Completing Completeness) (i)

$\tau'_1 = [\Omega_2][\&_1]$  
Similar

$[\Omega_2]\Gamma_2, x : \tau'_1 \vdash [\Omega_2, x : \&_1] \gamma$  
By def. of context application

$[\Omega_2, x : \tau'_1 \gamma]([\Gamma_2, x : \&_1] \gamma) \vdash [\Omega_2]e_0 \leftarrow [\Omega_2][\&_2] \gamma$  
By above equalities

$\text{dom}(\Gamma) = \text{dom}(\Omega)$  
Given

$\text{dom}(\Gamma_2, x : \&_1 \gamma) = \text{dom}(\Omega_2, x : \tau'_1 \gamma)$  
By def. of $\Gamma_2$ and $\Omega_2$

$\Gamma_2, x : \&_1 \gamma \vdash e_0 \leftarrow [\Gamma_2, x : \&_1] \gamma \vdash \Delta^+$  
By i.h.

$\Delta^+ \rightarrow \Omega^+$  
By def. of $\Delta^+$

$\text{dom}(\Delta^+) = \text{dom}(\Omega^+)$  
""""""""""""

$\Omega_2 \rightarrow \Omega^+$  
By def. of $\Delta$ and $\Omega$

$\Delta \rightarrow \Omega'$  
By Lemma 51 (Typing Extension)

$\Omega' = (\Omega', x : \Delta, \Omega_Z)$  
By Lemma 22 (Extension Inversion) (v)

$\Omega \rightarrow \Omega_2$  
Above

$\Omega \rightarrow \Omega^+$  
By Lemma 33 (Extension Transitivity)

$\Omega \rightarrow \Omega'$  
By Lemma 22 (Extension Inversion) (v)

$\Gamma \vdash \lambda x. e_0 \leftarrow \& \gamma \vdash \Delta$  
By $\lambda I\&$

$\& = [\Gamma] \&$  
$\& \in \text{unsolved}(\Gamma)$

$\Gamma \vdash \lambda x. e_0 \leftarrow [\Gamma] \& \gamma \vdash \Delta$  
By above equality
Proof of [Theorem 11](Completeness of Algorithmic Typing)

### Case 1

\[ \Gamma \vdash \text{rec } x. \; [\Omega] v \iff [\Omega] A \ p \]

- **Case** \( [\Omega] \Gamma, x : [\Omega] A \ p \vdash [\Omega] v \iff [\Omega] A \ p \)

  - Subderivation

\[ [\Omega] \Gamma, x : [\Omega] A \ p = [\Omega, x : [\Omega] A \ p](\Gamma, x : [\Gamma] A \ p) \]

- By definition of context application

\[ [\Omega, x : [\Omega] A \ p](\Gamma, x : [\Gamma] A \ p) \vdash [\Omega] v \iff [\Omega] A \ p \]

- By above equality

\[ \Gamma \rightarrow \Omega \]

- Given

\[ \Gamma, x : [\Gamma] A \ p \rightarrow \Omega, x : [\Omega] A \ p \]

- By \( \vartriangleright \) \( \text{Var} \)

\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \]

- Given

\[ \text{dom}(\Gamma, x : [\Gamma] A \ p) = \Omega, x : [\Omega] A \ p \]

- By def. of \( \text{dom}(\cdot) \)

\[ \Gamma, x : [\Gamma] A \ p \vdash v \iff [\Gamma] A \ p \vdash \Delta' \]

- By i.h.

\[ \Delta' \rightarrow \Omega_0' \]

-"

\[ \text{dom}(\Delta') = \text{dom}(\Omega_0') \]

-"

\[ \Omega, x : [\Omega] A \ p \rightarrow \Omega_0' \]

-"

\[ \Omega_0' = (\Omega', x : [\Omega] A \ p, \Theta) \]

- By Lemma 22 (Extension Inversion) (v)

\[ \Gamma, x : [\Gamma] A \ p \rightarrow \Delta' \]

- By Lemma 51 (Typing Extension)

\[ \Delta' = (\Delta, x : \cdots, \Theta) \]

- By Lemma 22 (Extension Inversion) (v)

\[ \Delta, x : \cdots, \Theta \rightarrow \Omega', x : [\Omega] A \ p, \Theta \]

- By above equalities

\[ \Delta \rightarrow \Omega' \]

- By Lemma 22 (Extension Inversion) (v)

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]

-"

\[ \Gamma, x : [\Gamma] A \ p \vdash v \iff [\Gamma] A \ p \vdash \Delta, x : [\Gamma] A \ p, \Theta \]

- By above equality

\[ \Gamma \vdash \text{rec } x. v \iff [\Gamma] A \ p \vdash \Delta \]

- By \( \text{Rec} \)

### Case 2

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \Rightarrow A \ q \]

- Subderivation

\[ [\Omega] \Gamma \vdash [\Omega] s_0 : A \Rightarrow C \ [p] \]

- By definition of context application

\[ [\Omega] \Gamma \vdash [\Omega] (e_0 \ s_0) \Rightarrow C \ p \]

- Decl \( \rightarrow \) \( E \)

\[ \Gamma \vdash e_0 \Rightarrow A \ q \]

- By i.h.

\[ \Theta \rightarrow \Omega_{\Theta} \]

-"

\[ \text{dom}(\Theta) = \text{dom}(\Omega_{\Theta}) \]

-"

\[ \Omega \rightarrow \Omega_{\Theta} \]

-"

\[ A = [\Omega_{\Theta}] A' \]

-"

\[ A' = [\Theta] A' \]

-"
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ \Gamma \rightarrow \Omega \]

Given

\[ [\Omega] \Gamma = [\Omega_0] \Theta \]

By Lemma 57 (Multiple Confluence)

\[ [\Omega] \Gamma \vdash [\Omega] s_0 : A \triangleright C [p] \]

Subderivation

\[ [\Omega_0] \Theta \vdash [\Omega] s_0 : [\Omega_0] A' \triangleright C [p] \]

By above equalities

\[ \Theta \vdash s_0 : [\Theta] A' \triangleright C' [p] \vdash \Delta \]

By i.h.

\[ C' = [\Delta] C' \]

""

\[ \Delta \rightarrow \Omega' \]

""

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]

""

\[ \Omega_0 \rightarrow \Omega' \]

""

\[ C = [\Omega'] C' \]

By above equality

\[ \Theta \vdash s_0 : A' \triangleright C' [p] \vdash \Delta \]

By Lemma 33 (Extension Transitivity)

\[ \Omega \rightarrow \Omega' \]

""

\[ \Gamma \vdash e_0 s_0 \Rightarrow C' p \vdash \Delta \]

By \( \rightarrow E \)
Case: for all $C_2$.

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A ! \Rightarrow C \quad \text{if} \quad [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A ! \Rightarrow C_2 \ \text{then} \ C_2 = [\Omega] \]

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A ! \Rightarrow C \quad \text{if} \quad [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A ! \Rightarrow C_2 \]

\[ \Gamma \rightarrow \Omega \quad \text{Given} \]
\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A ! \Rightarrow C \quad \text{Subderivation} \]
\[ \Gamma \vdash s : [\Gamma] A ! \Rightarrow C' \ \text{if} \Delta \quad \text{By i.h.} \]
\[ \Delta \rightarrow \Omega' \quad \]
\[ \Omega \rightarrow \Omega' \quad \]
\[ \text{dom}(\Delta) = \text{dom}(\Omega') \quad \]
\[ \text{C} = [\Omega'] C' \quad \]
\[ \text{C}' = [\Delta] C' \quad \]

Suppose, for a contradiction, that $\text{FEV}(\Delta C') \neq \emptyset$.
That is, there exists some $\hat{\beta} \in \text{FEV}(\Delta C')$.

\[ \Delta \rightarrow \Omega_2 \quad \text{By Lemma 60 (Split Solutions)} \]
\[ \Omega_2 \delta \quad \text{By construction of } \Omega \text{ canonical} \]

Choose $\hat{\alpha}_R$ such that $\hat{\alpha}_R \in \text{FEV}(C')$ and either $\hat{\alpha}_R = \hat{\alpha}$ or $\hat{\alpha}_R \in \text{FEV}(\Delta \hat{\alpha}_R)$.
Then either $\hat{\alpha}_R = \hat{\alpha}$, or $\hat{\alpha}_R$ is declared to the right of $\hat{\alpha}$ in $\Delta$.

\[ [\Omega_2] C' \neq [\Omega'] C' \quad \text{From (NEQ) and (EQ)} \]
\[ \Gamma \vdash s : [\Gamma] A ! \Rightarrow C' \ \text{if} \Delta \quad \text{By Theorem 8} \]
\[ \Gamma \vdash s : [\Gamma] A ! \Rightarrow C' \ \text{if} \Delta \quad \text{By Lemma 13 (Right-Hand Substitution for Typing)} \]

\[ \Delta = ([\Delta_L] * [\Delta_R]) \quad \text{Property of } \subseteq \]
\[ (\Gamma \# \cdot) \rightarrow (\Delta_L * [\Delta_R]) \quad \text{By Lemma 72 (Separation—Main) (Spines)} \]
\[ \text{FEV}(C') \subseteq \text{dom}(\Delta_R) \quad \text{By Definition 5} \]
\[ \hat{\alpha}_R \in \text{FEV}(C') \quad \text{Above} \]
\[ \hat{\alpha}_R \in \text{dom}(\Delta_R) \quad \text{Property of } \subseteq \]
\[ \text{dom}(\Delta_L) \cap \text{dom}(\Delta_R) = \emptyset \quad \Delta \text{ well-formed} \]
\[ \hat{\alpha}_R \notin \text{dom}(\Delta_L) \quad \text{Above} \]
\[ \text{dom}(\Gamma) \subseteq \text{dom}(\Delta_L) \quad \text{Property of } \subseteq \]
\[ \hat{\alpha}_R \notin \text{dom}(\Gamma) \quad \text{By Definition 5} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

\[ \Omega_2 \vdash [\Omega_2] s : [\Omega_2] [\Gamma] A \mapsto [\Omega_2] C' \not \in \Gamma \] Above
\[ \Omega_2 \text{ and } \Omega_1 \text{ differ only at } \alpha \] Above
\[ \text{FEV}([\Gamma] A) = \emptyset \] Above
\[ [\Omega_2] [\Gamma] A = [\Omega_1][\Gamma] A \] By preceding two lines
\[ \Gamma \vdash [\Gamma] A \text{ type} \quad \text{By Lemma } 33 \text{ (Extension Transitivity)} \]
\[ \Gamma \mapsto \Omega_2 \] \[ \Omega_2 \vdash [\Gamma] A \text{ type} \quad \text{By Lemma } 38 \text{ (Extension Weakening (Types))} \]
\[ \text{dom}(\Omega_2) = \text{dom}(\Omega_1) \] \[ \Omega_1 \text{ and } \Omega_2 \text{ differ only at } \alpha \] \[ \Omega_1 \vdash [\Gamma] A \text{ type} \quad \text{By Lemma } 18 \text{ (Equal Domains)} \]
\[ \Gamma \vdash [\Gamma] A \text{ type} \quad \text{By Lemma } 38 \text{ (Extension Weakening (Types))} \]
\[ \Omega \vdash [\Gamma] A \text{ type} \quad \text{By Lemma } 55 \text{ (Completing Completeness) (ii) twice} \]
\[ [\Omega_1][\Gamma] A = [\Omega_1][\Gamma] A = [\Omega][\Gamma] A \] \[ \text{By Lemma } 29 \text{ (Substitution Monotonicity) (iii)} \]
\[ = [\Omega] A \] \[ \text{By Lemma } 38 \text{ (Extension Weakening (Types))} \]
\[ \text{FEV}([\Delta] C') = \emptyset \] \[ \text{By contradiction} \]
\[ s \vdash [\Gamma] A \mapsto C' \not \in [\Gamma] A \mapsto \Delta \] \[ \text{By SpineRecover} \]

\[ \text{Case} \]
\[ \Omega \vdash [\Omega] s : [\Omega] A \mapsto C q \quad \text{DeclSpinePass} \]
\[ \text{Subderivation} \]
\[ s \vdash [\Gamma] A \mapsto C' q \mapsto [\Delta] \] \[ \text{By i.h.} \]
\[ \Delta \mapsto \Omega' \] "
\[ \text{dom}(\Delta) = \text{dom}(\Omega') \] "
\[ \Omega \mapsto \Omega' \] "
\[ C' = [\Delta] C' \] "
\[ C = [\Omega'] C' \] "

We distinguish cases as follows:

- If \( p = \not \) or \( q = 1 \), then we can just apply \text{SpinePass}.
  \[ s \vdash [\Gamma] A \mapsto C' \not \in \Gamma \] \[ \text{By SpinePass} \]

- Otherwise, \( p = ! \) and \( q = \not \). If \text{FEV}(C) \neq \emptyset , we can apply \text{SpinePass} as above. If \text{FEV}(C) = \emptyset , then we instead apply \text{SpineRecover}.
  \[ s \vdash [\Gamma] A \mapsto C' \not \in [\Gamma] A \mapsto \Delta \] \[ \text{By SpineRecover} \]

Here, \( q' = ! \) and \( q = \not \), so \( q' \subseteq q \).
• Case

\[
\begin{align*}
\text{DeclEmptySpine} & & \\
\varepsilon & & \\
\Gamma \vdash \cdot : [\Omega]A \quad p \gg [\Omega]A \quad p & & \quad \text{By EmptySpine} \\
\varepsilon & & \\
[\Gamma]A & = [\Gamma][\Gamma]A & & \quad \text{By idempotence of substitution} \\
\varepsilon & & \\
\Gamma & \rightarrow \Omega & & \quad \text{Given} \\
\varepsilon & & \\
dom(\Gamma) & = \dom(\Omega) & & \quad \text{Given} \\
\varepsilon & & \\
[\Omega][\Gamma]A & = [\Omega]A & & \quad \text{By Lemma 29 (Substitution Monotonicity) (iii)} \\
\varepsilon & & \\
\Omega & \rightarrow \Omega & & \quad \text{By Lemma 32 (Extension Reflexivity)} \\
\end{align*}
\]

• Case

\[
\begin{align*}
\text{Decl→Spine} & & \\
\varepsilon & & \\
[\Omega][\Gamma] & \vdash [\Omega]e_0 & & \leftrightarrow [\Omega]A_1 \quad q \quad & & \quad \text{Subderivation} \\
\varepsilon & & \\
\Gamma & \vdash e_0 & & \leftrightarrow A_1 \quad q & & \indent \Theta \quad \text{By i.h.} \\
\varepsilon & & \\
\Theta & \rightarrow \Omega_\Theta & & \quad \" \\
\varepsilon & & \\
\Omega & \rightarrow \Omega_\Theta & & \quad \" \\
\varepsilon & & \\
A & = [\Omega_\Theta]A' & & \quad \" \\
\varepsilon & & \\
A' & = [\Theta]A' & & \quad \" \\
\varepsilon & & \\
\text{Subderivation} & & \\
\Gamma & \vdash s_0 : [\Omega]A_2 \quad q \gg B \quad p & & \quad \text{By i.h.} \\
\varepsilon & & \\
\Delta & \rightarrow \Omega' & & \quad \" \\
\varepsilon & & \\
dom(\Delta) & = \dom(\Omega') & & \quad \" \\
\varepsilon & & \\
\Omega & \rightarrow \Omega' & & \quad \" \\
\varepsilon & & \\
B' & = [\Delta]B' & & \quad \" \\
\varepsilon & & \\
B & = [\Omega']B' & & \quad \" \\
\varepsilon & & \\
\Gamma & \vdash e_0 \quad s_0 : A_1 \rightarrow A_2 \quad q \gg B \quad p \rightarrow \Delta & & \quad \text{By →Spine} \\
\end{align*}
\]
• Case \([\Omega]\Gamma \vdash [\Omega]\Pi \text{ true} \quad [\Omega]\Gamma \vdash [\Omega]e \equiv [\Omega]A_0 \text{ p}\)

\[
[\Omega]\Gamma \vdash [\Omega]e \equiv ([\Omega]A_0) \land [\Omega]\Pi \text{ p}
\]

\text{Decl/\top}

If \(e\) not a case, then:

\[
[\Omega]\Gamma \vdash [\Omega]\Pi \text{ true} \quad \Gamma \vdash \Pi \text{ true } \vdash \Theta
\]

\[
\Theta \rightarrow \Omega_0'
\]

\[
\Omega \rightarrow \Omega_0'
\]

\[
\Gamma \rightarrow \Omega
\]

\[
\Gamma \rightarrow \Omega_0'
\]

\[
[\Omega]\Gamma = [\Omega]\Omega
\]

\[
= [\Omega_0'][\Omega_0'
\]

\[
= [\Omega_0'][\Theta
\]

\[
\Gamma \vdash A_0 \land \Pi \text{ p type}
\]

\[
\Gamma \vdash A_0 \text{ p type}
\]

\[
[\Omega]A_0 = [\Omega_0']A_0
\]

\text{Subderivation}

\[
\Theta \vdash e \equiv \Theta A_0 \text{ p } \vdash \Delta
\]

\text{By i.h.}

\[
\Delta \rightarrow \Omega'
\]

\[
\text{dom}(\Delta) = \text{dom}(\Omega')
\]

\[
\Omega_0' \rightarrow \Omega'
\]

\[
\Omega \rightarrow \Omega'
\]

\text{By Lemma 33 (Extension Transitivity)}

\[
\Gamma \vdash e \equiv A_0 \land \Pi \vdash \Delta
\]

\text{By \top}

Otherwise, we have \(e = \text{case}(e_0, \Pi)\). Let \(n\) be the height of the given derivation.

\[
\begin{align*}
n - 1 & \quad [\Omega]\Gamma \vdash [\Omega](\text{case}(e_0, \Pi)) \equiv [\Omega]A_0 \text{ p} & \text{Subderivation} \\
n - 2 & \quad [\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow B ! & \text{By Lemma 62 (Case Invertibility)} \\
n - 2 & \quad [\Omega]\Gamma \vdash [\Omega]\Pi \vdash B \equiv [\Omega]A_0 \text{ p} & \text{"} \\
n - 2 & \quad [\Omega]\Gamma \vdash [\Omega]\Pi \text{ covers B} & \text{"} \\
n - 1 & \quad [\Omega]\Gamma \vdash [\Omega]\Pi \text{ true} & \text{Subderivation} \\
n - 1 & \quad [\Omega]\Gamma \vdash [\Omega]\Pi \vdash B \equiv ([\Omega]A_0) \land ([\Omega]\Pi \text{ p}) & \text{By Lemma 61 (Interpolating With and Exists) (1)} \\
n - 1 & \quad [\Omega]\Gamma \vdash [\Omega]\Pi \vdash [\Omega](A_0 \land \Pi) \text{ p} & \text{By def. of subst.}
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash e_0 \Rightarrow B' ! \vdash \Theta & \text{By i.h.} \\
\Theta & \rightarrow \Omega_0' & \text{"} \\
\Omega & \rightarrow \Omega_0' & \text{"} \\
B & = [\Omega_0']B' & \text{"} \\
= [\Omega_0'][\Theta]B' & \text{By Lemma 30 (Substitution Invariance)}
\end{align*}
\]

\[
[\Omega]\Gamma = [\Omega_0'][\Theta
\]

\[
[\Omega](A_0 \land \Pi) = [\Omega_0'](A_0 \land \Pi)
\]

\text{By Lemma 57 (Multiple Confluence)}

\text{By Lemma 55 (Completing Completeness) (ii)}

\[
\begin{align*}
n - 1 & \quad [\Omega_0'][\Theta] \vdash [\Omega]\Pi \vdash [\Omega_0'][\Theta]B' \equiv [\Omega_0'](A_0 \land \Pi) \text{ p} & \text{By above equalities} \\
\Theta & \vdash [\Theta]B' \equiv A_0 \land \Pi \vdash \Delta & \text{By i.h.} \\
\Delta & \rightarrow \Omega' & \text{"} \\
\text{dom}(\Delta) & = \text{dom}(\Omega') & \text{"} \\
\Omega_0' & \rightarrow \Omega' & \text{"}
\end{align*}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ \Theta \vdash \Pi \text{ covers } [\Theta]B' \quad \text{By Theorem 10} \]
\[ \Omega \rightarrow \Omega' \quad \text{By Lemma 33 (Extension Transitivity)} \]
\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff A_0 \land P \dashv \Delta \quad \text{By Case} \]

- Case [DeclNil]: Similar to the first part of the [Decl\∧] case.

- Case

Let \( \Omega' = (\Omega, \alpha, A : N = t_2) \).

\[ \begin{align*}
\Omega' &\vdash (\Omega) t = \text{succ}(t_2) \quad \text{true} & \text{Subderivation} \\
[\Omega'](\Gamma, \alpha, A : N) &\vdash (\Omega) t = [\Omega'] \text{succ}(\alpha) \quad \text{true} & \text{Defs. of extension and subst.} \\
\Gamma, \alpha, A : N &\vdash t = \text{succ}(\alpha) \quad \vdash \Gamma' & \text{By Lemma 95 (Completeness of Checkprop)} \\
\Gamma' &\rightarrow \Omega_0' & \text{"} \\
\Omega' &\rightarrow \Omega_0' & \text{"} \\
\Gamma, \alpha, A : N &\vdash \Gamma' & \text{By Lemma 47 (Checkprop Extension)} \\
\Gamma, \alpha, A : N &\vdash \Omega_0' & \text{By Lemma 33 (Extension Transitivity)} \\
\Omega, \Gamma, \alpha, A : N &\vdash \Omega \quad \text{By Lemma 54 (Completing Stability)} \\
\Omega, \Gamma, \alpha, A : N &\vdash \Omega + & \text{By def. of context application} \\
\Omega, \Gamma, \alpha, A : N &\vdash [\Omega] \text{[\Omega +]} & \text{By Lemma 55 (Completing Completeness) (iii)} \\
\Omega, \Gamma, \alpha, A : N &\vdash [\Omega] A_0 & \text{By def. of context application} \\
\Omega, \Gamma, \alpha, A : N &\vdash [\Omega +] A_0 & \text{By Lemma 55 (Completing Completeness) (ii)} \\
\Omega, \Gamma, \alpha, A : N &\vdash [\Omega +] A_0 p & \text{By above equalities} \\
\Gamma', \alpha, A : N &\vdash [\Gamma'] A_0 p \dashv \Theta & \text{By i.h.} \\
\Theta &\rightarrow \Omega_0'' & \text{"} \\
\Omega_0' &\rightarrow \Omega_0'' & \text{"} \\
\Theta, \alpha, A : N &\vdash [\Theta] A_0 p \dashv \Delta, \Gamma, \alpha, A' & \text{By i.h.} \\
\Delta, \alpha, A' &\rightarrow \Omega'' & \text{"} \\
\text{dom}(\Delta, \alpha, A') &\dashv \text{dom}(\Omega'') & \text{"} \\
\Omega'' &\rightarrow \Omega'' & \text{"} \\
\Omega'' &\rightarrow \Omega'' & \text{By Lemma 22 (Extension Inversion) (ii)} \\
\Delta &\rightarrow \Omega' & \text{"} \\
\text{dom}(\Delta) &\dashv \text{dom}(\Omega') & \text{"} \\
\Gamma', \alpha, A' &\rightarrow \Omega' & \text{By Lemma 33 (Extension Transitivity)} \\
\Omega &\rightarrow \Omega' & \text{By Lemma 22 (Extension Inversion) (ii)} \\
\Gamma, \alpha, A : N &\vdash \text{case}(e_0, \Pi) \dashv A_0 \land P \dashv \Delta & \text{By Cons} \\
\end{align*} \]

- Case

\[ \begin{align*}
[\Omega] \Gamma &\vdash (\Omega) e_1 \iff A_1 p & \text{[Decl\times]} \\
[\Omega] \Gamma &\vdash (\Omega) e_2 \iff A_2 p & \text{[Decl\times]} \\
[\Omega] \Gamma &\vdash (\Omega) e_1, (\Omega) e_2 \iff A_1' \times A_2' p & \text{[Decl\times]} \\
\end{align*} \]
Either $[\Gamma]A = A_1 \times A_2$ or $[\Gamma]A = \hat{\alpha} \in \text{unsolved}(\Gamma)$.

- In the first case $(\Gamma)A = A_1 \times A_2$, we have $A'_1 = [\Omega]A_1$ and $A'_2 = [\Omega]A_2$.

\[
\begin{align*}
[\Omega]\Gamma \vdash [\Omega]e_1 \leftarrow A'_1 \Gamma & \quad \text{Subderivation} \\
[\Omega]\Gamma \vdash [\Omega]e_1 \leftarrow [\Omega]A_1 \Gamma & = A'_1 \\
\Gamma \vdash e_1 \leftarrow [\Gamma]A_1 \Gamma \vdash \Theta & \quad \text{By i.h.} \\
\Theta \rightarrow \Omega & \quad \text{"} \\
\text{dom(}\Theta\text{)} = \text{dom(}\Omega\Theta\text{)} & \quad \text{"} \\
\text{dom(}\Theta\text{)} = \text{dom(}\Omega\Theta\text{)} & \quad \text{By def. of subst.} \\
\end{align*}
\]

\[
\begin{align*}
[\Omega]\Gamma \vdash [\Omega]e_2 \leftarrow A'_2 \Gamma & \quad \text{Subderivation} \\
[\Omega]\Gamma \vdash [\Omega]e_2 \leftarrow [\Omega]A_2 \Gamma & = A'_2 \\
\Gamma \rightarrow \Theta & \quad \text{By Lemma 51 (Typing Extension)} \\
[\Omega]\Gamma = [\Omega\Theta]\Theta & \quad \text{By Lemma 51 (Multiple Confluence)} \\
[\Omega]A_2 = [\Omega\Theta]A_2 & \quad \text{By Lemma 55 (Completing Completeness) (ii)} \\
[\Omega\Theta]\Theta \vdash [\Omega]e_2 \leftarrow [\Omega\Theta]A_2 \Gamma & = \text{By above equalities} \\
\Theta \vdash e_2 \leftarrow [\Gamma]A_2 \Gamma \vdash \Delta & \quad \text{By i.h.} \\
\Theta \vdash e_2 \leftarrow [\Gamma]A_2 \Gamma \vdash \Delta & \quad \text{By i.h.} \\
\Delta \rightarrow \Omega' & \quad \text{"} \\
\text{dom(}\Delta\text{)} = \text{dom(}\Omega'\text{)} & \quad \text{"} \\
\text{dom(}\Delta\text{)} = \text{dom(}\Omega'\text{)} & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Omega_\Theta \rightarrow \Omega' & \quad \text{"} \\
\Omega \rightarrow \Omega' & \quad \text{By def. of subst.} \\
\Gamma \vdash \langle e_1, e_2 \rangle \leftarrow ([\Gamma]A_1) \times ([\Gamma]A_2) \Gamma \vdash \Delta & \quad \text{By def. of subst.} \\
\end{align*}
\]

- In the second case, where $[\Gamma]A = \hat{\alpha}$, combine the corresponding subcase for $\text{Decl}+\text{I_k}$ with some straightforward additional reasoning about contexts (because here we have two subderivations, rather than one).

\[
\begin{align*}
\text{Case} \quad [\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow C ! & \quad [\Omega]\Gamma \vdash [\Omega]\Pi : C \leftarrow [\Omega]A \Gamma \quad [\Omega]\Gamma \vdash [\Omega]\Pi \text{ covers } C \\
[\Omega]\Gamma \vdash [\Omega]e_0, [\Omega]\Pi \leftarrow [\Omega]A \Gamma & \quad \text{DeclCase} \\
\end{align*}
\]

\[
\begin{align*}
[\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow C ! & \quad \text{Subderivation} \\
\Gamma \vdash e_0 \Rightarrow C ! \vdash \Theta & \quad \text{By i.h.} \\
\Theta \rightarrow \Omega & \quad \text{Subderivation} \\
\text{dom(}\Theta\text{)} = \text{dom(}\Omega\Theta\text{)} & \quad \text{"} \\
\text{dom(}\Theta\text{)} = \text{dom(}\Omega\Theta\text{)} & \quad \text{"} \\
\text{C = }\text{dom(}\Theta\text{)}\text{C} & \quad \text{"} \\
\Theta \vdash C ! \text{ type} & \quad \text{By Lemma 63 (Well-Formed Outputs of Typing)} \\
\text{FEV(C')} = \emptyset & \quad \text{By inversion} \\
[\Omega\Theta]\text{C'} = C' & \quad \text{By a property of substitution}
\end{align*}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

\[ \Gamma \rightarrow \Omega \] Given
\[ \Delta \rightarrow \Omega \] Given
\[ \Theta \rightarrow \Omega \] By Lemma 33 (Extension Transitivity)
\[ [\Omega]\Gamma = [\Omega]\Theta = [\Omega]\Delta \] By Lemma 56 (Confluence of Completeness)
\[ \Gamma \rightarrow \Theta \] By Lemma 51 (Typing Extension)
\[ \Gamma \rightarrow \Theta_\Theta \] By Lemma 33 (Extension Transitivity)
\[ [\Omega]\Gamma = [\Theta_\Theta]\Theta \] By Lemma 57 (Multiple Confluence)
\[ \Gamma \vdash A \text{ type} \] Given + inversion
\[ \Omega \vdash A \text{ type} \] By Lemma 38 (Extension Weakening (Types))
\[ [\Omega]A = [\Theta_\Theta]A \] By Lemma 55 (Completing Completeness) (ii)
\[ [\Omega]\Gamma \vdash [\Theta_\Theta]\Pi \vdash C \iff [\Pi]A \vdash \Delta \] Subderivation
\[ \Theta \vdash \Pi \vdash C \iff [\Theta]A \vdash \Delta \] By i.h. (v)
\[ \Delta \rightarrow \Omega' \] "
\[ \text{dom}(\Delta) = \text{dom}(\Omega') \] "
\[ \Omega_\Theta \rightarrow \Omega \] "
\[ \Omega \rightarrow \Omega' \] By Lemma 33 (Extension Transitivity)
\[ [\Omega]\Gamma \vdash [\Theta_\Theta]\Pi \text{ covers } C \] Subderivation
\[ [\Omega]\Gamma = [\Theta_\Theta]\Delta \] Above
\[ = [\Omega']\Delta \] By Lemma 57 (Multiple Confluence)
\[ [\Omega']\Delta \vdash [\Theta_\Theta]\Pi \text{ covers } C' \] By above equalities
\[ \Delta \rightarrow \Omega' \] By Lemma 33 (Extension Transitivity)
\[ \Gamma \vdash C' \text{ ! type} \] Given
\[ \Gamma \rightarrow \Delta \] By Lemma 51 (Typing Extension) & 33 (Extension Transitivity)
\[ \Delta \vdash C' \text{ ! type} \] By Lemma 41 (Extension Weakening for Principal Typing)
\[ [\Delta]C' = C' \] By FEV(C') = \emptyset and a property of subst.
\[ \Delta \vdash \Pi \text{ covers } C' \] By Theorem 10
\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff [\Gamma]A \vdash \Delta \] By Case

\[ [\Omega]\Gamma \vdash [\Omega]e_1 \iff A_1 \] Subderivation
\[ [\Omega]\Gamma \vdash [\Omega]e_2 \iff A_2 \] By above equality
\[ [\Omega]\Gamma \vdash ([\Omega]e_1, [\Omega]e_2) \iff A_1 \times A_2 \] Decl×
\[ [\Omega]A \] Given

Either \( A = \tilde{\alpha} \) where \( [\Omega]\tilde{\alpha} = A_1 \times A_2 \), or \( A = A'_1 \times A'_2 \) where \( A_1 = [\Omega]A'_1 \) and \( A_2 = [\Omega]A'_2 \). In the former case \( A = \tilde{\alpha} \).

We have \( [\Omega]\tilde{\alpha} = A_1 \times A_2 \). Therefore \( A_1 = [\Omega]A'_1 \) and \( A_2 = [\Omega]A'_2 \). Moreover, \( \Gamma = \Gamma_0[\tilde{\alpha} : \kappa] \).

\[ [\Omega]\Gamma \vdash [\Omega]e_1 \iff [\Omega]A'_1 \] Subderivation
Let \( \Gamma' = \Gamma_0[\tilde{\alpha}_1 : \kappa, \tilde{\alpha}_2 : \kappa, \tilde{\alpha} : \kappa = \tilde{\alpha}_1 + \tilde{\alpha}_2] \).
\[ [\Omega]\Gamma = [\Omega]\Gamma' \] By def. of context substitution
\[ [\Omega]\Gamma' \vdash [\Omega]e_1 \iff [\Omega]A'_1 \] By above equality
\[ \Gamma' \vdash e_1 \iff [\Gamma']A'_1 \vdash \Theta \] By i.h.
\[ \Theta \rightarrow \Omega_1 \] "
\[ \Omega \rightarrow \Omega_1 \] "
\[ \text{dom}(\Theta) = \text{dom}(\Omega_1) \] "
\[ [\Omega]\Gamma \vdash [\Omega]e_2 \iff [\Omega]A'_2 \] Subderivation
Case \( \text{DeclMatch} \): Apply rule \text{DeclMatch}.

Case \( \text{DeclMatchSeq} \): Apply the i.h. twice, along with standard lemmas.

Case \( \text{DeclMatchEmpty} \): Apply the i.h. twice, along with standard lemmas.

Case \( \text{DeclMatchBase} \): Apply the i.h. (i) and rule \text{MatchBase}.

Case \( \text{DeclMatchUnit} \): Apply the i.h. and rule \text{MatchUnit}.

Case \( \text{DeclMatch=} \): By i.h. and rule \text{Match=}.

Case \( \text{DeclMatch} \times \): By i.h. and rule \text{Match} \times.

Case \( \text{DeclMatch}+\times \): By i.h. and rule \text{Match}+\times.

Case \( \text{DeclMatch}+\times \): By i.h. and rule \text{Match}+\times.

In the latter case (\( A = A'_1 \times A'_2 \)):

\[
\begin{align*}
\Omega \Gamma \vdash \Omega e_1 \equiv A_1 p & \quad \text{Subderivation} \\
\Omega \Gamma \vdash \Omega e_1 \equiv \Omega A'_1 p & \quad A_1 = \Omega A'_1 \\
\Gamma \vdash e_1 \equiv \Gamma A'_1 p \vdash \Theta & \quad \text{By i.h.} \\
\Theta \rightarrow \Omega_0 & \quad "" \\
dom(\Theta) = dom(\Omega_0) & \quad "" \\
\Omega \rightarrow \Omega_0 & \quad ""
\end{align*}
\]

\[
\begin{align*}
\Omega \Gamma \vdash \Omega e_2 \equiv A_2 p & \quad \text{Subderivation} \\
\Omega \Gamma \vdash \Omega e_2 \equiv \Omega A'_2 p & \quad A_2 = \Omega A'_2 \\
\Gamma \vdash A'_1 \times A'_2 p \text{ type} & \quad \text{Given (} A = A'_1 \times A'_2 \text{)} \\
\Gamma \vdash A'_2 \text{ type} & \quad \text{By inversion} \\
\Gamma \rightarrow \Omega & \quad \text{Given} \\
\Omega_0 \rightarrow \Omega_0 & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Omega_0 \rightarrow A'_2 \text{ type} & \quad \text{By Lemma 38 (Extension Weakening (Types))} \\
\Omega \Gamma \vdash \Omega e_2 \equiv \Omega_0 A'_2 p & \quad \text{By Lemma 55 (Completing Completeness)} \\
\Omega \Gamma \vdash \Omega e_2 \equiv \Omega_0 \Theta A'_2 p & \quad \text{By Lemma 29 (Substitution Monotonicity)} \\
\Theta \vdash e_2 \equiv \Theta A'_2 p \vdash \Delta & \quad \text{By i.h.} \\
\Delta \rightarrow \Omega' & \quad "" \\
dom(\Delta) = dom(\Omega') & \quad "" \\
\Omega_0 \rightarrow \Omega' & \quad "" \\
\Omega \rightarrow \Omega' & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma \vdash (e_1, e_2) \equiv (\Omega A_1) \times (\Omega A_2) p \vdash \Delta & \quad \text{By \times} \\
\Gamma \vdash (e_1, e_2) \equiv \Omega (A_1 \times A_2) p \vdash \Delta & \quad \text{By def. of substitution}
\end{align*}
\]

Now we turn to parts (v) and (vi), completeness of matching.

- **Case** \( \text{DeclMatchEmpty} \): Apply rule \text{MatchEmpty}.
- **Case** \( \text{DeclMatchSeq} \): Apply the i.h. twice, along with standard lemmas.
- **Case** \( \text{DeclMatchBase} \): Apply the i.h. (i) and rule \text{MatchBase}.
- **Case** \( \text{DeclMatchUnit} \): Apply the i.h. and rule \text{MatchUnit}.
- **Case** \( \text{DeclMatch=} \): By i.h. and rule \text{Match=}.
- **Case** \( \text{DeclMatch} \times \): By i.h. and rule \text{Match} \times.
- **Case** \( \text{DeclMatch}+\times \): By i.h. and rule \text{Match}+\times.
Proof of Theorem 11 (Completeness of Algorithmic Typing)

- Case \( \Gamma \vdash (\varphi) \) : types

\[
\frac{\Gamma \vdash \varphi \Rightarrow e :: [\Omega][A], [\Omega][\bar{A}] \leftarrow [\Omega][C] \quad \text{DeclMatch} }{\Gamma \vdash \varphi \Rightarrow e :: (([\Omega][A \wedge [\Omega][P]]), [\Omega][\bar{A}] \leftarrow [\Omega][C] \quad \text{DeclMatch\wedge} }
\]

To apply the i.h. (vi), we will show (1) \( \Gamma \vdash A, \bar{A} \) ! types, (2) \( \Gamma \vdash P \) prop, (3) \( \text{FEV}(P) = \emptyset \), (4) \( \Gamma \vdash C \) p type, (5) \( [\Omega]\Gamma / [\Omega][P] \vdash \varphi \Rightarrow [\Omega]e :: [\Omega]A \leftarrow [\Omega]C \), and (6) \( p' \subseteq p \).

- Case \( \text{DeclMatchNeg} \)
  By i.h. and rule \text{MatchNeg}.

- Case \( \text{DeclMatchWild} \)
  By i.h. and rule \text{MatchWild}.

- Case \( \text{DeclMatchNil} \)
  Similar to the \( \text{DeclMatch\wedge} \) case.

- Case \( \text{DeclMatchCons} \)
  Similar to the \( \text{DeclMatch\wedge} \) and \( \text{DeclMatch\wedge} \) cases.

- Case \( \text{mgu}([\Omega][\sigma], [\Omega][\tau]) = \bot \)

\[
\frac{[\Omega]\Gamma / [\Omega][\sigma] = [\Omega][\tau] \vdash [\Omega][\varphi] \Rightarrow e :: [\Omega][\bar{A}] \leftarrow [\Omega][C] \quad \text{DeclMatch\wedge} }{[\Omega]\Gamma / [\Omega][\sigma] = [\Omega][\tau] \vdash \varphi \Rightarrow e :: [\Omega][\bar{A}] \leftarrow [\Omega][C] \quad \text{DeclMatch\wedge} }
\]

\[
\frac{\begin{array}{l}
\begin{array}{l}
\Gamma \quad \text{Given} \\
\text{FEV}(\sigma = \tau) = \emptyset \\
[\Omega][\sigma] = [\Gamma][\sigma] \\
[\Omega][\tau] = [\Gamma][\tau] \\
mgu([\Omega][\sigma], [\Omega][\tau]) = \bot \\
\end{array} \\
\begin{array}{l}
\begin{array}{l}
\Gamma \quad \text{Given} \\
\Gamma / [\sigma \equiv [\tau \vdash [\Gamma][\varphi] \Rightarrow e :: [\Gamma][\bar{A}] \leftarrow [\Gamma][C] \quad \text{DeclMatch\wedge} \\
\end{array} \\
\begin{array}{l}
\begin{array}{l}
\Omega \quad \text{By Lemma 39 (Principal Agreement) (i)} \\
\end{array} \\
\begin{array}{l}
\begin{array}{l}
\text{dom}(\Gamma) = \text{dom}(\Omega) \\
\end{array} \\
\end{array} \\
\end{array} \\
\end{array}
\end{array}
\]

- Case \( \text{mgu}([\Omega][\sigma], [\Omega][\tau]) = \emptyset \)

\[
\frac{\begin{array}{l}
\begin{array}{l}
\theta([\Omega][\Gamma] \vdash \theta(\varphi) \Rightarrow e :: [\Omega][\bar{A}] \leftarrow [\Omega][C] \quad \text{DeclMatchUnify} \\
\end{array} \\
\begin{array}{l}
\begin{array}{l}
\text{mgu}([\Omega][\sigma], [\Omega][\tau]) = \bot \\
\text{By above equalities} \\
\begin{array}{l}
\begin{array}{l}
\Gamma / [\sigma \equiv [\tau \vdash [\Gamma][\varphi] \Rightarrow e :: [\Gamma][\bar{A}] \leftarrow [\Gamma][C] \quad \text{DeclMatch\wedge} \\
\end{array} \\
\begin{array}{l}
\begin{array}{l}
\Omega \quad \text{By Lemma 32 (Extension Reflexivity)} \\
\end{array} \\
\begin{array}{l}
\begin{array}{l}
\text{dom}(\Gamma) = \text{dom}(\Omega) \\
\end{array} \\
\end{array} \\
\end{array} \\
\end{array} \\
\end{array} \\
\end{array}
\end{array}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ ([\Omega] \sigma = [\Gamma] \sigma) \text{ and } ([\Omega] \tau = [\Gamma] \tau) \]

As in DeclMatch, case

\[ \text{mg}u([\Omega] \sigma, [\Omega] \tau) = \emptyset \]

Given

\[ \text{mg}u([\Gamma] \sigma, [\Gamma] \tau) = \emptyset \]

By above equalities

\[ \Gamma / \sigma \triangleq \tau : \kappa \rightarrow (\Gamma, \Theta) \]

By Lemma 92 (Completeness of Elimeq) (1)

\[ \Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n) \]

"''

\[ [\Gamma, \Theta] u = \theta([\Gamma] u) \]

" for all \( \Gamma \vdash u : \kappa \)

\[ \theta([\Omega] \Gamma) \vdash \theta(\rho \Rightarrow [\Omega] e) : \theta([\Omega] \vec{A}) \iff \theta([\Omega] C) \]

Subderivation

\[ \theta([\Omega] \vec{A}) = [\Omega, \uparrow p, \Theta] \vec{A} \]

By Lemma 93 (Substitution Upgrade) (i) (over \( \vec{A} \))

\[ \theta([\Omega] C) = [\Omega, \uparrow p, \Theta] C \]

By Lemma 93 (Substitution Upgrade) (i)

\[ \theta(\rho \Rightarrow [\Omega] e) = [\Omega, \uparrow p, \Theta] (\rho \Rightarrow e) \]

By Lemma 93 (Substitution Upgrade) (iv)

\[ [\Omega, \uparrow p, \Theta](\Gamma, \uparrow p, \Theta) \vdash [\Omega, \uparrow p, \Theta](\rho \Rightarrow e) : [\Omega, \uparrow p, \Theta] \vec{A} \iff [\Omega, \uparrow p, \Theta] C \]

By above equalities

\[ \Gamma, \uparrow p, \Theta \vdash (\rho \Rightarrow e) : [\Gamma, \uparrow p, \Theta] \vec{A} \iff [\Gamma, \uparrow p, \Theta] C \]\n
By i.h.

\[ \Delta, \uparrow p, \Theta \vdash \Omega', \uparrow p, \Omega'' \]

"''

\[ \Omega, \uparrow p, \Theta \vdash \Omega', \uparrow p, \Omega'' \]

"''

\[ \text{dom}(\Delta, \uparrow p, \Delta') = \text{dom}(\Omega', \uparrow p, \Omega'') \]

By Lemma 22 (Extension Inversion) (ii)

\[ \Delta \rightarrow \Omega' \]

By Lemma 22 (Extension Inversion) (ii)

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]

By Lemma 22 (Extension Inversion) (ii)

\[ \Omega \rightarrow \Omega' \]

By Lemma 22 (Extension Inversion) (ii)

\[ \Gamma / [\Gamma] \sigma = [\Gamma] \tau \vdash \rho \Rightarrow e : [\Gamma] \vec{A} \iff [\Gamma] C \]

By MatchUnify