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

Jana Dunfield Neelakantan R. Krishnaswami
August 15, 2020

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 8
2 List of Judgments 18

A Properties of the Declarative System 19
1 Lemma (Declarative Well-foundedness) .......................................................... 19
2 Lemma (Declarative Weakening) ................................................................. 19
3 Lemma (Declarative Term Substitution) ..................................................... 19
4 Lemma (Reflexivity of Declarative Subtyping) .......................................... 19
5 Lemma (Subtyping Inversion) ................................................................. 19
6 Lemma (Subtyping Polarity Flip) .............................................................. 19
7 Lemma (Transitivity of Declarative Subtyping) ....................................... 20

B Substitution and Well-formedness Properties 20
8 Lemma (Substitution—Well-formedness) ................................................. 20
9 Lemma (Uvar Preservation) ................................................................. 20
10 Lemma (Sorting Implies Typing) ............................................................ 20
11 Lemma (Right-Hand Substitution for Sorting) ........................................ 20
12 Lemma (Right-Hand Substitution for Propositions) ............................... 20
13 Lemma (Right-Hand Substitution for Typing) ......................................... 20
14 Lemma (Substitution for Sorting) .......................................................... 20
15 Lemma (Substitution for Prop Well-Formedness) .................................... 20
16 Lemma (Substitution for Type Well-Formedness) .................................. 20
17 Lemma (Substitution Stability) ............................................................. 20
18 Lemma (Equal Domains) ................................................................. 20

C Properties of Extension 20
19 Lemma (Declaration Preservation) .......................................................... 20
20 Lemma (Declaration Order Preservation) ................................................. 20
21 Lemma (Reverse Declaration Order Preservation) .................................. 20
22 Lemma (Extension Inversion) .............................................................. 21
## F Decidability of Instantiation

- Lemma (Left Unsolvedness Preservation) ............................................. 26
- Lemma (Left Free Variable Preservation) ............................................. 26
- Lemma (Instantiation Size Preservation) ............................................. 26
- Lemma (Decidability of Instantiation) ............................................. 26

## G Separation

- Lemma (Transitivity of Separation) ............................................. 26
- Lemma (Separation Truncation) ............................................. 26
- Lemma (Separation for Auxiliary Judgments) ............................................. 26
- Lemma (Separation for Subtyping) ............................................. 27
- Lemma (Separation—Main) ............................................. 27

## H Decidability of Algorithmic Subtyping

### H.1 Lemmas for Decidability of Subtyping

- Lemma (Substitution Isn’t Large) ............................................. 28
- Lemma (Instantiation Solves) ............................................. 28
- Lemma (Checkeq Solving) ............................................. 28
- Lemma (Prop Equiv Solving) ............................................. 28
- Lemma (Equiv Solving) ............................................. 28
- Lemma (Decidability of Propositional Judgments) ............................................. 28
- Lemma (Decidability of Equivalence) ............................................. 28

### H.2 Decidability of Subtyping

- Theorem (Decidability of Subtyping) ............................................. 28

### H.3 Decidability of Matching and Coverage

- Lemma (Decidability of Guardedness Judgment) ............................................. 28
- Lemma (Decidability of Expansion Judgments) ............................................. 28
- Lemma (Expansion Shrinks Size) ............................................. 29
- Theorem (Decidability of Coverage) ............................................. 29

### H.4 Decidability of Typing

- Theorem (Decidability of Typing) ............................................. 29

## I Determinacy

- Lemma (Determinacy of Auxiliary Judgments) ............................................. 30
- Lemma (Determinacy of Equivalence) ............................................. 30
- Theorem (Determinacy of Subtyping) ............................................. 30
- Theorem (Determinacy of Typing) ............................................. 30

## J Soundness

### J.1 Soundness of Instantiation

- Lemma (Soundness of Instantiation) ............................................. 31

### J.2 Soundness of Checkeq

- Lemma (Soundness of Checkeq) ............................................. 31

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

- Lemma (Soundness of Propositional Equivalence) ............................................. 31
- Lemma (Soundness of Algorithmic Equivalence) ............................................. 31

### J.4 Soundness of Checkprop

- Lemma (Soundness of Checkprop) ............................................. 31

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

- Lemma (Soundness of Equality Elimination) ............................................. 31

### J.6 Soundness of Subtyping

- Theorem (Soundness of Algorithmic Subtyping) ............................................. 31

### J.7 Soundness of Typing

- Theorem (Soundness of Match Coverage) ............................................. 31
- Lemma (Well-formedness of Algorithmic Typing) ............................................. 31
# CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
<th>Section/Lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Theorem (Eagerness of Types)</td>
</tr>
<tr>
<td>9</td>
<td>Theorem (Soundness of Algorithmic Typing)</td>
</tr>
<tr>
<td>K</td>
<td>Completeness</td>
</tr>
<tr>
<td>K.1</td>
<td>Completeness of Auxiliary Judgments</td>
</tr>
<tr>
<td>92</td>
<td>Lemma (Completeness of Instantiation)</td>
</tr>
<tr>
<td>93</td>
<td>Lemma (Completeness of Checkeq)</td>
</tr>
<tr>
<td>94</td>
<td>Lemma (Completeness of Elimeq)</td>
</tr>
<tr>
<td>95</td>
<td>Lemma (Substitution Upgrade)</td>
</tr>
<tr>
<td>96</td>
<td>Lemma (Completeness of Propequiv)</td>
</tr>
<tr>
<td>97</td>
<td>Lemma (Completeness of Checkprop)</td>
</tr>
<tr>
<td>K.2</td>
<td>Completeness of Equivalence and Subtyping</td>
</tr>
<tr>
<td>98</td>
<td>Lemma (Completeness of Equiv)</td>
</tr>
<tr>
<td>10</td>
<td>Theorem (Completeness of Subtyping)</td>
</tr>
<tr>
<td>K.3</td>
<td>Completeness of Typing</td>
</tr>
<tr>
<td>99</td>
<td>Lemma (Variable Decomposition)</td>
</tr>
<tr>
<td>100</td>
<td>Lemma (Pattern Decomposition and Substitution)</td>
</tr>
<tr>
<td>101</td>
<td>Lemma (Pattern Decomposition Functionality)</td>
</tr>
<tr>
<td>102</td>
<td>Lemma (Decidability of Variable Removal)</td>
</tr>
<tr>
<td>103</td>
<td>Lemma (Variable Inversion)</td>
</tr>
<tr>
<td>11</td>
<td>Theorem (Completeness of Match Coverage)</td>
</tr>
<tr>
<td>12</td>
<td>Theorem (Completeness of Algorithmic Typing)</td>
</tr>
</tbody>
</table>

# Proofs

## A' Properties of the Declarative System

<table>
<thead>
<tr>
<th>Page</th>
<th>Lemma/Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>Proof of Lemma (Declarative Well-foundedness)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Declarative Weakening)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Declarative Term Substitution)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Reflexivity of Declarative Subtyping)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Subtyping Inversion)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Subtyping Polarity Flip)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Transitivity of Declarative Subtyping)</td>
</tr>
</tbody>
</table>

## B' Substitution and Well-formedness Properties

<table>
<thead>
<tr>
<th>Page</th>
<th>Lemma/Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Proof of Lemma (Substitution—Well-formedness)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Uvar Preservation)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Sorting Implies Typing)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Substitution for Sorting)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Substitution for Propositions)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Substitution for Typing)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution for Sorting)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution for Prop Well-Formedness)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution for Type Well-Formedness)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution Stability)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Equal Domains)</td>
</tr>
</tbody>
</table>

## C' Properties of Extension

<table>
<thead>
<tr>
<th>Page</th>
<th>Lemma/Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>Proof of Lemma (Declaration Preservation)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Declaration Order Preservation)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Reverse Declaration Order Preservation)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Inversion)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Deep Evar Introduction)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Parallel Admissibility)</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Parallel Extension Solution)</td>
</tr>
<tr>
<td>Proof of Lemma</td>
<td>Page</td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>(Parallel Variable Update)</td>
<td>67</td>
</tr>
<tr>
<td>(Substitution Monotonicity)</td>
<td>67</td>
</tr>
<tr>
<td>(Substitution Invariance)</td>
<td>70</td>
</tr>
<tr>
<td>(Soft Extension)</td>
<td>70</td>
</tr>
<tr>
<td>(Split Extension)</td>
<td>70</td>
</tr>
<tr>
<td>(Extension Reflexivity)</td>
<td>71</td>
</tr>
<tr>
<td>(Extension Transitivity)</td>
<td>72</td>
</tr>
<tr>
<td>(Suffix Weakening)</td>
<td>74</td>
</tr>
<tr>
<td>(Extension Weakening (Sorts))</td>
<td>74</td>
</tr>
<tr>
<td>(Extension Weakening (Props))</td>
<td>74</td>
</tr>
<tr>
<td>(Extension Weakening (Types))</td>
<td>75</td>
</tr>
<tr>
<td>(Principal Agreement)</td>
<td>75</td>
</tr>
<tr>
<td>(Right-Hand Subst. for Principal Typing)</td>
<td>76</td>
</tr>
<tr>
<td>(Extension Weakening for Principal Typing)</td>
<td>76</td>
</tr>
<tr>
<td>(Inversion of Principal Typing)</td>
<td>77</td>
</tr>
<tr>
<td>(Instantiation Extension)</td>
<td>78</td>
</tr>
<tr>
<td>(Elimeq Extension)</td>
<td>78</td>
</tr>
<tr>
<td>(Elimprop Extension)</td>
<td>78</td>
</tr>
<tr>
<td>(Checkeq Extension)</td>
<td>80</td>
</tr>
<tr>
<td>(Checkprop Extension)</td>
<td>80</td>
</tr>
<tr>
<td>(Prop Equivalence Extension)</td>
<td>81</td>
</tr>
<tr>
<td>(Equivalence Extension)</td>
<td>81</td>
</tr>
<tr>
<td>(Subtyping Extension)</td>
<td>82</td>
</tr>
<tr>
<td>(Typing Extension)</td>
<td>83</td>
</tr>
<tr>
<td>(Context Partitioning)</td>
<td>84</td>
</tr>
<tr>
<td>(Completing Stability)</td>
<td>84</td>
</tr>
<tr>
<td>(Completing Completeness)</td>
<td>85</td>
</tr>
<tr>
<td>(Confluence of Completeness)</td>
<td>86</td>
</tr>
<tr>
<td>(Multiple Confluence)</td>
<td>86</td>
</tr>
<tr>
<td>(Canonical Completion)</td>
<td>86</td>
</tr>
<tr>
<td>(Split Solutions)</td>
<td>86</td>
</tr>
<tr>
<td>(Interpolating With and Exists)</td>
<td>87</td>
</tr>
<tr>
<td>(Case Invertibility)</td>
<td>87</td>
</tr>
<tr>
<td>(Well-Formed Outputs of Typing)</td>
<td>88</td>
</tr>
<tr>
<td>(Left Unsolvedness Preservation)</td>
<td>89</td>
</tr>
<tr>
<td>(Left Free Variable Preservation)</td>
<td>89</td>
</tr>
<tr>
<td>(Instantiation Size Preservation)</td>
<td>91</td>
</tr>
<tr>
<td>(Decidability of Instantiation)</td>
<td>92</td>
</tr>
<tr>
<td><strong>G' Separation</strong></td>
<td>93</td>
</tr>
<tr>
<td>-------------------</td>
<td>----</td>
</tr>
<tr>
<td>68</td>
<td>Proof of Lemma (Transitivity of Separation)</td>
</tr>
<tr>
<td>69</td>
<td>Proof of Lemma (Separation Truncation)</td>
</tr>
<tr>
<td>70</td>
<td>Proof of Lemma (Separation for Auxiliary Judgments)</td>
</tr>
<tr>
<td>71</td>
<td>Proof of Lemma (Separation for Subtyping)</td>
</tr>
<tr>
<td>72</td>
<td>Proof of Lemma (Separation—Main)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>H' Decidability of Algorithmic Subtyping</strong></th>
<th>103</th>
</tr>
</thead>
<tbody>
<tr>
<td>H'.1 Lemmas for Decidability of Subtyping</td>
<td>103</td>
</tr>
<tr>
<td>73</td>
<td>Proof of Lemma (Substitution Isn’t Large)</td>
</tr>
<tr>
<td>74</td>
<td>Proof of Lemma (Instantiation Solves)</td>
</tr>
<tr>
<td>75</td>
<td>Proof of Lemma (Checkeq Solving)</td>
</tr>
<tr>
<td>76</td>
<td>Proof of Lemma (Prop Equiv Solving)</td>
</tr>
<tr>
<td>77</td>
<td>Proof of Lemma (Equiv Solving)</td>
</tr>
<tr>
<td>78</td>
<td>Proof of Lemma (Decidability of Propositional Judgments)</td>
</tr>
<tr>
<td>79</td>
<td>Proof of Lemma (Decidability of Equivalence)</td>
</tr>
<tr>
<td>H'.2 Decidability of Subtyping</td>
<td>108</td>
</tr>
<tr>
<td>80</td>
<td>Proof of Theorem (Decidability of Subtyping)</td>
</tr>
<tr>
<td>H'.3 Decidability of Matching and Coverage</td>
<td>110</td>
</tr>
<tr>
<td>81</td>
<td>Proof of Lemma (Decidability of Guardedness Judgment)</td>
</tr>
<tr>
<td>82</td>
<td>Proof of Lemma (Decidability of Expansion Judgments)</td>
</tr>
<tr>
<td>83</td>
<td>Proof of Lemma (Expansion Shrinks Size)</td>
</tr>
<tr>
<td>84</td>
<td>Proof of Theorem (Decidability of Coverage)</td>
</tr>
<tr>
<td>H'.4 Decidability of Typing</td>
<td>113</td>
</tr>
<tr>
<td>85</td>
<td>Proof of Theorem (Decidability of Typing)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>I' Determinacy</strong></th>
<th>115</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>Proof of Lemma (Determinacy of Auxiliary Judgments)</td>
</tr>
<tr>
<td>84</td>
<td>Proof of Lemma (Determinacy of Equivalence)</td>
</tr>
<tr>
<td>85</td>
<td>Proof of Theorem (Determinacy of Subtyping)</td>
</tr>
<tr>
<td>86</td>
<td>Proof of Theorem (Determinacy of Typing)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>J' Soundness</strong></th>
<th>119</th>
</tr>
</thead>
<tbody>
<tr>
<td>J'.1 Instantiation</td>
<td>119</td>
</tr>
<tr>
<td>85</td>
<td>Proof of Lemma (Soundness of Instantiation)</td>
</tr>
<tr>
<td>86</td>
<td>Proof of Lemma (Soundness of Checkeq)</td>
</tr>
<tr>
<td>87</td>
<td>Proof of Lemma (Soundness of Propositional Equivalence)</td>
</tr>
<tr>
<td>88</td>
<td>Proof of Lemma (Soundness of Algorithmic Equivalence)</td>
</tr>
<tr>
<td>J'.2 Soundness of Checkprop</td>
<td>123</td>
</tr>
<tr>
<td>89</td>
<td>Proof of Lemma (Soundness of Checkprop)</td>
</tr>
<tr>
<td>J'.3 Soundness of Eliminations (Equality and Proposition)</td>
<td>124</td>
</tr>
<tr>
<td>90</td>
<td>Proof of Lemma (Soundness of Equality Elimination)</td>
</tr>
<tr>
<td>91</td>
<td>Proof of Theorem (Soundness of Algorithmic Subtyping)</td>
</tr>
<tr>
<td>J'.4 Soundness of Typing</td>
<td>129</td>
</tr>
<tr>
<td>92</td>
<td>Proof of Theorem (Soundness of Match Coverage)</td>
</tr>
<tr>
<td>93</td>
<td>Proof of Lemma (Well-formedness of Algorithmic Typing)</td>
</tr>
<tr>
<td>94</td>
<td>Proof of Theorem (Eagerness of Types)</td>
</tr>
<tr>
<td>95</td>
<td>Proof of Theorem (Soundness of Algorithmic Typing)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>K' Completeness</strong></th>
<th>149</th>
</tr>
</thead>
<tbody>
<tr>
<td>K'.1 Completeness of Auxiliary Judgments</td>
<td>149</td>
</tr>
<tr>
<td>92</td>
<td>Proof of Lemma (Completeness of Instantiation)</td>
</tr>
<tr>
<td>93</td>
<td>Proof of Lemma (Completeness of Checkeq)</td>
</tr>
<tr>
<td>94</td>
<td>Proof of Lemma (Completeness of Elimeq)</td>
</tr>
<tr>
<td>95</td>
<td>Proof of Lemma (Substitution Upgrade)</td>
</tr>
</tbody>
</table>
Proof of Lemma (Completeness of Propequiv) ................. 156
Proof of Lemma (Completeness of Checkprop) .................. 156
Proof of Lemma (Completeness of Equiv) ......................... 157
Proof of Theorem (Completeness of Subtyping) .................. 160

K’.2 Completeness of Equivalence and Subtyping ............... 157
Proof of Lemma (Completeness of Equiv) ......................... 157
Proof of Theorem (Completeness of Subtyping) .................. 160

K’.3 Completeness of Typing ..................................... 165
Proof of Lemma (Variable Decomposition) ....................... 165
Proof of Lemma (Pattern Decomposition and Substitution) ... 165
Proof of Lemma (Pattern Decomposition Functionality) ...... 165
Proof of Lemma (Decidability of Variable Removal) .......... 166
Proof of Lemma (Variable Inversion) .............................. 166
Proof of Theorem (Completeness of Match Coverage) ........ 166
Proof of Theorem (Completeness of Algorithmic Typing) ..... 169
1 Figures

We repeat some figures from the main paper. In Figures 8a and 14a, we include rules omitted from the main paper for space reasons.

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

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

\[
\Gamma \vdash e \Rightarrow A \quad \text{Under context } \Gamma, \text{ expression } e \text{ synthesizes output type } A
\]

\[
\begin{align*}
x : A & \mathbin{\in} \Gamma \\
\Gamma \vdash x \Rightarrow A \quad & \text{DeclVar} \\
\Gamma \vdash A \text{ type} & \\
\Gamma \vdash e \triangleleft A ! & \quad \text{DeclAnno} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma, v \mathbin{\vdash} \tau & \\
\Gamma \vdash e \triangleleft \tau & \quad \text{Decl\&I} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma, v \mathbin{\vdash} \tau, \alpha : \kappa & \\
\Gamma \vdash e \triangleleft (\exists \alpha : \kappa, A) \quad & \text{Decl\&I} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma, x : A & \mathbin{\vdash} e \triangleleft A ! \\
\Gamma \vdash \lambda x, e \Rightarrow A \Rightarrow B \quad & \text{Decl\&I} \\
\Gamma \vdash e \triangleleft A_k \quad & \text{Decl\&I_k} \\
\Gamma \vdash \text{inj}_k e \triangleleft A_1 + A_2 \quad & \text{Decl\&I_k} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash t = \text{zero} \quad & \text{DeclNil} \\
\Gamma \vdash \text{nil} \triangleleft (\text{Vec } t A) \quad & \text{Decl\&I} \\
\Gamma \vdash e \Rightarrow A \quad & \text{Decl\&E} \\
\forall B, \text{ if } \Gamma \vdash e \Rightarrow B \quad & \text{Decl\&E} \\
\Gamma \vdash \text{case } e, \Pi \triangleleft C \quad & \text{Decl\&E} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s : A & \Rightarrow C \Rightarrow q \\
\Gamma \vdash s : A & \Rightarrow C \Rightarrow q \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash t \mathbin{\triangleright} \text{true} & \\
\Gamma \vdash e \mathbin{\triangleright} A \quad & \text{Decl\&Spine} \\
\Gamma \vdash e \mathbin{\triangleright} (A \triangleright B) & \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e \mathbin{\triangleright} A \\
\Gamma \vdash e \mathbin{\triangleright} B & \\
\end{align*}
\]

\[
\begin{align*}
\forall C', \text{ if } \Gamma \vdash s : A \Rightarrow C' \Rightarrow q & \\
\Gamma \vdash s \Rightarrow A ! \Rightarrow C' \Rightarrow q & \text{DeclSpineRecover} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma / P \vdash e & \triangleleft C \\
mgu(\sigma, \tau) = \perp & \text{DeclCheck\&} \\
\Gamma / (\sigma = \tau) \vdash e & \triangleleft C \\
\end{align*}
\]

Figure 6a: Declarative typing, including rules omitted from main paper.
Γ ⊢ e ≜ A \ p ∪ \Delta

Under input context Γ, expression e checks against input type A, with output context Δ

Under input context Γ, expression e synthesizes output type A, with output context Δ

Γ ⊢ e ⇒ A \ q ∪ \Theta

Θ ⊢ e ≜ \oplus : \{\Theta\} A q ∪ \Theta

Sub

\Gamma, x: A p + e ⊢ B p + \Delta

Γ ⊢ \lambda x.e ≜ A → B p + \Delta

E

Γ, x: A p + e ⊢ B p + \Delta, x: A p, \Theta

Γ ⊢ \lambda x.e ≜ A → B p + \Delta

\Gamma, x: A p + e ⊢ B p + \Delta, x: A p, \Theta

Γ ⊢ \lambda x.e ≜ A → B p + \Delta

Case

\Gamma, \lambda x.e ≜ A → B p + \Delta

Γ ⊢ \lambda x.e ≜ A → B p + \Delta

Cons

Γ ⊢ s : A p + \Delta

Γ ⊢ \lambda x.e ≜ A → B p + \Delta

Γ, x : \Theta ⊢ e = e : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

Γ, x : \Theta ⊢ e = e : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

S

γ \vec{\Theta} : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

(\lambda x.e) : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

S

γ \vec{\Theta} : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

(\lambda x.e) : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

S

γ \vec{\Theta} : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

(\lambda x.e) : \Theta \vec{\Theta}\ vec{C} q ∪ \Delta

S

Figure 14a: Algorithmic typing, including rules omitted from main paper
Figure 16: Sorting; well-formedness of propositions, types, and contexts in the declarative system
Under context $\Gamma$, term $\tau$ has sort $\kappa$

$\Gamma \vdash \tau : \kappa$

$\frac{(u : \kappa) \in \Gamma}{\Gamma \vdash u : \kappa} \text{ VarSort}$

$\frac{(\alpha : \kappa = \tau) \in \Gamma}{\Gamma \vdash \alpha : \kappa} \text{ SolvedVarSort}$

$\frac{}{\Gamma \vdash 1 : \star} \text{ UnitSort}$

$\frac{}{\Gamma \vdash \tau_1 : \star}$

$\frac{}{\Gamma \vdash \tau_2 : \star} \text{ BinSort}$

$\frac{}{\Gamma \vdash \tau_1 + \tau_2 : \star} \text{ ZeroSort}$

$\frac{}{\Gamma \vdash \text{zero} : \mathbb{N}} \text{ SuccSort}$

Under context $\Gamma$, proposition $P$ is well-formed

$\Gamma \vdash P \text{ prop}$

$\frac{}{\Gamma \vdash t : \mathbb{N}} \text{ EqProp}$

Under context $\Gamma$, type $A$ is well-formed

$\Gamma \vdash A \text{ type}$

$\frac{(u : \star) \in \Gamma}{\Gamma \vdash u \text{ type}} \text{ VarWF}$

$\frac{(\alpha : \star = \tau) \in \Gamma}{\Gamma \vdash \alpha \text{ type}} \text{ SolvedVarWF}$

$\frac{}{\Gamma \vdash 1 : \star \text{ UnitWF}}$  

$\frac{}{\Gamma \vdash \tau_1 + \tau_2 : \star \text{ BinWF}}$

$\frac{}{\Gamma \vdash \text{zero} : \mathbb{N}} \text{ SuccWF}$

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash B \text{ type}} \in \{\to, \times, +\} \text{ BinWF}$

$\frac{}{\Gamma \vdash \text{Vec} \ t \ A \text{ type}} \text{ VecWF}$

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \forall \alpha : \kappa. A \text{ type}} \text{ ForallWF}$

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \exists \alpha : \kappa. A \text{ type}} \text{ ExistsWF}$

$\frac{\Gamma \vdash P \text{ prop}}{\Gamma \vdash A \text{ type}} \text{ ImpliesWF}$

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \land P \text{ type}} \text{ WithWF}$

Under context $\Gamma$, type $A$ is well-formed and respects principality $p$

$\Gamma \vdash A \text{ p type}$

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \text{FEV}(\Gamma[A]) = \emptyset} \text{ PrincipalWF}$

$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \not! \text{ type}} \text{ NonPrincipalWF}$

Under context $\Gamma$, types in $\vec{A}$ are well-formed [with principality $p$]

$\Gamma \vdash \vec{A} \text{ [p] types}$

for all $A \in \vec{A}$. $\Gamma \vdash A \text{ type}$

$\frac{}{\Gamma \vdash \vec{A} \text{ types}} \text{ TypevecWF}$

for all $A \in \vec{A}$. $\Gamma \vdash A \text{ p type}$

$\frac{}{\Gamma \vdash \vec{A} \text{ p types}} \text{ PrincipalTypevecWF}$

Algorithmic context $\Gamma$ is well-formed

$\Gamma \text{ ctx}$

$\frac{\text{EmptyCtx}}{\Gamma \vdash \text{ctx}}$

$\frac{\Gamma \vdash A \text{ type} \quad \text{x} \notin \text{dom}(\Gamma)}{\Gamma, x : A \not! \text{ ctx}} \text{ HypCtx}$

$\frac{\Gamma \vdash A \text{ type} \quad \text{x} \notin \text{dom}(\Gamma)}{\Gamma, \text{FEV}(\Gamma[A]) = \emptyset} \text{ HypCtx}$

$\frac{\Gamma \vdash A \text{ type} \quad \text{u} \notin \text{dom}(\Gamma)}{\Gamma, \text{VarCtx}}$

$\frac{\Gamma, \alpha : \kappa \in \Gamma \quad (\alpha = -) \notin \Gamma}{\Gamma, \alpha = \tau \text{ ctx}} \text{ EqnVarCtx}$

$\frac{\Gamma \vdash \tau : \kappa \quad \text{eqnVarCtx}}{\Gamma, \text{MarkerCtx}}$

$\frac{\Gamma \vdash \tau : \kappa \quad \text{u} \notin \Gamma}{\Gamma, \text{MarkerCtx}}$

Figure 17: 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 \( \Delta \)

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

\( \Gamma / P \vdash \Delta \perp \) Incorporate hypothesis \( P \) into \( \Gamma \), producing \( \Delta \) or inconsistency \( \perp \)

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

Figure 18: Checking and assuming propositions

\( \Gamma \vdash t_1 \triangleq t_2 : \kappa \vdash \Delta \) Check that \( t_1 \) equals \( t_2 \), taking \( \Gamma \) to \( \Delta \)

\[
\begin{align*}
\Gamma \vdash u \triangleq u : \kappa \vdash \Gamma & \quad \text{CheckeqVar} \\
\Gamma \vdash 1 \triangleq 1 : \ast \vdash \Gamma & \quad \text{CheckeqUnit} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \tau_1 \triangleq \tau_1' : \ast \vdash \Theta & \quad \Theta \vdash (\Theta)\tau_2 \triangleq (\Theta)\tau_2' : \ast \vdash \Delta & \quad \text{CheckeqBin} \\
\Gamma \vdash (\tau_1 \oplus \tau_2) \triangleq (\tau_1' \oplus \tau_2') : \ast \vdash \Delta & \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash \text{zero} \triangleq \text{zero} : N \vdash \Gamma & \quad \text{CheckeqZero} \\
\Gamma \vdash \text{succ}(t_1) \triangleq \text{succ}(t_2) : N \vdash \Delta & \quad \text{CheckeqSucc} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma[\alpha : \kappa] \vdash \alpha := t : \kappa \vdash \Delta & \quad \text{CheckeqInstL} \\
\Gamma[\alpha : \kappa] \vdash \alpha \triangleq t : \kappa \vdash \Delta & \quad \text{CheckeqInstR} \\
\end{align*}
\]

Figure 19: Checking equations

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

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

Figure 20: Head constructor clash
Figure 21: Eliminating equations

$Γ / σ ≃ τ : κ ⊣ Δ \downarrow$

Unify $σ$ and $τ$, taking $Γ$ to $Δ$, or to inconsistency $\bot$

$Γ / α ⊣ α : κ ⊣ Γ$

ElimeqUvarRef

$Γ / zero ≃ zero : N ⊣ Γ$

ElimeqZero

$α \notin FV(τ) \quad (α = -) \notin Γ$

ElimeqUvarL

$t ≠ α \quad α ∈ FV(τ)$

ElimeqUvarL

$Γ / τ ⊣ α : κ ⊣ \bot$

ElimeqUvarR

$Γ / 1 ≃ 1 : * ⊣ Γ$

ElimeqUnit

$Γ / (τ_1 ⊕ τ_2) ≃ (* τ_1') : * ⊣ Δ \downarrow$

ElimeqBin

$Γ / (τ_1 ⊕ τ_2) ≃ (τ_1' ⊕ τ_2) : * ⊣ Δ \downarrow$

ElimeqBinBot

$σ \# τ$

ElimeqClash
Under input context $\Gamma$, type $A$ is a subtype of $B$, with output context $\Delta$

$$\Delta\mid A \subsetneq^P B \mid \Delta$$

B not headed by $\forall$

$$\Gamma \vdash \forall \alpha : \kappa. A \subsetneq^\Delta \Gamma, \alpha \Theta \quad \Gamma \vdash \forall \alpha : \kappa. A \subsetneq^\Delta \Gamma \quad \vdash : \text{Equiv}$$

A not headed by $\exists$

$$\Gamma, \beta, \beta : \kappa \vdash A \subsetneq^\Delta \exists \beta \vdash \Gamma, \beta \Theta \quad \Gamma \vdash \exists \beta : \kappa. B \vdash \Delta \quad \vdash : \exists \text{R}$$

$$\Gamma, \alpha : \kappa \vdash A \subsetneq^\Delta B \vdash \Delta, \alpha : \kappa \Theta \quad \Gamma \vdash \exists \alpha : \kappa. A \subsetneq^\Delta B \vdash \Delta \quad \vdash : \exists \text{L}$$

$$\Gamma \vdash A \subsetneq B \vdash \Delta$$

$$\Gamma \vdash A \subsetneq B \vdash \Delta$$

$$\Gamma \vdash A \subsetneq B \vdash \Delta$$

$$\Gamma \vdash A \subsetneq B \vdash \Delta$$

$$\Gamma \vdash A \subsetneq B \vdash \Delta$$

$$\Gamma \vdash A \subsetneq B \vdash \Delta$$

$$\Delta \mid P \equiv Q \mid \Delta$$

Under input context $\Gamma$, check that $P$ is equivalent to $Q$ with output context $\Delta$

$$\Gamma \vdash t_1 \equiv t_2 : \text{N} \vdash \Theta \quad \Theta \vdash [\Theta] t_1' \equiv [\Theta] t_2' : \text{N} \vdash \Delta \quad \equiv \text{PropEq}$$

$$\Delta \mid A \equiv B \mid \Delta$$

Under input context $\Gamma$, check that $A$ is equivalent to $B$ with output context $\Delta$

$$\Gamma \vdash \alpha \equiv \alpha \dashv \Gamma \equiv \text{Var}$$

$$\Gamma \vdash \alpha \equiv \alpha \dashv \Gamma \equiv \text{Exvar}$$

$$\Gamma \vdash 1 \equiv 1 \dashv \Gamma \equiv \text{Unit}$$

$$\Gamma \vdash A_1 \equiv B_1 \vdash \Theta \quad \Theta \vdash [\Theta] A_2 \equiv [\Theta] B_2 \vdash \Delta \quad \equiv \oplus$$

$$\Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \vdash \Delta \quad \equiv \oplus$$

$$\Gamma, \alpha : \kappa \vdash A \equiv B \vdash \Delta, \alpha : \kappa, \Delta' \equiv \forall$$

$$\Gamma \vdash (\forall \alpha : \kappa. A) \equiv (\forall \alpha : \kappa. B) \vdash \Delta \quad \equiv \forall$$

$$\Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta] A \equiv [\Theta] B \vdash \Delta \quad \equiv \ominus$$

$$\Gamma \vdash (P \ominus A) \equiv (Q \ominus B) \vdash \Delta \quad \equiv \ominus$$

$$\tilde{\alpha} \notin \text{FV}(\tau) \quad \Gamma[\tilde{\alpha}] \vdash \tilde{\alpha} : \tau \vdash \Delta \quad \equiv \text{InstantiateL}$$

$$\Gamma[\tilde{\alpha}] \vdash \tilde{\alpha} \equiv \tau \vdash \Delta \quad \equiv \text{InstantiateR}$$

Figure 22: Algorithmic subtyping and equivalence
\(\Gamma \vdash \vec{\alpha} := t : \kappa \rightarrow \Delta\) Under input context \(\Gamma\), instantiate \(\vec{\alpha}\) such that \(\vec{\alpha} = t\) with output context \(\Delta\)

\[
\begin{align*}
\Gamma_0 \vdash \tau : \kappa \\
\Gamma_0, \vec{\alpha} : \kappa, \Gamma_1 \vdash \vec{\alpha} := \tau : \kappa \rightarrow \Gamma_0, \vec{\alpha} : \kappa = \tau, \Gamma_1 & \quad \text{InstSolve} \\
\vec{\beta} \in \text{unsolved}(\Gamma[\vec{\alpha} : \kappa][\vec{\beta} : \kappa]) \\
\Gamma[\vec{\alpha} : \kappa][\vec{\beta} : \kappa] \vdash \vec{\alpha} := \vec{\beta} : \kappa \rightarrow \Gamma[\vec{\alpha} : \kappa][\vec{\beta} : \kappa = \vec{\alpha}] & \quad \text{InstReach} \\
\Gamma[\vec{\alpha}_2 : \star, \vec{\alpha}_1 : \star = \vec{\alpha}_1 \oplus \vec{\alpha}_2] \vdash \vec{\alpha}_1 := \tau_1 : \star \rightarrow \Theta & \quad \Theta \vdash \vec{\alpha}_2 := [\Theta]\tau_2 : \star \rightarrow \Delta & \quad \text{InstBin} \\
\Gamma[\vec{\alpha} : \star] \vdash \vec{\alpha} := \tau_1 \oplus \tau_2 : \star \rightarrow \Delta
\end{align*}
\]

\(\Gamma[\vec{\alpha} : \kappa] \vdash \vec{\alpha} := \text{zero} : \kappa \rightarrow \Gamma[\vec{\alpha} : \kappa = \text{zero}] \quad \text{InstZero} \)

\(\Gamma[\vec{\alpha} : \kappa] \vdash \vec{\alpha} := \text{succ} : \kappa \rightarrow \Gamma[\vec{\alpha} : \kappa = \text{succ}(\vec{\alpha})] \quad \text{InstSucc} \)

\(\Gamma[\vec{\alpha} : \kappa] \vdash \vec{\alpha} := \text{succ} : \kappa \rightarrow \Gamma[\vec{\alpha} : \kappa = \text{succ}(\tau_1) : \kappa \rightarrow \Delta] \quad \text{InstSucc} \)

Figure 23: Instantiation
1 Figures

Under context $\Gamma$, check branches $\Pi$ with patterns of type $\bar{A}$ and bodies of type $C$

\[
\Gamma \vdash \Pi :: \bar{A} \bot \leq C \ p \vdash \Delta
\]

Under context $\Gamma$, incorporate proposition $P$ while checking branches $\Pi$ with patterns of type $\bar{A}$ and bodies of type $C$

\[
\Gamma / P \vdash \Pi :: \bar{A} \bot \leq C \ p \vdash \Delta
\]

Figure 24: Algorithmic pattern matching
Figure 25: Algorithmic match coverage
## 2 List of Judgments

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 16</td>
</tr>
<tr>
<td>$\Psi \vdash P \prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Psi \vdash A \type$</td>
<td>Type is well-formed</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Psi \vdash \vec{A} \types$</td>
<td>Type vector is well-formed</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Psi \ctx$</td>
<td>Declarative context is well-formed</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Psi \vdash A \leq_P B$</td>
<td>Declarative subtyping</td>
<td>Figure 4</td>
</tr>
<tr>
<td>$\Psi \vdash P \true$</td>
<td>Declarative truth</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash e \Leftarrow A \leftarrow P$</td>
<td>Declarative checking</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash e \Rightarrow A \leftarrow P$</td>
<td>Declarative synthesis</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \leftarrow P \Rightarrow C \q$</td>
<td>Declarative spine typing</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \leftarrow P \Rightarrow C \ceil{q}$</td>
<td>Declarative spine typing, recovering principality</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi :: \vec{A} \leftarrow C \leftarrow P$</td>
<td>Declarative pattern matching</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi :: \vec{A} \leftarrow C \leftarrow P$</td>
<td>Declarative proposition assumption</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi \covers \vec{A} \leftarrow C \leftarrow P$</td>
<td>Declarative match coverage</td>
<td>Figure 8</td>
</tr>
<tr>
<td>$\Gamma \vdash \tau : \kappa$</td>
<td>Index term/monotype is well-formed</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash P \prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash A \type$</td>
<td>Polytype is well-formed</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \ctx$</td>
<td>Algorithmic context is well-formed</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash A &lt;: P \B \leftarrow \Delta$</td>
<td>Algorithmic subtyping</td>
<td>Figure 22</td>
</tr>
<tr>
<td>$\Gamma \vdash A &lt;: B \leftarrow \Delta$</td>
<td>Algorithmic proposition assumption</td>
<td>Figure 22</td>
</tr>
<tr>
<td>$\Gamma \vdash A \equiv B \leftarrow \Delta$</td>
<td>Equivalence of types</td>
<td>Figure 22</td>
</tr>
<tr>
<td>$\Gamma \vdash \alpha \Leftarrow t : \kappa \leftarrow \Delta$</td>
<td>Instantiate</td>
<td>Figure 23</td>
</tr>
<tr>
<td>$\Gamma \vdash e \Leftarrow A \prop \Leftarrow \Delta$</td>
<td>Algorithmic checking</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Gamma \vdash e \Rightarrow A \prop \Leftarrow \Delta$</td>
<td>Algorithmic synthesis</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Gamma \vdash s : A \prop \Rightarrow C \q \leftarrow \Delta$</td>
<td>Algorithmic spine typing</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Gamma \vdash s : A \prop \Rightarrow C \ceil{q} \leftarrow \Delta$</td>
<td>Algorithmic spine typing, recovering principality</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi :: \vec{A} \q \Leftarrow C \prop \Leftarrow \Delta$</td>
<td>Algorithmic pattern matching</td>
<td>Figure 24</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi :: \vec{A} \Leftarrow C \prop \Leftarrow \Delta$</td>
<td>Algorithmic pattern matching (assumption)</td>
<td>Figure 24</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi \covers \vec{A} \q$</td>
<td>Algorithmic match coverage</td>
<td>Figure 25</td>
</tr>
<tr>
<td>$\Gamma \rightarrow \Delta$</td>
<td>Context extension</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$[\Omega] \Gamma$</td>
<td>Apply complete context</td>
<td>Figure 13</td>
</tr>
</tbody>
</table>
A Properties of the Declarative System

Lemma 1 (Declarative Well-foundedness). 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 $\Psi \vdash s : A p \gg B \lceil q \rceil$
(vi) pattern matching $\Psi \vdash \Pi :: \vec{A} ! \Leftarrow C p$
(vii) pattern matching, equality elimination $\Psi / P \vdash \Pi :: \vec{A} ! \Leftarrow C p$

Lemma 2 (Declarative Weakening). Go to proof
(i) If $\Psi_0, \Psi_1 \vdash t : \kappa$ then $\Psi_0, \Psi, \Psi_1 \vdash t : \kappa$.
(ii) If $\Psi_0, \Psi_1 \vdash P \ prop$ then $\Psi_0, \Psi, \Psi_1 \vdash P \ prop$.
(iii) If $\Psi_0, \Psi_1 \vdash P \ true$ then $\Psi_0, \Psi, \Psi_1 \vdash P \ true$.
(iv) If $\Psi_0, \Psi_1 \vdash A \ type$ then $\Psi_0, \Psi, \Psi_1 \vdash A \ type$.

Lemma 3 (Declarative Term Substitution). 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 $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P \ prop$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]P \ prop$.
3. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \ type$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A \ type$.
4. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \leq^P B$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A \leq^P [t/\alpha]B$.
5. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P \ true$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]P \ true$.

Lemma 4 (Reflexivity of Declarative Subtyping). Go to proof
Given $\Psi \vdash A \ type$, we have that $\Psi \vdash A \leq^P A$.

Lemma 5 (Subtyping Inversion). 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). 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 $\Psi \vdash A \leq^P B$ then $A = B$. 
Lemma 7 (Transitivity of Declarative Subtyping). \[ \text{Go to proof} \] Given $\Psi \vdash A$ type and $\Psi \vdash B$ type and $\Psi \vdash C$ type:

(i) If $D_1 : \Psi \vdash A \leq^P B$ and $D_2 : \Psi \vdash B \leq^P C$
then $\Psi \vdash A \leq^P C$.

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, $\text{FEV}(e) = \emptyset$.

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

(ii) If $\Gamma \vdash P$ prop and $\Gamma \vdash \tau \ p$ type then $\Gamma \vdash [\tau/\alpha]P$ prop.
Moreover, if $p = !$ and $\text{FEV}([\Gamma]P) = \emptyset$ then $\text{FEV}([\Gamma][\tau/\alpha]P) = \emptyset$.

Lemma 9 (Uvar Preservation). \[ \text{Go to proof} \] If $\Delta \longrightarrow \Omega$ then:

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

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

Lemma 10 (Sorting Implies Typing). \[ \text{Go to proof} \] If $\Gamma \vdash t : \ast$ then $\Gamma \vdash t$ type.

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

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

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

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

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

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

Lemma 17 (Substitution Stability). \[ \text{Go to proof} \] 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). \[ \text{Go to proof} \] If $\Omega_1 \vdash A$ type and $\text{dom}(\Omega_1) = \text{dom}(\Omega_2)$ then $\Omega_2 \vdash A$ type.

C Properties of Extension

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

Lemma 20 (Declaration Order Preservation). \[ \text{Go to proof} \] If $\Gamma \longrightarrow \Delta$ 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). \[ \text{Go to proof} \] If $\Gamma \longrightarrow \Delta$ 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 $[\Gamma]A = [\Gamma][\Theta](\Lambda)$ and $[\Gamma]A = [\Theta](\Gamma)\Lambda$.

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 $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ where $\Delta' < D$.

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

(ii) If $D :: \Gamma_0, \bullet u, \Gamma_1 \rightarrow \Delta$

then there exist unique $\Delta_0$ and $\Delta_1$

such that $\Delta = (\Delta_0, \bullet u, \Delta_1)$ and $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ where $\Delta' < D$.

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

Moreover, if dom($\Gamma_0, \bullet u, \Gamma_1$) = dom($\Delta$) then dom($\Gamma_0$) = 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 $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]\tau = [\Delta_0]\tau'$ where $\Delta' < D$.

(iv) If $D :: \Gamma_0, \& : \kappa = \tau, \Gamma_1 \rightarrow \Delta$

then there exist unique $\Delta_0, \tau'$, and $\Delta_1$

such that $\Delta = (\Delta_0, \& : \kappa = \tau', \Delta_1)$ and $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]\tau = [\Delta_0]\tau'$ where $\Delta' < D$.

(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 $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ and $[\Delta_0]A = [\Delta_0]A'$ where $\Delta' < D$.

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

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

(vi) If $D :: \Gamma_0, \& : \kappa, \Gamma_1 \rightarrow \Delta$

then either

- there exist unique $\Delta_0, \tau'$, and $\Delta_1$

  such that $\Delta = (\Delta_0, \& : \kappa = \tau', \Delta_1)$ and $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ where $\Delta' < D$,

  or

- there exist unique $\Delta_0$ and $\Delta_1$

  such that $\Delta = (\Delta_0, \& : \kappa, \Delta_1)$ and $\Delta' :: \Gamma_0 \rightarrow \Delta_0$ where $\Delta' < D$.

Lemma 23 (Deep Evar Introduction). Go to proof

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

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

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

Lemma 24 (Soft Extension). Go to proof

If $\Delta$ and $\Gamma, \Theta$ ctx and $\Theta$ is soft, then there exists $\Omega$ such that dom($\Theta$) = 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 \( \Gamma \longrightarrow \Omega \) and \( (\Gamma, \Theta) \) is well-formed, then \( \Gamma, \Theta \longrightarrow \Omega, \Theta \).

Proof. By induction on \( \Theta \), following the definition of \(|-|\) and applying the rules for \( \longrightarrow \).

Lemma 26 (Parallel Admissibility). Go to proof

If \( \Gamma_l \longrightarrow \Delta_l \) and \( \Gamma_l, \Gamma_r \longrightarrow \Delta_l, \Delta_r \) then:

(i) \( \Gamma_l, \hat{\alpha} : \kappa, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa, \Delta_r \)

(ii) If \( \Delta_l \vdash \tau' : \kappa \) then \( \Gamma_l, \hat{\alpha} : \kappa, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa = \tau', \Delta_r \).

(iii) If \( \Gamma_l \vdash \tau : \kappa \) and \( \Delta_l \vdash \tau' \) type and \( |\Delta_l|\tau = |\Delta_l|\tau' \), then \( \Gamma_l, \hat{\alpha} : \kappa = \tau, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa = \tau', \Delta_r \).

Lemma 27 (Parallel Extension Solution). Go to proof

If \( \Gamma_l, \hat{\alpha} : \kappa, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa = \tau', \Delta_r \) and \( \Gamma_l \vdash \tau : \kappa \) and \( |\Delta_l|\tau = |\Delta_l|\tau' \) then \( \Gamma_l, \hat{\alpha} : \kappa = \tau, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa = \tau', \Delta_r \).

Lemma 28 (Parallel Variable Update). Go to proof

If \( \Gamma_l, \hat{\alpha} : \kappa, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa = \tau_0, \Delta_r \) and \( \Gamma_l \vdash \tau_1 : \kappa \) and \( |\Delta_l|\tau_0 = |\Delta_l|\tau_1 = |\Delta_l|\tau_2 \) then \( \Gamma_l, \hat{\alpha} : \kappa = \tau_1, \Gamma_r \longrightarrow \Delta_l, \hat{\alpha} : \kappa = \tau_2, \Delta_r \).

Lemma 29 (Substitution Monotonicity). Go to proof

(i) If \( \Gamma \longrightarrow \Delta \) and \( \Gamma \vdash t : \kappa \) then \( |\Delta|\Gamma t = |\Delta|t \).

(ii) If \( \Gamma \longrightarrow \Delta \) and \( \Gamma \vdash \text{prop} \) then \( |\Delta|\Gamma \text{prop} = |\Delta|\text{prop} \).

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

Lemma 30 (Substitution Invariance). Go to proof

(i) If \( \Gamma \longrightarrow \Delta \) and \( \Gamma \vdash t : \kappa \) and \( \text{FEV}([\Gamma]t) = \emptyset \) then \( |\Delta|\Gamma t = |\Gamma|t \).

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

(iii) If \( \Gamma \longrightarrow \Delta \) 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 if, for all \( (\hat{\alpha} : \kappa = t) \) and \( (\alpha = t) \in \Omega \), the solution \( t \) is ground \( (\text{FEV}(t) = \emptyset) \).

Lemma 31 (Split Extension). Go to proof

If \( \Delta \longrightarrow \Omega \)
and \( \hat{\alpha} \in \text{unsolved}(\Delta) \)
and \( \Omega = \Omega_1[\hat{\alpha} : \kappa = t_1] \)
and \( \Omega \) is canonical (Definition 3)
and \( \Omega \vdash t_2 : \kappa \)
them \( \Delta \longrightarrow \Omega_1[\hat{\alpha} : \kappa = t_2] \).
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 \( \Gamma \vdash P \text{ true} \vdash \Delta \) and \( \Gamma \rightarrow \Gamma' \) then there exists \( \Delta' \) such that \( \Delta \rightarrow \Delta' \) and \( \Gamma' \vdash P \text{ true} \vdash \Delta' \).

Counterexample: Suppose \( \hat{\alpha} \vdash \hat{\alpha} = 1 \text{ true} \vdash \hat{\alpha} = 1 \) and \( \hat{\alpha} \rightarrow (\hat{\alpha} = (1 \rightarrow 1)) \). 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 \( \text{FEV}(P) = \emptyset \) 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 \( \Gamma \vdash (P \lor A) \) 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 \( \Gamma \vdash \alpha : \tau : \kappa \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).
C.5 Equivalence Extends

Lemma 44 (Elimeq Extension).  \[ \text{If } \Gamma \vdash t : \kappa \rightarrow \Delta \text{ then there exists } \Theta \text{ such that } \Gamma, \Theta \rightarrow \Delta. \]

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

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

Lemma 47 (Checkprop Extension).  \[ \text{If } \Gamma \vdash P \text{ true } \rightarrow \Delta \text{ then } \Gamma \rightarrow \Delta. \]

Lemma 48 (Prop Equivalence Extension).  \[ \text{If } \Gamma \vdash A \equiv B \rightarrow \Delta \text{ then } \Gamma \rightarrow \Delta. \]

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

C.6 Subtyping Extends

Lemma 50 (Subtyping Extension).  \[ \text{If } \Gamma \vdash A <: \equiv B \rightarrow \Delta \text{ then } \Gamma \rightarrow \Delta. \]

C.7 Typing Extends

Lemma 51 (Typing Extension).  \[ \text{If } \Gamma \vdash e \Leftarrow A \rightarrow \Delta \text{ or } \Gamma \vdash e \Rightarrow A \rightarrow \Delta \text{ or } \Gamma \vdash s : A \rightarrow B \rightarrow \Delta \text{ or } \Gamma \vdash \Pi : A \rightarrow C \rightarrow \Delta \text{ or } \Gamma / P \vdash \Pi :: \bar{A} \rightarrow \Delta \rightarrow \Delta \text{ then } \Gamma \rightarrow \Delta. \]

C.8 Unfiled

Lemma 52 (Context Partitioning).  \[ \text{If } \Delta, \omega, \Theta \rightarrow \Omega, \omega, \Omega_p \text{ then there is a } \Psi \text{ such that } [\Omega, \omega, \Omega_p] (\Delta, \omega, \Theta) = [\Omega] \Delta, \Psi. \]

Lemma 53 (Softness Goes Away).
\[ \text{If } \Delta, \Theta \rightarrow \Omega, \Omega_p \text{ where } \Delta \rightarrow \Omega \text{ and } \Theta \text{ is soft, then } [\Omega, \Theta] = [\Omega]. \]

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

Lemma 54 (Completing Stability).  \[ \text{If } \Gamma \rightarrow \Omega \text{ then } [\Omega] \Gamma = [\Omega] \Omega. \]

Lemma 55 (Completing Completeness).  \[ \text{If } \Omega \rightarrow \Omega' \text{ and } \Omega \vdash t : \kappa \text{ then } [\Omega] t = [\Omega'] t. \]

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

Lemma 57 (Multiple Confluence).  \[ \text{If } \Delta \rightarrow \Omega \text{ and } \Omega \rightarrow \Omega' \text{ and } \Delta' \rightarrow \Omega' \text{ then } [\Omega] \Delta = [\Omega'] \Delta'. \]
Lemma 58 (Bundled Substitution for Sorting). If $\Gamma \vdash t : \kappa$ and $\Gamma \rightarrow \Omega$ then $[\Omega]\Gamma \vdash [\Omega]t : \kappa$.

Proof.

Given

By Lemma 36 (Extension Weakening (Sorts))

By Lemma 14 (Substitution for Sorting)

By Lemma 32 (Extension Reflexivity)

By Lemma 56 (Confluence of Completeness)

$\square$

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 $\hat{\alpha} : \kappa = \tau$ 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). If $\Delta \rightarrow \Omega$ and $\bar{\alpha} \in \text{unsolved}(\Delta)$ then there exists $\Omega_1 = \Omega_1[\bar{\alpha} : \kappa = t_1]$ such that $\Omega_1 \rightarrow \Omega$ and $\Omega_2 = \Omega_2[\bar{\alpha} : \kappa = t_2]$ where $\Delta \rightarrow \Omega_2$ and $t_2 \neq t_1$ and $\Omega_2$ is canonical.

D Internal Properties of the Declarative System

Lemma 61 (Interpolating With and Exists). (1) If $D :: \Psi \vdash \Pi :: \bar{\alpha} ! \leftarrow C \ᾶ p$ and $\Psi \vdash P_0 \text{ true}$

then $D' :: \Psi \vdash \Pi :: \bar{\alpha} ! \leftarrow C \land P_0 \ p$.

(2) If $D :: \Psi \vdash \Pi :: \bar{\alpha} ! \leftarrow [\tau/\alpha]C_0 \ p$ and $\Psi \vdash [\tau/\alpha]C_0 \ p : \kappa$

then $D' :: \Psi \vdash \Pi :: \bar{\alpha} ! \leftarrow (\exists \alpha : \kappa.C_0) \ p$.

In both cases, the height of $D'$ is one greater than the height of $D$.
Moreover, similar properties hold for the eliminating judgment $\Psi / P \vdash \Pi :: \bar{\alpha} ! \leftarrow C \ p$.

Lemma 62 (Case Invertibility). If $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C \ p$

then $\Psi \vdash e_0 \Rightarrow \Lambda !$ and $\Psi \vdash \Pi :: \Lambda ! \leftarrow C \ p$ and $\Psi \vdash \Pi \text{ covers } \Lambda !$

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). (Spines) If $\Gamma \vdash s : A \ q \Rightarrow C \ p \rightarrow \Delta$ or $\Gamma \vdash s : A \ q \Rightarrow C \ [p] \rightarrow \Delta$

and $\Gamma \vdash A \ q \text{ type}$

then $\Delta \vdash C \ p \text{ type}$.

(Synthesis) If $\Gamma \vdash e \Rightarrow A \ p \rightarrow \Delta$

then $A \vdash p \text{ type}$.
F  Decidability of Instantiation

Lemma 64 (Left Unsolvedness Preservation). If $\Gamma_0, \alpha : \kappa \vdash \Delta$ and $\beta \in \text{unsolved}(\Gamma_0)$ then $\beta \in \text{unsolved}(\Delta)$.

Lemma 65 (Left Free Variable Preservation). If $\Gamma_0, \alpha : \kappa, \Gamma_1 \vdash \Delta$ and $\beta \not\in \text{FV}([\Gamma]s)$ and $\beta \in \text{unsolved}(\Gamma_0)$ and $\beta \not\in \text{FV}(\Delta|s)$, then $\beta \not\in \text{FV}(\Delta|s)$.

Lemma 66 (Instantiation Size Preservation). If $\Gamma_0, \alpha : \kappa, \Gamma_1 \vdash \Delta$ and $\beta \not\in \text{FV}(\Delta|s)$, then $|\Gamma|s = |\Delta|s|$, where $|C|$ is the plain size of the term $C$.

Lemma 67 (Decidability of Instantiation). If $\Gamma = \Gamma_0 : \kappa'$ and $\Gamma \vdash t : \kappa$ such that $[\Gamma]t = t$ and $\alpha \not\in \text{FV}(t)$, then:

1. Either there exists $\Delta$ such that $\Gamma_0 [\alpha : \kappa'] \vdash \Delta : \kappa - \Delta$, or not.

G  Separation

Definition 4 (Separation). An algorithmic context $\Gamma$ is separable and written $\Gamma_L * \Gamma_R$ if (1) $\Gamma = (\Gamma_L, \Gamma_R)$ and (2) for all $(\alpha : \kappa = \tau) \in \Gamma_R$ it is the case that $\text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$.

Any context $\Gamma$ is separable into, at least, $\ast \Gamma$ and $\Gamma \ast$.

Definition 5 (Separation-Preserving Extension). The separated context $\Gamma_L * \Gamma_R$ extends to $\Delta_L * \Delta_R$, written

$$(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$$

if $(\Gamma_L, \Gamma_R) \rightarrow (\Delta_L, \Delta_R)$ and $\text{dom}(\Gamma_L) \subseteq \text{dom}(\Delta_L)$ and $\text{dom}(\Gamma_R) \subseteq \text{dom}(\Delta_R)$.

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 $(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$ rather than $(\Gamma_L * \Gamma_R) \rightarrow (\Delta_L * \Delta_R)$, because only $\xrightarrow{\ast}$ includes the domain conditions. For example, $(\alpha * \beta) \rightarrow (\alpha, \beta = \alpha) *$, but the variable $\beta$ has “crossed over” to the left of $*$ in the context $(\alpha, \beta = \alpha) *$.

Lemma 68 (Transitivity of Separation). $\xrightarrow{\ast}$

If $(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Theta_L * \Theta_R)$ and $(\Theta_L * \Theta_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$

then $(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$.

Lemma 69 (Separation Truncation). $\xrightarrow{\ast}$

If $H$ has the form $\alpha : \kappa$ or $\uparrow \alpha$ or $\uparrow p$ or $\alpha : A p$

and $(\Gamma_L * (\Gamma_R, H)) \xrightarrow{\ast} (\Delta_L * \Delta_R)$

then $(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$ where $\Delta_R = (\Delta_0, H, \Theta)$.

Lemma 70 (Separation for Auxiliary Judgments). $\xrightarrow{\ast}$

(i) If $\Gamma_L * \Gamma_R \vdash \sigma \equiv \tau : \kappa - \Delta$

and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$

then $\Delta = (\Delta_L * \Delta_R)$ and $(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$.

(ii) If $\Gamma_L * \Gamma_R \vdash p$ true $\rightarrow \Delta$

and $\text{FEV}(P) \subseteq \text{dom}(\Gamma_R)$

then $\Delta = (\Delta_L * \Delta_R)$ and $(\Gamma_L * \Gamma_R) \xrightarrow{\ast} (\Delta_L * \Delta_R)$.

August 15, 2020
(iii) If $\Gamma_L \vdash \Gamma_R / \sigma \triangleq \tau : \kappa \vdash \Delta$
and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset$
then $\Delta = (\Delta_L \ast (\Delta_R, \Theta))$ and $(\Gamma_L \ast (\Gamma_R, \Theta)) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(iv) If $\Gamma_L \ast \Gamma_R / \Pi \vdash \Delta$
and $\text{FEV}(\Pi) = \emptyset$
then $\Delta = (\Delta_L \ast (\Delta_R, \Theta))$ and $(\Gamma_L \ast (\Gamma_R, \Theta)) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(v) If $\Gamma_L \ast \Gamma_R / \alpha := \tau : \kappa \vdash \Delta$
and $(\text{FEV}(\tau) \cup \{\alpha\}) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(vi) If $\Gamma_L \ast \Gamma_R / \Pi \vdash \Delta$
and $\text{FEV}(\Pi) \cup \text{FEV}(\Delta) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(vii) If $\Gamma_L \ast \Gamma_R / A \equiv B \vdash \Delta$
and $\text{FEV}(A) \cup \text{FEV}(B) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

Lemma 71 (Separation for Subtyping). Go to proof
If $\Gamma_L \ast \Gamma_R / A \vdash^P B \vdash \Delta$
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 $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

Lemma 72 (Separation—Main). Go to proof
(Spines) If $\Gamma_L \ast \Gamma_R / s : A \triangleright C \vdash \Delta$
or $\Gamma_L \ast \Gamma_R / s : A \triangleright C [\triangleright q] \vdash \Delta$
and $\Gamma_L \ast \Gamma_R / A \vdash \Delta$
and $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$ and $\text{FEV}(C) \subseteq \text{dom}(\Delta_R)$.

(Checking) If $\Gamma_L \ast \Gamma_R / e \equiv C \vdash \Delta$
and $\Gamma_L \ast \Gamma_R / C \vdash \Delta$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(Synthesis) If $\Gamma_L \ast \Gamma_R / e \Rightarrow C \vdash \Delta$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(Match) If $\Gamma_L \ast \Gamma_R / \Pi : \bar{A} q \equiv C \vdash \Delta$
and $\text{FEV}(A) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

(Match Elim.) If $\Gamma_L \ast \Gamma_R / \Pi : \bar{A} ! \equiv C \vdash \Delta$
and $\text{FEV}(\Pi) = \emptyset$
and $\text{FEV}(A) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\rightarrow^e (\Delta_L \ast \Delta_R)$.

H Decidability of Algorithmic Subtyping

Definition 6. The following connectives are large:

\[ \forall \quad \exists \quad \land \]
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 $\alpha$ is denoted by $\#_{\text{large}}(\alpha)$.

### H.1 Lemmas for Decidability of Subtyping

**Lemma 73** (Substitution Isn’t Large). Go to proof
For all contexts $\Theta$, we have $\#_{\text{large}}([\Theta]\alpha) = \#_{\text{large}}(\alpha)$.

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

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

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

**Lemma 77** (Equiv Solving). Go to proof
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). Go to proof
The following judgments are decidable, with $\Delta$ as output in (1)–(3), and $\Delta \bot$ 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. $\Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta$
2. $\Gamma \vdash P \text{ true} \vdash \Delta$
3. $\Gamma \vdash P \equiv Q \vdash \Delta$
4. $\Gamma / \sigma \equiv t : \kappa \vdash \Delta \bot$
5. $\Gamma / P \equiv \Delta \bot$

**Lemma 79** (Decidability of Equivalence). Go to proof
Given a context $\Gamma$ and types $A, B$ such that $\Gamma \vdash A \text{ type}$ and $\Gamma \vdash B \text{ type}$ 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). Go to proof
Given a context $\Gamma$ and types $A, B$ such that $\Gamma \vdash A \text{ type}$ and $\Gamma \vdash B \text{ type}$ and $[\Gamma] A = A$ and $[\Gamma] B = B$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash A <: B \vdash \Delta$.

### H.3 Decidability of Matching and Coverage

**Lemma 80** (Decidability of Guardedness Judgment). Go to proof
For any set of branches $\Pi$, the relation $\Pi \text{ guarded}$ is decidable.

**Lemma 81** (Decidability of Expansion Judgments). Go to proof
Given branches $\Pi$, it is decidable whether:

1. there exists a unique $\Pi'$ such that $\Pi \prec \Pi'$;
2. there exist unique $\Pi_L$ and $\Pi_R$ such that $\Pi \prec\prec \Pi_L \parallel \Pi_R$;
3. there exists a unique $\Pi'$ such that $\Pi \prec\prec \Pi'$;

August 15, 2020
(4) there exists a unique $\Pi'$ such that $\Pi \overset{1}{\rightarrow} \Pi'$.

(5) there exist unique $\Pi_1$ and $\Pi_2$ such that $\Pi \overset{\text{vec}}{\rightarrow} \Pi_1 \parallel \Pi_2$.

**Lemma 82** (Expansion Shrinks Size). Go to proof

We define the size of a pattern $|p|$ as follows:

- $|x| = 0$
- $|\_| = 0$
- $|(p, p')| = 1 + |p| + |p'|$
- $|\Omega| = 0$
- $|\text{inj}_1 p| = 1 + |p|$
- $|\text{inj}_2 p| = 1 + |p|$
- $|\square| = 1$
- $|p :: p'| = 1 + |p| + |p'|$

We lift size to branches $\pi = \vec{p} \Rightarrow e$ as follows:

- $|p_1, \ldots, p_n \Rightarrow e| = |p_1| + \ldots + |p_n|$

We lift size to branch lists $\Pi = \pi_1 \mid \ldots \mid \pi_n$ as follows:

- $|\pi_1 \mid \ldots \mid \pi_n| = |\pi_1| + \ldots + |\pi_n|$

Now, the following properties hold:

1. If $\Pi \overset{\text{ex}}{\rightarrow} \Pi'$ then $|\Pi| = |\Pi'|$.
2. If $\Pi \overset{1}{\rightarrow} \Pi'$ then $|\Pi| = |\Pi'|$.
3. If $\Pi \overset{\text{vec}}{\rightarrow} \Pi'$ then $|\Pi| \leq |\Pi'|$.
4. If $\Pi \overset{\text{L}}{\rightarrow} \Pi_1 \parallel \Pi_R$ then $|\Pi| \leq |\Pi_1|$ and $|\Pi| \leq |\Pi_2|$.
5. If $\Pi \overset{\text{vec}}{\rightarrow} \Pi_1 \parallel \Pi_2$ then $|\Pi_1| \leq |\Pi|$ and $|\Pi_2| \leq |\Pi|$.
6. If $\Pi$ guarded and $\Pi \overset{\text{vec}}{\rightarrow} \Pi_1 \parallel \Pi_2$ then $|\Pi_1| < |\Pi|$ and $|\Pi_2| < |\Pi|$.

**Theorem 2** (Decidability of Coverage). Go to proof

Given a context $\Gamma$, branches $\Pi$ and types $A$, it is decidable whether $\Gamma \vdash \Pi$ covers $\bar{A}$ if $q$ is derivable.

**H.4 Decidability of Typing**

**Theorem 3** (Decidability of Typing). Go to proof

(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 $\Gamma \vdash e \Rightarrow A \ p \vdash \Delta$.

(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 $\Gamma \vdash s : A \ p \Rightarrow B \ q \vdash \Delta$.

(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 $\Gamma \vdash e \Leftarrow B \ p \vdash \Delta$.

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

Also, if given a proposition $P$ as well, it is decidable whether there exists $\Delta$ such that $\Gamma / P \vdash \Pi :: \bar{A} \ ! \Leftarrow C \ p \vdash \Delta$. 
I Determinacy

Lemma 83 (Determinacy of Auxiliary Judgments). Go to proof

1. Elimeq: Given \( \Gamma, \sigma, t, \kappa \) such that \( \text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset \) and \( D_1 \vdash \Gamma / \sigma \Rightarrow t : \kappa \vdash \Delta^1 \) and \( D_2 \vdash \Gamma / \sigma \Rightarrow t : \kappa \vdash \Delta^2 \), 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 \( \Gamma \vdash \kappa \) and \( \hat{\alpha} \notin \text{FV}(t) \) and \( D_1 \vdash \Gamma / \hat{\alpha} \Rightarrow t : \kappa \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / \hat{\alpha} \Rightarrow t : \kappa \vdash \Delta_2 \), 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 \( D_1 \vdash \Gamma / \hat{\alpha} \Rightarrow \hat{\beta} : \kappa \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / \hat{\beta} \Rightarrow \hat{\alpha} : \kappa \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

4. Checkeq: Given \( \Gamma, \sigma, t, \kappa \) such that \( D_1 \vdash \Gamma / \sigma \Rightarrow t : \kappa \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / \sigma \Rightarrow t : \kappa \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

5. Elimprop: Given \( \Gamma, P \) such that \( D_1 \vdash \Gamma / P \vdash \Delta^1 \) and \( D_2 \vdash \Gamma / P \vdash \Delta^2 \), it is the case that \( \Delta^1 = \Delta^2 \).

6. Checkprop: Given \( \Gamma, P \) such that \( D_1 \vdash \Gamma / P \) true \( \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / P \) true \( \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

Lemma 84 (Determinacy of Equivalence). Go to proof

1. Propositional equivalence: Given \( \Gamma, P, Q \) such that \( D_1 \vdash \Gamma / P \equiv Q \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / P \equiv Q \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

2. Type equivalence: Given \( \Gamma, A, B \) such that \( D_1 \vdash \Gamma / A \equiv B \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / A \equiv B \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

Theorem 4 (Determinacy of Subtyping). Go to proof

1. Subtyping: Given \( \Gamma, e, A, B \) such that \( D_1 \vdash \Gamma / e : A \lessdot P \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / e : A \lessdot P \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

Theorem 5 (Determinacy of Typing). Go to proof

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

2. Synthesis: Given \( \Gamma, e \) such that \( D_1 \vdash \Gamma / e \Rightarrow B_1 \vdash P_1 \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / e \Rightarrow B_2 \vdash P_2 \vdash \Delta_2 \), 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 \( D_1 \vdash \Gamma / e : A \vdash C_1 \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / e : A \vdash C_2 \vdash \Delta_2 \), 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 \( \Gamma / e : A \vdash C_k \vdash \Delta_k \).

4. Match judgments:
   Given \( \Gamma, \Pi, \tilde{A}, p, C \) such that \( D_1 \vdash \Gamma / \Pi / e : \tilde{A} \vdash q \vdash C \vdash P \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / \Pi / e : \tilde{A} \vdash q \vdash C \vdash P \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

Given \( \Gamma, \Pi, \tilde{A}, p, C \) such that \( D_1 \vdash \Gamma / P \vdash \tilde{A} ! \vdash C \vdash P \vdash \Delta_1 \) and \( D_2 \vdash \Gamma / P \vdash \tilde{A} ! \vdash C \vdash P \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
J Soundness

J.1 Soundness of Instantiation

Lemma 85 (Soundness of Instantiation). Go to proof
If $\Gamma \vdash \alpha := \tau : \kappa \rightarrow \Delta$ and $\alpha \notin \text{FV}(\Gamma \tau)$ and $\Gamma \tau = \tau$ and $\Delta \rightarrow \Omega$ then $\Omega \alpha = [\Omega] \tau$.

J.2 Soundness of Checkeq

Lemma 86 (Soundness of Checkeq). Go to proof
If $\Gamma \vdash \sigma \triangleright t : \kappa \rightarrow \Delta$ where $\Delta \rightarrow \Omega$ then $\Omega \sigma = \Omega t$.

J.3 Soundness of Equivalence (Propositions and Types)

Lemma 87 (Soundness of Propositional Equivalence). Go to proof
If $\Gamma \vdash P \equiv Q \rightarrow \Delta$ where $\Delta \rightarrow \Omega$ then $\Omega P = \Omega Q$.

Lemma 88 (Soundness of Algorithmic Equivalence). Go to proof
If $\Gamma \vdash A \equiv B \rightarrow \Delta$ where $\Delta \rightarrow \Omega$ then $\Omega A = \Omega B$.

J.4 Soundness of Checkprop

Lemma 89 (Soundness of Checkprop). Go to proof
If $\Gamma \vdash P \text{true} \rightarrow \Delta$ and $\Delta \rightarrow \Omega$ then $\Psi \vdash \Omega P \text{true}$.

J.5 Soundness of Eliminations (Equality and Proposition)

Lemma 90 (Soundness of Equality Elimination). Go to proof
If $\Gamma \vdash \sigma = \sigma : \kappa$ and $\Gamma \vdash \sigma : \kappa$ and $\Gamma \vdash \text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset$, then:

1. If $\Gamma / \sigma \triangleright t : \kappa \rightarrow \Delta$ 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 $\Omega \triangleright t' : \kappa'$,
   it is the case that $[\Omega, \Theta] t' = [\emptyset] [\Omega] t'$, where $\emptyset = \text{mgu}(\sigma, t)$.

2. If $\Gamma / \sigma \triangleright t : \kappa \rightarrow \perp$ then mgu($\sigma, t$) = $\perp$ (that is, no most general unifier exists).

J.6 Soundness of Subtyping

Theorem 6 (Soundness of Algorithmic Subtyping). Go to proof
If $[\Gamma] A = A$ and $[\Gamma] B = B$ and $\Gamma \vdash B \text{ type}$ and $\Gamma \vdash A \text{ type}$ and $\Delta \rightarrow \Omega$ and $\Gamma \vdash A <:\! P \rightarrow \Delta$ then $\Omega \Delta \vdash \Omega \succeq \Omega A <:\! P \rightarrow \Omega B$.

J.7 Soundness of Typing

Theorem 7 (Soundness of Match Coverage). Go to proof
1. If $\Gamma \vdash \Pi$ covers $\tilde{\alpha} q$ and $\Gamma \vdash \tilde{\alpha} q$ types and $[\Gamma] \tilde{\alpha} = \tilde{\alpha}$ and $\Gamma \rightarrow \Omega$ then $\Omega [\Gamma] \Pi$ covers $\tilde{\alpha} q$.

2. If $\Gamma / \Pi$ covers $\tilde{\alpha} !$ and $\Gamma \rightarrow \Omega$ and $\Gamma \vdash \tilde{\alpha} !$ types and $[\Gamma] \tilde{\alpha} = \tilde{\alpha}$ and $[\Gamma] \Pi = \Pi$ then $\Omega [\Gamma] / \Pi$ covers $\tilde{\alpha} !$.

Lemma 91 (Well-formedness of Algorithmic Typing). Go to proof
Given $\Gamma \ \text{ctx}$:
(i) If $\Gamma \vdash e \Rightarrow A \ p \vdash \Delta$ then $\Delta \vdash A \ p \text{ type}$.

(ii) If $\Gamma \vdash s : A \ p \Rightarrow B \ q \vdash \Delta$ and $\Gamma \vdash A \ p \text{ type}$ then $\Delta \vdash B \ q \text{ type}$.

**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.

\[
\langle e/s/\Pi, \text{ordinary spine judgment} < \text{recovering spine judgment}, \text{height}(\mathcal{D}) \rangle
\]

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 $\langle =, =, < \rangle$, 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: $\langle <, -, - \rangle$.

**Algorithmic rules (soundness cases):**

- $\text{Var}, \text{I}, \text{I}^\alpha, \text{EmptySpine}$ and $\text{Nil}$ have no premises, or only auxiliary judgments as premises.
- $\text{Sub} \langle =, =, < \rangle$
- $\text{Anno} \langle <, -, - \rangle$
- $\forall \text{Spine} \exists \text{II} \langle =, =, < \rangle$
- $\exists \text{II} \langle =, =, < \rangle$
- $\exists \text{SpineRecover} \langle =, <, - \rangle$
- $\exists \text{SpinePass} \langle =, <, - \rangle$
- $\Rightarrow \text{I} \langle =, =, < \rangle$
- $\Rightarrow \text{Rec} \langle <, -, - \rangle$
- $\Rightarrow \text{SpineRecover} \langle =, <, - \rangle$
- $\Rightarrow \text{SpinePass} \langle =, <, - \rangle$
- $\Rightarrow \text{Cons} \langle <, -, - \rangle$
- $^\alpha \text{Spine} \langle =, =, < \rangle$
- $\text{Case} \langle <, -, - \rangle$

**Declarative rules (completeness cases):**

- $\text{DeclVar}, \text{DeclI}, \text{DeclEmptySpine}$ and $\text{DeclNil}$ have no premises, or only auxiliary judgments as premises.
- $\text{DeclSub} \langle =, =, < \rangle$
- $\text{DeclAnno} \langle <, -, - \rangle$
J.7 Soundness of Typing

• \( \text{Decl} \forall I, \text{Decl} \forall \text{Spine}, \text{Decl} \exists I, \text{Decl} \rightarrow I, \text{Decl} \rightarrow E, \text{DeclRec} : \langle =, =, < \rangle \)

• \( \text{DeclSpineRecover} : \langle =, <, - \rangle \)

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

• \( \text{Decl} \rightarrow \text{Spine}, \text{Decl} \rightarrow I, \text{Decl} \rightarrow E, \text{DeclRec}, \text{DeclSpineRecover}, \text{DeclSpinePass} : \langle <, -, - \rangle \)

Definition 8 (Eagerness).
A derivation \( D \) whose conclusion is \( J \) is eager if:

(i) \( J = \Gamma \vdash e \Leftarrow A \ p \vdash \Delta \)
    if \( \Gamma \vdash A \ p \vdash \Delta \) and \( A = [\Gamma]A \)
    implies that
    every subderivation of \( D \) is eager.

(ii) \( J = \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \)
    if \( A = [\Delta]A \)
    and every subderivation of \( D \) is eager.

(iii) \( J = \Gamma \vdash s : A \ p \gg B \ q \vdash \Delta \)
    if \( \Gamma \vdash A \ p \vdash \Delta \) and \( A = [\Gamma]A \)
    implies that
    \( B = [\Delta]B \)
    and every subderivation of \( D \) is eager.

(iv) \( J = \Gamma \vdash s : A \ p \gg B \ [q] \vdash \Delta \)
    if \( \Gamma \vdash A \ p \vdash \Delta \) and \( A = [\Gamma]A \)
    implies that
    \( B = [\Delta]B \)
    and every subderivation of \( D \) is eager.

(v) \( J = \Gamma \vdash \Pi :: \vec{A} \ q \Leftarrow C \ p \vdash \Delta \)
    if \( \Gamma \vdash \vec{A} \ q \vdash \Delta \) and \( [\Gamma]\vec{A} = \vec{A} \) and \( \Gamma \vdash C \ p \vdash \Delta \)
    implies that
    every subderivation of \( D \) is eager.

(vi) \( J = \Gamma /P \vdash \Pi :: \vec{A} \ ! \Leftarrow C \ p \vdash \Delta \)
    if \( \Gamma \vdash \vec{A} \ ! \vdash \Delta \) and \( \Gamma \vdash P \ prop \) and \( [\Gamma]\vec{A} = \vec{A} \) and \( \Gamma \vdash C \ p \vdash \Delta \)
    implies that
    every subderivation of \( D \) is eager.

Theorem 8 (Eagerness of Types). Go to proof

(i) If \( D \) derives \( \Gamma \vdash e \Leftarrow A \ p \vdash \Delta \) and \( \Gamma \vdash A \ p \vdash \Delta \) then \( D \) is eager.

(ii) If \( D \) derives \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \) then \( D \) is eager.

(iii) If \( D \) derives \( \Gamma \vdash s : A \ p \gg B \ q \vdash \Delta \) and \( \Gamma \vdash A \ p \vdash \Delta \) then \( D \) is eager.

(iv) If \( D \) derives \( \Gamma \vdash s : A \ p \gg B \ [q] \vdash \Delta \) and \( \Gamma \vdash A \ p \vdash \Delta \) then \( D \) is eager.

(v) If \( D \) derives \( \Gamma \vdash \Pi :: \vec{A} \ q \Leftarrow C \ p \vdash \Delta \) and \( [\Gamma]\vec{A} = \vec{A} \) and \( \Gamma \vdash C \ p \vdash \Delta \) then \( D \) is eager.
(vi) If \( \mathcal{D} \) derives \( \Gamma / P \vdash \Pi : \bar{A} ! \equiv C p : \Delta \) and \( \Gamma \vdash P \) prop and \( \text{FEV}(P) = \emptyset \) and \( [\Gamma] P = P \) and \( \Gamma \vdash C p : \text{type} \) and \( \Gamma \vdash C p : \text{type} \) then \( \mathcal{D} \) is eager.

**Theorem 9** (Soundness of Algorithmic Typing). [Go to proof]

Given \( \Delta \rightarrow \Omega \):

(i) If \( \Gamma \vdash e \equiv A p : \Delta \) and \( \Gamma \vdash A p : \text{type} \) and \( A = [\Gamma] A \) then \( [\Omega] \Delta \vdash [\Omega] e \equiv [\Omega] A p . \)

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

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

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

(v) If \( \Gamma \vdash \Pi : \bar{A} ! \equiv C p : \Delta \) and \( \Gamma \vdash \bar{A} ! \) types and \( [\Gamma] \bar{A} = \bar{A} \) and \( \Gamma \vdash C p : \text{type} \) then \( p \vdash [\Omega] \Delta \vdash [\Omega] \Pi ! \equiv [\Omega] \bar{A} q [\Omega] C . \)

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

K Completeness

K.1 Completeness of Auxiliary Judgments

**Lemma 92** (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 \( \Delta \notin \text{unsolved}(\Gamma) \) and \( \Delta \notin \text{FV}(\tau) \):

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

**Lemma 93** (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 \( \Gamma \vdash [\Gamma] \sigma = [\Gamma] \tau : \kappa \rightarrow \Delta \) where \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \rightarrow \Omega' \).

**Lemma 94** (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 \( \text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset \) then:

1. If \( \text{mgu}(\sigma, t) = \emptyset \) then \( \Gamma / \sigma \downarrow t \vdash \kappa \rightarrow \bot \).
   - 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 \( [\Gamma] u = 0([\Gamma] u) \).

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

**Lemma 95** (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 \( [\Gamma] u = 0([\Gamma] u) \), then:

1. If \( \Gamma \vdash A \) type then \( [\Gamma, \Delta] A = 0([\Gamma] A) \).

2. If \( \Gamma \rightarrow \Omega \) then \( [\Omega] \Gamma = 0([\Omega] \Gamma) \).

3. If \( \Gamma \rightarrow \Omega \) then \( [\Omega, \Delta] [\Gamma, \Delta] = 0([\Omega] \Gamma) \).
Lemma 96 (Completeness of Propequiv). \[\text{Go to proof}\]
Given \(\Gamma \rightarrow \Omega\) and \(\Gamma \vdash P\) prop and \(\Gamma \vdash Q\) prop and \([\Omega]P = [\Omega]Q\) then \(\Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \vdash \Delta\) where \(\Delta \rightarrow \Omega'\) and \(\Omega \rightarrow \Omega'\).

Lemma 97 (Completeness of Checkprop). \[\text{Go to proof}\]
If \(\Gamma \rightarrow \Omega\) and \(\text{dom}(\Gamma) = \text{dom}(\Omega)\) and \(\Gamma \vdash P\) prop and \([\Omega]P = P\) and \([\Omega]P\) 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 98 (Completeness of Equiv). \[\text{Go to proof}\]
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 \(\Gamma \vdash [\Gamma]A \equiv [\Gamma]B \vdash \Delta\).

Theorem 10 (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 \([\Omega]A = [\Omega]B\) then there exist \(\Delta\) and \(\Omega'\) such that \(\Delta \rightarrow \Omega'\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash [\Gamma]A \triangleleft^P [\Gamma]B \vdash \Delta\).

K.3 Completeness of Typing

Lemma 99 (Variable Decomposition). \[\text{Go to proof}\]
If \(\Pi \stackrel{\text{var}}{\rightarrow} \Pi',\) then
1. if \(\Pi \xrightarrow{1} \Pi''\) then \(\Pi'' = \Pi'\).
2. if \(\Pi \xrightarrow{\tau} \Pi'''\) then there exists \(\Pi''\) such that \(\Pi''' \stackrel{\text{var}}{\rightarrow} \Pi''\) and \(\Pi'' \stackrel{\text{var}}{\rightarrow} \Pi',\)
3. if \(\Pi \xrightarrow{\tau} \Pi_L \parallel \Pi_R\) then \(\Pi_L \stackrel{\text{var}}{\rightarrow} \Pi'\) and \(\Pi_R \stackrel{\text{var}}{\rightarrow} \Pi',\)
4. if \(\Pi \stackrel{\text{vec}}{\rightarrow} \Pi[i]\parallel \Pi[i]\) then \(\Pi' = \Pi[i]\).

Lemma 100 (Pattern Decomposition and Substitution). \[\text{Go to proof}\]
1. If \(\Pi \stackrel{\text{var}}{\rightarrow} \Pi'\) then \([\Omega]\Pi \stackrel{\text{var}}{\rightarrow} [\Omega]\Pi'.\)
2. If \(\Pi \xrightarrow{1} \Pi'\) then \([\Omega]\Pi \xrightarrow{1} [\Omega]\Pi'.\)
3. If \(\Pi \xrightarrow{\tau} \Pi'\) then \([\Omega]\Pi \xrightarrow{\tau} [\Omega]\Pi'.\)
4. If \(\Pi \xrightarrow{1} \Pi_1 \parallel \Pi_2\) then \([\Omega]\Pi \xrightarrow{1} [\Omega]\Pi_1 \parallel [\Omega]\Pi_2.\)
5. If \(\Pi \stackrel{\text{vec}}{\rightarrow} \Pi_1 \parallel \Pi_2\) then \([\Omega]\Pi \stackrel{\text{vec}}{\rightarrow} [\Omega]\Pi_1 \parallel [\Omega]\Pi_2.\)
6. If \([\Omega]\Pi \stackrel{\text{var}}{\rightarrow} \Pi'\) then there is \(\Pi''\) such that \([\Omega]\Pi'' = \Pi'\) and \(\Pi \stackrel{\text{var}}{\rightarrow} \Pi''.\)
7. If $[\Omega] \Pi \vdash^1 \Pi'$ then there is $\Pi''$ such that $[\Omega] \Pi'' = \Pi'$ and $\Pi \vdash^1 \Pi''$.

8. If $[\Omega] \Pi \vdash^{\bowtie} \Pi'$ then there is $\Pi''$ such that $[\Omega] \Pi'' = \Pi'$ and $\Pi \vdash^{\bowtie} \Pi''$.

9. If $[\Omega] \Pi \models^{\bowtie} \Pi'_1 \parallel \Pi'_2$ then there are $\Pi_1$ and $\Pi_2$ such that $[\Omega] \Pi_1 = \Pi'_1$ and $[\Omega] \Pi_2 = \Pi'_2$ and $\Pi \vdash^{\bowtie} \Pi_1 \parallel \Pi_2$.

10. If $[\Omega] \Pi \models^{\bowtie} \Pi'_1 \parallel \Pi'_2$ then there are $\Pi_1$ and $\Pi_2$ such that $[\Omega] \Pi_1 = \Pi'_1$ and $[\Omega] \Pi_2 = \Pi'_2$ and $\Pi \models^{\bowtie} \Pi_1 \parallel \Pi_2$.

**Lemma 101 (Pattern Decomposition Functionality).** Go to proof

1. If $\Pi \models^{\bowtie} \Pi'$ and $\Pi \models^{\bowtie} \Pi''$ then $\Pi' = \Pi''$.

2. If $\Pi \vdash^1 \Pi'$ and $\Pi \vdash^1 \Pi''$ then $\Pi' = \Pi''$.

3. If $\Pi \vdash^{\bowtie} \Pi'$ and $\Pi \vdash^{\bowtie} \Pi''$ then $\Pi' = \Pi''$.

4. If $\Pi \vdash^{\bowtie} \Pi_1 \parallel \Pi_2$ and $\Pi \vdash^{\bowtie} \Pi'_1 \parallel \Pi'_2$ then $\Pi_1 = \Pi'_1$ and $\Pi_2 = \Pi'_2$.

5. If $\Pi \models^{\bowtie} \Pi_1 \parallel \Pi_2$ and $\Pi \models^{\bowtie} \Pi'_1 \parallel \Pi'_2$ then $\Pi_1 = \Pi'_1$ and $\Pi_2 = \Pi'_2$.

**Lemma 102 (Decidability of Variable Removal).** Go to proof For all $\Pi$, either there exists a $\Pi'$ such that $\Pi \vdash^{\bowtie} \Pi'$ or there does not.

**Lemma 103 (Variable Inversion).** Go to proof

1. If $\Pi \models^{\bowtie} \Pi'$ and $\Psi \vdash \Pi$ covers $A, \bar{A} q$ then $\Psi \vdash \Pi'$ covers $\bar{A} q$.

2. If $\Pi \models^{\bowtie} \Pi'$ and $\Gamma \vdash \Pi$ covers $A, \bar{A} q$ then $\Gamma \vdash \Pi'$ covers $\bar{A} q$.

**Theorem 11 (Completeness of Match Coverage).** Go to proof

1. If $\Gamma \vdash \bar{A} q$ types and $[\Gamma] \bar{A} = \bar{A}$ and (for all $\Omega$ such that $\Gamma \rightarrow \Omega$, we have $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega] \bar{A} q$) then $\Gamma \vdash \Pi$ covers $\bar{A} q$.

2. If $[\Gamma] \bar{A} = \bar{A}$ and $[\Gamma] \bar{P} = \bar{P}$ and $\Gamma \vdash \bar{A} !$ types and (for all $\Omega$ such that $\Gamma \rightarrow \Omega$, we have $[\Omega] \Gamma / [\Omega] \Pi$ covers $[\Omega] \bar{A} !$) then $\Gamma / \bar{P} \vdash \Pi$ covers $\bar{A} !$.

**Theorem 12 (Completeness of Algorithmic Typing).** 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 $p' \sqsubseteq p$
then there exist $\Delta$ and $\Omega'$
such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$
and $\Gamma \vdash e \leftrightarrow [\Gamma] A p' \vdash \Delta$.

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

(iv) If $\Gamma \vdash A p$ type and $[\Omega] \Gamma \vdash [\Omega] s : [\Omega] A p \Rightarrow B [q]$ and $p' \sqsubseteq p$
then there exist $\Delta$, $\Omega'$, $B'$, and $q' \sqsubseteq q$
such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma] A p' \Rightarrow B' [q'] \vdash \Delta$ and $B' = [\Delta] B'$ and $B = [\Omega'] B'$.
(v) If $Γ ⊢ \vec{A}$ types and $Γ ⊢ C p$ type and $[Ω]|Γ ⊢ [Ω]|Π :: [Ω]|\vec{A} q ⇐ [Ω]|C p$ and $p' ⊆ p$
then there exist $Δ, Ω'$, and $C$
such that $Δ \rightarrowΩ'$ and $\text{dom}(Δ) = \text{dom}(Ω')$ and $Ω \rightarrowΩ'$
and $Γ ⊢ Π :: [Γ]|\vec{A} q ⇐ [Γ]|C p' \rightarrow Δ$.

(vi) If $Γ ⊢ \vec{A}$ types and $Γ ⊢ P$ prop and $\text{FEV}(P) = \emptyset$ and $Γ ⊢ C p$ type
and $[Ω]|Γ / [Ω]|P ⊢ [Ω]|Π :: [Ω]|\vec{A} q ⇐ [Ω]|C p$
and $p' ⊆ p$
then there exist $Δ, Ω'$, and $C$
such that $Δ \rightarrowΩ'$ and $\text{dom}(Δ) = \text{dom}(Ω')$ and $Ω \rightarrowΩ'$
and $Γ / [Γ]|P ⊢ Π :: [Γ]|\vec{A} q ⇐ [Γ]|C p' \rightarrow Δ$. 
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 $\Psi \vdash e \Rightarrow Bp$
(ii) checking $\Psi \vdash e \Leftarrow Ap$
(iii) checking, equality elimination $\Psi/P \vdash e \Leftarrow Cp$
(iv) ordinary spine $\Psi \vdash s : Ap \gg Bq$
(v) recovery spine $\Psi \vdash s : Ap \gg B[q]$
(vi) pattern matching $\Psi \vdash \Pi :: \vec{A} ! \Leftarrow Cp$
(vii) pattern matching, equality elimination $\Psi/P \vdash \Pi :: \vec{A} ! \Leftarrow Cp$

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:

- synthesis judgment at $n$
- checking judgments at $n$
- ordinary spine judgment at $n$
- recovery spine judgment at $n$
- match judgments at $n$
- synthesis judgment at $n + 1$

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:

- synthesis judgment at $n$
- (checking judgment at $n$ with $\text{large}(A) = 1$
- (checking judgment at $n$ with $\text{large}(A) = 2$
- (checking judgment at $n$ with $\text{large}(A) = 2$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 1$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 1$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 2$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 2$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 2$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 2$
- (match judgment at $n$ with $\text{large}(\vec{A}) = 2$
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 $\text{DeclSpine}$, $\text{Decl→Spine}$, and $\text{Decl→Spine}$ 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, $|θ(e)| = |e|$, 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, $Ψ \vdash A \text{ type}$, $Ψ \vdash τ : κ$, and $Ψ \vdash P \text{ prop}$ 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 $\text{Decl→I}$ 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 $\text{DeclV}$ depends on a checking judgment at $n$ with fewer large connectives. (3)
  
  Rule $\text{DeclE}$ depends on a checking judgment at $n$ with fewer large connectives. (3)
  
  Rule $\text{Decl∧}$ depends on a checking judgment at $n$ with fewer large connectives. (3)
  
  Rules $\text{Decl→I}$, $\text{DeclRec}$, $\text{Decl→I}$, $\text{Decl×I}$, 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 $Ψ / P \vdash e \leftarrow A p$ 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 $\text{DeclCheck⊥}$ has no nontrivial premises.
  
  Rule $\text{DeclCheckUnify}$ depends on a checking judgment: Since $|θ(e)| = |e|$, this checking judgment is at $n$. Since the mgu $θ$ is over monotypes, $\#\text{large}(θ(A)) = \#\text{large}(A)$. 
• **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 \texttt{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 \#\text{large}(\Lambda).

**Proof.** Rule \texttt{Decl\backslash Spine} depends on an ordinary spine judgment of size \( n \), with a type that has fewer large connectives. (c)
Rule \texttt{Decl\lor Spine} depends on an ordinary spine judgment of size \( n \), with a type that has fewer large connectives. (c)
Rule \texttt{DeclEmptySpine} has no premises.
Rule \texttt{Decl\neg 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 \texttt{DeclSpineRecover} and \texttt{DeclSpinePass} depend only on ordinary spine judgments at \( n \).

• **Match judgments**

**Claim:** For all \( n \geq 1 \), the match judgment \( \Psi \vdash \Pi :: \tilde{A} ! \iff C p \) 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 \( \tilde{A} \), and
(d) matchelim judgments at size \( n \) with fewer large connectives in \( \tilde{A} \).

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

• **Matchelim judgments**

**Claim:** For all \( n \geq 1 \), the matchelim judgment \( \Psi / \Pi \vdash P :: \tilde{A} ! \iff C p \) over \( \Psi \) of size \( n \) depends only on match judgments with the same number of large connectives in \( \tilde{A} \).

**Proof.** Rule \texttt{DeclMatch\perp} has no nontrivial premises.
Rule \texttt{DeclMatchUnify} depends on a match judgment with the same number of large connectives (similar to \texttt{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 $\Psi_0, \alpha : \kappa, \Psi_1 \vdash \text{prop}$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]\text{prop}$.
3. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash \text{type}$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]\text{type}$.
4. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash \text{A} \leq \text{P} \text{B}$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]\text{A} \leq \text{P} [t/\alpha]\text{B}$.
5. If $\Psi_0, \alpha : \kappa, \Psi_1 \vdash \text{true}$ then $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]\text{true}$.

Proof. By induction on the derivation of the substitutee.

Lemma 4 (Reflexivity of Declarative Subtyping).
Given $\Psi \vdash \text{A} \text{type}$, we have that $\Psi \vdash \text{A} \leq \text{P} \text{A}$.

Proof. By induction on $\text{A}$, writing $p$ for the sign of the subtyping judgment.
Our induction metric is the number of quantifiers on the outside of $\text{A}$, plus one if the polarity of $\text{A}$ and the subtyping judgment do not match up (that is, if $\text{neg}(\text{A})$ and $p = +$, or $\text{pos}(\text{A})$ and $p = -$).

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

- **Case** $\text{A} = \exists b : \kappa. \text{B}$ and $p = +$:
  $\Psi, b : \kappa \vdash \text{B} \leq^+ \text{B}$  
  $\Psi, b : \kappa \vdash b : \kappa$  
  $\Psi, b : \kappa \vdash \exists b : \kappa. \text{B}$  
  By rule $\leq \text{∃R}$

- **Case** $\text{A} = \exists b : \kappa. \text{B}$ and $p = -$:
  $\Psi \vdash \exists b : \kappa. \text{B} \leq^+ \exists b : \kappa. \text{B}$  
  By i.h. (polarities match)

- **Case** $\text{A} = \forall b : \kappa. \text{B}$ and $p = +$:
  $\Psi \vdash \forall b : \kappa. \text{B} \leq^- \forall b : \kappa. \text{B}$  
  By $\leq^-$

- **Case** $\text{A} = \forall b : \kappa. \text{B}$ and $p = -$:
  $\Psi, b : \kappa \vdash \forall b : \kappa. \text{B} \leq^- \forall b : \kappa. \text{B}$  
  By rule $\leq \forall \text{R}$

Lemma 5 (Subtyping Inversion).

- If $\Psi \vdash \exists \alpha : \kappa. \text{A} \leq^+ \text{B}$ then $\Psi, \alpha : \kappa \vdash \text{A} \leq^+ \text{B}$.
- If $\Psi \vdash \text{A} \leq^- \forall \beta : \kappa. \text{B}$ then $\Psi, \beta : \kappa \vdash \text{A} \leq^- \text{B}$.
Proof. By a routine induction on the subtyping derivations.

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

- 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 $\Psi \vdash A \leq^P B$ then $A = B$.

Proof. By a routine induction on the subtyping derivations.

**Lemma 7 (Transitivity of Declarative Subtyping).**

Given $\Psi \vdash A$ type and $\Psi \vdash B$ type and $\Psi \vdash C$ type:

(i) If $D_1 : \Psi \vdash A \leq^P B$ and $D_2 : \Psi \vdash B \leq^P C$ then $\Psi \vdash A \leq^P C$.

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 $D_1$:

  - Case $\begin{array}{c}
  \Psi \vdash \tau : \kappa_1 \\
  \Psi \vdash [\tau/\alpha]A' \leq^- B
  \end{array}$

    $\begin{array}{c}
    \Psi \vdash \forall \alpha : \kappa_1. A' \leq^- B \\
    \leq \forall L
  \end{array}$

    Subderivation

    Subderivation

    Given

    By i.h. (A lost a quantifier)

    By rule $\leq \forall L$

  - Case $\begin{array}{c}
  \Psi, \beta : \kappa_2 \vdash A \leq^- B'
  \end{array}$

    $\begin{array}{c}
    \Psi \vdash A \leq^- \forall \beta : \kappa_2. B' \\
    \leq \forall R
  \end{array}$

  We case-analyze $D_2$:

  * Case $\begin{array}{c}
  \Psi \vdash \tau : \kappa_2 \\
  \Psi \vdash [\tau/\beta]B' \leq^- C
  \end{array}$

    $\begin{array}{c}
    \Psi \vdash \forall \beta : \kappa_2. B' \leq^- C \\
    \leq \forall L
  \end{array}$

    By Lemma 5 (Subtyping Inversion) on $D_1$

    Subderivation

    Subderivation of $D_2$

    By Lemma 3 (Declarative Term Substitution)

    By i.h. (B lost a quantifier)
* Case \( \Psi, c : \kappa_3 \vdash B \leq C' \) 
\[
\frac{\Psi \vdash B \leq \forall c : \kappa_3. C'}{\leq \forall R}
\]

\( \frac{\Psi \vdash A \leq B}{\text{Given}} \)
\( \frac{\Psi, c : \kappa_3 \vdash A \leq B}{\text{By Lemma 2 (Declarative Weakening)}} \)
\( \frac{\Psi, c : \kappa_3 \vdash B \leq C'}{\text{Subderivation}} \)
\( \frac{\Psi, c : \kappa_3 \vdash A \leq C'}{\text{By i.h. (C lost a quantifier)}} \)
\( \frac{\Psi \vdash B \leq \forall c : \kappa_3. C'}{\leq \forall R} \)

• Case nonpos(B), polarity = +:
Now we case-analyze \( D_1 \):

– Case \( \frac{\Psi, \alpha : \tau \vdash A' \leq^+ B}{\Psi \vdash \exists \alpha : \kappa_1. A' \leq^+ B} \leq \exists \) 

\( \frac{\Psi, \alpha : \tau \vdash A' \leq^+ B}{\text{Subderivation}} \)
\( \frac{\Psi, \alpha : \tau \vdash B \leq^+ C}{\text{By Lemma 2 (Declarative Weakening) (D_2)}} \)
\( \frac{\Psi, \alpha : \tau \vdash A' \leq^+ C}{\text{By i.h. (A lost a quantifier)}} \)
\( \frac{\Psi \vdash \exists \alpha : \kappa_1. A' \leq^+ C}{\leq \exists} \)

– Case \( \frac{\Psi \vdash A \leq^+ B \quad \text{nonpos(A)} \quad \text{nonpos(B)}}{\Psi \vdash A \leq^+ B} \leq \)

Now we case-analyze \( D_2 \):

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

\( \frac{\Psi \vdash A \leq^+ B}{\text{Given}} \)
\( \frac{\Psi \vdash \tau : \kappa_3}{\text{Subderivation of } D_2} \)
\( \frac{\Psi \vdash B \leq^+ [\tau/c]C'}{\text{Subderivation of } D_2} \)
\( \frac{\Psi \vdash A \leq^+ [\tau/c]C'}{\text{By i.h. (C lost a quantifier)}} \)
\( \frac{\Psi \vdash A \leq^+ \exists c : \kappa_3. C'}{\leq \exists R} \)

* Case \( \frac{\Psi \vdash B \leq C \quad \text{nonpos(B)} \quad \text{nonpos(C)}}{\Psi \vdash B \leq C} \leq \)

\( \frac{\Psi \vdash A \leq C \quad \text{Subderivation of } D_1}{} \)
\( \frac{\Psi \vdash B \leq C \quad \text{Subderivation of } D_2}{\leq} \)
\( \frac{\Psi \vdash A \leq C \quad \text{By i.h. (D_1 and D_2 smaller)}}{\leq} \)
• Case $B = \exists \beta : \kappa_2 . B'$, polarity = $+$:
  
  Now we case-analyze $D_2$:

  $\vdash N \tau : \kappa_3 \quad \vdash B \leq^+ \tau/\alpha C'$

  $\vdash B \leq^+ \exists\alpha : \kappa_3 . C' \quad \leq^R$

  \[
  \begin{array}{l}
  \vdash \tau : \kappa_3 \quad \vdash B \leq^+ \tau/\alpha C' \\
  \vdash \exists\alpha : \kappa_3 . C' \quad \leq^R \\
  \end{array}
  \]

  $\vdash \tau : \kappa_3$ \quad Subderivation of $D_2$
  $\vdash B \leq^+ \tau/\alpha C'$ \quad Subderivation of $D_2$
  $\vdash A \leq^+ B$ \quad Given
  $\vdash A \leq^+ \tau/\alpha C'$ \quad By i.h. (C lost a quantifier)
  $\vdash A \leq^+ C$ \quad By rule $\leq^R$

  $\vdash \exists \beta : \kappa_2 . B' \leq^+ C$ \quad $\leq^R$

  Now we case-analyze $D_1$:

  $\star$ Case $\vdash N \tau : \kappa_2 \quad \vdash A \leq^+ B$' \quad $\leq^R$

  $\vdash A \leq^+ \exists\beta : \kappa_2 . B'$ \quad $\leq^R$

  $\vdash \tau : \kappa_2$ \quad Subderivation of $D_2$
  $\vdash A \leq^+ \tau/\beta B'$ \quad Subderivation of $D_1$
  $\vdash [\tau/\beta]B' \leq^+ C$ \quad By Lemma $\star$ (Declarative Term Substitution)
  $\vdash A \leq^+ C$ \quad By i.h. (B lost a quantifier)

  $\star$ Case $\vdash N \alpha : \kappa_1 \vdash A \leq^+ B$ \quad $\leq^R$

  $\vdash \exists\alpha : \kappa_1 . A' \leq^+ B$ \quad $\leq^R$

  $\vdash \tau : \kappa_1$ \quad Subderivation of $D_1$
  $\vdash A \leq^+ \exists\alpha : \kappa_1 . A'$ \quad $\leq^R$
  $\vdash A \leq^+ B$ \quad Given
  $\vdash A \leq^+ \tau/\alpha B'$ \quad Subderivation of $D_1$
  $\vdash [\tau/\alpha]B' \leq^+ C$ \quad By Lemma $\star$ (Declarative Weakening)
  $\vdash A \leq^+ C$ \quad By i.h. (A lost a quantifier)
  $\vdash \exists\alpha : \kappa_1 . A' \leq^+ C$ \quad By $\leq^L$

• Case $\text{nonneg}(B)$, polarity = $-$:
  
  We case-analyze $D_2$:

  $\vdash N \tau : \kappa_3 \quad \vdash B \leq^+ C'$ \quad $\leq^R$

  $\vdash B \leq^+ \exists\tau : \kappa_3 . C' \quad \leq^R$

  $\vdash \tau : \kappa_3$ \quad Subderivation of $D_2$
  $\vdash \tau : \kappa_3 \quad \vdash A \leq^+ B$ \quad By Lemma $\star$ (Declarative Weakening)
  $\vdash \tau : \kappa_3 \quad \vdash A \leq^+ C'$ \quad By i.h. (C lost a quantifier)
  $\vdash A \leq^+ \forall c : \kappa_3 . C'$ \quad By $\leq^R$
Proof of Lemma 7 (Transitivity of Declarative Subtyping).

\[
\frac{\Psi \vdash B \leq^+ C \quad \text{nonneg}(B) \quad \text{nonneg}(C)}{\Psi \vdash B \leq C} \quad \leq^+
\]

We case-analyze \(\mathcal{D}_1\):

\* Case \(\Psi \vdash \tau : \kappa_1 \quad \Psi \vdash [\tau/\alpha]A' \leq - B\)

\[
\Psi \vdash \forall \alpha : \kappa_1. A' \leq - B \quad \leq \forall L
\]

\(\Psi \vdash B \leq C\) \quad Given
\(\Psi \vdash \tau : \kappa_1\) \quad Subderivation of \(\mathcal{D}_1\)
\(\Psi \vdash [\tau/\alpha]A' \leq - B\) \quad Subderivation of \(\mathcal{D}_1\)
\(\Psi \vdash [\tau/\alpha]A' \leq C\) \quad By i.h. (\(A\) lost a quantifier)
\(\Psi \vdash \forall \alpha : \kappa_1. A' \leq C\) \quad By \(\leq \forall L\)

\* Case \(\Psi \vdash A \leq^+ B\) \quad \text{nonpos}(A) \quad \text{nonpos}(B)

\[
\Psi \vdash A \leq - B \quad \leq \n\]

\(\Psi \vdash A \leq^+ B\) \quad Subderivation of \(\mathcal{D}_1\)
\(\Psi \vdash B \leq^+ C\) \quad Subderivation of \(\mathcal{D}_2\)
\(\Psi \vdash A \leq^+ C\) \quad By i.h. (\(D_1\) and \(D_2\) smaller)
\(\text{nonneg}(A)\) \quad Subderivation of \(\mathcal{D}_2\)
\(\text{nonneg}(C)\) \quad Subderivation of \(\mathcal{D}_2\)
\(\Psi \vdash A \leq - C\) \quad By \(\leq \n\)

\[\square\]

**B’ Substitution and Well-formedness Properties**

**Lemma 8** (Substitution—Well-formedness).

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

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

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

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

**Lemma 9** (Uvar Preservation).

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

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

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

**Proof.** By induction on \(\Omega\), following the definition of context application (Figure 13). \[\square\]

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

**Proof.** By induction on the given derivation. All cases are straightforward. \[\square\]

**Lemma 11** (Right-Hand Substitution for Sorting). \(\Gamma \vdash t : \kappa \) then \(\Gamma|\Gamma|t : \kappa\).

**Proof.** By induction on \(|\Gamma|t|\) (the size of \(t\) under \(\Gamma\)).
Proof of Lemma 11 (Right-Hand Substitution for Sorting).

Proof. By induction on $|\Gamma|\vdash t : \kappa$.

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 $u : \kappa \in \Omega$
  $\Omega \vdash u : \kappa$
  \textbf{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 \textbf{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), $\Omega \vdash [\Omega] \tau : \kappa$.

• Case $\beta : \kappa = \tau \in \Omega$
  $\Omega \vdash \beta : \kappa$
  \textbf{SolvedVarSort}

  $\beta : \kappa = \tau \in \Omega$
  \textbf{Subderivation}

  $\Omega = (\Omega_L, \beta : \kappa = \tau, \Omega_R)$
  \textbf{Decomposing $\Omega$}

  $\Omega_L \vdash \tau : \kappa$
  \textbf{By implicit assumption that $\Omega$ is well-formed}

  $\Omega_L, \beta : \kappa = \tau, \Omega_R \vdash \tau : \kappa$
  \textbf{By Lemma 34 (Suffix Weakening)}

  $\Omega \vdash [\Omega] \tau : \kappa$
  \textbf{By Lemma 11 (Right-Hand Substitution for Sorting)}

  $\Rightarrow$

  $[\Omega] \Omega \vdash [\Omega] \beta : \kappa$
  $[\Omega] \tau = [\Omega] \beta$

• Case $\Omega \vdash 1 : \star$
  \textbf{UnitSort}

Since $1 = [\Omega] 1$, applying \textbf{UnitSort} gives the result.

• Case $\Omega \vdash \tau_1 : \star$
  $\Omega \vdash \tau_2 : \star$
  $\Omega \vdash \tau_1 \oplus \tau_2 : \star$
  \textbf{BinSort}

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

• Case $\Omega \vdash \text{zero} : \mathbb{N}$
  \textbf{ZeroSort}

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

• Case $\Omega \vdash t : \mathbb{N}$
  $\Omega \vdash \text{succ}(t) : \mathbb{N}$
  \textbf{SuccSort}

By i.h., rule \textbf{SuccSort}, and the definition of substitution.

\hfill $\square$

\textbf{Lemma 15 (Substitution for Prop Well-Formedness).}

\textit{If $\Omega \vdash P \text{ prop}$ then $[\Omega] \Omega \vdash [\Omega] P \text{ prop}$.}

\textbf{Proof.} Only one rule derives this judgment form:

• Case $\Omega \vdash t : \mathbb{N}$
  $\Omega \vdash t' : \mathbb{N}$
  $\Omega \vdash t = t' : \mathbb{N}$
  \textbf{EqProp}
Proof of Lemma 15 \textbf{(Substitution for Prop Well-Formedness)}

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

\[ \Omega \vdash A \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A \text{ type} \]
\[ [\Omega]\Omega \vdash ([\Omega] t) = ([\Omega] t') \]
\[ \Omega \vdash t' : N \]
\[ [\Omega]\Omega \vdash [\Omega] t' : N \]
\[ [\Omega]\Omega \vdash (\forall \alpha : \kappa. [\Omega] A) : \kappa' \]
\[ [\Omega]\Omega \vdash (\forall \alpha : \kappa. [\Omega] A) : \kappa' \]
\[ \Omega \vdash A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A_0 : \kappa' \]
\[ [\Omega]\Omega \vdash (\forall \alpha : \kappa. [\Omega] A_0) : \kappa' \]
\[ [\Omega]\Omega \vdash (\forall \alpha : \kappa. [\Omega] A_0) : \kappa' \]
\[ \Omega \vdash P \text{ prop} \]
\[ [\Omega]\Omega \vdash [\Omega] P \text{ prop} \]
\[ \Omega \vdash P \supset A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] P \supset A_0 \text{ type} \]

\[ \Omega \vdash A_1 \text{ type} \]
\[ \Omega \vdash A_2 \text{ type} \]
\[ \Omega \vdash A_1 \oplus A_2 \text{ type} \]
\[ \Omega \vdash A_1 \oplus A_2 \text{ type} \]

\[ \Omega \vdash \forall \alpha : \kappa. A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] \forall \alpha : \kappa. A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] \forall \alpha : \kappa. A_0 \text{ type} \]

\[ \Omega \vdash P \text{ prop} \]
\[ [\Omega]\Omega \vdash [\Omega] P \text{ prop} \]
\[ \Omega \vdash P \supset A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] P \supset A_0 \text{ type} \]

\[ \Omega \vdash A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A_0 \text{ type} \]

\[ \Omega \vdash A \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A \text{ type} \]

\[ \Omega \vdash (\forall \alpha : \kappa. [\Omega] A_0) \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] (\forall \alpha : \kappa. [\Omega] A_0) \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] (\forall \alpha : \kappa. [\Omega] A_0) \text{ type} \]

\[ \Omega \vdash A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A_0 \text{ type} \]
\[ [\Omega]\Omega \vdash [\Omega] A_0 \text{ type} \]
Proof of Lemma 16 (Substitution for Type Well-Formedness).

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

49

- Case \( \Omega \vdash P \; \text{prop} \quad \Omega \vdash A_0 \; \text{type} \)
  \[ \frac{} {\Omega \vdash A_0 \wedge P \; \text{type}} \text{WithWF} \]

  Similar to the \text{ImpliesWF} case.

\begin{lemma}
(ImpliesWF)\
If \( (\Omega,\Omega Z) \) is well-formed and \( \Omega Z \) is soft and \( \Omega \vdash A \; \text{type} \) then \[ [\Omega]A = [\Omega,\Omega Z]A. \]
\end{lemma}

\begin{proof}
By induction on \( \Omega Z \).
Since \( \Omega Z \) is soft, either (1) \( \Omega Z = \cdot \) (and the result is immediate) or (2) \( \Omega Z = (\Omega',\hat{\alpha} : \kappa) \) or (3) \( \Omega Z = (\Omega',\hat{\alpha} : \kappa = t) \). However, according to the grammar for complete contexts such as \( \Omega Z \), (2) is impossible.

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

\begin{lemma}
(ImpliesWF)\
If \( \Omega_1 \vdash A \; \text{type} \) and \( \text{dom}(\Omega_1) = \text{dom}(\Omega_2) \) then \( \Omega_2 \vdash A \; \text{type} \).
\end{lemma}

\begin{proof}
By induction on the given derivation. \( \square \)
\end{proof}

\section{Properties of Extension}

\begin{lemma}
(Declaration Preservation). If \( \Gamma \longrightarrow \Delta \) and \( u \) is declared in \( \Gamma \), then \( u \) is declared in \( \Delta \).
\end{lemma}

\begin{proof}
By induction on the derivation of \( \Gamma \longrightarrow \Delta \).
- Case \( \cdot \longrightarrow \text{Id} \)
  \[
  \text{This case is impossible, since by hypothesis } u \text{ is declared in } \Gamma. \]

- Case \( \Gamma \longrightarrow \Delta \quad [\Delta]A = [\Delta']A' \)
  \[ \frac{} {\Gamma, x : A \longrightarrow \Delta, x : A'} \text{Var} \]
  
  - 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 \( \Gamma \longrightarrow \Delta \quad \Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa \)
  \[ \frac{} {\Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa} \text{Uvar} \]
  Similar to the \text{Var} case.

- Case \( \Gamma \longrightarrow \Delta \quad \Gamma, \hat{\alpha} : \kappa \longrightarrow \Delta, \hat{\alpha} : \kappa \)
  \[ \frac{} {\Gamma, \hat{\alpha} : \kappa \longrightarrow \Delta, \hat{\alpha} : \kappa} \text{Unsolved} \]
  Similar to the \text{Var} case.

- Case \( \Gamma \longrightarrow \Delta \quad [\Delta]t = [\Delta']t' \)
  \[ \frac{} {\Gamma, \hat{\alpha} : \kappa = t \longrightarrow \Delta, \hat{\alpha} : \kappa = t'} \text{Solved} \]
  Similar to the \text{Var} case.

Proof of Lemma 19 (Declaration Preservation).
Proof of Lemma 19 (Declaration Preservation)

• Case $\Gamma \rightarrow \Delta$ 
  $[\Delta]t = [\Delta]t'$
  $\Gamma, \alpha = t \rightarrow \Delta, \alpha = t'$
  $\rightarrow$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 $\Gamma \rightarrow \Delta$
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa$
  $\rightarrow$Marker

  Similar to the $\rightarrow$Eqn case.

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

  Similar to the $\rightarrow$Var case.

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

  Similar to the $\rightarrow$Add case.

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 $\rightarrow$Id

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

• Case $\Gamma \rightarrow \Delta$
  $[\Delta]A = [\Delta]A'$
  $\Gamma, x : A \rightarrow \Delta, x : A'$
  $\rightarrow$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 $\Gamma \rightarrow \Delta$
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa$
  $\rightarrow$Uvar

  Similar to the $\rightarrow$Var case.
Proof of Lemma 20 (Declaration Order Preservation)

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ \Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \] Unsolved

  Similar to the \( \rightarrow \text{Var} \) case.

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ |\Delta| t = |\Delta| t' \]
  \[ \Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa = t' \rightarrow \] Solved

  Similar to the \( \rightarrow \text{Var} \) case.

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' = t \rightarrow \] Solve

  Similar to the \( \rightarrow \text{Var} \) case.

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ |\Delta| t = |\Delta| t' \]
  \[ \Gamma, \alpha = t \rightarrow \Delta, \alpha = t' \rightarrow \] Eqn

  The equation \( \alpha = t \) does not declare any variables, so \( u \) and \( v \) must be 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 \( \Delta, \alpha : \kappa = t' \).

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ \Gamma, \alpha \rightarrow \Delta, \alpha \rightarrow \] Marker

  Similar to the \( \rightarrow \text{Eqn} \) case.

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ \Gamma \rightarrow \Delta, \alpha : \kappa \rightarrow \] Add

  By i.h., \( u \) is declared to the left of \( v \) in \( \Delta \).
  Therefore \( u \) is declared to the left of \( v \) in \( (\Delta, \alpha : \kappa) \).

• Case
  \[ \Gamma \rightarrow \Delta \]
  \[ \Gamma \rightarrow \Delta, \alpha : \kappa \rightarrow \] AddSolved

  Similar to the \( \rightarrow \text{Add} \) case.

\[ \square \]

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 \( \Delta \), 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 \( u \) 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 \( \Gamma \).

\[ \square \]

Lemma 22 (Extension Inversion).

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

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

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

(v) If $D \vdash \Gamma_0, x : A, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0$, $A'$, and $\Delta_1$
such that $\Delta = (\Delta_0, x : A', \Delta_1)$ and $D' \vdash \Gamma_0 \rightarrow \Delta_0$ 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 $D \vdash \Gamma_0, \hat{x} : \kappa, \Gamma_1 \rightarrow \Delta$ then either

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

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 red 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 red 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** $\Gamma \rightarrow \Delta$ | $[\Delta] A = [\Delta] A'$
  - ![Inference Rule](image)
  - $\Gamma, x : A \rightarrow \Delta, x : A'$
  - $\Gamma_0, \alpha : \kappa, \Gamma_1$
  - $(\Gamma, x : A) = (\Gamma_0, \alpha : \kappa, \Gamma_1)$
    - Given
  - $(\Gamma, x : A) = (\Gamma_0, \alpha : \kappa, \Gamma'_1, x : A)$
    - Since the last element must be equal
  - $(\Gamma, x : A) = (\Gamma_0, \alpha : \kappa, \Gamma'_1, x : A)$
    - By transitivity
  - $\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma'_1)$
    - 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.
  - $\Gamma_0 \rightarrow \Delta_0$
    - ""
  - if $\Gamma'_1$ soft then $\Delta_1$ soft
    - ""
  - $(\Delta, x : A') = (\Delta_0, \alpha : \kappa, \Delta_1, x : A')$
    - By congruence
  - if $\Gamma'_1, x : A$ soft then $\Delta_1, x : A'$ soft
    - Since $\Gamma'_1, x : A$ is not soft
Proof of Lemma 22 (Extension Inversion)

There are two cases:

- Case $\alpha : \kappa = \beta : \kappa'$:
  
  - $(\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma, \alpha : \kappa, \Gamma_1')$ where $\Gamma_0 = \Gamma$ and $\Gamma_1 = \cdot$.
  
  - $(\Delta_0, \alpha : \kappa, \Delta_1) = (\Delta, \alpha : \kappa, \Delta_1')$ where $\Delta_0 = \Delta$ and $\Delta_1 = \cdot$.
  
  - if $\Gamma_1$ soft then $\Delta_1$ soft since $\cdot$ is soft.

- Case $\alpha \neq \beta$:
  
  - $(\Gamma_0, \alpha : \kappa, \Gamma_1') = (\Gamma_0, \alpha : \kappa, \Gamma_1')$ Given
  
  - $(\Delta_0, \alpha : \kappa, \Delta_1) = (\Delta_0, \alpha : \kappa, \Delta_1')$ Since the last element must be equal
  
  - $\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1')$ 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.
  
  - if $\Gamma_1'$ soft then $\Delta_1'$ soft $''$

  - $(\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa')$ By congruence
  
  - if $\Gamma_1', \beta : \kappa'$ soft then $\Delta_1, \beta : \kappa'$ soft Since $\Gamma_1', \beta : \kappa'$ is not soft.

- Case $\Gamma \rightarrow \Delta$
  
  - $(\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma_0, \alpha : \kappa, \Gamma_1)$ Given
  
  - $(\Gamma_0, \alpha : \kappa, \Gamma_1', \kappa : \kappa')$ Since the last element must be equal
  
  - $\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1')$ 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.
  
  - if $\Gamma_1'$ soft then $\Delta_1$ soft $''$

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

  Suppose $\Gamma_1', \hat{\alpha} : \kappa'$ soft.
  
  - $\Gamma_1'$ soft By definition of softness
  
  - $\Delta_1$ soft By induction
  
  - $\Delta_1$ soft By definition of softness
  
  - if $\Gamma_1', \hat{\alpha} : \kappa'$ soft then $\Delta_1, \hat{\alpha} : \kappa'$ soft Implication introduction

- Case $\Gamma \rightarrow \Delta$
  
  - $[\Delta]t = [\Delta]t'$ Solved
Similar to the \(\rightarrow\)Unsolved\) case.

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

  \[|\Delta|t = |\Delta|t'\]

  \[\frac{(\Gamma, \beta = t) = (\Gamma_0, \alpha : \kappa, \Gamma_1)}{\Gamma_0, \alpha : \kappa, \Gamma_1 \rightarrow \Delta, \beta = t'} \rightarrow\text{Eqn}\]

  Given

  Since the last element must be equal

  By injectivity of syntax

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

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

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

  By equality

  By i.h.

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

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

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

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

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

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

  By equality

  By i.h.

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

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

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

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

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

  By equality

  By i.h.

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

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

Suppose \(\Gamma_1\) soft.

\[\Delta_1 \text{ soft}\]

\[\Delta_1, \hat{\alpha} : \kappa' \text{ soft}\]

By definition of softness

\[\text{if } \Gamma_1 \text{ soft then } \Delta_1, \hat{\alpha} : \kappa' \text{ soft} \quad \text{Implication introduction}\]
Proof of **Lemma 22** (Extension Inversion)  

\[ \text{dom} \]

(iii) We have \( (\Delta_0, \alpha : \kappa, \Delta_1, \hat{\alpha} : \kappa' = t) \) By congruence of equality

Suppose \( \Delta_1 \) soft.

\( (\Delta_1, \hat{\alpha} : \kappa' = t) \) soft  
By definition of softness

\[ \begin{align*}
\text{if} \ & \Delta_1, \hat{\alpha} : \kappa' = t \text{ soft} \\
\implies & \text{Implication introduction}
\end{align*} \]

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

(iii) We have \( \Gamma_0, \alpha = \tau, \Gamma_1 \rightarrow \Delta \).
\[(\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \beta : \kappa')\]  
\[= (\Gamma_0, \alpha = \tau, \Gamma'_1, \beta : \kappa')\]  
\[\Gamma = (\Gamma_0, \alpha = \tau, \Gamma'_1)\]  
Given  
Since the final elements must be equal  
By injectivity of context syntax 

\[\Delta = (\Delta_0, \alpha = \tau', \Delta_1)\]  
By i.h.  
""  
""  
\[\langle \Delta, \beta : \kappa' \rangle = (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa')\]  
By congruence of equality

- **Case**  
  \[\Gamma \rightarrow \Delta\]  
  \[\Delta_\alpha = \alpha_\tau\]  
  \[\rightarrow \var\]  
  \[\Gamma_0, \alpha = \tau, \Gamma_1\]  
  Similar to the \[\rightarrow \var\] case.

- **Case**  
  \[\Gamma \rightarrow \Delta\]  
  \[\Gamma_\alpha \rightarrow \Delta_\alpha\]  
  \[\rightarrow \text{Marker}\]  
  \[\Gamma_0, \alpha = \tau, \Gamma_1\]  
  Similar to the \[\rightarrow \var\] case.

- **Case**  
  \[\Gamma \rightarrow \Delta\]  
  \[\Delta_\alpha = \alpha_\tau\]  
  \[\rightarrow \text{Unsolved}\]  
  \[\Gamma_0, \alpha = \tau, \Gamma_1\]  
  Similar to the \[\rightarrow \var\] case.

- **Case**  
  \[\Gamma \rightarrow \Delta\]  
  \[\Delta_\alpha = \alpha_\tau\]  
  \[\rightarrow \text{Solved}\]  
  \[\Gamma_0, \alpha = \tau, \Gamma_1\]  
  Similar to the \[\rightarrow \var\] case.

- **Case**  
  \[\Gamma \rightarrow \Delta\]  
  \[\Delta_\alpha = \alpha_\tau\]  
  \[\rightarrow \text{Solve}\]  
  \[\Gamma_0, \alpha = \tau, \Gamma_1\]  
  Similar to the \[\rightarrow \var\] case.

- **Case**  
  \[\Gamma \rightarrow \Delta\]  
  \[\Delta_\alpha = \alpha_\tau\]  
  \[\rightarrow \text{Eqn}\]  
  \[\Gamma_0, \alpha = \tau, \Gamma_1\]  
  There are two cases:

  - **Case** \(\alpha = \beta\):
    \[\tau = t\] and \(\Gamma_1 = \cdot\) and \(\Gamma_0 = \Gamma\)  
    By injectivity of syntax  
    \[\rightarrow \text{Subderivation (\(\Gamma_0 = \Gamma\) and let \(\Delta_0 = \Delta\))}\]  
    \[\rightarrow \text{where \(\Delta_1 = \cdot\)}\]  
    \[\rightarrow \text{By premise \([\Delta]t = [\Delta]t'\)}\]

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

Proof of Lemma 22 (Extension Inversion) lem:extension-inversion

\((\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \beta = t)\) \hspace{1cm} \text{Given}
\((\Gamma_0, \alpha = \tau, \Gamma_1') = (\Gamma, \beta = t)\) \hspace{1cm} \text{Since the final elements must be equal}
\(\Gamma = (\Gamma_0, \alpha = \tau, \Gamma_1')\) \hspace{1cm} \text{By injectivity of context syntax}

\(\Delta = (\Delta_0, \alpha = \tau', \Delta_1)\) \hspace{1cm} \text{By i.h.}

\[\Delta_0] \tau = [\Delta_0] \tau'\]  
\[\Gamma_0 \rightarrow \Delta_0\]  
\([\Delta, \beta = t'] = (\Delta_0, \alpha = \tau', \Delta_1, \beta = t')\) \hspace{1cm} \text{By congruence of equality}

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

\(\Gamma, \beta : \kappa' \rightarrow Add\)

\(\Delta = (\Delta_0, \alpha = \tau', \Delta_1)\) \hspace{1cm} \text{By i.h.}

\[\Delta_0] \tau = [\Delta_0] \tau'\]  
\[\Gamma_0 \rightarrow \Delta_0\]  
\([\Delta, \beta : \kappa' = t] = (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa' = t)\) \hspace{1cm} \text{By congruence of equality}

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

\(\Gamma, \beta : \kappa' \rightarrow AddSolved\)

\(\Delta = (\Delta_0, \alpha = \tau', \Delta_1)\) \hspace{1cm} \text{By i.h.}

\[\Delta_0] \tau = [\Delta_0] \tau'\]  
\[\Gamma_0 \rightarrow \Delta_0\]  
\([\Delta, \beta : \kappa' = t] = (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa' = t)\) \hspace{1cm} \text{By congruence of equality}

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

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

\(\Gamma, \beta : \kappa' \rightarrow Uvar\)

\((\Gamma_0, \alpha : \kappa = \tau, \Gamma_1) = (\Gamma, \beta : \kappa')\) \hspace{1cm} \text{Given}
\((\Gamma_0, \alpha : \kappa = \tau, \Gamma_1') = (\Gamma, \beta : \kappa')\) \hspace{1cm} \text{Since the final elements must be equal}
\(\Gamma = (\Gamma_0, \alpha : \kappa = \tau, \Gamma_1')\) \hspace{1cm} \text{By injectivity of context syntax}

\(\Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1)\) \hspace{1cm} \text{By i.h.}

\[\Delta_0] \tau = [\Delta_0] \tau'\]  
\[\Gamma_0 \rightarrow \Delta_0\]  
\([\Delta, \beta : \kappa' = t] = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa' = t)\) \hspace{1cm} \text{By congruence of equality}

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

\([\Delta] A = [\Delta] A'\) \hspace{1cm} \text{Given}
\([\Delta_0, \alpha : \kappa = \tau, \Gamma_1'] = (\Gamma, \beta : \kappa' = t)\) \hspace{1cm} \text{Since the final elements must be equal}
\(\Gamma = (\Gamma_0, \alpha : \kappa = \tau, \Gamma_1')\) \hspace{1cm} \text{By injectivity of context syntax}

\(\Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1)\) \hspace{1cm} \text{By i.h.}

\[\Delta_0] \tau = [\Delta_0] \tau'\]  
\[\Gamma_0 \rightarrow \Delta_0\]  
\([\Delta, \beta : \kappa' = t] = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa' = t)\) \hspace{1cm} \text{By congruence of equality}

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

\([\Delta] A = [\Delta] A'\) \hspace{1cm} \text{Given}
\([\Gamma_0, \alpha : \kappa = \tau, \Gamma_1'] = (\Gamma, \beta : \kappa' = t)\) \hspace{1cm} \text{Since the final elements must be equal}
\(\Gamma = (\Gamma_0, \alpha : \kappa = \tau, \Gamma_1')\) \hspace{1cm} \text{By injectivity of context syntax}

\(\Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1)\) \hspace{1cm} \text{By i.h.}

\[\Delta_0] \tau = [\Delta_0] \tau'\]  
\[\Gamma_0 \rightarrow \Delta_0\]  
\([\Delta, \beta : \kappa' = t] = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa' = t)\) \hspace{1cm} \text{By congruence of equality}

Similar to the \(\rightarrow Uvar\) case.
Proof of Lemma 22 (Extension Inversion) lem:extension-inversion

• Case \[ \Gamma \rightarrow \Delta \]
\[ \Gamma', \beta \rightarrow \Delta', \beta \]

Similar to the \[ \rightarrow \text{Uvar} \] case.

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

Similar to the \[ \rightarrow \text{Uvar} \] case.

• Case \[ \Gamma \rightarrow \Delta \]
\[ [\Delta]t = [\Delta]t' \]
\[ \Gamma', \beta : \kappa' = t \rightarrow \Delta', \beta : \kappa' = t' \]

There are two cases.

– Case \[ \hat{\alpha} = \beta \]:
\[ \kappa' = \kappa \text{ and } t = \tau \text{ and } \Gamma_1 = \cdot \text{ and } \Gamma = \Gamma_0 \]
By injectivity of syntax
\[ \Rightarrow (\Delta, \beta : \kappa' = t') = (\Delta_0, \beta : \kappa' = \tau', \Delta_1) \]
where \( \tau' = t' \) and \( \Delta_1 = \cdot \) and \( \Delta = \Delta_0 \)
\[ \Rightarrow \Gamma_0 \rightarrow \Delta_0 \]
From subderivation \( \Gamma \rightarrow \Delta \)
\[ \Rightarrow [\Delta_0] \tau = [\Delta_0] \tau' \]
From premise \( [\Delta]t = [\Delta]t' \) and \( x \)

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

• Case \[ \Gamma \rightarrow \Delta \]
\[ [\Delta]t = [\Delta]t' \]
\[ \Gamma', \beta = t \rightarrow \Delta', \beta = t' \]

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

• Case \[ \Gamma \rightarrow \Delta \]
\[ \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \]

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

\( \Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \)  
\( \therefore [\Delta_0] \tau = [\Delta_0] \tau' \)
\( \therefore \Gamma_0 \rightarrow \Delta_0 \)
\( \therefore (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') \)  
By congruence of equality

\begin{itemize}
  \item Case

\[ \begin{array}{c}
  \Gamma \rightarrow \Delta \\
  \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \\
  \hline
  \Gamma_0, \hat{\beta} : \kappa' \rightarrow \Delta, \hat{\beta} : \kappa' = t \\
\end{array} \]

\( \Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \)  
By i.h.
\( [\Delta_0] \tau = [\Delta_0] \tau' \)
\( \Gamma_0 \rightarrow \Delta_0 \)
\( (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') \)  
By congruence of equality

\begin{itemize}
  \item Case

\[ \begin{array}{c}
  \Gamma \rightarrow \Delta \\
  \Gamma_0, \hat{\beta} : \kappa' \rightarrow \Delta, \hat{\beta} : \kappa' = t \\
  \hline
  \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \\
\end{array} \]

\( (\Gamma, \hat{\beta} : \kappa') = (\Gamma, \hat{\alpha} : \kappa = \tau, \Gamma_1) \)  
Given
\( = (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa') \)  
Since the last elements must be equal
\( \Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') \)  
By injectivity of syntax

\[ \Gamma \rightarrow \Delta \]
\( \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1' \rightarrow \Delta \)
\( \Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \)  
By equality
\( [\Delta_0] \tau = [\Delta_0] \tau' \)
\( \Gamma_0 \rightarrow \Delta_0 \)
\( (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') \)  
By congruence of equality

\end{itemize}

\( (v) \) We have \( \Gamma_0, x : A, \Gamma_1 \rightarrow \Delta \). 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 \( \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \rightarrow \Delta \).

\begin{itemize}
  \item Case

\[ \begin{array}{c}
  \Gamma \rightarrow \Delta \\
  \Gamma_0, \hat{\beta} : \kappa' \rightarrow \Delta, \hat{\beta} : \kappa' = t \\
  \hline
  \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \\
\end{array} \]

\( (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1) = (\Gamma, \hat{\beta} : \kappa') \)  
Given
\( = (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa') \)  
Since the final elements must be equal
\( \Gamma = (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') \)  
By injectivity of context syntax

By induction, there are two possibilities:

- \( \hat{\alpha} \) is not solved:
  \( \Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) \)  
  By i.h.
  \( \therefore \Gamma_0 \rightarrow \Delta_0 \)
  
- \( \Delta, \hat{\beta} : \kappa' \) = \( (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \hat{\beta} : \kappa') \)  
  By congruence of equality
\end{proof}
Proof of Lemma 22 (Extension Inversion) lem:extension-inversion

- \( \hat{\alpha} \) is solved:
  \[ \Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \]
  By i.h.

  \[ \Gamma_0 \rightarrow \Delta_0 \] 

  \[ (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') \]
  By congruence of equality

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

  \[ [\Delta] = [\Delta] \]

  \[ \Gamma, x : A \rightarrow \Delta, x : A' \] 

  \[ \Gamma_0, \& \kappa, \Gamma_1 \] 

  Similar to the \[ \rightarrow \text{UVar} \] case.

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

  \[ \Gamma, \triangleright \beta \rightarrow \Delta, \triangleright \beta \] 

  \[ \Gamma_0, \& \kappa, \Gamma_1 \] 

  Similar to the \[ \rightarrow \text{UVar} \] case.

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

  \[ [\Delta] \tau = [\Delta] \tau' \]

  \[ \Gamma, \hat{\beta} = t \rightarrow \Delta, \hat{\beta} = t' \] 

  \[ \rightarrow \text{Eqn} \] 

  Similar to the \[ \rightarrow \text{UVar} \] case.

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

  \[ [\Delta] \tau = [\Delta] \tau' \]

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

  \[ \rightarrow \text{Solved} \] 

  Similar to the \[ \rightarrow \text{UVar} \] case.

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

  \[ \Gamma, \hat{\beta} : \kappa' \rightarrow \Delta, \hat{\beta} : \kappa' \] 

  \[ \rightarrow \text{Unsolved} \] 

  - Case \( \hat{\alpha} \neq \hat{\beta} \):

    \[ (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1) = (\Gamma, \hat{\beta} : \kappa') \]

    Given

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

    Since the final elements must be equal

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

    By injectivity of context syntax

  By induction, there are two possibilities:

  * \( \hat{\alpha} \) is not solved:

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

    By i.h.

    \[ \Gamma_0 \rightarrow \Delta_0 \]

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

    By congruence of equality

  * \( \hat{\alpha} \) is solved:

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

    By i.h.

    \[ \Gamma_0 \rightarrow \Delta_0 \]

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

    By congruence of equality

  - Case \( \hat{\alpha} = \hat{\beta} \):

    \[ \Gamma \rightarrow \Delta = \Delta_0 \]

    \[ \hat{\alpha} = \hat{\beta} \]
\[
\kappa' = \kappa \text{ and } \Gamma_0 = \Gamma \text{ and } \Gamma_1 = \cdot \quad \text{By injectivity of syntax}
\]
\[
\Delta, \beta : \kappa' = (\Delta_0, \alpha : \kappa, \Delta_1)
\]
\[
\Gamma_0 \to \Delta_0
\]
\[
\text{From premise } \Gamma \to \Delta
\]

- **Case**

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

\[
\Delta, \beta : \kappa' \to \Delta_0 \quad \text{Add}
\]

By induction, there are two possibilities:

- \(\alpha\) is not solved:

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

\[
\Gamma_0 \to \Delta_0
\]

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

- \(\alpha\) is solved:

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

\[
\Gamma_0 \to \Delta_0
\]

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

- **Case**

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

\[
\Delta, \beta : \kappa' \to \Delta_0 \quad \text{AddSolved}
\]

By induction, there are two possibilities:

- \(\alpha\) is not solved:

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

\[
\Gamma_0 \to \Delta_0
\]

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

- \(\alpha\) is solved:

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

\[
\Gamma_0 \to \Delta_0
\]

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

- **Case**

\[
\Gamma \to \Delta
\]

\[
\Delta, \beta : \kappa' \to \Delta_0 \quad \text{Solve}
\]

By induction, there are two possibilities:

- 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:
Lemma 23 (Deep Evar Introduction). (i) If $\Gamma_0, \Gamma_1$ is well-formed and $\alpha$ is not declared in $\Gamma_0, \Gamma_1$ then $\Gamma_0, \Gamma_1 \vdash \Gamma_0, \alpha : \kappa, \Gamma_1$.

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

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

Proof.

(i) Assume that $\Gamma_0, \Gamma_1$ is well-formed. We proceed by induction on $\Gamma_1$.

- Case $\Gamma_1 = \cdot$:
  
  $\Gamma_0 \ctx \quad \text{Given}$
  
  $\alpha \notin \text{dom}(\Gamma_0) \quad \text{Given}$
  
  $\Gamma_0, \alpha : \kappa \ctx \quad \text{By rule VarCtx}$
  
  $\Gamma_0 \rightarrow \Gamma_0 \quad \text{By Lemma 32 (Extension Reflexivity)}$
  
  $\Gamma_0 \rightarrow \Gamma_0, \alpha : \kappa \quad \text{By rule $\rightarrow$Add}$

- Case $\Gamma_1 = \Gamma_1', x : A$:
  
  $\Gamma_0, \Gamma_1', x : A \ctx \quad \text{Given}$
  
  $\Gamma_0, \Gamma_1' \ctx \quad \text{By inversion}$
  
  $x \notin \text{dom}(\Gamma_0, \Gamma_1') \quad \text{By inversion (1)}$
  
  $\Gamma_0, \Gamma_1' \vdash A \text{ type} \quad \text{By inversion}$
  
  $\alpha \notin \text{dom}(\Gamma_0, \Gamma_1', x : A) \quad \text{Given}$
  
  $\alpha \neq x \quad \text{By inversion (2)}$
  
  $\Gamma_0, \alpha : \kappa, \Gamma_1' \ctx \quad \text{By i.h.}$
  
  $\Gamma_0, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1' \quad \text{"}$
  
  $\Gamma_0, \alpha : \kappa, \Gamma_1' \vdash A \text{ type} \quad \text{By Lemma 36 (Extension Weakening (Sorts))}$
  
  $x \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma_1') \quad \text{By (1) and (2)}$
  
  $\Gamma_0, \Gamma_1', x : A \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1', x : A \quad \text{By $\rightarrow$Var}
Proof of Lemma 23 (Deep Evar Introduction)

- Case $\Gamma_1 = \Gamma_1', \beta : \kappa'$:
  \[
  \begin{align*}
  \Gamma_0, \Gamma_1', \beta : \kappa' : \text{ctx} & \quad \text{Given} \\
  \Gamma_0, \Gamma_1' : \text{ctx} & \quad \text{By inversion} \\
  \beta \not\in \text{dom}(\Gamma_0, \Gamma_1') & \quad \text{By inversion (1)} \\
  \alpha \not\in \text{dom}(\Gamma_0, \Gamma_1', \beta : \kappa') & \quad \text{Given} \\
  \alpha \neq \beta & \quad \text{By inversion (2)} \\
  \Gamma_0, \beta : \kappa, \Gamma_1' : \text{ctx} & \quad \text{By i.h.} \\
  \Gamma_0, \Gamma_1' \rightarrow \Gamma_0, \beta : \kappa, \Gamma_1' & \quad "" \\
  \beta \not\in \text{dom}(\Gamma_0, \beta : \kappa, \Gamma_1') & \quad \text{By (1) and (2)} \\
  \end{align*}
  \]
  \[\Gamma_0, \Gamma_1', \beta : \kappa' \rightarrow \Gamma_0, \beta : \kappa, \Gamma_1' \quad \text{By } \rightarrow \text{Uvar} \]

- Case $\Gamma_1 = \Gamma_1', \hat{\beta} : \kappa'$:
  \[
  \begin{align*}
  \Gamma_0, \Gamma_1', \hat{\beta} : \kappa' : \text{ctx} & \quad \text{Given} \\
  \Gamma_0, \Gamma_1' : \text{ctx} & \quad \text{By inversion} \\
  \hat{\beta} \not\in \text{dom}(\Gamma_0, \Gamma_1') & \quad \text{By inversion (1)} \\
  \alpha \not\in \text{dom}(\Gamma_0, \Gamma_1', \hat{\beta} : \kappa') & \quad \text{Given} \\
  \alpha \neq \hat{\beta} & \quad \text{By inversion (2)} \\
  \Gamma_0, \alpha : \kappa, \Gamma_1' : \text{ctx} & \quad \text{By i.h.} \\
  \Gamma_0, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1' & \quad "" \\
  \hat{\beta} \not\in \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma_1') & \quad \text{By (1) and (2)} \\
  \end{align*}
  \]
  \[\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1' \quad \text{By } \rightarrow \text{Unsolved} \]

- Case $\Gamma_1 = (\Gamma_1', \hat{\beta} : \kappa')$:
  \[
  \begin{align*}
  \Gamma_0, \Gamma_1', \hat{\beta} : \kappa' = t : \text{ctx} & \quad \text{Given} \\
  \Gamma_0, \Gamma_1' : \text{ctx} & \quad \text{By inversion} \\
  \hat{\beta} \not\in \text{dom}(\Gamma_0, \Gamma_1') & \quad \text{By inversion (1)} \\
  \Gamma_0, \Gamma_1' \vdash t : \kappa' & \quad \text{By inversion} \\
  \alpha \not\in \text{dom}(\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' = t) & \quad \text{Given} \\
  \alpha \neq \hat{\beta} & \quad \text{By inversion (2)} \\
  \Gamma_0, \alpha : \kappa, \Gamma_1' : \text{ctx} & \quad \text{By i.h.} \\
  \Gamma_0, \Gamma_1' \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1' & \quad "" \\
  \Gamma_0, \alpha : \kappa, \Gamma_1' \vdash t : \kappa' & \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \\
  \hat{\beta} \not\in \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma_1') & \quad \text{By (1) and (2)} \\
  \end{align*}
  \]
  \[\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' = t \rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1' \quad \text{By } \rightarrow \text{Solved} \]

- Case $\Gamma_1 = (\Gamma_1', \beta = t)$:
Proof of Lemma 23 (Deep Evar Introduction) lem:deep-existent

\[ \Gamma_0, \Gamma_1', \beta = t \text{ ctx} \quad \text{Given} \]
\[ \Gamma_0, \Gamma_1' \text{ ctx} \quad \text{By inversion} \]
\[ \beta \notin \text{dom}(\Gamma_0, \Gamma_1') \quad \text{By inversion (1)} \]
\[ \Gamma_0, \Gamma_1' \vdash t : N \quad \text{By inversion} \]
\[ \hat{\alpha} \notin \text{dom}(\Gamma_0, \Gamma_1', \beta = t) \quad \text{Given} \]
\[ \hat{\alpha} \neq \beta \quad \text{By inversion (2)} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \text{ ctx} \quad \text{By i.h.} \]
\[ \Gamma_0, \Gamma_1' \rightarrow \Gamma_0, \Gamma_1' \quad \text{"} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \]
\[ \beta \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') \quad \text{By (1) and (2)} \]
\[ \Rightarrow \quad \Gamma_0, \Gamma_1', \beta = t \rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta = t \quad \text{By Marker} \]

- Case \( \Gamma_1 = (\Gamma_1', \bullet_\beta) \):

\[ \Gamma_0, \Gamma_1, \bullet_\beta \text{ ctx} \quad \text{Given} \]
\[ \Gamma_0, \Gamma_1' \text{ ctx} \quad \text{By inversion} \]
\[ \beta \notin \text{dom}(\Gamma_0, \Gamma_1') \quad \text{By inversion (1)} \]
\[ \hat{\alpha} \notin \text{dom}(\Gamma_0, \Gamma_1', \bullet_\beta) \quad \text{Given} \]
\[ \hat{\alpha} \neq \beta \quad \text{By inversion (2)} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \text{ ctx} \quad \text{By i.h.} \]
\[ \Gamma_0, \Gamma_1' \rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \quad \text{"} \]
\[ \beta \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') \quad \text{By (1) and (2)} \]
\[ \Rightarrow \quad \Gamma_0, \Gamma_1', \bullet_\beta \rightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \bullet_\beta \quad \text{By Marker} \]

(ii) Assume \( \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \text{ ctx} \). We proceed by induction on \( \Gamma_1 \):

- Case \( \Gamma_1 = \cdot \):

\[ \Gamma_0 \vdash t : \kappa \quad \text{Given} \]
\[ \Gamma_0, \Gamma_1 \text{ ctx} \quad \text{Given} \]
\[ \Gamma_0 \text{ ctx} \quad \text{Since } \Gamma_1 = \cdot \]
\[ \Gamma_0 \rightarrow \Gamma_0 \quad \text{By Lemma 32 (Extension Reflexivity)} \]
\[ \Gamma_0, \hat{\alpha} : \kappa \rightarrow \Gamma_0, \hat{\alpha} : \kappa = t \quad \text{By rule Var} \]
\[ \Rightarrow \quad \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \rightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \quad \text{Since } \Gamma_1 = \cdot \]

- Case \( \Gamma_1 = (\Gamma_1', \alpha : A) \):

\[ \Gamma_0 \vdash t : \kappa \quad \text{Given} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', x : A \text{ ctx} \quad \text{Given} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \text{ ctx} \quad \text{By inversion} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash A \text{ type} \quad \text{By inversion} \]
\[ x \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') \quad \text{By inversion (1)} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \rightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \quad \text{By i.h.} \]
\[ \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \vdash A \text{ type} \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \]
\[ x \notin \text{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1') \quad \text{Since this is the same domain as (1)} \]
\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', x : A \rightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1, x : A \quad \text{By rule Var} \]

- Case \( \Gamma_1 = (\Gamma_1', \beta : \kappa') \):
\[
\Gamma_0 \vdash t : \kappa \\
\Gamma_0, \alpha : \kappa, \Gamma'_0, \beta : \kappa \equiv ctx \\
\Gamma_0, \alpha : \kappa, \Gamma'_0 \text{ ctx} \\
\beta \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma'_0) \\
\Gamma_0, \alpha : \kappa, \Gamma' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \\
\beta \notin \text{dom}(\Gamma_0, \alpha : \kappa = t, \Gamma'_0) \\
\Gamma_0, \alpha : \kappa, \Gamma'_0, \beta : \kappa' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, \beta : \kappa' 
\]

\begin{itemize}
  \item Case $\Gamma_1 = (\Gamma'_0, \beta : \kappa')$:
    \[
    \Gamma_0 \vdash t : \kappa \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0, \beta : \kappa' \equiv ctx \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0 \text{ ctx} \\
    \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma'_0) \\
    \Gamma_0, \alpha : \kappa, \Gamma' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \\
    \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa = t, \Gamma'_0) \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0, \beta : \kappa' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, \beta : \kappa' 
    \]
    \begin{itemize}
      \item Given
      \item Given
      \item By inversion
      \item By inversion (1)
      \item By i.h.
      \item By i.h.
      \item By rule \text{Unsolvable}
    \end{itemize}
  \item Case $\Gamma_1 = (\Gamma'_0, \beta : \kappa' = t')$:
    \[
    \Gamma_0 \vdash t' : \kappa \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0, \beta : \kappa' = t' \equiv ctx \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0 \text{ ctx} \\
    \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma'_0) \\
    \Gamma_0, \alpha : \kappa, \Gamma' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \\
    \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa = t, \Gamma'_0) \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0, \beta : \kappa' = t' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, \beta : \kappa' = t' 
    \]
    \begin{itemize}
      \item By Lemma \text{36} (Extension Weakening (Sorts))
      \item Given
      \item Given
      \item By inversion
      \item By inversion (1)
      \item By i.h.
      \item By i.h.
      \item By rule \text{Solved}
    \end{itemize}
  \item Case $\Gamma_1 = (\Gamma'_0, \beta = t')$:
    \[
    \Gamma_0 \vdash t' : \kappa \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0, \beta = t' \equiv ctx \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0 \text{ ctx} \\
    \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa, \Gamma'_0) \\
    \Gamma_0, \alpha : \kappa, \Gamma' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1 \\
    \beta \notin \text{dom}(\Gamma_0, \alpha : \kappa = t, \Gamma'_0) \\
    \Gamma_0, \alpha : \kappa, \Gamma'_0, \beta = t' \rightarrow \Gamma_0, \alpha : \kappa = t, \Gamma_1, \beta = t' 
    \]
    \begin{itemize}
      \item By Lemma \text{36} (Extension Weakening (Sorts))
      \item Given
      \item Given
      \item By inversion
      \item By inversion (1)
      \item By i.h.
      \item By i.h.
      \item By rule \text{Eqn}
    \end{itemize}
  \item Case $\Gamma_1 = (\Gamma'_0, \beta \equiv \beta)$:
\end{itemize}
Proof of **Lemma 23** (Deep Evar Introduction).

\[ \Gamma_0 \vdash t : \kappa \]
\[ \Gamma_0, \beta : \kappa, \Gamma'_1 \vdash \text{ctx} \]
\[ \Gamma_0, \beta : \kappa, \Gamma'_1 \vdash \text{ctx} \]
\[ \beta \notin \text{dom} (\Gamma_0, \beta : \kappa, \Gamma'_1) \]
\[ \beta \notin \text{dom} (\Gamma_0, \beta : \kappa = t, \Gamma'_1) \]
\[ \Gamma_0, \beta : \kappa, \Gamma'_1, \beta \vdash \text{ctx} \]

(iii) Apply parts (i) and (ii) as lemmas, then Lemma 33 (Extension Transitivity).

**Lemma 26** (Parallel Admissibility).

If \( \Gamma_L \rightarrow \Delta_L \) and \( \Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta_R \) then:

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

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

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

Proof. By induction on \( \Delta_R \). As always, we assume that all contexts mentioned in the statement of the lemma are well-formed. Hence, \( \beta \notin \text{dom} (\Gamma_L) \cup \text{dom} (\Gamma_R) \cup \text{dom} (\Delta_L) \cup \text{dom} (\Delta_R) \).

(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 that 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 \( \Delta_L \) again. It’s easy to avoid decomposing \( \Delta_L \): when \( \Delta_R = \), 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 \( \Delta_R = \)\**: We have \( \Gamma_L \rightarrow \Delta_L \). Applying \( \text{Unsolved} \) to that derivation gives the result.

- **Case \( \Delta_R = (\Delta'_R, \beta) : \kappa = t \)**: We have \( \beta \neq \beta \) by the well-formedness assumption.

The concluding rule of \( \Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta'_R, \beta \) must have been \( \text{Unsolved} \) or \( \text{Add} \). In both cases, the result follows by i.h. and applying \( \text{Unsolved} \) or \( \text{Add} \). Note: In \( \text{Unsolved} \), the left-hand context doesn’t change, so we clearly maintain \( \Gamma_L \rightarrow \Delta_L \). In \( \text{Add} \), we can properly apply the i.h. because \( \Gamma_R \neq \). Suppose, for a contradiction, that \( \Gamma_R = \). Then \( \Gamma_L = (\Gamma'_L, \beta) \). It was given that \( \Gamma_L \rightarrow \Delta_L \), that is, \( \Gamma'_L, \beta \rightarrow \Delta_L \). 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 \( \Gamma_R \neq \).

- **Case \( \Delta_R = (\Delta'_R, \beta : \kappa = t) \)**: We have \( \beta \neq \beta \) by the well-formedness assumption.

The concluding rule must have been \( \text{Solved} \) or \( \text{AddSolved} \). In each case, apply the i.h. and then the corresponding rule. (In \( \text{Solved} \) and \( \text{Add} \), use Lemma 19 (Declaration Preservation) to show \( \Gamma_R \neq \)).

- **Case \( \Delta_R = (\Delta'_R, \beta : \kappa = \tau) \)**: The concluding rule must have been \( \text{Eqn} \). The result follows by i.h. and applying \( \text{Eqn} \).

- **Case \( \Delta_R = (\Delta'_R, \beta : \kappa) \)**: Similar to the previous case, with rule \( \text{Marker} \).

- **Case \( \Delta_R = (\Delta'_R, \beta : A) \)**: Similar to the previous case, with rule \( \text{Var} \).

(ii) Similar to part (i), except that when \( \Delta_R = \), apply rule \( \text{Solve} \)
(iii) Similar to part (i), except that when $\Delta_R = \cdot$, apply rule $\rightarrow$Solved using the given equality to satisfy the second premise.

Lemma 27 (Parallel Extension Solution).

If $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$ and $\Gamma_L \vdash \tau : \kappa$ and $[\Delta_L]\tau = [\Delta_L]\tau'$
then $\Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R \rightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

Proof. By induction on $\Delta_R$.

In the case where $\Delta_R = \cdot$, we know that rule $\rightarrow$Solve must have concluded the derivation (we can use Lemma 19 (Declaration Preservation) to get a contradiction that rules out $\rightarrow$AddSolved); then we have a subderivation $\Gamma_L \rightarrow \Delta_L$, to which we can apply $\rightarrow$Solved.

Lemma 28 (Parallel Variable Update).

If $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_0, \Delta_R$ and $\Gamma_L \vdash \tau_1 : \kappa$ and $[\Delta_L]\tau_0 = [\Delta_L]\tau_1 = [\Delta_L]\tau_2$
then $\Gamma_L, \hat{\alpha} : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_2, \Delta_R$.

Proof. By induction on $\Delta_R$. Similar to the proof of Lemma 27 (Parallel Extension Solution), but applying $\rightarrow$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 $\Gamma \vdash A$ type then $[\Delta][\Gamma]A = [\Delta]A$.

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 $\Gamma \vdash t : \kappa$.

  − Case $\hat{\alpha} : \kappa \in \Gamma$

    \[
    \Gamma \vdash \hat{\alpha} : \kappa \quad \text{VarSort}
    \]

    $[\Gamma]\hat{\alpha} = \hat{\alpha}$ Since $\hat{\alpha}$ is not solved in $\Gamma$

    $[\Delta]\hat{\alpha} = [\Delta]\hat{\alpha}$ Reflexivity

    $= [\Delta][\Gamma]\hat{\alpha}$ By above equality

  − Case $(\alpha : \kappa) \in \Gamma$

    \[
    \Gamma \vdash \alpha : \kappa \quad \text{VarSort}
    \]

Consider whether or not there is a binding of the form $(\alpha = \tau) \in \Gamma$.

* Case $(\alpha = \tau) \in \Gamma$:
Proof of Lemma 29 (Substitution Monotonicity)  

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

By Lemma 22 (Extension Inversion) (i)

\[ \Gamma_0 \Rightarrow \Delta_0 \]

D' < D

(1) \([\Delta_0] \tau' = [\Delta_0] \tau\]

By i.h.

(2) \([\Delta_0][\Gamma_0] \alpha = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0, \alpha = \tau, \Gamma_1] \alpha\]

By definition

Since \( \alpha \notin \text{dom}(\Gamma_1) \)

\([\Delta_0][\Gamma_0] \tau\]

Since \( \text{FV}([\Gamma_0] \tau) \cap \text{dom}(\Delta_1) = \emptyset \)

By (2) and (1)

\([\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0, \alpha = \tau, \Gamma_1] \alpha\]

By definition of substitution

Since \( \text{FV}([\Delta_0] \tau) \cap \text{dom}(\Delta_1) = \emptyset \)

By definition of \( \Delta \)

* Case \( (\alpha = \tau) \notin \Gamma \):

\([\Gamma] \alpha = \alpha\]

By definition of substitution

\([\Delta][\Gamma] \alpha = [\Delta] \alpha\]

Apply \([\Delta]\) to both sides

– Case \( \Gamma, \hat{\alpha} : \kappa \vdash \Gamma_1 \vdash \hat{\alpha} : \kappa \)  

SolvedVarSort

Similar to the VarSort case.

– Case \( \Gamma \vdash 1 : \Delta \)

SolvedVarSort

\([\Delta] 1 = 1 = [\Delta][\Gamma] 1\]

Since \( \text{FV}(1) = \emptyset \)

– Case \( \Gamma \vdash \tau_1 : \Delta \)  

SolvedVarSort

\([\Delta][\Gamma] \tau_1 = [\Delta] \tau_1\]

By i.h.

\([\Delta][\Gamma] \tau_2 = [\Delta] \tau_2\]

By i.h.

\([\Delta][\Gamma] (\tau_1 \oplus \tau_2) = [\Delta] (\tau_1 \oplus \tau_2)\]

By congruence of equality

Definition of substitution

– Case \( \Gamma \vdash \text{zero} : \kappa \)  

\([\Delta] \text{zero} = \text{zero} = [\Delta][\Gamma] \text{zero}\]

Since \( \text{FV}(\text{zero}) = \emptyset \)

– Case \( \Gamma \vdash t : \kappa \)

\([\Delta][\Gamma] t = [\Delta] t\]

By i.h.

\(\text{succ}([\Delta][\Gamma] t) = \text{succ}([\Delta] t)\]

By congruence of equality

\([\Delta][\Gamma] \text{succ}(t) = [\Delta] \text{succ}(t)\]

By definition of substitution

Proof of Lemma 29 (Substitution Monotonicity)  

lem:substitution-monotonicity
Proof of **Lemma 29** (*Substitution Monotonicity*)

*Proof of Part (ii):* We have a derivation of $\Gamma \vdash P$ *prop*, and will use the previous part as a lemma.

- **Case**

  $\Gamma \vdash t : N \quad \Gamma \vdash t' : N \quad \text{EqProp}$

  $[\Delta][\Gamma]t = [\Delta]t$ \quad By part (i)

  $[\Delta][\Gamma]t' = [\Delta]t'$ \quad By part (i)

  $([\Delta][\Gamma]t = [\Delta][\Gamma]t') = ([\Delta]t = [\Delta]t')$ \quad By congruence of equality

  $[\Delta][\Gamma](t = t') = [\Delta](t = t')$ \quad Definition of substitution

*Proof of Part (iii):* By induction on the derivation of $\Gamma \vdash A$ *type*, using the previous parts as lemmas.

- **Case**

  $(u : \ast) \in \Gamma \quad \text{VarWF}$

  $\Gamma \vdash u : \ast$ \quad By rule *VarSort*

  $[\Delta][\Gamma]u = [\Delta]u$ \quad By part (i)

- **Case**

  $(\check{\alpha} : \ast = \tau) \in \Gamma \quad \text{SolvedVarWF}$

  $\Gamma \vdash \check{\alpha} : \ast$ \quad By rule *SolvedVarSort*

  $[\Delta][\Gamma]\check{\alpha} = [\Delta]\check{\alpha}$ \quad By part (i)

- **Case**

  $\Gamma \vdash 1$ *type* \quad \text{UnitWF}

  $\Gamma \vdash 1 : \ast$ \quad By rule *UnitSort*

  $[\Delta][\Gamma]1 = [\Delta]1$ \quad By part (i)

- **Case**

  $\Gamma \vdash A_1$ *type* \quad $\Gamma \vdash A_2$ *type* \quad \text{BinWF}

  $[\Delta][\Gamma]A_1 = [\Delta]A_1$ \quad By i.h.

  $[\Delta][\Gamma]A_2 = [\Delta]A_2$ \quad By i.h.

  $[\Delta][\Gamma](A_1 \oplus A_2) = [\Delta](A_1 \oplus A_2)$ \quad By congruence of equality

  $[\Delta][\Gamma](\forall \alpha : \kappa. A_0) = [\Delta](\forall \alpha : \kappa. A_0)$ \quad By definition of substitution

- **Case** \text{VecWF} \quad Similar to the *BinWF* case.

- **Case**

  $\Gamma, \alpha : \kappa \vdash A_0$ *type* \quad \text{ForallWF}

  $\Gamma \vdash \forall \alpha : \kappa. A_0$ *type* \quad Given

  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa$ \quad By rule *Uvar*

  $[\Delta, \alpha : \kappa][\Gamma, \alpha : \kappa]A_0 = [\Delta, \alpha : \kappa]A_0$ \quad By i.h.

  $[\Delta][\Gamma]A_0 = [\Delta]A_0$ \quad By definition of substitution

  $\forall \alpha : \kappa. [\Delta][\Gamma]A_0 = \forall \alpha : \kappa. [\Delta]A_0$ \quad By congruence of equality

  $[\Delta][\Gamma](\forall \alpha : \kappa. A_0) = [\Delta](\forall \alpha : \kappa. A_0)$ \quad By definition of substitution
Proof of Lemma 29 (Substitution Monotonicity) \[\text{lem:substitution-monotonicity}\]

- Case ExistsWF: Similar to the ForallWF case.

- Case \(\Gamma \vdash P \text{ prop} \quad \Gamma \vdash A_0 \text{ type} \):
  \[
  \frac{\Gamma \vdash P \supset A_0 \text{ type}}{\Gamma \vdash P \supset A_0 \text{ type}} \text{ImpliesWF}
  \]
  |\(\Delta|\Gamma|P = |\Delta|P\) By part (ii)
  |\(\Delta|\Gamma|A_0 = |\Delta|A_0\) By i.h.
  |\(\Delta|\Gamma|P \supset |\Delta|P \supset |\Delta|A_0\) By congruence of equality
  |\(\Delta|\Gamma|P \supset A_0) = |\Delta|P \supset |\Delta|A_0)\) Definition of substitution

- Case \(\Gamma \vdash P \text{ prop} \quad \Gamma \vdash A_0 \text{ type} \):
  \[
  \frac{\Gamma \vdash A_0 \wedge P \text{ type}}{\Gamma \vdash A_0 \wedge P \text{ type}} \text{WithWF}
  \]
  Similar to the ImpliesWF case.

Lemma 30 (Substitution Invariance).

(i) If \(\Gamma \rightarrow \Delta\) and \(\Gamma \vdash t : \kappa\) and FEV(\(\Gamma|t\)) = \emptyset then \(|\Delta|\Gamma|t = |\Gamma|t|.

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

(iii) If \(\Gamma \rightarrow \Delta\) and \(\Gamma \vdash A \text{ type}\) and 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', \& : \kappa = t|):
  \]
  \[
  \\frac{\Gamma, \Theta' \rightarrow \Gamma, \Omega'}{\Theta, \& : \kappa = t \rightarrow \Delta, \Omega', \& : \kappa = t \rightarrow \Omega} \text{ By i.h. Solved}
  \]

- Case \(\Theta = (\Theta', \& : \kappa):\)
  \[
  \frac{\Gamma, \Theta' \rightarrow \Gamma, \Omega'}{\Theta, \& : \kappa \rightarrow \Delta, \Omega', \& : \kappa = t \rightarrow \Omega} \text{ By i.h. Solve}
  \]

Lemma 31 (Split Extension).
If \(\Delta \rightarrow \Omega\)
and \(\& \in \text{unsolved}(\Delta]\)
and \(\Omega = \Omega_1[\& : \kappa = t_1]\)
and \(\Omega\) is canonical (Definition 3)
and \(\Omega \vdash t_2 : \kappa\)
then \(\Delta \rightarrow \Omega_1[\& : \kappa = t_2].\)
Proof of Lemma 31 (Split Extension)

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 $\Omega \vdash t_2 : \kappa$. \qed

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
  - $\text{EmptyCtx}$
    - $\rightarrow \cdot$ By rule $\rightarrow\text{Id}$
  
  - Case $\Gamma \text{ctx} \quad x \not\in \text{dom}(\Gamma) \quad \Gamma \vdash A$ type
    - $\Gamma, x : A\ \text{ctx}$ by HypCtx
    - $\Gamma \rightarrow \Gamma$ By i.h.
    - $[\Gamma]A = [\Gamma]A$ By reflexivity
    - $\Gamma, x : A \rightarrow \Gamma, x : A$ By rule $\rightarrow\text{Var}$

  
  - Case $\Gamma \text{ctx} \quad u : \kappa \not\in \text{dom}(\Gamma)$
    - $\Gamma, u : \kappa\ \text{ctx}$ by VarCtx
    - $\Gamma \rightarrow \Gamma$ By i.h.
    - $\Gamma, u : \kappa \rightarrow \Gamma, u : \kappa$ By rule $\rightarrow\text{Uvar}$ or $\rightarrow\text{Unsolved}$

  
  - Case $\Gamma \text{ctx} \quad \hat{\alpha} \not\in \text{dom}(\Gamma)$
    - $\Gamma, \hat{\alpha} : \kappa = t\ \text{ctx}$ by SolvedCtx
    - $\Gamma \rightarrow \Gamma$ By i.h.
    - $[\Gamma]t = [\Gamma]t$ By reflexivity
    - $\Gamma, \hat{\alpha} : \kappa = t \rightarrow \Gamma, \hat{\alpha} : \kappa = t$ By rule $\rightarrow\text{Solved}$

  
  - Case $\Gamma \text{ctx} \quad \alpha \in \Gamma \quad (\alpha = -) \not\in \Gamma \quad \Gamma \vdash \tau : \kappa$
    - $\Gamma, \alpha = \tau\ \text{ctx}$ by EqnVarCtx
    - $\Gamma \rightarrow \Gamma$ By i.h.
    - $[\Gamma]t = [\Gamma]t$ By reflexivity
    - $\Gamma, \alpha = t \rightarrow \Gamma, \alpha = t$ By rule $\rightarrow\text{Eqn}$
Proof of Lemma 32 (Extension Reflexivity).

\[ \Gamma \text{ctx} \not\exists u \not\in \Gamma \]
\[ \Gamma, \Gamma ; \not\exists u \text{ctx} \]

Case \[ \Gamma \Gamma, \Gamma ; \not\exists u \text{ctx} \]

By i.h.

\[ \Gamma, \Gamma ; \not\exists u \text{ctx} \]

By rule \[ \not\exists \text{Marker} \]

\[ \Gamma \Gamma \]

Lemma 33 (Extension Transitivity).

If \( D : \Gamma \rightarrow \Theta \) and \( D' : \Theta \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).

Proof. By induction on \( D' \).

Case \[ \Theta \rightarrow \Delta \]

By inversion on \( D \)

\[ \Theta \rightarrow \Delta \]

By rule \[ \not\exists \text{Id} \]

\[ \Gamma = \cdot \]

Since \( \Gamma = \Delta = \cdot \)

Case \[ \Theta' \rightarrow \Delta' \]

By congruence of equality

By Lemma 29 (Substitution Monotonicity)

\[ [\Delta']A'' = [\Delta']A \]

By premise \( [\Delta']A = [\Delta']A' \)

\[ \Gamma', x : A'' \rightarrow \Delta', x : A' \]

By \[ \not\exists \text{Var} \]

Case \[ \Theta' \rightarrow \Delta' \]

By inversion on \( D \)

\[ [\Theta']A'' = [\Theta]A \]

By inversion on \( D \)

\[ \Gamma' \rightarrow \Theta' \]

By i.h.

\[ \Gamma' \rightarrow \Delta' \]

By \[ \not\exists \text{Uvar} \]

Case \[ \Theta' \rightarrow \Delta' \]

By inversion on \( D \)

\[ \Gamma' \rightarrow \Theta' \]

By i.h.

\[ \Gamma' \rightarrow \Delta' \]

By \[ \not\exists \text{Uvar} \]

Case \[ \Theta' \rightarrow \Delta' \]

By inversion on \( D \)

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

By \[ \not\exists \text{Uvar} \]

Case \[ \Theta' \rightarrow \Delta' \]

By \[ \not\exists \text{Var} \]

Two rules could have concluded \( D : \Gamma \rightarrow (\Theta', \alpha : \kappa) \):
Proof of Lemma 33 (Extension Transitivity)

\begin{proof}

- Case \( \Gamma' \rightarrow \Theta' \)
  \[
  \Gamma', \hat{\alpha} : \kappa \rightarrow \Theta', \hat{\alpha} : \kappa \\
  \Gamma
  \]
  \( \Gamma' \rightarrow \Delta' \)

  By i.h.

  \( \Gamma', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa \)
  By rule \( \rightarrow \text{Add} \)

- Case \( \Gamma \rightarrow \Theta' \)
  \[
  \Gamma \rightarrow \Theta', \hat{\alpha} : \kappa \\
  \rightarrow \text{Add}
  \]
  \( \Gamma \rightarrow \Delta' \)

  By i.h.

  \( \Gamma \rightarrow \Delta', \hat{\alpha} : \kappa \)
  By rule \( \rightarrow \text{Add} \)

- Case \( \Theta' \rightarrow \Delta' \) \( \Delta'[\mathbf{t}] = \Delta'[\mathbf{t}'] \)
  \[
  \Theta', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa = \mathbf{t'} \\
  \rightarrow \text{Solved}
  \]

  Two rules could have concluded \( \mathcal{D} :: \Gamma \rightarrow (\Theta', \hat{\alpha} : \kappa = \mathbf{t}) \):

- Case \( \Gamma' \rightarrow \Theta' \) \( \Theta'[\mathbf{t}''] = \Theta'[\mathbf{t}] \)
  \[
  \Gamma', \hat{\alpha} : \kappa = \mathbf{t''} \rightarrow \Theta', \hat{\alpha} : \kappa = \mathbf{t} \\
  \rightarrow \text{Solved}
  \]

  \( \Gamma' \rightarrow \Delta' \)

  By i.h.

  \( \Theta'[\mathbf{t}'] = \Theta'[\mathbf{t}] \)
  Premise

  \( \Delta'[\Theta'[\mathbf{t}'] = \Delta'[\Theta'[\mathbf{t}] \)
  Applying \( \Delta' \) to both sides

  \( \Delta'[\mathbf{t}'] = \Delta'[\mathbf{t}] \)
  By Lemma 29 (Substitution Monotonicity)

  \( \Delta'[\mathbf{t}] = \Delta'[\mathbf{t}'] \)
  By premise \( \Delta'[\mathbf{t}] = \Delta'[\mathbf{t}'] \)

  \( \Gamma', \hat{\alpha} : \kappa = \mathbf{t''} \rightarrow \Delta', \hat{\alpha} : \kappa = \mathbf{t} \)
  By rule \( \rightarrow \text{Solved} \)

- Case \( \Gamma \rightarrow \Theta' \)
  \[
  \Gamma \rightarrow \Theta', \hat{\alpha} : \kappa = \mathbf{t} \\
  \rightarrow \text{AddSolved}
  \]

  \( \Gamma \rightarrow \Delta' \)

  By i.h.

  \( \Gamma \rightarrow \Delta', \hat{\alpha} : \kappa = \mathbf{t'} \)
  By rule \( \rightarrow \text{AddSolved} \)

- Case \( \Theta' \rightarrow \Delta' \) \( \Delta'[\mathbf{t}] = \Delta'[\mathbf{t}'] \)
  \[
  \Theta', \hat{\alpha} = \mathbf{t} \\
  \rightarrow \text{Eqn}
  \]

  \( \Theta', \hat{\alpha} = \mathbf{t'} \)
  By \( \Delta'[\mathbf{t}] = \Delta'[\mathbf{t}'] \)

\end{proof}
\[ \Gamma = (\Gamma', \alpha = t') \]  
By inversion on \( D \)

\[ \Gamma' \rightarrow \Theta' \]  
By inversion on \( D \)

\[ [\Theta'][t'''] = [\Theta'][t] \]  
By inversion on \( D \)

\[ [\Delta'][t'''] = [\Delta'][\Theta']t \]  
Applying \( \Delta' \) to both sides

\[ \Gamma' \rightarrow \Delta' \]  
By i.h.

\[ [\Delta'][t'''] = [\Delta'][t'] \]  
By Lemma 29 (Substitution Monotonicity)

\[ \Gamma', \alpha = t'' \rightarrow \Delta', \alpha = t' \]  
By rule \( \rightarrow \text{Eqn} \)

**Case**

\[ \Theta \rightarrow \Delta' \]

\[ \Theta \rightarrow \Delta', \alpha : \kappa \rightarrow \text{Add} \]

\[ \Gamma \rightarrow \Delta' \]  
By i.h.

\[ \Gamma \rightarrow \Delta', \alpha : \kappa \rightarrow \text{Add} \]

**Case**

\[ \Theta \rightarrow \Delta' \]

\[ \Theta \rightarrow \Delta', \alpha : \kappa = t \rightarrow \text{AddSolved} \]

\[ \Gamma \rightarrow \Delta' \]  
By i.h.

\[ \Gamma \rightarrow \Delta', \alpha : \kappa = t \rightarrow \text{AddSolved} \]

**Case**

\[ \Theta' \rightarrow \Delta' \]

\[ \Theta' \rightarrow \Delta', \alpha : \kappa = t \rightarrow \text{AddSolved} \]

\[ \Gamma = \Gamma', \alpha \rightarrow \text{Marker} \]

\[ \Theta \rightarrow \Delta' \]  
By inversion on \( D \)

\[ \Gamma' \rightarrow \Theta' \]  
By inversion on \( D \)

\[ \Gamma' \rightarrow \Delta' \]  
By i.h.

\[ \Gamma', \alpha \rightarrow \Delta', \alpha \rightarrow \text{Uvar} \]

\[ \Box \]

### 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 $\text{EqProp}$, and Lemma 36 (Extension Weakening (Sorts)) twice.

Lemma 38 (Extension Weakening (Types)). If $\Gamma \vdash A$ a type and $\Gamma \rightarrow \Delta$ then $\Delta \vdash A$ a type.

Proof. By a straightforward induction on $\Gamma \vdash A$ a type.

In the $\text{VarWF}$ case, use Lemma 22 (Extension Inversion) (i) or (v). In the $\text{SolvedVarWF}$ case, use Lemma 22 (Extension Inversion) (iv).

In the other cases, apply the i.h. and/or (for $\text{ImpliesWF}$ and $\text{WithWF}$) Lemma 37 (Extension Weakening (Props)) to all subderivations, then apply the rule.

C’3 Principal Typing Properties

Lemma 39 (Principal Agreement).

(i) If $\Gamma \vdash A$ a type and $\Gamma \rightarrow \Delta$ then $\Delta \vdash A = [\Gamma] A$.

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

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

Part (i):

- Case $\Gamma_0 \rightarrow \Delta_0$ \quad $[\Delta_0] t = [\Delta_0] t'$
  $\Gamma_0, \alpha = t \rightarrow \Delta_0, \alpha = t'$ \quad $\text{Eqn}$

If $\alpha \notin \text{FV}(A)$, then:

$[\Gamma_0, \alpha = t] A = [\Gamma_0] A$ \quad By def. of subst.

$= [\Delta_0] A$ \quad By i.h.

$= [\Delta_0, \alpha = t'] A$ \quad By def. of subst.

Otherwise, $\alpha \in \text{FV}(A)$.

$\Gamma_0 \vdash t$ a type \quad $\Gamma$ is well-formed

$\Gamma_0 \vdash [\Gamma_0] t$ a type \quad By Lemma 13 (Right-Hand Substitution for Typing)

Suppose, for a contradiction, that $\text{FEV}([\Gamma_0] t) \neq \emptyset$.

Since $\alpha \in \text{FV}(A)$, we also have $\text{FEV}([\Gamma] A) \neq \emptyset$, a contradiction.
Proof of Lemma 39 (Principal Agreement)

\[ \text{lem:substitution-tpp-stable} \]

**FEV**\([\Gamma_0 | t] \neq \emptyset \)
- Assumption (for contradiction)

\[ \Gamma_0 | t = \Gamma | \alpha \]
- By def. of subst.

**FEV**\([\Gamma | \alpha] \neq \emptyset \)
- By above equality

\[ \alpha \in \text{FV}(A) \]
- Above

**FEV**\([\Gamma | A] \neq \emptyset \)
- By a property of subst.

\[ \Gamma \vdash A \text{ type} \]
- Given

**FEV**\([\Gamma | A] = \emptyset \)
- By inversion

\[ \Rightarrow \]

**FEV**\([\Gamma_0 | t] = \emptyset \)
- By contradiction

\[ \Gamma_0 \vdash t \text{ type} \]
- By **PrincipalWF**

\[ [\Gamma_0 | t] = [\Delta_0 | t] \]
- By i.h.

**FEV**\([\Gamma_0 | t] = \emptyset \)
- By above equality

\[ \Gamma_0 \vdash [\Delta_0 | t] \text{ type} \]
- By above equality

\[ [\Gamma_0 | t] / \alpha A \]
- By Lemma 8 (Substitution—Well-formedness) (i)

\[ [\Gamma_0 | t] \]
- By above equality

\[ \Gamma_0 \vdash [\Delta_0 | t] / \alpha A \]
- By above equality

\[ = [\Delta_0 | [\Delta_0 | t] / \alpha A] \]
- By above equality

\[ = [\Delta_0 | [\Delta_0 | t / \alpha] A] \]
- By above equality

\[ = [\Delta_0 | [\Delta_0 | t / \alpha] A] \]
- By def. of subst.

\[ \Rightarrow \]

**FEV**\([\Gamma_0 | t] = \emptyset \)
- By contradiction

\[ \Gamma_0 \vdash A \text{ type} \]
- By inversion

**FEV**\([\Gamma_0 | A] = \emptyset \)
- By above equality

\[ \Rightarrow \]

\[ \Gamma_0 \vdash [\Delta_0 | t] / \alpha A \]
- By above equality

\[ = [\Delta_0 | [\Delta_0 | t'] / \alpha A] \]
- By above equality

\[ = [\Delta_0 | [\Delta_0 | t'] / \alpha A] \]
- By def. of subst.

\[ \Rightarrow \]

**FEV**\([\Gamma_0 | t] = \emptyset \)
- By contradiction

\[ \Gamma_0 \vdash t \text{ type} \]
- By **PrincipalWF**

**FEV**\([\Gamma_0 | t] = \emptyset \)
- By above equality

\[ \Gamma_0 \vdash [\Delta_0 | t] / \alpha A \]
- By above equality

**FEV**\([\Delta_0 | t] / \alpha A \]
- By def. of subst.

- **Case** \(--\rightarrow\text{Solved}\), \(--\rightarrow\text{Solve}\), \(--\rightarrow\text{Add}\), \(--\rightarrow\text{Solved}\) Similar to the \(--\rightarrow\text{Eqn}\) case.

- **Case** \(--\rightarrow\text{Id}\), \(--\rightarrow\text{Var}\), \(--\rightarrow\text{Uvar}\), \(--\rightarrow\text{Unsolved}\), \(--\rightarrow\text{Marker}\)

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 \( \Gamma \vdash \Gamma | A \text{ type} \).

**Proof.** By cases of \( p \):

- **Case** \( p = ! \):

  \[ \Gamma \vdash A \text{ type} \]
  - By inversion

  **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)

  **FEV**\([\Gamma | \Gamma] A = \emptyset \)
  - By inversion

  \[ \Gamma \vdash [\Gamma | \Gamma] A \text{ type} \]
  - By rule **PrincipalWF**

- **Case** \( p = / \!

  \[ \Gamma \vdash A \text{ type} \]
  - By inversion

  \[ \Gamma \vdash [\Gamma | A] \text{ type} \]
  - By Lemma 13 (Right-Hand Substitution for Typing)

  \[ \Gamma \vdash A \text{ / type} \]
  - By rule **NonPrincipalWF**

**Lemma 41** (Extension Weakening for Principal Typing). If \( \Gamma \vdash A \text{ type} \) and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash A \text{ type} \).

**Proof of Lemma 41** (Extension Weakening for Principal Typing)
Proof. By cases of \( p \):

- Case \( p = \j \):
  \[
  \begin{array}{l}
  \Gamma \vdash A \text{ type} & \text{By inversion} \\
  \Delta \vdash A \text{ type} & \text{By Lemma 38 (Extension Weakening (Types))} \\
  \Delta \vdash A \ j \text{ type} & \text{By rule NonPrincipalWF} \\
  \end{array}
  \]

- Case \( p = ! \):
  \[
  \begin{array}{l}
  \Gamma \vdash A \text{ type} & \text{By inversion} \\
  \text{FEV}(\Gamma|A) = \emptyset & \text{By inversion} \\
  \Delta \vdash A \text{ type} & \text{By Lemma 38 (Extension Weakening (Types))} \\
  \Delta \vdash \Delta|A \text{ type} & \text{By Lemma 13 (Right-Hand Substitution for Typing)} \\
  \Delta|A = [\Gamma]A & \text{By Lemma 30 (Substitution Invariance)} \\
  \text{FEV}(\Delta|A) = \emptyset & \text{By congruence of equality} \\
  \Delta \vdash \Delta|A \ j \text{ type} & \text{By rule PrincipalWF} \\
  \end{array}
  \]

Lemma 42 (Inversion of Principal Typing).

1. If \( \Gamma \vdash (A \rightarrow B) \ p \text{ type} \) then \( \Gamma \vdash A \ p \text{ type} \) and \( \Gamma \vdash B \ p \text{ type} \).
2. If \( \Gamma \vdash (P \supset A) \ p \text{ type} \) then \( \Gamma \vdash P \ p \text{ prop} \) and \( \Gamma \vdash A \ p \text{ type} \).
3. If \( \Gamma \vdash (A \land P) \ p \text{ type} \) then \( \Gamma \vdash P \ p \text{ prop} \) and \( \Gamma \vdash A \ p \text{ type} \).

Proof. Proof of part 1:
We have \( \Gamma \vdash A \rightarrow B \ p \text{ type} \).

- Case \( p = \j \):
  \[
  \begin{array}{l}
  1 \ 
  \Gamma \vdash A \rightarrow B \text{ type} & \text{By inversion} \\
  \Gamma \vdash A \text{ type} & \text{By inversion on 1} \\
  \Gamma \vdash B \text{ type} & \text{By inversion on 1} \\
  \Gamma \vdash A \ j \text{ type} & \text{By rule NonPrincipalWF} \\
  \Gamma \vdash B \ j \text{ type} & \text{By rule NonPrincipalWF} \\
  \end{array}
  \]

- Case \( p = ! \):
  \[
  \begin{array}{l}
  1 \ 
  \ 
  \text{FEV}(\Gamma|A) = \emptyset & \text{By inversion on } \Gamma \vdash A \rightarrow B \ j \text{ type} \\
  = \text{FEV}(\Gamma|A) \cup \text{FEV}(\Gamma|B) & \text{By definition of substitution} \\
  = \text{FEV}(\Gamma|A) \cup \text{FEV}(\Gamma|B) & \text{By definition of FEV}(-) \\
  \end{array}
  \]
  \[
  \begin{array}{l}
  \text{FEV}(\Gamma|A) = \text{FEV}(\Gamma|B) = \emptyset & \text{By properties of empty sets and unions} \\
  \Gamma \vdash A \text{ type} & \text{By inversion on 1} \\
  \Gamma \vdash B \text{ type} & \text{By inversion on 1} \\
  \Gamma \vdash A \ j \text{ type} & \text{By rule PrincipalWF} \\
  \Gamma \vdash B \ j \text{ type} & \text{By rule PrincipalWF} \\
  \end{array}
  \]

Part 2: We have \( \Gamma \vdash P \supset A \ p \text{ type} \). Similar to Part 1.
Part 3: We have \( \Gamma \vdash A \land P \ p \text{ type} \). Similar to Part 2.
C’.4 Instantiation Extends

Lemma 43 (Instantiation Extension).
If $\Gamma \vdash \alpha := \tau : \kappa \rightarrow \Delta$ then $\Gamma \rightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case** $\Gamma \vdash \tau : \kappa$
  
  $\Gamma \vdash \varepsilon : \kappa \rightarrow \Gamma \vdash \varepsilon : \kappa \rightarrow \varepsilon \vdash \tau : \kappa \rightarrow \Delta$  
  
  By Lemma 23 (Deep Evar Introduction) (ii).

- **Case** $\beta \in \text{unsolved}((\Gamma_0 \vdash \alpha : \kappa) \cap (\beta : \kappa))$
  
  $\Gamma_0 \vdash \alpha : \kappa \rightarrow \beta : \kappa \rightarrow \Gamma_0 \vdash \alpha : \kappa \rightarrow \beta : \kappa = \alpha$
  
  By Lemma 23 (Deep Evar Introduction) (ii).

- **Case** $\Gamma_0 \vdash \alpha_2 : * \vdash \alpha_1 : * \vdash \alpha_1 : * \rightarrow \alpha_1 := \tau_1 : * \rightarrow \Theta \vdash \alpha_2 := [\Theta] \tau_2 : * \rightarrow \Delta$
  
  Subderivation

  $\Theta \vdash \alpha_2 := [\Theta] \tau_2 : * \rightarrow \Delta$
  
  By i.h.

  $\Theta \rightarrow \Delta$
  
  By i.h.

  $\Gamma_0 \vdash \alpha_2 : * \vdash \alpha_1 : * \vdash \alpha_1 := \tau_1 : * \rightarrow \Theta \vdash \alpha_2 := [\Theta] \tau_2 : * \rightarrow \Delta$
  
  By Lemma 33 (Extension Transitivity)

- **Case** $\Gamma_0 \vdash \alpha : * \rightarrow \Delta$
  
  By Lemma 33 (Extension Transitivity)

- **Case** $\Gamma_0 \vdash \alpha : N \vdash \alpha := \text{zero} : N \rightarrow \Delta$
  
  InstZero

  Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case** $\Gamma [\alpha_1 : N, \alpha : N = \text{succ} (\alpha_1)] \vdash \alpha_1 := t_1 : N \rightarrow \Delta$
  
  InstSucc

  By reasoning similar to the InstBin case.

C’.5 Equivalence Extends

Lemma 44 (Elimeq Extension).
If $\Gamma / s \doteq t : \kappa \rightarrow \Delta$ 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**
  \[ \Gamma / \alpha \equiv \alpha : \kappa \vdash \Gamma \]
  \( \text{ElimeqUvarRefl} \)
  Since \( \Delta = \Gamma \), applying Lemma 32 (Extension Reflexivity) suffices (let \( \Theta = \cdot \)).

- **Case**
  \[ \Gamma / \text{zero} \equiv \text{zero} : \mathbb{N} \vdash \Gamma \]
  \( \text{ElimeqZero} \)
  Similar to the \( \text{ElimeqUvarRefl} \) case.

- **Case**
  \[ \Gamma / \sigma \equiv t : \mathbb{N} \vdash \Delta \]
  \( \text{ElimeqSucc} \)
  Follows by i.h.

- **Case**
  \[ \Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := t : \kappa \vdash \Gamma \]
  \( \text{ElimeqInstL} \)
  \[ \Gamma \vdash \hat{\alpha} := t : \kappa \vdash \Delta \] Subderivation
  \[ \Gamma \rightarrow \Delta \] By Lemma 43 (Instantiation Extension)
  Let \( \Theta = \cdot \).
  \( \Rightarrow \Gamma, \Theta \rightarrow \Delta \) By \( \Theta = \cdot \).

- **Case**
  \[ \alpha \notin \text{FV}(\Gamma[t]) \quad (\alpha = -) \notin \Gamma \]
  \[ \Gamma / \alpha \equiv t : \kappa \vdash \Gamma, \alpha = t \]
  \( \text{ElimeqUvarL} \)
  Let \( \Theta \) be \( (\alpha = t) \).
  \( \Rightarrow \Gamma, \alpha = t \rightarrow \Gamma, \alpha = t \) By Lemma 32 (Extension Reflexivity)

- **Cases** \( \text{ElimeqInstR} \), \( \text{ElimeqUvarR} \)
  Similar to the respective L cases.

- **Case**
  \[ \sigma \neq t \]
  \[ \Gamma / \sigma \equiv t : \kappa \vdash \bot \]
  \( \text{ElimeqClash} \)
  The statement says that the output is a (consistent) context \( \Delta \), so this case is impossible.

Lemma 45 (Elimprop Extension).
*If \( \Gamma / P \vdash \Delta \) 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 \( \Delta \).
**Proof of Lemma 45** (Elimprop Extension)

\[
\frac{\Gamma / \sigma \vdash t : N \vdash \Delta}{\Gamma / \sigma \vdash t : \Delta} \text{ ElimpropEq}
\]

\[
\Gamma / \sigma \vdash t : N \vdash \Delta \quad \text{Subderivation}
\]

\[
\Rightarrow \Gamma, \Theta \rightarrow \Delta \quad \text{By Lemma 44 (Elimeq Extension)}
\]

\[
\square
\]

**Lemma 46** (Checkeq Extension).
If \(\Gamma \vdash A \equiv B \vdash \Delta\) then \(\Gamma \rightarrow \Delta\).

**Proof.** By induction on the given derivation.

- **Case**
  \[
  \Gamma \vdash u \equiv u : \kappa \rightarrow \Gamma \text{ CheckeqVar}
  \]
  Since \(\Delta = \Gamma\), applying Lemma 32 (Extension Reflexivity) suffices.

- **Cases** CheckeqUnit, CheckeqZero
  Similar to the CheckeqVar case.

- **Case**
  \[
  \Gamma \vdash \tau_1 \equiv \tau_1' \rightarrow \Theta \quad \Theta \vdash [\Theta] \tau_2 \equiv [\Theta] \tau_2' : \star \vdash \Delta \quad \text{CheckeqBin}
  \]
  \[
  \Gamma \rightarrow \Theta \quad \text{By i.h.}
  \]
  \[
  \Theta \rightarrow \Delta \quad \text{By i.h.}
  \]
  \[
  \Rightarrow \Gamma \rightarrow \Delta \quad \text{By Lemma 33 (Extension Transitivity)}
  \]

- **Case**
  \[
  \Gamma \vdash \sigma \equiv t : N \vdash \Delta \quad \text{CheckeqSucc}
  \]
  \[
  \Rightarrow \Gamma \rightarrow \Delta \quad \text{Subderivation}
  \]
  \[
  \text{By i.h.}
  \]

- **Case**
  \[
  \Gamma_0[\tilde{\alpha}] \vdash \tilde{\alpha} := t : \kappa \vdash \Delta \quad \tilde{\alpha} \notin FV(\Gamma_0[\tilde{\alpha}]t) \quad \text{CheckeqInstL}
  \]
  \[
  \Rightarrow \Gamma_0[\tilde{\alpha}] \vdash \tilde{\alpha} := t : \kappa \vdash \Delta \quad \text{Subderivation}
  \]
  \[
  \Rightarrow \Gamma_0[\tilde{\alpha}] \rightarrow \Delta \quad \text{By Lemma 43 (Instantiation Extension)}
  \]

- **Case** CheckeqInstR
  Similar to the CheckeqInstL case.

\[
\square
\]

**Lemma 47** (Checkprop Extension).
If \(\Gamma \vdash P \text{ true } \vdash \Delta\) then \(\Gamma \rightarrow \Delta\).

**Proof.** By induction on the given derivation.
### 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**
  \[
  \frac{\Gamma \vdash \sigma_1 \equiv \tau_1 : N \vdash \Theta \quad \Theta \vdash \sigma_2 \equiv \tau_2 : N \vdash \Delta}{\Gamma \vdash (\sigma_1 = \sigma_2) \equiv (\tau_1 = \tau_2) \vdash \Delta} \quad \equiv \text{PropEq}
  \]
  
  Subderivation
  \[
  \Gamma \rightarrow \Theta \quad \text{By Lemma 46 (Checkeq Extension)}
  \]
  
  \[
  \Theta \vdash \sigma_2 \equiv \tau_2 : N \vdash \Delta \quad \text{Subderivation}
  \]
  
  \[
  \Theta \rightarrow \Delta \quad \text{By Lemma 46 (Checkeq Extension)}
  \]
  
  \[
  \Gamma \rightarrow \Delta \quad \text{By Lemma 46 (Checkeq Extension)}
  \]

- **Case**

**Lemma 49 (Equivalence Extension)**

If \( \Gamma \vdash A \equiv B \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).

**Proof.** By induction on the given derivation.

- **Case**
  \[
  \frac{\Gamma \vdash \alpha \equiv \alpha : \Delta}{\Gamma \vdash \alpha \equiv \alpha \vdash \Delta} \quad \equiv \text{Var}
  \]

  Here \( \Delta = \Gamma \), so Lemma 32 (Extension Reflexivity) suffices.

- **Case**
  \[
  \frac{\Gamma \vdash \& \equiv \& : \Delta}{\Gamma \vdash \& \equiv \& \vdash \Delta} \quad \equiv \text{Exvar}
  \]

  Similar to the \( \equiv \text{Var} \) case.

- **Case**
  \[
  \frac{\Gamma \vdash 1 \equiv 1 : \Delta}{\Gamma \vdash 1 \equiv 1 \vdash \Delta} \quad \equiv \text{Unit}
  \]

  Similar to the \( \equiv \text{Var} \) case.

- **Case**
  \[
  \frac{\Gamma \vdash A_1 \equiv B_1 \vdash \Theta \quad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta}{\Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \vdash \Delta} \quad \equiv \oplus
  \]

  Subderivation
  \[
  \Gamma \rightarrow \Theta \quad \text{By i.h.}
  \]

  \[
  \Theta \vdash (\Theta)A_2 \equiv (\Theta)B_2 \vdash \Delta \quad \text{Subderivation}
  \]

  \[
  \Theta \rightarrow \Delta \quad \text{By i.h.}
  \]

  \[
  \Gamma \rightarrow \Delta \quad \text{By Lemma 33 (Extension Transitivity)}
  \]

**Proof of Lemma 49 (Equivalence Extension)**

lem: equiv-extension
Case $\equiv \text{Vec}$: Similar to the $\equiv$ case.

Cases $\equiv \equiv \land$: Similar to the $\equiv$ case, but with Lemma 48 (Prop Equivalence Extension) on the first premise.

Case $\Gamma, \alpha : \kappa \vdash A_0 \equiv B \vdash \Delta, \alpha : \kappa, \Delta'$

\[
\begin{align*}
\Gamma \vdash \forall \alpha : \kappa. A_0 &\equiv \forall \alpha : \kappa. B \vdash \Delta \\
\Gamma, \alpha : \kappa \vdash A_0 &\equiv B \vdash \Delta, \alpha : \kappa, \Delta'
\end{align*}
\]

Subderivation

By i.h.

By Lemma 22 (Extension Inversion) (i)

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.}

C.6 Subtyping Extends

Lemma 50 (Subtyping Extension). If $\Gamma \vdash A <: B \vdash \Delta$ then $\Gamma \rightarrow \Delta$.

Proof. By induction on the given derivation.

Case $\Gamma, \alpha : \kappa \vdash \lceil \alpha \rceil A_0 <: B \vdash \Delta, \alpha : \kappa, \Theta$

\[
\begin{align*}
\Gamma \vdash \forall \alpha : \kappa. A_0 &<: B \vdash \Delta \\
\Gamma, \alpha : \kappa \vdash [\alpha / \alpha] A_0 &<: B \vdash \Delta, \alpha : \kappa, \Theta
\end{align*}
\]

Subderivation

By i.h. (i)

By Lemma 22 (Extension Inversion) (ii)

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.

Case $\equiv \equiv \land$: Similar to the $\equiv$ case, but using part (i) of Lemma 22 (Extension Inversion).

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.

Case $\equiv \equiv \land$: Similar to the $\equiv$ case.
C'.7 Typing Extends

Lemma 51 (Typing Extension).
If $\Gamma \vdash e \equiv A \ P \vdash \Delta$
or $\Gamma \vdash e \Rightarrow A \ p \vdash \Delta$
or $\Gamma \vdash s : A \ p \Rightarrow B \ q \vdash \Delta$
or $\Gamma \vdash \Pi ::  A \ q \equiv C \ p \vdash \Delta$
then $\Gamma \Rightarrow \Delta$.

Proof. By induction on the given derivation.

- Match judgments:
  In rule MatchEmpty $\Delta = \Gamma$, so the result follows by Lemma 32 (Extension Reflexivity).
  Rules MatchBase, Match×, Match+ and 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 MatchSeq, Lemma 33 (Extension Transitivity) is also needed.
  In rule Match⇒ apply the i.h., then use Lemma 22 (Extension Inversion) (i).
  Match∧ Use the i.h.
  MatchNeg Use the i.h. and Lemma 22 (Extension Inversion) (v).
  Match⊥ Immediate by Lemma 32 (Extension Reflexivity).
  MatchUnify

- Synthesis, checking, and spine judgments: In rules Var, 1, EmptySpine, and ⊃ I ⊥ the output context $\Delta$ is exactly $\Gamma$, so the result follows by Lemma 32 (Extension Reflexivity).
  – Case ∀I: Use the i.h. and Lemma 33 (Extension Transitivity).
  – Case ∀Spine: By →Add, $\Gamma \Rightarrow \Gamma, \alpha : \kappa$.
    The result follows by i.h. and Lemma 33 (Extension Transitivity).
  – Cases ∧ Spine Use Lemma 47 (Checkprop Extension), the i.h., and Lemma 33 (Extension Transitivity).
  – Cases Nil Cons Using reasoning found in the ∧I and ⊃I cases.
  – Case ⊃I
    $\Gamma, \triangleright, \Theta' \leadsto \Theta$ By Lemma 45 (Elimprop Extension)
    $\Theta \leadsto \Delta, \triangleright, \Delta'$ By i.h.
    $\Gamma, \triangleright, \Theta' \leadsto \Delta, \triangleright, \Delta'$ By Lemma 33 (Extension Transitivity)
    $\Rightarrow \Gamma \Rightarrow \Delta$ By Lemma 22 (Extension Inversion)

- Cases ¬I Rec Use the i.h. and Lemma 22 (Extension Inversion).
- Cases Sub Anno →E Spine +Ik ×I Use the i.h., and Lemma 33 (Extension Transitivity) as needed.
- Case 1I By Lemma 23 (Deep Evar Introduction) (ii).
C'.8 Unfiled

Lemma 52 (Context Partitioning).
\[ \Delta, \Theta \vdash \Omega, \Theta \rightarrow \Omega, \Theta, \Omega \]
then there is a \( \Psi \) such that \( \Delta, \Theta, \Omega \vdash \Omega, \Delta, \Theta = \Omega, \Delta, \Theta \).

Proof. By induction on the given derivation.

- Case \( \rightarrow \text{Id} \): Impossible: \( \Delta, \Theta \) cannot have the form \( \cdot \).

- Case \( \rightarrow \text{Var} \): We have \( \Omega = (\Omega', x : A) \) and \( \Theta = (\Theta', x : A') \). By i.h., there is \( \Psi' \) such that \( \Omega, \Theta \vdash \Omega, \Theta' \). Then by the definition of context application, \( \Omega, \Theta, \Omega' \vdash \Omega, \Theta', x : A \). Let \( \Psi = (\Psi', x : \Omega') \).

- Case \( \rightarrow \text{Uvar} \): Similar to the \( \rightarrow \text{Var} \) case, with \( \Psi = (\Psi', \alpha : \kappa) \).

- Cases \( \rightarrow \text{Eqn} \rightarrow \text{Unsolved} \rightarrow \text{Solved} \rightarrow \text{Add} \rightarrow \text{AddSolved} \rightarrow \text{Marker} \).

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).
\[ \Gamma \rightarrow \Omega \]
then \( \Omega \vdash \Omega \).

Proof. By induction on the derivation of \( \Gamma \rightarrow \Omega \).

- Case \( \rightarrow \text{Id} \)

Immediate.

- Case \( \Gamma \rightarrow \Omega \)

By i.h.

- Case \( \Gamma \rightarrow \Omega \)

Subderivation

By congruence of equality

By definition of substitution

Similar to \( \rightarrow \text{Var} \)
Proof of Lemma 54 (Completing Stability).

\[ \text{Case } \Gamma_0 \rightarrow \Omega_0 \]
\[ \Gamma_0, \alpha : \kappa \rightarrow \Omega_0, \alpha : \kappa \rightarrow \text{Unsolved} \]

Similar to \( \rightarrow \text{Var} \)

\[ \text{Case } \Gamma_0 \rightarrow \Omega_0 \]
\[ [\Omega_0]t = [\Omega_0]t' \]
\[ \Gamma_0, \alpha : \kappa = t \rightarrow \Omega_0, \alpha : \kappa = t' \rightarrow \text{Solved} \]

Similar to \( \rightarrow \text{Var} \)

\[ \text{Case } \Gamma_0 \rightarrow \Omega_0 \]
\[ \Gamma_0, \alpha : \kappa \rightarrow \Omega_0, \alpha : \kappa \rightarrow \text{Marker} \]

Similar to \( \rightarrow \text{Var} \)

\[ \text{Case } \Gamma_0 \rightarrow \Omega_0 \]
\[ \Gamma_0, \beta : \kappa' \rightarrow \Omega_0, \beta : \kappa' = t \rightarrow \text{Solve} \]

Similar to \( \rightarrow \text{Var} \)

\[ \text{Case } \Gamma_0 \rightarrow \Omega_0 \]
\[ [\Omega_0]t' = [\Omega_0]t \]
\[ \Gamma_0, \alpha = t' \rightarrow \Omega_0, \alpha = t \rightarrow \text{Eqn} \]

\[ \Gamma_0 \rightarrow \Omega_0 \quad \text{Subderivation} \]
\[ [\Omega_0]t' = [\Omega_0]t \quad \text{Subderivation} \]
\[ [\Omega_0]t = [\Omega_0]t' \quad \text{By i.h.} \]
\[ [\Omega_0][\alpha / \alpha'] = [\Omega_0][\alpha / \alpha'] \quad \text{By congruence of equality} \]
\[ [\Omega_0, \alpha = t](\Gamma_0, \alpha = t') = [\Omega_0, \alpha = t](\Omega_0, \alpha = t) \quad \text{By definition of context substitution} \]

\[ \text{Case } \Gamma \rightarrow \Omega_0 \]
\[ \Gamma \rightarrow \Omega_0, \alpha : \kappa \rightarrow \text{Add} \]
\[ \Gamma \rightarrow \Omega_0 \quad \text{Subderivation} \]
\[ [\Omega_0][\alpha] = [\Omega_0][\alpha] \quad \text{By i.h.} \]
\[ [\Omega_0, \alpha : \kappa][\alpha] = [\Omega_0, \alpha : \kappa](\Omega_0, \alpha : \kappa) \quad \text{By definition of context substitution} \]

\[ \text{Case } \Gamma \rightarrow \Omega_0 \]
\[ \Gamma \rightarrow \Omega_0, \alpha : \kappa \rightarrow \text{AddSolved} \]

Similar to the \( \rightarrow \text{Add} \) case.

\[ \square \]

Lemma 55 (Completing Completeness).

(i) 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](lem:substitution-monotonicity)) (i), $[\Omega' \mid t = [\Omega'] \mid t]$. Now we need to show $[\Omega'] \mid t = [\Omega] \mid t$. Considered as a substitution, $\Omega'$ is the identity everywhere except existential variables $\alpha$ and universal variables $\kappa$. First, since $\Omega$ is complete, $[\Omega] \mid t$ has no free existentials. Second, universal variables free in $[\Omega] \mid t$ 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 15 that $\Omega'$ also lacks an equation, so applying $\Omega'$ also leaves $\alpha$ alone.

  Transitivity of equality gives $[\Omega'] \mid t = [\Omega] \mid t$.

• **Part (ii):** Similar to part (i), using Lemma 29 ([Substitution Monotonicity](lem:substitution-monotonicity)) (iii) instead of (i).

• **Part (iii):** By induction on the given derivation of $\Omega \rightarrow \Omega'$.

  Only cases $\rightarrow \text{Id}$, $\rightarrow \text{Var}$, $\rightarrow \text{Unvar}$, $\rightarrow \text{Eqn}$, $\rightarrow \text{Solved}$, $\rightarrow \text{AddSolved}$, 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.**

\[
\begin{align*}
\Delta_1 \rightarrow \Omega & \quad \text{Given} \\
[\Omega] \Delta_1 & = [\Omega] \Omega & \text{By Lemma 54 (Completing Stability)} \\
\Delta_2 \rightarrow \Omega & \quad \text{Given} \\
[\Omega] \Delta_2 & = [\Omega] \Omega & \text{By Lemma 54 (Completing Stability)} \\
[\Omega] \Delta_1 & = [\Omega] \Delta_2 & \text{By transitivity of equality}
\end{align*}
\]

**Lemma 57 (Multiple Confluence).**
If $\Delta \rightarrow \Omega$ and $\Delta \rightarrow \Omega'$ and $\Delta \rightarrow \Omega'$ then $[\Omega] \Delta = [\Omega'] \Delta'$.

**Proof.**

\[
\begin{align*}
\Delta \rightarrow \Omega & \quad \text{Given} \\
[\Omega] \Delta & = [\Omega] \Omega & \text{By Lemma 54 (Completing Stability)} \\
\Omega & \rightarrow \Omega' & \text{Given} \\
[\Omega] \Omega & = [\Omega'] \Omega' & \text{By Lemma 55 (Completing Completeness) (iii)} \\
& = [\Omega'] \Delta' & \text{By Lemma 54 (Completing Stability) (}\Delta' \rightarrow \Omega' \text{ given)}
\end{align*}
\]

**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 $\hat{\alpha} : \kappa = \tau$ and $\alpha = \tau$ 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 $\dot{\alpha} \in \text{unsolved}(\Delta)$
then there exists $\Omega_1 = \Omega_{\text{canon}}^\dot{\alpha} : \kappa = t_1$ such that $\Omega_1 \rightarrow \Omega$ and $\Omega_2 = \Omega_{\text{canon}}^\dot{\alpha} : \kappa = t_2$ 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 $\Omega_{\text{canon}} = \Omega_{\text{canon}}^\dot{\alpha} : \kappa = t_1$, where $\text{FEV}(t_1) = \emptyset$. Therefore $\Omega_{\text{canon}}^\dot{\alpha} : \kappa = t_1 \rightarrow \Omega$.

Now choose $t_2$ as follows:
• If \( \kappa = \ast \), let \( t_2 = t_1 \rightarrow t_1 \).

• If \( \kappa = \mathbb{N} \), let \( t_2 = \text{succ}(t_1) \).

Thus, \( \ast \neq t_1 \). Let \( \Omega_2 = \Omega_1'[\alpha : \kappa = t_2] \).

\[ \Delta \rightarrow \Omega_2 \quad \text{By Lemma 31 (Split Extension)} \]

\section*{D' Internal Properties of the Declarative System}

\textbf{Lemma 61} (Interpolating With and Exists).

\begin{enumerate}
\item If \( D :: \Psi \vdash \Pi :: \vec{A}! \Leftarrow C \) and \( \Psi \vdash P_0 \text{ true} \),
  then \( D' :: \Psi \vdash \Pi :: \vec{A}! \Leftarrow C \land P_0 \).
\item If \( D :: \Psi \vdash \Pi :: \vec{A}! \Leftarrow \tau/\alpha C_0 \) and \( \Psi \vdash \tau \vdash \kappa \),
  then \( D' :: \Psi \vdash \Pi :: \vec{A}! \Leftarrow (\exists \alpha : \kappa. C_0) \).
\end{enumerate}

In both cases, the height of \( D' \) is one greater than the height of \( D \).
Moreover, similar properties hold for the eliminating judgment \( \Psi / P \vdash \Pi :: \vec{A}! \Leftarrow C \).

\textbf{Proof.} By induction on the given match derivation.

In the \texttt{DeclMatchBase} case, for part (1), apply rule \[ \land I \].
For part (2), apply rule \[ \exists I \].

In the \texttt{DeclMatchNeg} case, part (1), use Lemma 2 (Declarative Weakening) (iii).
In part (2), use Lemma 2 (Declarative Weakening) (i). \[ \square \]

\textbf{Lemma 62} (Case Invertibility).

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

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

\begin{itemize}
\item \textbf{Case} \( \Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A \quad \Psi \vdash A \Leftarrow B \)
  \[ \Psi \vdash \text{case}(e_0, \Pi) \Leftarrow B \quad \text{DeclSub} \]

Impossible, because \( \Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A \) is not derivable.

\item \textbf{Cases} \[ \texttt{Decl}\land, \texttt{Decl}\exists \]
  Impossible: these rules have a value restriction, but a case expression is not a value.

\item \textbf{Case} \( \Psi \vdash P \text{ true} \quad \Psi \vdash \text{case}(e_0, \Pi) \Leftarrow C_0 \)
  \[ \Psi \vdash \text{case}(e_0, \Pi) \Leftarrow C_0 \land P \quad \text{Decl}\land \]

\begin{itemize}
\item \( \ast \neq n - 1 \quad \Psi \vdash e_0 \Rightarrow A! \) \quad By i.h.
  \item \( \ast \neq n - 1 \quad \Psi \vdash \Pi :: A \Leftarrow C_0 \) \quad "\#"
  \item \( \ast \neq n - 1 \quad \Psi \vdash \Pi \text{ covers } A \) \quad "\#"
  \item \( \leq n - 1 \quad \Psi \vdash P \text{ true} \) \quad Subderivation
  \item \( \ast \neq n \quad \Psi \vdash \Pi :: A \Leftarrow C_0 \land P \) \quad By Lemma 61 (Interpolating With and Exists) (1)
\end{itemize}

\[ \square \]
Proof of Lemma 62 (Case Invertibility)

• Case $\Psi \vdash \tau : \kappa \quad \Psi \vdash \text{case}(e_0, \Pi) \iff [\tau/\alpha]C_0$

\[
\Psi \vdash \text{case}(e_0, \Pi) \iff \exists \alpha : \kappa. C_0 \quad \text{Decl} \exists
\]

\[\text{Infer: } \Psi \vdash e_0 \Rightarrow A ! \quad \text{By i.h.} \]
\[\Psi \vdash \Pi :: A \leftarrow C_0 \quad \text{"} \]
\[\Psi \vdash \Pi \text{ covers } A \quad \text{"} \]
\[\Psi \vdash \tau : \kappa \quad \text{Subderivation} \]
\[\Psi \vdash \Pi :: A \iff \exists \alpha : \kappa. C_0 \quad \text{By Lemma 61 [Interpolating With and Exists] (2)} \]

The heights of the derivations are similar to those in the \text{Decl} \land I case.

• Cases $\text{Decl} \land I, \text{Decl} \rightarrow I, \text{DeclRec}, \text{Decl} + I_k, \text{Decl} \times I, \text{DeclNil}, \text{DeclCons}$

Impossible, because in these rules $e$ cannot have the form $\text{case}(e_0, \Pi)$.

• Case $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A ! \quad \Psi \vdash \Pi :: A \leftarrow C \quad \Psi \vdash \Pi \text{ covers } A !$

\[
\Psi \vdash \text{case}(e_0, \Pi) \iff C \quad \text{DeclCase}
\]

Immediate.

\[\square\]

E' Miscellaneous Properties of the Algorithmic System

Lemma 63 (Well-Formed Outputs of Typing).

(Spines) If $\Gamma \vdash s : A q \gg C p \vdash \Delta$ or $\Gamma \vdash s : A q \gg C [p] \vdash \Delta$

and $\Gamma \vdash A q \text{ type}$

then $\Delta \vdash C p \text{ type}.$

(Synthesis) If $\Gamma \vdash e \Rightarrow A p \vdash \Delta$

then $A \vdash p \text{ type}.$

Proof. By induction on the given derivation.

• Case $\text{Anno}$ Use Lemma 51 (Typing Extension) and Lemma 41 (Extension Weakening for Principal Typing).

• Case $\text{Spine}$ We have $\Gamma \vdash (\forall \alpha : \kappa. A_0) \quad q \text{ type}.$

By inversion, $\Gamma, \alpha : \kappa \vdash A_0 q \text{ type}.$

By properties of substitution, $\Gamma, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A_0 q \text{ type}.$

Now apply the i.h.

• Case $\text{SpineRecover}$ Use Lemma 42 (Inversion of Principal Typing) (2), Lemma 47 (Checkprop Extension), and Lemma 41 (Extension Weakening for Principal Typing).

• Case $\text{SpinePass}$ By i.h.

• Case $\text{EmptySpine}$ Immediate.

• Case $\text{SpineRec}$ Use Lemma 42 (Inversion of Principal Typing) (1), Lemma 51 (Typing Extension), and Lemma 41 (Extension Weakening for Principal Typing).

• Case $\text{SpineAdd}$ Show that $\hat{\alpha}_1 \to \hat{\alpha}_2$ 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 $\Gamma_0 \vdash \tau : \kappa$
  
  \[
  \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \vdash \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1
  \]
  
  Immediate, since to the left of $\hat{\alpha}$, the contexts $\Delta$ and $\Gamma$ are the same.

- Case $\hat{\beta} \in \text{unsolved}(\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])$
  
  \[
  \Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} := \beta : \kappa \vdash \Gamma'[\hat{\alpha}][\hat{\beta} : \kappa]
  \]
  
  Immediate, since to the left of $\hat{\alpha}$, the contexts $\Delta$ and $\Gamma$ are the same.

- Case $\Gamma_0, \hat{\alpha}_2 : *, \hat{\alpha}_1 : *, \hat{\alpha} : * \equiv \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \vdash \hat{\alpha}_1 := \tau_1 : * \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : * \vdash \Delta$
  
  \[
  \Gamma_0, \hat{\alpha} : *, \Gamma_1 \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : * \vdash \Delta
  \]
  
  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:
  \[
  \begin{align*}
  \Gamma_0, \hat{\alpha}_2 : *, \hat{\alpha}_1 : *, \hat{\alpha} : * & \equiv \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \vdash \hat{\alpha}_1 := \tau_1 : * \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : * \vdash \Delta \\
  \Theta & \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : * \vdash \Delta
  \end{align*}
  \]
  
  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 $\Gamma_0, \hat{\alpha}_2 : *$, $\hat{\alpha}_1 : *, \hat{\alpha} \equiv \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1$.
  
  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), $\hat{\beta} \in \text{unsolved}(\Delta)$.

- Case $\Gamma'[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{zero} : N \vdash \Gamma'[\hat{\alpha} : N = \text{zero}]
  \]
  
  Immediate, since to the left of $\hat{\alpha}$, the contexts $\Delta$ and $\Gamma$ are the same.

- Case $\Gamma[\hat{\alpha}_1 : N, \hat{\alpha} : N = \text{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : N \vdash \Delta$
  
  \[
  \Gamma[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{succ}(t_1) : N \vdash \Delta
  \]
  
  We have $\hat{\beta} \in \text{unsolved}(\Gamma_0)$. Therefore $\hat{\beta} \in \text{unsolved}(\Gamma_0, \hat{\alpha}_1 : N)$. By i.h., $\hat{\beta} \in \text{unsolved}(\Delta)$.

Lemma 65 (Left Free Variable Preservation). If $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := t : \kappa \vdash \Delta$ and $\hat{\beta} \in \text{unsolved}(\Gamma_0)$ and $\hat{\beta} \not\in \text{FV}(\Gamma[s])$ and $\hat{\beta} \not\in \text{unsolved}(\Gamma_0)$ and $\hat{\beta} \not\in \text{FV}(\Delta[s])$.

Proof. By induction on the given instantiation derivation.
Proof of Lemma 65 (Left Free Variable Preservation)

Case

\[ \Gamma_0 \vdash \tau : \kappa \]

\[ \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Gamma_0, \hat{\alpha} : \kappa = \tau_1, \Gamma_1 \]

**InstSolve**

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

\[ \hat{\gamma} \in \text{unsolved}(\Gamma[\hat{\alpha} : \kappa][\hat{\gamma} : \kappa]) \]

\[ \Gamma[\hat{\alpha} : \kappa][\hat{\gamma} : \kappa] \vdash \hat{\alpha} := \hat{\gamma} : \kappa \rightarrow \Gamma[\hat{\alpha} : \kappa][\hat{\gamma} : \kappa] \]

**InstReach**

Since \( \Delta \) differs from \( \Gamma \) only in solving \( \hat{\gamma} \) to \( \hat{\alpha} \), applying \( \Delta \) to a type will not introduce a \( \hat{\beta} \). We have \( \hat{\beta} \notin FV(\Gamma|\sigma) \), so \( \hat{\beta} \notin FV(\Delta|\sigma) \).

Case

\[ \Gamma' \]

\[ \Gamma[\hat{\alpha}_2 : \times, \hat{\alpha}_1 : \times, \hat{\alpha} : \times = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 := \tau_1 : \times \rightarrow \Theta \]

\[ \Theta \vdash \hat{\alpha}_2 := (\Theta|\tau_2 : \times \rightarrow \Delta) \]

**InstBin**

We have \( \Gamma' \vdash \sigma \) type and \( \hat{\alpha} \notin FV(\Gamma'|\sigma) \) and \( \hat{\beta} \notin FV(\Gamma'|\sigma) \).

By weakening, we get \( \Gamma' \vdash \sigma : \kappa' \); since \( \hat{\alpha} \notin FV(\Gamma|\sigma) \) 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 \( \hat{\beta} \in (\Gamma_0, \hat{\alpha}_2 : \times) \).

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 \( \hat{\alpha}_2 \in \text{unsolved}(\Gamma_0, \hat{\alpha}_2 : \times) \).

By Lemma 64 (Left Unsolvedness Preservation), \( \hat{\alpha}_2 \in \text{unsolved}(\Theta) \).

Therefore \( \Theta \) has the form \( (\Theta_0, \hat{\alpha}_2 : \times, \Theta_1) \).

Since \( \hat{\beta} \neq \hat{\alpha}_2 \), we know that \( \hat{\beta} \) is declared to the left of \( \hat{\alpha}_2 \) in \( (\Gamma_0, \hat{\alpha}_2 : \times) \), 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

\[ \Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \text{zero} : \mathbb{N} \rightarrow \Gamma[\hat{\alpha} : \mathbb{N} = \text{zero}] \]

**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

\[ \Gamma[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \text{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : \mathbb{N} \rightarrow \Delta \]

\[ \Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \text{succ}(t_1) : \mathbb{N} \rightarrow \Delta \]

**InstSucc**
Proof of Lemma 65 (Left Free Variable Preservation).

\[ \Gamma \vdash \sigma : \kappa \quad \text{Given} \]
\[ \Theta \vdash \sigma : \kappa \quad \text{By weakening} \]
\[ \hat{\alpha} \notin FV(\Gamma) \quad \text{Given} \]
\[ \hat{\alpha} \notin FV(\Gamma) \quad \text{and} \Theta \text{ only solves} \hat{\alpha} \]
\[ \Theta = (\Gamma_0, \hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \text{succ}(\hat{\alpha}_1), \Gamma_1) \quad \text{Given} \]
\[ \hat{\beta} \notin \text{unsolved}(\Gamma_0) \quad \text{Given} \]
\[ \hat{\beta} \notin \text{unsolved}(\Gamma_0, \hat{\alpha}_1 : \mathbb{N}) \quad \text{fresh} \hat{\alpha}_1 \]
\[ \hat{\beta} \notin FV(\Gamma) \quad \text{Given} \]
\[ \hat{\beta} \notin FV(\Theta) \quad \text{fresh} \hat{\alpha}_1 \]
\[ \text{Eq.} \quad \hat{\beta} \notin FV(\Delta) \quad \text{By i.h.} \]

Lemma 66 (Instantiation Size Preservation). If \( \Gamma_0, \hat{\alpha}, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Delta \) and \( \Gamma \vdash s : \kappa' \) and \( \hat{\alpha} \notin FV(\Gamma) \), then \( |\Gamma| = |\Delta| \), where \( |C| \) is the plain size of the term \( C \).

**Proof.** By induction on the given derivation.

- **Case**
  \[
  \Gamma_0 \vdash \tau : \kappa \\
  \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Delta \\
  \Gamma \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Delta \\
  \text{(InstSolve)}
  \]

  Since \( \Delta \) differs from \( \Gamma \) only in solving \( \hat{\alpha} \), and we know \( \hat{\alpha} \notin FV(\Gamma) \), we have \( |\Delta| = |\Gamma| \); therefore \( |\Delta| = |\Gamma| \).

- **Case**
  \[
  \Gamma' \vdash \hat{\alpha} : \mathbb{N} \\
  \Gamma' \vdash \hat{\alpha} := \text{zero} : \mathbb{N} \\
  \Gamma' \vdash \hat{\alpha} := \text{zero} : \mathbb{N} = \text{zero} \\
  \text{(InstZero)}
  \]

  Similar to the InstSolve case.

- **Case**
  \[
  \Gamma' \vdash [\hat{\alpha} : \kappa] \mid \hat{\beta} : \kappa \mid \hat{\alpha} := \hat{\beta} : \kappa \rightarrow \Delta \\
  \Gamma' \vdash [\hat{\alpha} : \kappa] \mid \hat{\alpha} := \hat{\beta} : \kappa = \hat{\alpha} \\
  \text{(InstReach)}
  \]

  Here, \( \Delta \) differs from \( \Gamma \) only in solving \( \hat{\beta} \) to \( \hat{\alpha} \). However, \( \hat{\alpha} \) has the same size as \( \hat{\beta} \), so even if \( \hat{\beta} \notin FV(\Gamma) \), we have \( |\Delta| = |\Gamma| \).

- **Case**
  \[
  \Gamma' \vdash [\hat{\alpha}_2 : \tau_1 \rightarrow \Theta] \\
  \hat{\alpha}_1 := \tau_1 : \star \rightarrow \Theta \\
  \Theta \vdash \hat{\alpha}_2 := \tau_2 : \star \rightarrow \Delta \\
  \Gamma \vdash [\hat{\alpha}_1 : \star \rightarrow \Theta] \\
  \text{(InstBin)}
  \]

  We have \( \Gamma \vdash \sigma : \kappa' \) and \( \hat{\alpha} \notin FV(\Gamma) \).

  Since \( \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{dom}(\Gamma) \), we have \( \hat{\alpha}, \hat{\alpha}_1, \hat{\alpha}_2 \notin FV(\Gamma) \).

  By Lemma 23 (Deep Evar Introduction), \( \Gamma[\hat{\alpha} : \star] \rightarrow \Gamma' \).

  By Lemma 36 (Extension Weakening (Sorts)), \( \Gamma \vdash \sigma : \kappa' \).

  Since \( \hat{\alpha} \notin FV(\sigma) \), it follows that \( |\Gamma'| = |\Gamma| \), and so \( |\Gamma'| = |\Gamma| \).

  By induction on the first premise, \( |\Gamma'| = |\Theta| \).

  By Lemma 20 (Declaration Order Preservation), since \( \hat{\alpha}_2 \) is declared to the left of \( \hat{\alpha}_1 \) in \( \Gamma' \), we have
that \( \hat{\alpha}_2 \) is declared to the left of \( \hat{\alpha}_1 \) in \( \Theta \).
By Lemma 64 (Left Unresolvedness Preservation), since \( \hat{\alpha}_2 \in \text{unsolved}(\Gamma') \), it is unresolved in \( \Theta \): that is, \( \Theta = (\Theta_0, \hat{\alpha}_2 : \star, \Theta_1) \).
By Lemma 43 (Instantiation Extension), we have \( \Gamma' \rightarrow \Theta \).
By Lemma 36 (Extension Weakening (Sorts)), \( \Theta \vdash \sigma : \kappa' \).
Since \( \hat{\alpha}_2 \notin \text{FV}(\{\Gamma'\}\sigma) \), Lemma 65 (Left Free Variable Preservation) gives \( \hat{\alpha}_2 \notin \text{FV}(\Theta\sigma) \).
By induction on the second premise, \( |\Theta\sigma| = |\Delta\sigma| \), and by transitivity of equality, \( |\Gamma\sigma| = |\Delta\sigma| \).

\[\text{Case} \quad \Gamma' \quad \frac{\Gamma[\hat{\alpha} : \star] \vdash \sigma : \kappa'}{\Gamma[\hat{\alpha} : \star] \vdash \sigma : \kappa'} \quad \text{Given} \]
\[\hat{\alpha} \notin |\Gamma[\hat{\alpha} : \star]| \sigma \quad \text{Given} \]
\[\Gamma[\hat{\alpha} : \star] \rightarrow \Gamma' \quad \text{By Lemma 23 (Deep Evar Introduction)} \]
\[\Gamma' \vdash \sigma : \kappa' \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \]
\[|\Gamma'\sigma| = |\Gamma[\hat{\alpha} : \star]| \sigma \quad \text{By congruence of equality} \]
\[\hat{\alpha}_1 \notin |\Gamma'\sigma| \quad \text{Since } |\Gamma'\sigma| = |\Gamma[\hat{\alpha} : \star]| \sigma, \text{ and } \hat{\alpha}_1 \notin \text{dom}(\Gamma[\hat{\alpha} : \star]) \]
\[|\Gamma'\sigma| = |\Theta\sigma| \quad \text{By i.h.} \]
\[|\Gamma[\hat{\alpha} : \star]| \sigma = |\Theta\sigma| \quad \text{By transitivity of equality} \]

\[\square\]

**Lemma 67** (Decidability of Instantiation). If \( \Gamma = \Gamma_0[\hat{\alpha} : \kappa'] \) and \( \Gamma \vdash t : \kappa \) such that \( |\Gamma|t = t \) and \( \hat{\alpha} \notin \text{FV}(t) \), then:

(1) Either there exists \( \Delta \) such that \( \Gamma_0[\hat{\alpha} : \kappa'] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta \), or not.

**Proof.** By induction on the derivation of \( \Gamma \vdash t : \kappa \).

- **Case** \( (u : \kappa) \in \Gamma \)

\[\frac{\text{VarSort}}{\Gamma_L, \hat{\alpha} : \kappa', \Gamma_R \vdash u : \kappa} \]
If \( \kappa \neq \kappa' \), no rule matches and no derivation exists.
Otherwise:
- If \( (u : \kappa) \in \Gamma_L \), we can apply rule \text{InstReach}.
- If \( u \) is some unsolved existential variable \( \hat{\beta} \) and \( (\hat{\beta} : \kappa) \in \Gamma_R \), then we can apply rule \text{InstSolve}.
- Otherwise, \( u \) is declared in \( \Gamma_R \) and is a universal variable; no rule matches and no derivation exists.

- **Case** \( (\hat{\beta} : \kappa = \tau) \in \Gamma \)

\[\frac{\text{SolvedVarSort}}{\Gamma \vdash \hat{\beta} : \kappa} \]
By inversion, \( (\hat{\beta} : \kappa = \tau) \in \Gamma \), but \( |\Gamma|\hat{\beta} = \hat{\beta} \) is given, so this case is impossible.

- **Case** \text{UnitSort}
If \( \kappa' = \star \), then apply rule \text{InstSolve}. Otherwise, no rule matches and no derivation exists.

- **Case**

\[\frac{\text{BinSort}}{\Gamma \vdash \tau_1 : \star, \Gamma \vdash \tau_2 : \star} \]
\[\frac{\text{InstReach}}{\Gamma_L, \hat{\alpha} : \kappa', \Gamma_R \vdash \tau_1 \oplus \tau_2 : \star} \]

\[\square\]
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 $\vec{\alpha} \notin FV([\Gamma](\tau_1 \oplus \tau_2))$.
If $\Gamma_L \vdash \tau_1 \oplus \tau_2 : \star$, then we have a derivation by $\text{InstSolve}$.

If not, the only other rule whose conclusion matches $\tau_1 \oplus \tau_2$ is $\text{InstBin}$.

First, consider whether $\Gamma_L, \vec{\alpha}_2 : \star, \vec{\alpha}_1 : \star, \vec{\alpha} : \vec{\alpha}_1 \oplus \vec{\alpha}_2, \Gamma_R \vdash \vec{\alpha}_1 \leftarrow t : \star \vdash \Theta$. It is decidable.

By definition of substitution, $[\Gamma](\vec{\tau}) = ([\Gamma] \vec{\tau}_1) \cup ([\Gamma] \vec{\tau}_2)$. Since $[\Gamma](\vec{\tau}_1 \oplus \vec{\tau}_2) = \vec{\tau}_1 \oplus \vec{\tau}_2$, we have $[\Gamma']\vec{\tau}_1 = \vec{\tau}_1$ and $[\Gamma']\vec{\tau}_2 = \vec{\tau}_2$.

By weakening, $\Gamma_L, \vec{\alpha}_2 : \star, \vec{\alpha}_1 : \star, \vec{\alpha} : \vec{\alpha}_1 \oplus \vec{\alpha}_2, \Gamma_R \vdash \vec{\alpha}_1 \vdash t : \star \vdash \Theta$. Since $\Gamma \vdash \tau_1 : \star$ and $\Gamma \vdash \tau_2 : \star$, we have $\vec{\alpha}_1, \vec{\alpha}_2 \notin FV(\vec{\tau}_1) \cup FV(\vec{\tau}_2)$.

By i.h., either there exists $\tau$ s.t. $\Gamma_L, \vec{\alpha}_2 : \star, \vec{\alpha}_1 : \star, \vec{\alpha} : \vec{\alpha}_1 \oplus \vec{\alpha}_2, \Gamma_R \vdash \vec{\alpha}_1 \vdash \tau_1 : \star \vdash \Theta$, or not.

If not, then no derivation by $\text{InstBin}$ exists.

Otherwise, there exists such a $\Theta$. By Lemma 64 (Left Unsolvedness Preservation), we have $\vec{\alpha}_2 \in \text{unsolved}(\Theta)$.

By Lemma 65 (Left Free Variable Preservation), we know that $\vec{\alpha}_2 \notin FV(\Theta)\vec{\tau}_2$.

Substitution is idempotent, so $\Theta \vdash \Theta \vec{\tau}_2 = \Theta \vec{\tau}_2$.

By i.h., either there exists $\Delta$ such that $\Theta \vdash \vec{\alpha}_2 := \Theta \vec{\tau}_2 : \kappa \vdash \Delta$, or not.

If not, no derivation by $\text{InstBin}$ exists.

Otherwise, there exists such a $\Delta$. By rule $\text{InstBin}$, we have $\Gamma \vdash \vec{\alpha} := t : \kappa \vdash \Delta$.

• Case $\Gamma \vdash \text{zero} : \mathbb{N}$ $\text{ZeroSort}$

If $\kappa' \neq \mathbb{N}$, then no rule matches and no derivation exists. Otherwise, apply rule $\text{InstSolve}$.

• Case $\Gamma \vdash \text{succ}(t_0) : \mathbb{N}$ $\text{SuccSort}$

If $\kappa' \neq \mathbb{N}$, then no rule matches and no derivation exists. Otherwise:
If $\Gamma_L \vdash \text{succ}(t_0) : \mathbb{N}$, then we have a derivation by $\text{InstSolve}$.

If not, the only other rule whose conclusion matches $\text{succ}(t_0)$ is $\text{InstSucc}$.

The remainder of this case is similar to the $\text{BinSort}$ case, but shorter. \qed

G’ Separation

Lemma 68 (Transitivity of Separation).

If $(\Gamma_L \ast \Gamma_R) \xrightarrow{\star} (\Theta_L \ast \Theta_R)$ and $(\Theta_L \ast \Theta_R) \xrightarrow{\star} (\Delta_L \ast \Delta_R)$
then $(\Gamma_L \ast \Gamma_R) \xrightarrow{\star} (\Delta_L \ast \Delta_R)$.

Proof.

\[
\begin{align*}
(\Gamma_L \ast \Gamma_R) & \xrightarrow{\star} (\Theta_L \ast \Theta_R) & \text{Given} \\
(\Gamma_L, \Gamma_R) & \longrightarrow (\Theta_L, \Theta_R) & \text{By Definition 5} \\
\Gamma_L & \subseteq \Theta_L \text{ and } \Gamma_R \subseteq \Theta_R & " \\
(\Theta_L \ast \Theta_R) & \xrightarrow{\star} (\Delta_L \ast \Delta_R) & \text{Given} \\
(\Theta_L, \Theta_R) & \longrightarrow (\Delta_L, \Delta_R) & \text{By Definition 5} \\
\Theta_L & \subseteq \Delta_L \text{ and } \Theta_R \subseteq \Delta_R & " \\
(\Gamma_L, \Gamma_R) & \longrightarrow (\Delta_L, \Delta_R) & \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma_L & \subseteq \Delta_L \text{ and } \Gamma_R \subseteq \Delta_R & \text{By transitivity of } \subseteq \\
\therefore (\Gamma_L \ast \Gamma_R) & \xrightarrow{\star} (\Delta_L \ast \Delta_R) & \text{By Definition 5}
\end{align*}
\]

\qed
Lemma 69 (Separation Truncation).
If $H$ has the form $\alpha : \kappa$ or $\Gamma^\alpha$ or $\Gamma_0^\alpha$ or $x : A \cdot p$
and $(\Gamma_1 \cdot (\Gamma_R, H)) \vdash \Delta$ where $\Delta_R = (\Delta_0, H, \Theta)$
then $(\Gamma_1 \cdot (\Gamma_R)) \vdash \Delta$.

Proof. By induction on $\Delta_R$
If $\Delta_R = (\ldots, H)$, we have $(\Gamma_1 \cdot (\Gamma_R, H)) \vdash \Delta$, and inversion on $\rightarrow \text{Uvar}$ (if $H$ is $\alpha : \kappa$, or the corresponding rule for other forms) gives the result (with $\Theta = \cdot$).
Otherwise, proceed into the subderivation of $(\Gamma_1, \Gamma_R, \alpha : \kappa) \vdash \Delta$, with $\Delta_R = (\Delta_0', \Delta')$ where $\Delta'$ is a single declaration. Use the i.h. on $\Delta_0'$, producing some $\Theta'$. Finally, let $\Theta = (\Theta', \Delta')$.

Lemma 70 (Separation for Auxiliary Judgments).

(i) If $\Gamma_L \cdot \Gamma_R \vdash \sigma \equiv \tau : \kappa \vdash \Delta$
and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \cdot \Delta_R)$ and $(\Gamma_1 \cdot (\Gamma_R)) \vdash \Delta$.

(ii) If $\Gamma_L \cdot \Gamma_R \vdash P \vdash \Delta$
and $\text{FEV}(P) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \cdot \Delta_R)$ and $(\Gamma_1 \cdot (\Gamma_R)) \vdash \Delta$.

(iii) If $\Gamma_L \cdot \Gamma_R / \sigma \equiv \tau : \kappa \vdash \Delta$
and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset$
then $\Delta = (\Delta_L \cdot (\Delta_R, \Theta))$ and $(\Gamma_1 \cdot (\Gamma_R, \Theta)) \vdash \Delta$.

(iv) If $\Gamma_L \cdot \Gamma_R / P \vdash \Delta$
and $\text{FEV}(P) = \emptyset$
then $\Delta = (\Delta_L \cdot (\Delta_R, \Theta))$ and $(\Gamma_1 \cdot (\Gamma_R, \Theta)) \vdash \Delta$.

(v) If $\Gamma_L \cdot \Gamma_R \vdash \hat{\alpha} := \tau : \kappa \vdash \Delta$
and $(\text{FEV}(\tau) \cup \{\hat{\alpha}\}) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \cdot (\Delta_R))$ and $(\Gamma_1 \cdot (\Gamma_R)) \vdash \Delta$.

(vi) If $\Gamma_L \cdot \Gamma_R \vdash A \equiv B \vdash \Delta$
and $\text{FEV}(A) \cup \text{FEV}(B) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \cdot (\Delta_R))$ and $(\Gamma_1 \cdot (\Gamma_R)) \vdash \Delta$.

Proof. Part (i): By induction on the derivation of the given checkeq judgment. Cases [CheckeqVar] [CheckeqUnit] and [CheckeqZero] are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). For case [CheckeqSucc], apply the i.h. For cases [CheckeqInstL] and [CheckeqInstR] use the i.h. (v). For case [CheckeqBin] use reasoning similar to that in the $\Delta$ case of Lemma 77 (Separation---Main) (transitivity of separation, and applying $\Theta$ in the second premise).

Part (ii), (iv), (v), (vi), i.e., pi: Use the i.h. (i).

Part (i), (iii), elimp: Cases [ElimeqUvarRef] [ElimeqUnit] and [CheckeqZero] are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). Cases [ElimeqUvarL] [ElimeqUvarR] [ElimeqUnit] and [ElimeqBinBot] are impossible (we have $\Delta$, not $\perp$). For case [ElimeqSucc], apply the i.h. The case for [ElimeqBin] is similar to the case [CheckeqBin] in part (i). For cases [ElimeqUvarL] and [ElimeqUvarR] we have $\Delta = (\Gamma_1, \Gamma_R, \alpha : \tau)$ which, since $\text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$, ensures that $(\Gamma_1 \cdot (\Gamma_R, \alpha : \tau)) \vdash (\Delta_L \cdot (\Delta_R, \alpha : \tau))$.

Part (iii), i.e., instjudg:

- Case [InstSolve] Here, $\Gamma = (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1)$ and $\Delta = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1)$. We have $\hat{\alpha} \in \text{dom}(\Gamma_R)$, so the declaration $\hat{\alpha} : \kappa$ is in $\Gamma_R$. Since $\text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$, the context $\Delta$ maintains the separation.
Proof of Lemma 70 (Separation for Auxiliary Judgments). Lemma 70 (Separation for Auxiliary Judgments) states that if \( \Gamma_L \ast \Gamma_R \vdash A \triangleleft P \; B \dashv \Delta \) 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 \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\text{InstRev}} (\Delta_L \ast \Delta_R) \) and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \).

**Proof.** By induction on the given derivation. In the \( \triangleleft \text{Equiv} \) 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).

1. **(Spines)** If \( \Gamma_L \ast \Gamma_R \vdash s : A \triangleright C \; q \dashv \Delta \) or \( \Gamma_L \ast \Gamma_R \vdash s : A \triangleright C \; \lbrack q \rbrack \dashv \Delta \) and \( \Gamma_L \ast \Gamma_R \vdash A \triangleright P \) type and \( \text{FEV}(A) \subseteq \text{dom}(\Gamma_R) \) then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\text{InstRev}} (\Delta_L \ast \Delta_R) \) and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \).

2. **(Checking)** If \( \Gamma_L \ast \Gamma_R \vdash e \iff C \; p \dashv \Delta \) and \( \Gamma_L \ast \Gamma_R \vdash C \; p \triangleright P \) type and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \) then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\text{InstRev}} (\Delta_L \ast \Delta_R) \).

3. **(Synthesis)** If \( \Gamma_L \ast \Gamma_R \vdash e \Rightarrow A \; p \dashv \Delta \) then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\text{InstRev}} (\Delta_L \ast \Delta_R) \).

4. **(Match)** If \( \Gamma_L \ast \Gamma_R \vdash \Pi :: A \; q \iff C \; p \dashv \Delta \) and \( \text{FEV}(\vec{A}) = \emptyset \) and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \) then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{\text{InstRev}} (\Delta_L \ast \Delta_R) \).
(Match Elim.) If $\Gamma_L \ast \Gamma_R / \Pi \vdash \bar{A} ! \leftrightarrow C p \vdash \Delta$
and $\text{FEV}(P) = \emptyset$
and $\text{FEV}(\bar{A}) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \vdash_{\bar{A}} (\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 $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$.
- **Case Match\land:** Here we use the i.h. (Match Elim.), which requires that $\text{FEV}(P) = \emptyset$, which motivates $\text{FEV}(\bar{A}) = \emptyset$.
- **Case MatchNeg:** In its premise, this rule appends a type $A \in \bar{A}$ to $\Gamma_R$ and claims it is principal ($z : A!$), which motivates $\text{FEV}(\bar{A} = \emptyset)$.

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 | I | ] ⇒ [I]:** 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**

  $\Gamma_L \ast \Gamma_R \vdash \cdot : A \ p \triangleright \bigtriangledown \bigtriangledown p \triangleright \bigtriangledown q \vdash \Gamma_L \ast \Gamma_R$

  Let $\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$.
We have $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$. Since $\Delta_R = \Gamma_R$ and $C = A$, it is immediate that $\text{FEV}(C) \subseteq \text{dom}(\Delta_R)$.

- **Case**

  $\Gamma_L \ast \Gamma_R \vdash e \Rightarrow A \ q \vdash \emptyset \vdash B \ p \vdash \Delta$

  By i.h., $\emptyset = (\emptyset_L \ast \emptyset_R)$ and $(\Gamma_L \ast \Gamma_R) \vdash_{\emptyset} (\emptyset_L \ast \emptyset_R)$.
By Lemma 71 (Separation for Subtyping), $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Theta_L \ast \Theta_R) \vdash_{\emptyset} (\Delta_L \ast \Delta_R)$.
By Lemma 68 (Transitivity of Separation), $(\Gamma_L \ast \Gamma_R) \vdash_{\emptyset} (\Delta_L \ast \Delta_R)$.

- **Case**

  $\Gamma \vdash A \ type \quad \Gamma \vdash e \Leftarrow [\Gamma] A! \vdash \Delta$

  By i.h.; since $\text{FEV}(A) = \emptyset$, the condition on the (Checking) part is trivial.

- **Case**

  $\Gamma[\bar{A} : \bar{A}] \vdash () \Leftarrow \bar{\bar{A}} \vdash \Delta[\bar{A} : \bar{A} = 1]$

  Adding a solution with a ground type cannot destroy separation.

- **Case**

  $\nu \text{chk-I} \quad \Gamma_L, \Gamma_R, \alpha : \kappa \vdash v \Leftarrow A_0 \ p \vdash \Delta, \alpha : \kappa, \emptyset$

  $\Gamma_L, \Gamma_R \vdash v \Leftarrow \forall \alpha : \kappa. A_0 \ p \vdash \Delta$
Proof of Lemma 72 (Separation—Main) 

\[ \text{Proof of Lemma 72 (Separation—Main)} \]

[lem:separation-main]

- Case  
  \[ \Gamma_L, \Gamma_R, \hat{\alpha} : \kappa \vdash e \ s : [\hat{\alpha}/\alpha]A_0 \Rightarrow C q \vdash \Delta \]
  \[ \Gamma_L, \Gamma_R \vdash e \ s : \forall \alpha : \kappa.A_0 p \Rightarrow C q \vdash \Delta \]

  \[ \text{FEV}(\forall \alpha : \kappa.A_0) \subseteq \text{dom}(\Gamma_R) \]
  \[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R, \alpha : \kappa) \]
  \[ (\Delta, \alpha : \kappa, \Theta) = (\Delta_L * \Delta_R^\kappa) \]
  \[ (\Gamma_L * (\Gamma_R, \alpha : \kappa)) \xrightarrow{\kappa} (\Delta_L * \Delta_R^\kappa) \]
  \[ \Delta_L^\kappa = (\Delta_R, \alpha : \kappa, \Theta) \]
  \[ (\Delta, \alpha : \kappa, \Theta) = (\Delta_L * \Delta_R^\kappa) \]
  \[ = (\Delta_L, \Delta_R, \alpha : \kappa) \]

  \[ \Delta = (\Delta_L, \Delta_R) \]

  \[ \alpha \text{ not multiply declared} \]

- Case  
  \[ \Gamma_L, \Gamma_R, \hat{\alpha} : \kappa \vdash e \ s : [\hat{\alpha}/\alpha]A_0 \Rightarrow C q \vdash \Delta \]
  \[ \Gamma_L, \Gamma_R \vdash e \ s : \forall \alpha : \kappa.A_0 p \Rightarrow C q \vdash \Delta \]

  \[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R) \]
  \[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R, \hat{\alpha} : \kappa) \]
  \[ \Delta = (\Delta_L * \Delta_R) \]
  \[ (\Gamma_L * (\Gamma_R, \hat{\alpha} : \kappa)) \xrightarrow{\kappa} (\Delta_L * \Delta_R) \]
  \[ \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \]
  \[ \text{dom}(\Gamma_L) \subseteq \text{dom}(\Delta_L) \]
  \[ \text{dom}(\Gamma_R, \hat{\alpha} : \kappa) \subseteq \text{dom}(\Delta_R) \]
  \[ \text{dom}(\Gamma_R) \cup (\hat{\alpha}) \subseteq \text{dom}(\Delta_R) \]
  \[ \text{dom}(\Gamma_R) \subseteq \text{dom}(\Delta_R) \]
  \[ (\Gamma_L, \Gamma_R) \rightarrow (\Delta_L, \Delta_R) \]
  \[ (\Gamma_L, \Gamma_R) \rightarrow (\Delta_L, \Delta_R) \]

  \[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R) \]
  \[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R, \hat{\alpha} : \kappa) \]
  \[ \Delta = (\Delta_L * \Delta_R) \]
  \[ (\Gamma_L * (\Gamma_R, \hat{\alpha} : \kappa)) \xrightarrow{\kappa} (\Delta_L * \Delta_R) \]

- Case  
  \[ e \text{ not a case} \]
  \[ \Gamma_L, \Gamma_R \vdash e \ s : (A_0 \land P) p \vdash \Delta \]
  \[ \Gamma_L * \Gamma_R \vdash e \ s : (A_0 \land P) p \vdash \Delta \]

  \[ \text{FEV}(A_0 \land P) \subseteq \text{dom}(\Gamma_R) \]
  \[ \text{FEV}(P) \subseteq \text{dom}(\Gamma_R) \]
  \[ \Theta = (\Theta_L * \Theta_R) \]
  \[ (\Gamma_L * \Gamma_R) \xrightarrow{\kappa} (\Theta_L * \Theta_R) \]

\[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R) \]
\[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R, \hat{\alpha} : \kappa) \]
\[ \Delta = (\Delta_L * \Delta_R) \]
\[ (\Gamma_L * (\Gamma_R, \hat{\alpha} : \kappa)) \xrightarrow{\kappa} (\Delta_L * \Delta_R) \]

\[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R) \]
\[ \text{FEV}(\forall \alpha : \kappa A_0) \subseteq \text{dom}(\Gamma_R, \hat{\alpha} : \kappa) \]
\[ \Delta = (\Delta_L * \Delta_R) \]
\[ (\Gamma_L * (\Gamma_R, \hat{\alpha} : \kappa)) \xrightarrow{\kappa} (\Delta_L * \Delta_R) \]
Proof of Lemma 72 (Separation—Main) \lem:separation-main

\[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \]
\[ \text{dom}(\Gamma_R) \subseteq \text{dom}(\Theta_R) \]
\[ \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R) \]
\[ \text{FEV}(\Theta A_0) \subseteq \text{dom}(\Theta_R) \]
\[ \Gamma_L \ast \Gamma_R \vdash (A_0 \land P) \text{ p type} \]
\[ \Gamma_L \ast \Gamma_R \vdash A_0 \text{ p type} \]
\[ \Theta \vdash A_0 \text{ p type} \]
\[ \Theta \vdash (\Theta A_0) \text{ p type} \]

\[ \Delta = (\Delta_L \ast \Delta_R) \]
\[ (\Theta L \ast \Theta R) \vdash (\Delta L \ast \Delta_R) \]
\[ (\Gamma L \ast \Gamma_R) \vdash (\Delta L \ast \Delta_R) \]

Case \text{Nil} \quad \text{Similar to a section of the } \land \text{ case.}

Case \text{Cons} \quad \text{Similar to the } \land \text{ case, with an extra use of the i.h. for the additional second premise.}

By i.h.

By previous line

By inversion

By Lemma 41 (Extension Weakening for Principal Typing)

By Lemma 13 (Right-Hand Substitution for Typing)

By Lemma 68 (Transitivity of Separation)

Above

By Definition 5

Above and def. of FEV

Below

By Lemma 70 (Separation for Auxiliary Judgments) (iv)

By Lemma 42 (Inversion of Principal Typing) (2)

By Lemma 35 (Suffix Weakening)

By Lemmas 41 and 40

Immediate

By i.h.

By Lemma 69 (Separation Truncation)

Similar to the \land \text{ case}
Proof of Lemma 72 (Separation—Main) lemm:separation-main

\[
\begin{align*}
\Gamma_L \ast \Gamma_R \vdash (P \supset A_0) \text{ p type} & \quad \text{Given} \\
\Gamma_L \ast \Gamma_R \vdash P \text{ prop} & \quad \text{By inversion} \\
\Gamma_L, \Gamma_R \vdash \text{P true } \vdash \Theta & \quad \text{Subderivation} \\
\Theta = (\Theta_L \ast \Theta_R) & \quad \text{By Lemma 70 (Separation for Auxiliary Judgments) (i)} \\
(\Gamma_L \ast \Gamma_R) \overset{-\ast}{=} (\Theta_L \ast \Theta_R) & \quad \text{By Lemma 70 (Separation for Auxiliary Judgments) (ii)} \\
\end{align*}
\]

\[
\begin{align*}
\Theta \vdash e \ s : [\Theta]A_0 \ p \ \succ C \ q \vdash \Delta & \quad \text{Subderivation} \\
(\Delta_i \triangleright p, \Delta') = (\Delta_L \ast \Delta_R) & \quad \text{By i.h.} \\
(\Theta_L \ast \Theta_R) \overset{-\ast}{=} (\Delta_L \ast \Delta_R) & \quad \text{By i.h.} \\
\text{\textast \textasteriskcentered \textast \textasteriskcentered \textast \textasteriskcentered \textasteriskcentered FEV(C) \subseteq \text{dom}(\Delta_R)} & \quad \text{By Lemma 68 (Transitivity of Separation)} \\
\end{align*}
\]

• Case \( \Gamma_L, \Gamma_R, x: C \ p \vdash \nu \iff C \ p \vdash \Delta, x: C \ p, \Theta \):

\[
\begin{align*}
\Gamma_L, \Gamma_R \vdash \text{rec } x. \nu \iff C \ p \vdash \Delta & \quad \text{Rec} \\
\Gamma_L, \Gamma_R \vdash C \ p \vdash \Delta & \quad \text{Subderivation} \\
\text{FEV}(C) \subseteq \text{dom}(\Gamma_R) & \quad \text{By i.h.} \\
\Gamma_L \ast (\Gamma_L \ast C p) \vdash C \ p \vdash \Delta & \quad \text{Subderivation} \\
\text{FEV}(C) \subseteq \text{dom}(\Gamma_R) & \quad \text{By i.h.} \\
\Gamma_L, \Gamma_R, x: C \ p \vdash \nu \iff C \ p \vdash \Delta, x: C \ p, \Theta & \quad \text{Subderivation} \\
(\Delta, x: C \ p, \Theta) = (\Delta_L, \Delta_R) & \quad \text{By i.h.} \\
\text{FEV}(\Delta_L) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\text{FEV}(\Delta_R) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\Gamma_L \ast (\Gamma_L \ast C p) \vdash C \ p \vdash \Delta & \quad \text{Subderivation} \\
\Gamma_L, \Gamma_R, x: C \ p \vdash \nu \iff C \ p \vdash \Delta, x: C \ p, \Theta & \quad \text{Subderivation} \\
(\Delta, x: C \ p, \Theta) = (\Delta_L, \Delta_R) & \quad \text{By i.h.} \\
\text{FEV}(\Delta_L) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\text{FEV}(\Delta_R) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\Gamma_L \ast (\Gamma_L \ast C p) \vdash C \ p \vdash \Delta & \quad \text{Subderivation} \\
\text{FEV}(\Delta_L) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\text{FEV}(\Delta_R) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\end{align*}
\]

Similar to the \( \nu \) case.

• Case \( \Gamma_L, \Gamma_R, x: A \ p \vdash e \iff B \ p \vdash \Delta, x: A \ p, \Theta \):

\[
\begin{align*}
\Gamma_L, \Gamma_R \vdash \lambda x. e \iff A \rightarrow B \ p \vdash \Delta & \quad \text{Subderivation} \\
\Gamma_L \ast \Gamma_R \vdash (A \rightarrow B) \ p \text{ type} & \quad \text{Given} \\
\Gamma_L, \Gamma_R \vdash B \ p \vdash \Delta & \quad \text{Subderivation} \\
\text{FEV}(A \rightarrow B) \subseteq \text{dom}(\Gamma_R) & \quad \text{By i.h.} \\
\Gamma_L \ast (\Gamma_L \ast A p) \vdash B \ p \vdash \Delta & \quad \text{Subderivation} \\
\text{FEV}(A) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\Gamma_L, \Gamma_R, x: A \ p \vdash e \iff B \ p \vdash \Delta, x: A \ p, \Theta & \quad \text{Subderivation} \\
(\Delta, x: A \ p, \Theta) = (\Delta_L, \Delta_R) & \quad \text{By i.h.} \\
\text{FEV}(\Delta_L) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\text{FEV}(\Delta_R) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\Gamma_L, \Gamma_R, x: A \ p \vdash e \iff B \ p \vdash \Delta, x: A \ p, \Theta & \quad \text{Subderivation} \\
(\Delta, x: A \ p, \Theta) = (\Delta_L, \Delta_R) & \quad \text{By i.h.} \\
\text{FEV}(\Delta_L) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\text{FEV}(\Delta_R) \subseteq \text{dom}(\Gamma_R) & \quad \text{By def. of FEV} \\
\end{align*}
\]

Similar to the \( \nu \) case.

• Case \( \Gamma_L[\hat{x}_1 : \ast, \hat{x}_2 : \ast, \hat{x} : \ast \ast \ast = \hat{x}_1 \rightarrow \hat{x}_2], x : \hat{x}_1 \vdash e_0 \iff \hat{x}_2 \vdash \Delta, x : \hat{x}_1, \Delta' \)

\[
\begin{align*}
\Gamma_L, \Gamma_R \vdash \lambda x. e_0 \iff \hat{x} \vdash \Delta & \quad \text{Subderivation} \\
\Gamma_L \ast \Gamma_R \vdash \lambda x. e_0 \iff \hat{x} \vdash \Delta & \quad \text{Subderivation} \\
\end{align*}
\]

We have \( \Gamma_L \ast \Gamma_R = \Gamma_0[\hat{x} : \ast] \). We also have \( \text{FEV}(\hat{x}) \subseteq \text{dom}(\Gamma_R) \). Therefore \( \hat{x} \in \text{dom}(\Gamma_R) \) and

\[
\Gamma_0[\hat{x} : \ast] \quad = \quad \Gamma_L, \Gamma_2, \hat{x} : \ast, \Gamma_3
\]
Proof of Lemma 72 (Separation—Main)

where Γ_R = (Γ_2, α : *, Γ_3).

Then the input context in the premise has the following form:

Γ₀[⟨α₁:*⟩, ⟨α₂:*⟩, α:* = α₁→α₂], x : α₁ = Γ_L, Γ_2, α₁:*, α₂:* = α₁→α₂, Γ_3, x : α₁

Let us separate this context at the same point as Γ₀[⟨α⟩], that is, after Γ_L and before Γ_2, and call the resulting right-hand context Γ'_R. That is,

Γ₀[⟨α₁:*⟩, ⟨α₂:*⟩, α:* = α₁→α₂], x : α₁ = Γ_L * (Γ_2, ⟨α₁:*⟩, ⟨α₂:*⟩) = α₁→α₂, Γ_3, x : α₁

Let us separate this context at the same point as Γ₀[⟨α⟩], that is, after Γ_L and before Γ_2, and call the resulting right-hand context Γ'_R. That is,

Γ₀[⟨α₁:*⟩, ⟨α₂:*⟩, α:* = α₁→α₂], x : α₁ = Γ_L * (Γ_2, ⟨α₁:*⟩, ⟨α₂:*⟩) = α₁→α₂, Γ_3, x : α₁

FEV(α) ⊆ dom(Γ_R)

Given

Γ_L * Γ'_R ⊢ e₀ ↔ α₂ → Δ, x : α₁, Δ'

Subderivation

Γ_L * Γ'_R ⊢ α₂ f type

α₂ ∈ dom(Γ'_R)

FEV(α₂) ⊆ dom(Γ'_R)

(Δ, x : α₁, Δ') = (Δ_L, Δ'_R)

By i.h.

Γ_L * Γ'_R ⊢ α₁ = (Δ_L, Δ_R)

Similar to the ν₁ case

Γ_L * Γ'_R ⊢ α₂ = (Δ_L, Δ_R)

Similar to the ν₁ case

---

• Case

Γ ⊢ e ⇒ A p ⊢ Θ

Θ ⊢ s : [Θ] A p ⇒ C [q] ⊢ Δ

Γ ⊢ e s ⇒ C q ⊢ Δ

By i.h. and Lemma 68 (Transitivity of Separation), with Lemma 91 (Well-formedness of Algorithmic Typing) and Lemma 13 (Right-Hand Substitution for Typing).

• Case

Γ ⊢ s : A ! ⇒ C f ⊢ Δ

FEV([Δ]C) = ∅

Γ ⊢ s : A ! ⊢ C [q] ⊢ Δ

SpineRecover

Use the i.h.

• Case

Γ ⊢ s : A p ⇒ C q ⊢ Δ

(⟨p = f⟩ or (q = !) or (FEV([Δ]C) ≠ ∅))

Γ ⊢ s : A p ⇒ C [q] ⊢ Δ

SpinePass

Use the i.h.

• Case

Γ_L * Γ_R ⊢ e ⇐ A_1 p ⊢ Θ

Θ ⊢ s : [Θ] A_2 p ⇒ C q ⊢ Δ

Γ_L * Γ_R ⊢ e s : A_1 → A_2 p ⇒ C q ⊢ Δ

Spine

Use the i.h.
Proof of **Lemma 72** (Separation—Main) \( \text{lem:separation-main} \)

\[ \Gamma \vdash (A_1 \rightarrow A_2) \text{ p type} \quad \text{Given} \]
\[ \Gamma \vdash A_1 \text{ type} \quad \text{By inversion} \]
\[ \text{FEV}(A_1 \rightarrow A_2) \subseteq \text{dom}(\Gamma_R) \quad \text{Given} \]
\[ \text{FEV}(A_1) \subseteq \text{dom}(\Gamma_R) \quad \text{By def. of FEV} \]
\[ \Theta = (\Theta_L, \Theta_R) \quad \text{By i.h.} \]
\[ (\Gamma_L * \Gamma_R) \dashv\vdash (\Theta_L * \Theta_R) \quad " \]

\[ \Gamma \vdash A_2 \text{ type} \quad \text{By inversion} \]
\[ \Gamma \vdash [\Theta]A_2 \text{ type} \quad \text{By Lemma 13 (Right-Hand Substitution for Typing)} \]
\[ \text{FEV}(A_2) \subseteq \text{dom}(\Gamma_R) \quad \text{By def. of FEV} \]
\[ \Delta = (\Delta_L, \Delta_R) \quad \text{By i.h.} \]
\[ (\Theta_L * \Theta_R) \dashv\vdash (\Delta_L * \Delta_R) \quad " \]
\[ \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \quad " \]
\[ \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \quad \text{By Lemma 68 (Transitivity of Separation)} \]

• **Case**

\[ \Gamma \vdash e \leftarrow A_k \text{ p } \dashv \Delta \]
\[ \Gamma \vdash \text{inj}_k e \leftarrow A_1 + A_2 \text{ p } \dashv \Delta \]

Use the i.h. (inverting \( \Gamma \vdash (A_1 + A_2) \text{ p type} \)).

• **Case**

\[ \Gamma \vdash e_1 \leftarrow A_1 \text{ p } \dashv \Theta \quad \Theta \vdash e_2 \leftarrow [\Theta]A_2 \text{ p } \dashv \Delta \]
\[ \Gamma \vdash \langle e_1 , e_2 \rangle \leftarrow A_1 \times A_2 \text{ p } \dashv \Delta \]

\[ \Gamma \vdash (A_1 \times A_2) \text{ p type} \quad \text{Given} \]
\[ \Gamma \vdash A_1 \text{ p type} \quad \text{By inversion} \]
\[ \Gamma \vdash e_1 \leftarrow A_1 \text{ p } \dashv \Theta \quad \text{Subderivation} \]
\[ \Theta = (\Theta_L, \Theta_R) \quad \text{By i.h.} \]
\[ (\Gamma_L * \Gamma_R) \dashv\vdash (\Theta_L * \Theta_R) \quad " \]

\[ \Gamma \vdash A_2 \text{ type} \quad \text{By inversion} \]
\[ \Gamma \rightarrow \Theta \quad \text{By Lemma 51 (Typing Extension)} \]
\[ \Theta \vdash A_2 \text{ type} \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \]
\[ \Theta \vdash [\Theta]A_2 \text{ type} \quad \text{By Lemma 13 (Right-Hand Substitution for Typing)} \]
\[ \Theta \vdash e_2 \leftarrow [\Theta]A_2 \text{ p } \dashv \Delta \quad \text{Subderivation} \]

\[ \Delta = (\Delta_L, \Delta_R) \quad \text{By i.h.} \]
\[ (\Theta_L * \Theta_R) \dashv\vdash (\Delta_L * \Delta_R) \quad " \]
\[ (\Gamma_L * \Gamma_R) \dashv\vdash (\Delta_L * \Delta_R) \quad \text{By Lemma 68 (Transitivity of Separation)} \]
**Proof of Lemma 72** *(Separation—Main)*

**Case**

\[
\Gamma ([\hat{\alpha}_2:*], \hat{\alpha}_1:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2) \vdash e \iff \hat{\alpha}_1 \not\in \Theta \\
\Theta \vdash e_2 \iff [\Theta] \hat{\alpha}_2 \not\vdash \Delta
\]

We have \((\Gamma_L \ast \Gamma_R) = \Gamma_0 \hat{\alpha} : \star\). We also have FEV(\hat{\alpha}) \subseteq \text{dom}(\Gamma_R). Therefore \(\hat{\alpha} \in \text{dom}(\Gamma_R)\) and

\[
\Gamma_0[\hat{\alpha} : \star] = \Gamma_L, \Gamma_2, \hat{\alpha} : \star, \Gamma_3
\]

where \(\Gamma_R = (\Gamma_2, \hat{\alpha} : \star, \Gamma_3)\).

Then the input context in the premise has the following form:

\[
\Gamma_0[\hat{\alpha}_1:*; \hat{\alpha}_2:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2] = (\Gamma_L, \Gamma_2, \hat{\alpha}_1:*; \hat{\alpha}_2:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3)
\]

Let us separate this context at the same point as \(\Gamma_0[\hat{\alpha} : \star]\), that is, after \(\Gamma_L\) and before \(\Gamma_2\), and call the resulting right-hand context \(\Gamma'_R\):

\[
\Gamma_0[\hat{\alpha}_1:*; \hat{\alpha}_2:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2] = \Gamma_L \ast (\Gamma_2, \hat{\alpha}_1:*; \hat{\alpha}_2:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3)
\]

\[
\Gamma'_R = (\Gamma_2, \hat{\alpha} : \star, \Gamma_3)
\]

\[
\Gamma_L \ast \Gamma'_R = (\Gamma_L \ast (\Gamma_2, \hat{\alpha}_1:*; \hat{\alpha}_2:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3))
\]

By Lemma 23 *(Deep Evar Introduction)* (i), (ii) and the definition of separation, we can show

\[
(\Gamma_L \ast (\Gamma_2, \hat{\alpha} : \star, \Gamma_3)) \vdash (\Gamma_L \ast (\Gamma_2, \hat{\alpha}_1:*; \hat{\alpha}_2:*; \hat{\alpha}:* = \hat{\alpha}_1 \times \hat{\alpha}_2, \Gamma_3))
\]

\[
\ast \ast \Gamma_L \ast \Gamma_R \frac{\vdash (\Gamma_L \ast \Gamma'_R)}{\vdash (\Gamma_L \ast \Gamma'_R) \by \text{above equalities}} \by \text{Lemma 68 *(Transitivity of Separation)* twice}
\]

**Case**

\[
\Gamma(\hat{\alpha}_2:*; \hat{\alpha}_1:*; \hat{\alpha}:* = \hat{\alpha}_1 \rightarrow \hat{\alpha}_2) \vdash e \iff \hat{\alpha}_1 \not\in \Theta \\
\Theta \vdash e_2 \iff [\Theta] \hat{\alpha}_2 \not\vdash \Delta
\]

Similar to the \(\times \text{Inj}_E\) case, but simpler.

**Case**

\[
\Gamma(\hat{\alpha}_2:*; \hat{\alpha}_1:*; \hat{\alpha}:* = \hat{\alpha}_1 \rightarrow \hat{\alpha}_2) \vdash e \iff \hat{\alpha}_1 \not\in \Theta \\
\Theta \vdash e_2 \iff [\Theta] \hat{\alpha}_2 \not\vdash \Delta
\]

Similar to the \(\times \text{Inj}_E\) and \(\times \text{Spine}\) cases, except that (because we’re in the spine part of the lemma) we have to show that 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.
Lemma 73 (Substitution Isn't Large).
For all contexts \( \Theta \), we have \( \#\text{large}(\Theta|\Lambda) = \#\text{large}(\Lambda) \).

Proof. By induction on \( \Lambda \), following the definition of substitution.

Lemma 74 (Instantiation Solves).
If \( \Gamma \vdash \alpha := \tau : \kappa \rightarrow \Delta \) and \( |\Gamma|\tau = \tau \) and \( \alpha \not\in \text{FV}(|\Gamma|\tau) \) then \( |\text{unsolved}(\Gamma)| = |\text{unsolved}(\Delta)| + 1 \).

Proof. By induction on the given derivation.

- Case
  \[ \Gamma \vdash e : \Lambda \rightarrow \Theta \quad \Theta \vdash \Pi :: \Lambda q \equiv [\Theta]C \, p \rightarrow \Delta \quad \Pi \vdash [\Delta|A \, \text{covers } \Delta] \]
  \[ \Gamma \vdash \text{case}(e, \Pi) \equiv C \, p \rightarrow \Delta \]

  Use the i.h. and Lemma 68 (Transitivity of Separation).

H’ Decidability of Algorithmic Subtyping

H’.1 Lemmas for Decidability of Subtyping

Lemma 75 (Checkeq Solving).

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

Proof. By induction on the given derivation.

- **Case**
  \[
  \Gamma \vdash u \equiv u : \kappa \vdash \Delta
  \]

  Here \( \Delta = \Gamma \).

- **Cases**
  \[
  \checkeq\text{Var}, \checkeq\text{Unit}, \checkeq\text{Zero} \quad \text{Similar to the } \checkeq\text{Var} \text{ case.}
  \]

- **Case**
  \[
  \Gamma \vdash \sigma \equiv t : \mathbb{N} \vdash \Delta
  \]

  \[
  \Gamma \vdash \text{succ}(\sigma) \equiv \text{succ}(t) : \mathbb{N} \vdash \Delta
  \]

  Follows by i.h.

- **Case**
  \[
  \Gamma_0[\checkeq] \vdash \checkeq : t : \kappa \vdash \Delta
  \]

  \[
  \checkeq \notin \text{FV}(t)
  \]

  \[
  \Gamma_0[\checkeq] \vdash \checkeq : t : \kappa \vdash \Delta
  \]

  Subderivation

  \[
  \Delta = \Gamma \text{ or } |\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| - 1 \quad \text{By Lemma 74 (Instantiation Solves)}
  \]

- **Case**
  \[
  \Gamma[\checkeq : \kappa] \vdash \checkeq : t : \kappa \vdash \Delta
  \]

  Similar to the \checkeq\text{InstL} case.

- **Case**
  \[
  \Gamma \vdash \sigma_1 \equiv \tau_1 : * \vdash \Theta
  \]

  \[
  \Theta \vdash [\Theta] \sigma_2 \equiv [\Theta] \tau_2 : * \vdash \Delta
  \]

  \[
  \Gamma \vdash \sigma_1 + \sigma_2 \equiv \tau_1 + \tau_2 : * \vdash \Delta
  \]

  \[
  \checkeq\text{Bin}
  \]

  \[
  \Gamma \vdash \sigma_1 \equiv \tau_1 : * \vdash \Theta
  \]

  Subderivation

  \[
  \Theta = \Gamma \text{ or } |\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)| \quad \text{By i.h.}
  \]

  - \( \Theta = \Gamma; \)

  \[
  \Theta \vdash [\Theta] \sigma_2 \equiv [\Theta] \tau_2 : * \vdash \Delta
  \]

  Subderivation

  \[
  \Gamma \vdash [\Gamma] \sigma_2 \equiv [\Gamma] \tau_2 : * \vdash \Delta
  \]

  \[
  \Theta = \Gamma \quad \text{By } \Theta = \Gamma
  \]

  \[
  \Delta = \Gamma \text{ or } |\text{unsolved}(\Gamma)| = |\text{unsolved}(\Delta)| + 1 \quad \text{By i.h.}
  \]

  - \(|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|; \)

  \[
  \Theta \vdash [\Theta] \sigma_2 \equiv [\Theta] \tau_2 : * \vdash \Delta
  \]

  Subderivation

  \[
  \Delta = \Theta \text{ or } |\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)| \quad \text{By i.h.}
  \]

  If \( \Delta = \Theta \) then substituting \( \Delta \) for \( \Theta \) in \(|\text{unsolved}(\Theta)| < |\text{unsolved}(\Delta)| \) gives \(|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|. \)

  If \(|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)| \) then transitivity of \( < \) gives \(|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|. \)

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

Proof. Only one rule can derive the judgment:

- Case

  $\Gamma \vdash \sigma_1 \Downarrow t_1 : N \Downarrow \Theta \quad \Theta \vdash [\Theta]\sigma_2 \Downarrow [\Theta]t_2 : N \Downarrow \Delta$

  $\Gamma \vdash (\sigma_1 = \sigma_2) \equiv (t_1 = t_2) \Downarrow \Delta \quad \equiv\text{PropEq}$

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 $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$. 

\]

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

  $\Gamma \vdash \alpha \equiv \alpha \Downarrow \Gamma \quad \equiv\text{Var}$

  Here $\Delta = \Gamma$.

- Cases $\equiv\text{Exvar}$, $\equiv\text{Unit}$: Similar to the $\equiv\text{Var}$ case.

- Case

  $\Gamma \vdash A_1 \equiv B_1 \Downarrow \Theta \quad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \Downarrow \Delta$

  $\Gamma \vdash (A_1 \equiv A_2) \equiv (B_1 \equiv B_2) \Downarrow \Delta \quad \equiv\text{eq}$

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 $\equiv\text{Vec}$: Similar to the $\equiv\text{eq}$ case.

- Case

  $\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \Downarrow \Delta, \alpha : \kappa, \Delta'$

  $\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \Downarrow \Delta \quad \equiv\text{V} $

By i.h., either $(\Delta, \alpha : \kappa, \Delta') = (\Gamma, \alpha : \kappa)$, or $|\text{unsolved}(\Delta, \alpha : \kappa, \Delta')| < |\text{unsolved}(\Gamma, \alpha : \kappa)|$.

In the former case, Lemma 22 (Extension Inversion) (i) tells us that $\Delta' = \kappa$. Thus, $(\Delta, \alpha : \kappa) = (\Gamma, \alpha : \kappa)$, and so $\Delta = \Gamma$.

In the latter case, we have $|\text{unsolved}(\Delta, \alpha : \kappa, \Delta')| < |\text{unsolved}(\Gamma, \alpha : \kappa)|$, that is:

$|\text{unsolved}(\Delta)| + 0 + |\text{unsolved}(\Delta')| + 0 < |\text{unsolved}(\Gamma)| + 0$

Since $|\text{unsolved}(\Delta')|$ cannot be negative, we have $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

---

Proof of Lemma 77 (Equiv Solving) lem:equiv-solving
Proof of Lemma 77 (Equiv Solving) lem:equiv-solving

• Case

\[
\Gamma \vdash P \equiv Q \Theta \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta \Rightarrow
\]

\[
\Gamma \vdash A_0 \Rightarrow P \equiv Q \vdash B_0 \vdash \Delta \Rightarrow
\]

Similar to the \( \equiv \Theta \) case, but using Lemma 76 (Prop Equiv Solving) on the first premise instead of the i.h.

• Case

\[
\Gamma \vdash P \equiv Q \Theta \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta \Rightarrow
\]

\[
\Gamma \vdash A_0 \wedge P \equiv B_0 \wedge Q \vdash \Delta \Rightarrow
\]

Similar to the \( \equiv \wedge \) case.

• Case

\[
\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} : \tau : \vdash \Delta \quad \hat{\alpha} \notin FV(\tau) \Rightarrow
\]

\[
\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} \equiv \tau : \vdash \Delta \Rightarrow
\]

\[
\text{ InstantiateL}
\]

By Lemma 74 (Instantiation Solves), \(|\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| - 1\).

• Case

\[
\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} : \tau : \vdash \Delta \quad \hat{\alpha} \notin FV(\tau) \Rightarrow
\]

\[
\Gamma_0[\hat{\alpha}] \vdash \tau \equiv \hat{\alpha} : \vdash \Delta \Rightarrow
\]

\[
\text{ InstantiateR}
\]

Similar to the \( \equiv \text{ InstantiateL} \) case.

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) \( \Gamma \vdash \sigma \equiv t : \kappa : \vdash \Delta \)

(2) \( \Gamma \vdash P \text{ true : } \vdash \Delta \)

(3) \( \Gamma \vdash P \equiv Q : \vdash \Delta \)

(4) \( \Gamma / \sigma \equiv t : \kappa : \vdash \Delta^\perp \)

(5) \( \Gamma / P : \vdash \Delta^\perp \)

Proof. Since there is no mutual recursion between the judgments, we can prove their decidability in order, separately.

(1) Decidability of \( \Gamma \vdash \sigma \equiv t : \kappa : \vdash \Delta \): By induction on the sizes of \( \sigma \) and \( t \).

• Cases \( \text{ CheckeqVar } \text{ CheckeqUnit } \text{ CheckeqZero } \) No premises.

• Case \( \text{ CheckeqSucc } \) Both \( \sigma \) and \( t \) get smaller in the premise.

• Cases \( \text{ CheckeqInstL } \text{ CheckeqInstR } \) Follows from Lemma 67 (Decidability of Instantiation).

(2) Decidability of \( \Gamma \vdash P \text{ true : } \vdash \Delta \): 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, \( \equiv \text{PropEq} \) which has two premises of the form (1), so decidability follows from part (1).

(4) Decidability of \( \Gamma / \sigma \equiv t : \kappa : \vdash \Delta^\perp \): 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** [ElimeqUvarRef], [ElimeqZero]: No premises, and the output is the same as the input.

• **Case** [ElimeqClash]: The only premise is the clash judgment, which is clearly decidable. There is no output.

• **Case** [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, $[\Theta]\tau'_2 = [\Gamma]\tau'_2$, 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 $\Gamma$.

• **Case** [ElimeqBinBot]: The premise is invoked on subterms, and does not yield an output context.

• **Case** [ElimeqSucc]: Both $\sigma$ and $t$ get smaller. By i.h., the output context has fewer unsolved variables, if it exists.

• **Cases** [ElimeqInstL], [ElimeqInstR]: Follows from Lemma 67 (Decidability of Instantiation). Furthermore, by Lemma 74 (Instantiation Solves), instantiation solves a variable in the output.

• **Cases** [ElimeqUvarL], [ElimeqUvarR]: These rules have no nontrivial premises, and $\alpha$ is solved in the output context.

• **Cases** [ElimeqUvarL], [ElimeqUvarR]: These rules have no nontrivial premises, and produce the output context $\perp$.

(5) **Decidability of** $\Gamma / P \vdash \Delta^\perp$: 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 $\Gamma \vdash A$ type and $\Gamma \vdash B$ type and $[\Gamma]A = A$ and $[\Gamma]B = B$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash A \equiv B \vdash \Delta$.

**Proof.** Let the judgment $\Gamma \vdash A \equiv B \vdash \Delta$ 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** $\Gamma \vdash A_1 \equiv B_1 \vdash \Theta$, $\Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta$

  \[ \Gamma \vdash A_1 \uplus A_2 \equiv B_1 \uplus B_2 \vdash \Delta \]

  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 $\equiv\uplus$ where two of the types are monotypes.
• Case \( \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta' \)
\[
\frac{\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \vdash \Delta}{\Delta'}
\]

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
\[
\frac{\Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta}{\Gamma \vdash P \supset A_0 \equiv Q \supset B_0 \vdash \Delta}
\]

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.

• Case
\[
\frac{\Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta}{\Gamma \vdash A_0 \wedge P \equiv B_0 \wedge Q \vdash \Delta}
\]

Similar to the \( \supset \) case.

• Case
\[
\frac{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} := \tau : \star \vdash \Delta \quad \hat{\alpha} \notin FV(\tau)}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \equiv \tau \vdash \Delta}
\]

Follows from Lemma 67 (Decidability of Instantiation).

• Case \( \equiv \text{Instantiatel} \) Similar to the \( \equiv \text{Instantiatel} \) case.

\section*{H'.2 Decidability of Subtyping}

\textbf{Theorem 1 (Decidability of Subtyping).}

\textit{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 \( \Gamma \vdash A \ll < : P B \vdash \Delta \).}

\textbf{Proof.} Let the judgments be measured lexicographically by \( \#\text{large}(A) + \#\text{large}(B) \).

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 \( \Gamma \vdash A \ll < : P B \vdash \Delta \).

• Case \( A \) not headed by \( \forall / \exists \)
\[
\frac{B \text{ not headed by } \forall / \exists}{\Gamma \vdash B \equiv B \vdash \Delta}
\]

In this case, we appeal to Lemma 79 (Decidability of Equivalence).

• Case \( B \) not headed by \( \forall \)
\[
\frac{\Gamma, \varpi \vdash \hat{\alpha} : \kappa \vdash [\hat{\alpha} / \alpha]A \ll : B \vdash \Delta, \varpi, \Theta}{\Gamma \vdash \forall \alpha : \kappa. A \ll : B \vdash \Delta}
\]

The premise has one fewer quantifier.
• Case \( \Gamma, \beta : \kappa \vdash A <: - B \vdash \Delta, \beta : \kappa, \Theta \):

\[
\Gamma \vdash A <: - \forall \beta : \kappa. B \vdash \Delta.
\]

\(<:\forall R\) 

The premise has one fewer quantifier.

• Case \( \Gamma, \alpha : \kappa \vdash A <: + B \vdash \Delta, \alpha : \kappa, \Theta \):

\[
\Gamma \vdash \exists \alpha : \kappa. A <: + B \vdash \Delta \]

\(<:\exists L\)

The premise has one fewer quantifier.

• Case \( A \) not headed by \( \exists \)

\[
\Gamma, \triangleright \beta, \triangleright \beta : \kappa \vdash A <: - B' \vdash \Delta, \beta : \kappa, \Theta
\]

\[
\Gamma \vdash A <: - \exists \beta : \kappa. B \vdash \Delta
\]

\(<:\exists R\)

The premise has one fewer quantifier.

• Case

\[
\Gamma \vdash A <: - B \vdash \Delta
\]

\(\text{neg}(A)\) \(\text{nonpos}(B)\)

\[
\Gamma \vdash A <: + B \vdash \Delta
\]

\(\text{neg}(B)\) \(\text{nonpos}(B)\)

\(<:\exists L\)

Consider whether \( B \) is negative.

– Case \( \text{neg}(B)\):

\[
B = \forall \beta : \kappa. B'
\]

\[
\Gamma, \beta : \kappa \vdash A <: - B' \vdash \Delta, \beta : \kappa, \Theta
\]

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 \( \forall \).

\[
A = \forall \alpha : \kappa. A'
\]

\[
\Gamma, \triangleright \alpha, \triangleright \alpha : \kappa \vdash [\alpha/\alpha]A' <: - \vdash \Delta, \triangleright \alpha, \Theta
\]

Inversion on the premise

There is one fewer quantifier in the subderivation.

• Case

\[
\Gamma \vdash A <: - B \vdash \Delta
\]

\(\text{neg}(A)\) \(\text{nonpos}(B)\)

\[
\Gamma \vdash A <: + B \vdash \Delta
\]

\(\text{neg}(B)\) \(\text{nonpos}(A)\)

\(<:\exists R\)

\[
B = \forall \beta : \kappa. B'
\]

\[
\Gamma, \beta : \kappa \vdash A <: - B' \vdash \Delta, \beta : \kappa, \Theta
\]

Inversion on the premise

There is one fewer quantifier in the subderivation.

• Case

\[
\Gamma \vdash A <: + B \vdash \Delta
\]

\(\text{pos}(A)\) \(\text{nonneg}(B)\)

\[
\Gamma \vdash A <: - B \vdash \Delta
\]

\(\text{pos}(B)\) \(\text{nonneg}(A)\)

\(<:\exists L\)

This case is similar to the \( <:\exists R\) case.
Proof of Theorem 1 (Decidability of Subtyping)

\[ \Gamma \vdash A <: B \rightarrow \Delta \quad \text{nonneg}(A) \]
\[ \vdash B \rightarrow \Delta \quad \text{pos}(B) \]
\[ \Gamma \vdash A <: B \rightarrow \Delta \quad \text{\textless ; R} \]

This case is similar to the \[ \lll \text{ case.} \]

\[ \Box \]

H’3 Decidability of Matching and Coverage

Lemma 80 (Decidability of Guardedness Judgment).
For any set of branches \( \Pi \), the relation \( \Pi \) guarded is decidable.

Proof. This follows via a routine induction on \( \Pi \), counting the number of branch lists. \[ \Box \]

Lemma 81 (Decidability of Expansion Judgments).
Given branches \( \Pi \), it is decidable whether:

1. there exists a unique \( \Pi' \) such that \( \Pi \xrightarrow{\text{var}} \Pi' \);
2. there exist unique \( \Pi_L \) and \( \Pi_R \) such that \( \Pi \xrightarrow{\text{\lll}} \Pi_L \parallel \Pi_R \);
3. there exists a unique \( \Pi' \) such that \( \Pi \xrightarrow{\text{\llr}} \Pi' \).
4. there exists a unique \( \Pi' \) such that \( \Pi \xrightarrow{\text{\lll}} \Pi' \).
5. there exist unique \( \Pi_{(j)} \) and \( \Pi_{..} \) such that \( \Pi \xrightarrow{\text{\llr}} \Pi_{(j)} \parallel \Pi_{..} \).

Proof. In each part, by induction on \( \Pi \): Every rule either has no premises, or breaks down \( \Pi \) in its nontrivial premise. \[ \Box \]

Lemma 82 (Expansion Shrinks Size).
We define the size of a pattern \( |p| \) as follows:

\[
\begin{align*}
|x| &= 0 \\
|\_| &= 0 \\
|\langle p, p' \rangle| &= 1 + |p| + |p'| \\
|()| &= 0 \\
|\text{inj}_1 p| &= 1 + |p| \\
|\text{inj}_2 p| &= 1 + |p| \\
|\lll| &= 1 \\
|p \xrightarrow{p'}| &= 1 + |p| + |p'| \\
\end{align*}
\]

We lift size to branches \( \pi = p \Rightarrow e \) as follows:

\[ |p_1, \ldots, p_n \Rightarrow e| = |p_1| + \ldots + |p_n| \]

We lift size to branch lists \( \Pi = \pi_1 \ldots \pi_n \) as follows:

\[ |\pi_1 \ldots \pi_n| = |\pi_1| + \ldots + |\pi_n| \]

Now, the following properties hold:

1. If \( \Pi \xrightarrow{\text{var}} \Pi' \) then \( |\Pi| = |\Pi'| \).
2. If \( \Pi \xrightarrow{\text{\llr}} \Pi' \) then \( |\Pi| = |\Pi'| \).
3. If \( \Pi \xrightarrow{\text{\lll}} \Pi' \) then \( |\Pi| \leq |\Pi'| \).
4. If \( \Pi \overset{\rightarrow}{\rightarrow} \Pi_1 \| \Pi_2 \) then \( |\Pi_1| \leq |\Pi| \) and \( |\Pi| \leq |\Pi_2| \).

5. If \( \Pi \overset{\text{Vec}}{\rightarrow} \Pi_1 \| \Pi_2 \) then \( |\Pi_1| \leq |\Pi| \) and \( |\Pi_2| \leq |\Pi| \).

6. If \( \Pi \) guarded and \( \Pi \overset{\text{Vec}}{\rightarrow} \Pi_1 \| \Pi_2 \) then \( |\Pi_1| < |\Pi| \) and \( |\Pi_2| < |\Pi| \).

**Proof.** Properties 1-5 follow by a routine induction on the derivation of the expansion judgement. Since expansion only ever removes pattern constructors, and only ever adds wildcards, it never increases the size of the resulting branch list.

Case 6 for vectors proceeds by induction on the derivation of \( \Pi \) guarded, and furthermore depends upon the proof for case 5.

1. Case

\[
\begin{align*}
& \Gamma, p \Rightarrow e \mid \Pi \text{ guarded} \\
& \text{By inversion on the expansion derivation, we know } \Pi \overset{\text{Vec}}{\rightarrow} \Pi_1 \| \Pi_2. \\
& \text{By part 5, we know that } |\Pi_1| \leq |\Pi| \text{ and } |\Pi_2| \leq |\Pi|. \\
& \text{By the definition of size, we know that } |p, p', \vec{p} \Rightarrow e| < |\Pi_1, \vec{p} \Rightarrow e|. \\
& \text{Hence } |p, p', \vec{p} \Rightarrow e| < |\Pi_1, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi. \\
& \text{By the definition of size, we know that } |\Pi_1| < |\Pi_2, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi. \\
& \text{Hence } |\Pi| < |\Pi_1, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi.
\end{align*}
\]

2. Case

\[
\begin{align*}
& \emptyset, p : p', \vec{p} \Rightarrow e \mid \Pi \text{ guarded} \\
& \text{By inversion on the expansion derivation, we know } \Pi \overset{\text{Vec}}{\rightarrow} \Pi_1 \| \Pi_2. \\
& \text{By part 5, we know that } |\Pi_1| \leq |\Pi| \text{ and } |\Pi_2| \leq |\Pi|. \\
& \text{By the definition of size, we know that } |p : p', \vec{p} \Rightarrow e| < |\Pi_1, \vec{p} \Rightarrow e|. \\
& \text{Hence } |p : p', \vec{p} \Rightarrow e| < |\Pi_1, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi. \\
& \text{By the definition of size, we know that } |\Pi_1| < |\Pi_2, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi. \\
& \text{Hence } |\Pi| < |\Pi_1, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi.
\end{align*}
\]

3. Case

\[
\begin{align*}
& \Pi \text{ guarded} \\
& \emptyset, \vec{p} \Rightarrow e \mid \Pi \text{ guarded} \\
& \text{By inversion on the expansion derivation, we know } \Pi \overset{\text{Vec}}{\rightarrow} \Pi_1 \| \Pi_2. \\
& \text{By induction, } |\Pi_1| < |\Pi| \text{ and } |\Pi_2| < |\Pi|. \\
& \text{By the definition of size, } |\emptyset, \vec{p} \Rightarrow e| < |\Pi_1, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi. \\
& \text{By the definition of size, } |\Pi_1, \vec{p} \Rightarrow e| < |\Pi_2, \vec{p} \Rightarrow e| \Rightarrow e \mid \Pi.
\end{align*}
\]

4. Case

\[
\begin{align*}
& \Pi \text{ guarded} \\
& x, \vec{p} \Rightarrow e \mid \Pi \text{ guarded} \\
& \text{Similar to previous case.}
\end{align*}
\]

**Theorem 2 (Decidability of Coverage).**
Given a context \( \Gamma \), branches \( \Pi \) and types \( \vec{A} \), it is decidable whether \( \Gamma \vdash \Pi \text{ covers } \vec{A} q \) is derivable.
Proof. By induction on, lexicographically, (1) the size $|\Pi|$ of the branch list $\Pi$ and then (2) the number of $\land$ connectives in $\mathcal{A}$, and then (3) the size of $\mathcal{A}$, considered to be the sum of the sizes $|A|$ of each type $A$ in $\mathcal{A}$ (treating constraints $s=t$ as size 1).

(For $\text{CoversVar}$, $\text{Covers} \times$, $\text{CoversVec}$, $\text{CoversVec}'$, and $\text{Covers} \vdash$, we also use the appropriate part of Lemma 81 [Decidability of Expansion Judgments], as well as Lemma 82 [Expansion Shrinks Size].)

- **Case $\text{CoversEmpty}$**: No premises.
- **Case $\text{CoversVar}$**: The number of $\land$ connectives does not grow, and the size of the branch list stays the same, and $\mathcal{A}$ gets smaller.
- **Case $\text{Covers1}$**: The number of $\land$ connectives and the size of the branch list stays the same, and $\mathcal{A}$ gets smaller.
- **Case $\text{Covers} \land$**: The size of the branch list stays the same, and the number of $\land$ connectives in $\mathcal{A}$ goes down. This lets us analyze the two possibilities for the coverage-with-assumptions judgement:
  - **Case $\text{CoversEq}$**: The first premise is decidable by Lemma 78 [Decidability of Propositional Judgments] (4). The number of $\land$ connectives in $\mathcal{A}$ gets smaller (note that applying $\Delta$ as a substitution cannot add $\land$ connectives).
  - **Case $\text{CoversEqBot}$**: The premise is decidable by Lemma 78 [Decidability of Propositional Judgments] (4).
- **Case $\text{Covers} \land \bot$**: The size of the branch list stays the same, and the number of $\land$ connectives in $\mathcal{A}$ goes down.
- **Case $\text{Covers} \times$**: The size of the branch list does not grow, the number of $\land$ connectives stays the same, and $\mathcal{A}$ gets smaller, since $|A_1| + |A_2| < |A_1 \times A_2|$.
- **Case $\text{Covers} \vdash$**: Here we have $\mathcal{A} = (A_1 + A_2, \vec{B})$. In the first premise, we have $(A_1, \vec{B})$, which is smaller than $\mathcal{A}$, and in the second premise we have $(A_2, \vec{B})$, which is likewise smaller. (In both premises, the size of the branch list does not grow, and the number of $\land$ connectives stays the same.)
- **Case $\text{CoversVec}$**: Since $\Pi$ guarded is decidable, and $\Pi \lessdot \Pi_1 || \Pi_2$ is decidable, then we know that the size of the branch lists $\Pi_1$ and $\Pi_2$ is strictly smaller than $\Pi$.

This lets us analyze the two cases for each premise, noting that the assumption is decidable by Lemma 78 [Decidability of Propositional Judgments] (4).

  - **Case $\text{CoversEq}$**: The first premise (that $t=0$) is decidable by Lemma 78 [Decidability of Propositional Judgments] (4). The size of $\Pi_1$ is strictly smaller than $\Pi$'s size, so we can still appeal to induction (note $\Delta$ as a substitution cannot add change the size of a branch list).
  - **Case $\text{CoversEqBot}$**: Decidable by Lemma 78 [Decidability of Propositional Judgments] (4).

The cons case is nearly identical:

  - **Case $\text{CoversEq}$**: The first premise (that $t = \text{succ}(n)$) is decidable by Lemma 78 [Decidability of Propositional Judgments] (4). The size of $\Pi_1$ is strictly smaller than $\Pi$'s size, so we can still appeal to induction (note $\Delta$ as a substitution cannot add change the size of a branch list).
  - **Case $\text{CoversEqBot}$**: Decidable by Lemma 78 [Decidability of Propositional Judgments] (4).

- **Case $\text{CoversVec}$**: Since $\Pi$ guarded is decidable, and $\Pi \lessdot \Pi_1 || \Pi_2$ is decidable, then we know that the size of the branch lists $\Pi_1$ and $\Pi_2$ is strictly smaller than $\Pi$.
  - **Case $\text{Covers} \vdash$**: The size of the branch list does not grow, and $\mathcal{A}$ gets smaller.
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
    \( \Gamma \vdash s : B \vdash \Delta \).

(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 \( \Gamma \vdash \Pi :: \vec{A} \vdash C \vdash \Delta \).

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

Proof. For rules deriving judgments of the form

\[ \Gamma \vdash e \Rightarrow \Box \,
    \Gamma \vdash e \Leftarrow B \vdash \Box \,
    \Gamma \vdash s : B \vdash \Box \,
    \Gamma \vdash \Pi :: \vec{A} \vdash C \vdash \Box \]

(where we write “\( \Box \)” for parts of the judgments that are outputs), the following induction measure on such judgments is adequate to prove decidability:

\[ \langle e/s/\Pi, \Rightarrow , \Leftarrow / \Rightarrow , \#\text{large}(B), B \,
    \text{Match}, \vec{A}, \text{match judgment form} \rangle \]

where \( \langle \ldots \rangle \) 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,

\[ \Rightarrow \prec \Leftarrow / \Rightarrow / \text{Match} \]

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 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 EmptySpine: No premises.
Proof of Theorem 3 (Decidability of Typing)

- Case $\rightarrow$Spine: In each premise, the expression/spine gets smaller (we have $e$ and $s$ in the conclusion, $e$ in the first premise, and $s$ in the second premise).

- Case $\Var$: No nontrivial premises.

- Case $\Sub$: 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 $\Anno$: It is easy to show that the judgment $\Gamma \vdash A \uparrow$ type is decidable. The second premise types $e$, but the conclusion types $(e : A)$, so the first part of the measure gets smaller.

- Cases $\exists \alpha, \exists \alpha$, $\forall \alpha$: No premises.

- Case $\forall$: Both the premise and conclusion type $e$, and both are checking; however, $\#large(A_0) < \#large(\forall \alpha : \kappa. A_0)$, so the premise is smaller.

- Case $\forall$Spine: Both the premise and conclusion type $e$ and $s$, and both are spine judgments; however, $\#large(\rightarrow)$ decreases.

- Case $\And$: By Lemma 78 (Decidability of Propositional Judgments) (2), the first premise is decidable. For the second premise, $\#large(\Theta_A) = \#large(A_0) < \#large(A_0 \wedge P)$.

- Case $\Or$: Both the premise and conclusion type $e$, and both are checking; however, $\#large(\rightarrow)$ decreases so the premise is smaller.

- Case $\Or$: For the first premise, use Lemma 78 (Decidability of Propositional Judgments) (5). In the second premise, $\#large(\rightarrow)$ gets smaller (similar to the $\And$ case).

- Case $\And$: The premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (5).

- Case $\For$: Similar to the $\And$ case.

- Cases $\rightarrow \rightarrow \rightarrow \rightarrow$: In the premise, the term is smaller.

- Cases $\rightarrow \rightarrow \rightarrow$: In all premises, the term is smaller.

- Cases $\rightarrow \rightarrow \rightarrow \rightarrow$: In all premises, the term is smaller.

- Case $\Case$: 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 $\Match \rightarrow \Match \times \Match \rightarrow \Match \rightarrow \Match \rightarrow \\MatchNeg \MatchWild$: Smaller by part (2) of the measure.

- Case $\Match \wedge$: 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 $\Match \top$: For the premise, use Lemma 78 (Decidability of Propositional Judgments) (4).

- Case $\MatchUnify$: Lemma 78 (Decidability of Propositional Judgments) (4) shows that the first premise is decidable. The second premise has the same (single) branch and list of types, but is smaller by the “match judgment form” part of the measure.
I’ Determinacy

Lemma 83 (Determinacy of Auxiliary Judgments).

1. Elimeq: Given $\Gamma$, $\sigma$, $t$, $\kappa$ such that $\text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset$ and $D_1 :: \Gamma / \sigma \triangleright t : \kappa \vdash \Delta_1^{\perp}$ and $D_2 :: \Gamma / \sigma \triangleright t : \kappa \vdash \Delta_2^{\perp}$,
it is the case that $\Delta_1^{\perp} = \Delta_2^{\perp}$.

2. Instantiation: Given $\Gamma$, $\dot{\alpha}$, $\tau$, $t$, $\kappa$ such that $\dot{\alpha} \in \text{unsolved}(\Gamma)$ and $\Gamma \vdash t : \kappa$ and $\dot{\alpha} \notin \text{FV}(t)$ and $D_1 :: \Gamma \vdash \dot{\alpha} := t : \kappa \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \dot{\alpha} := t : \kappa \vdash \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

3. Symmetric instantiation:
   Given $\Gamma$, $\dot{\alpha}$, $\dot{\beta}$, $\kappa$ such that $\dot{\alpha}, \dot{\beta} \in \text{unsolved}(\Gamma)$ and $\dot{\alpha} \neq \dot{\beta}$
   and $D_1 :: \Gamma \vdash \dot{\alpha} := \dot{\beta} : \kappa \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \dot{\beta} := \dot{\alpha} : \kappa \vdash \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

4. Checkeq: Given $\Gamma$, $\sigma$, $t$, $\kappa$ such that $D_1 :: \Gamma \vdash \sigma \triangleright t : \kappa \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \sigma \triangleright t : \kappa \vdash \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

5. Elimprop: Given $\Gamma$, $P$ such that $D_1 :: \Gamma \vdash P \vdash \Delta_1$ and $D_2 :: \Gamma \vdash P \vdash \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

6. Checkprop: Given $\Gamma$, $P$ such that $D_1 :: \Gamma \vdash P \triangleright \Delta_1$ and $D_2 :: \Gamma \vdash P \triangleright \Delta_2$,
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 ElimeqUvarRefl applies if and only if $t = \alpha$. (Rule ElimeqClash cannot apply; rules ElimeqUvarL and ElimeqUvarR have a free variable condition; rules ElimeqUvarLR and ElimeqUvarRL have a condition that $\sigma \neq t$.)
In the remainder, assume $t \neq \alpha$.

- If $\alpha \in \text{FV}(t)$, then rule ElimeqUvarL applies, and no other rule applies (including ElimeqUvarR and ElimeqClash).
In the remainder, assume $\alpha \notin \text{FV}(t)$.

- Consider whether ElimeqUvarR applies. The conclusion matches if we have $t = \beta$ for some $\beta \neq \alpha$ (that is, $\sigma = \alpha$ and $t = \beta$). But ElimeqUvarR has a condition that $\beta \in \text{FV}(\sigma)$, and $\sigma = \alpha$, so the condition is not satisfied.

In the symmetric case, use the reasoning above, exchanging L’s and R’s in the rule names.

Proof of Part (2) (Instantiation).
Rule InstBin applies if and only if $t$ has the form $t_1 \oplus t_2$.
Rule InstZero applies if and only if $t$ has the form zero.
Rule InstSucc applies if and only if $t$ has the form succ$(t_0)$.
If $t$ has the form $\beta$, then consider whether $\beta$ is declared to the left of $\dot{\alpha}$ in the given context:

- If $\beta$ is declared to the left of $\dot{\alpha}$, then rule InstReach cannot be used, which leaves only InstSolve

- If $\beta$ is declared to the right of $\dot{\alpha}$, then InstSolve cannot be used because $\beta$ is not well-formed under $\Gamma_0$ (the context to the left of $\dot{\alpha}$ in InstSolve). That leaves only InstReach

- $\dot{\alpha}$ cannot be $\beta$, because it is given that $\dot{\alpha} \notin \text{FV}(t) = \text{FV}(\beta) = \{\beta\}$. 

Proof of Lemma 83 (Determinacy of Auxiliary Judgments) lem:aux-det
Proof of Part (3) (Symmetric instantiation).

Rule \texttt{CheckeqInstL} and \texttt{CheckeqInstR} cannot have been used in either derivation.

Suppose that \texttt{InstSolve} concluded \( \Delta_1 \). Then \( \Delta_1 \) is the same as \( \Gamma \) with \( \hat{\alpha} \) solved to \( \hat{\beta} \). Moreover, \( \hat{\beta} \) is declared to the left of \( \hat{\alpha} \) in \( \Gamma \). Thus, \texttt{InstSolve} cannot conclude \( \Delta_2 \). However, \texttt{InstReach} can conclude \( \Delta_2 \), but produces a context \( \Delta_2 \) which is the same as \( \Gamma \) but with \( \hat{\beta} \) solved to \( \hat{\alpha} \). Therefore \( \Delta_1 = \Delta_2 \).

The other possibility is that \texttt{InstReach} concluded \( \Delta_1 \). Then \( \Delta_1 \) is the same as \( \Gamma \) with \( \hat{\beta} \) solved to \( \hat{\alpha} \), with \( \hat{\alpha} \) declared to the left of \( \hat{\beta} \) in \( \Gamma \). Thus, \texttt{InstReach} cannot conclude \( \Delta_2 \). However, \texttt{InstSolve} can conclude \( \Delta_2 \), producing a context \( \Delta_2 \) which is the same as \( \Gamma \) but with \( \hat{\alpha} \) solved to \( \hat{\beta} \). Therefore \( \Delta_1 = \Delta_2 \).

Proof of Part (4) (Checkeq).

Rule \texttt{CheckeqVar} applies if and only if \( \sigma = t = \hat{\alpha} \) or \( \sigma = t = \alpha \) (note the free variable conditions in \texttt{CheckeqInstL} and \texttt{CheckeqInstR}).

- Rule \texttt{CheckeqUnit} applies if and only if \( \sigma = t = 1 \).
- Rule \texttt{CheckeqBin} applies if and only if \( \sigma \) and \( t \) are both headed by the same binary connective.
- Rule \texttt{CheckeqZero} applies if and only if \( \sigma = t = 0 \).
- Rule \texttt{CheckeqSucc} applies if and only if \( \sigma \) and \( t \) are headed by \texttt{succ}.

Now suppose \( \sigma = \hat{\alpha} \). If \( t \) is not an existential variable, then \texttt{CheckeqInstR} cannot be used, which leaves only \texttt{CheckeqInstL}. If \( t \) is an existential variable, that is, some \( \hat{\beta} \) (distinct from \( \hat{\alpha} \), and is unsolved, then both \texttt{CheckeqInstL} and \texttt{CheckeqInstR} apply, but by part (3), we get the same output context from each.

The \( t = \alpha \) subcase is similar.

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 84 (Determinacy of Equivalence).

1. Propositional equivalence: Given \( \Gamma, P, Q \) such that \( \Delta_1 :: \Gamma \vdash P \equiv Q \vdash \Delta_1 \) and \( \Delta_2 :: \Gamma \vdash P \equiv Q \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

2. Type equivalence: Given \( \Gamma, A, B \) such that \( \Delta_1 :: \Gamma \vdash A \equiv B \vdash \Delta_1 \) and \( \Delta_2 :: \Gamma \vdash A \equiv B \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

Proof of Part (1) (propositional equivalence). Only one rule derives judgments of this form; the result follows from Lemma 83 (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 \texttt{Exvar} applies (due to the free variable conditions in \texttt{InstantiateL} and \texttt{InstantiateR}).

If \( A \) and \( B \) are different unsolved existential variables, the judgment matches the conclusion of both \texttt{InstantiateL} and \texttt{InstantiateR} but by part (3) of Lemma 83 (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 \( \Delta_1 :: \Gamma \vdash A :^P B \vdash \Delta_1 \) and \( \Delta_2 :: \Gamma \vdash A :^P B \vdash \Delta_2 \), 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 $\Gamma \vdash A <_{\text{\small{\oplus}}} B \vdash \Delta_1$ and $\Gamma \vdash A <_{\text{\small{\ominus}}} B \vdash \Delta_2$, then we know the last rule ending the derivation of $D_1$ and $D_2$ must be:

$$
\begin{array}{c|c|c|c}
\forall & \exists & \text{other} \\
\hline
\forall & <_{\ominus} R & <_{\ominus} L & <_{\ominus} L \\
\exists & <_{\oplus} R & <_{\oplus} L & <_{\ominus} L \\
\text{other} & <_{\ominus} R & <_{\ominus} R & \text{Equiv} \\
\end{array}
$$

The only case in which there are two possible final rules is in the $\forall/\exists$ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A <_{\text{\small{\ominus}}} B \vdash \Delta_1$ and $\Gamma \vdash A <_{\text{\small{\oplus}}} B \vdash \Delta_2$.

• If $\Gamma \vdash A <_{\text{\small{\ominus}}} B \vdash \Delta_1$ and $\Gamma \vdash A <_{\text{\small{\ominus}}} B \vdash \Delta_2$, then we know the last rule ending the derivation of $D_1$ and $D_2$ must be:

$$
\begin{array}{c|c|c|c}
\forall & \exists & \text{other} \\
\hline
\forall & <_{\forall} R & <_{\forall} L & <_{\forall} L \\
\exists & <_{\exists} R & <_{\exists} L & <_{\ominus} R \\
\text{other} & <_{\ominus} R & <_{\ominus} R & \text{Equiv} \\
\end{array}
$$

The only case in which there are two possible final rules is in the $\forall/\exists$ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A <_{\text{\small{\ominus}}} B \vdash \Delta_1$ and $\Gamma \vdash A <_{\text{\small{\oplus}}} B \vdash \Delta_2$.

As a result, the result follows by a routine induction.

\[\square\]

**Theorem 5** (Determinacy of Typing).

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

2. Synthesis: Given $\Gamma, e$ such that $D_1 :: \Gamma \vdash e \Rightarrow B_1 \vdash p_1 \vdash \Delta_1$ and $D_2 :: \Gamma \vdash e \Rightarrow B_2 \vdash p_2 \vdash \Delta_2$, 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 $D_1 :: \Gamma \vdash e : A \vdash C_1 \vdash q_1 \vdash \Delta_1$ and $D_2 :: \Gamma \vdash e : A \vdash C_2 \vdash q_2 \vdash \Delta_2$, 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 $\Gamma \vdash e : A \vdash C_k [q_k] \vdash \Delta_k$.

4. Match judgments:

   Given $\Gamma, \Pi, \bar{A}, p, C$ such that $D_1 :: \Gamma \vdash \Pi \vdash \bar{A} \vdash C_1 \vdash p \vdash \Delta_1$ and $D_2 :: \Gamma \vdash \Pi \vdash \bar{A} \vdash C_2 \vdash p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

   Given $\Gamma, \Pi, \bar{A}, p, C$

   such that $D_1 :: \Gamma / P \vdash \Pi / \bar{A} \vdash C \vdash p \vdash \Delta_1$ and $D_2 :: \Gamma / P \vdash \Pi / \bar{A} \vdash C \vdash p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

\[\square\]

Proof of Part (1) (checking).

The rules with a checking judgment in the conclusion are: $\set{\text{thm:subtyping-det}}$

The table below shows which rules apply for given $e$ and $A$. The extra “chk-I$\Gamma$” column highlights the role of the “chk-I$\Gamma$” (“check-intro”) category of syntactic forms: we restrict the introduction rules for $\forall$ and $\exists$ to
type only these forms. For example, given \( e = x \) and \( A = (\forall \alpha : \kappa. A_0) \), we need not choose between \( \text{Sub} \) and \( \text{\neg\neg} \) the latter is ruled out by its \( \text{chk-I} \) premise.

\[
\begin{array}{cccccccccccc}
\lambda x. e_0 & \text{chk-I} & \forall & \exists & \land & \to & + & \times & 1 & \hat{\alpha} & \alpha & \text{Vec} \\
\text{rec } x. v & \text{Note 2} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} & \text{Rec} \\\n\text{inj}_k e_0 & \text{chk-I} & \lor & \land & \top & \bot & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} \\
\langle e_1, e_2 \rangle & \text{chk-I} & \lor & \land & \top & \bot & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} \\
(\_ ) & \text{chk-I} & \lor & \land & \top & \bot & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} \\
e & \text{[] } & \text{chk-I} & \lor & \land & \top & \bot & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} \\
e_1 : e_2 & \text{chk-I} & \lor & \land & \top & \bot & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} & \text{I} \\
\text{case}(e_0, \Pi) & \text{Note 3} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} \\
x & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} \\
(e_0 : A) & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} \\
e_1 s & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} \\
\end{array}
\]

Notes:

- **Note 1**: The choice between \( \exists \) and \( \neg\neg \) is resolved by Lemma 83 (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: \( \text{Sub} \), \( \text{\neg\neg} \), \( \rightarrow \), and \( \rightarrow \). 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 \( \rightarrow \).

**Proof of Part (3) (spine judgments)**. For the ordinary spine judgment, rule \( \text{EmptySpine} \) applies if and only if the given spine is empty. Otherwise, the choice of rule is determined by the head constructor of the input type: \( \rightarrow \text{Spine} \lor \neg\neg \text{Spine} \lor \lor \text{Spine} \lor \land \text{Spine} \).

For the principality-recovering spine judgment: If \( p = f \), only rule \( \text{SpinePass} \) applies. If \( p = l \) and \( q = l \), only rule \( \text{SpinePass} \) applies. If \( p = l \) and \( q = f \), then the result is determined by \( \text{FEV}(C) \): if \( \text{FEV}(C) = \emptyset \) then only \( \text{SpinePass} \) applies; otherwise, \( \text{FEV}(C) \neq \emptyset \) and only \( \text{SpinePass} \) applies.

**Proof of Part (4) (matching)**. First, the elimination judgment form \( \Gamma / P \vdash \ldots \) cannot be the case that both \( \Gamma / \sigma \vdash t : \kappa \vdash \bot \) and \( \Gamma / \sigma \vdash t : \kappa \vdash \top \), so either \( \text{Match} \) concludes both \( D_1 \) and \( D_2 \) (and the result follows), or \( \text{MatchUnify} \) concludes both \( D_1 \) and \( D_2 \) (in which case, apply the i.h.).

Now the main judgment form, without "/ \( P \)"; 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. So in the rest of this part, we assume \( \Pi \) has length one.

Moreover, \( \text{MatchBase} \) applies if and only if \( \hat{A} \) has length zero. So in the rest of this part, we assume the length of \( \hat{A} \) is at least one.
Let $A$ be the first type in $\vec{A}$. 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 ("∅") can apply, as shown in the following table:

$$
\begin{array}{cccccc}
\top & \wedge & + & \times & \text{Vec} & \text{other} \\
\begin{array}{l}
\text{inj}_k \rho_0 \\
\rho \left( \rho_1, \rho_2 \right) \\
z \\
\emptyset \\
\rho_1 :: \rho_2
\end{array} & \begin{array}{l}
\text{Match} \top \\
\text{Match} \wedge \\
\text{Match} + \\
\text{Match} \times \\
\text{Match}\vec{A}
\end{array} & \begin{array}{l}
\text{Match}\wedge \\
\text{Match} \top \\
\text{Match} + \\
\text{Match} \times \\
\text{Match}\vec{A}
\end{array} & \begin{array}{l}
\emptyset \\
\emptyset \\
\emptyset \\
\emptyset \\
\emptyset
\end{array} & \begin{array}{l}
\emptyset \\
\emptyset \\
\emptyset \\
\emptyset \\
\emptyset
\end{array} & \begin{array}{l}
\emptyset \\
\emptyset \\
\emptyset \\
\emptyset \\
\emptyset
\end{array}
\end{array}
$$

## J’ Soundness

### J’.1 Instantiation

**Lemma 85 (Soundness of Instantiation).**

If $\Gamma \vdash \star : \kappa \vdash \Delta$ and $\star \not\in \text{FV}(\Gamma)\tau$ and $[\Gamma]\tau = \tau$ and $\Delta \rightarrow \Omega$ then $[\Omega] \hat{\star} = [\Omega] \tau$.

**Proof.** By induction on the derivation of $\Gamma \vdash \star : \kappa \vdash \Delta$.

- **Case** $\Gamma_0 \vdash \star : \kappa$

  \[
  \frac{
  \Gamma_0, \hat{\star} : \kappa, \Gamma_1 \vdash \star := \kappa \vdash \Gamma_0, \hat{\star} : \kappa = \Gamma_1
  }{
  \Gamma, \hat{\star} : \kappa \vdash \star := \kappa
  } \text{InstSolve}
  \]

  $[\Delta] \hat{\star} = [\Delta] \tau$ By definition

  $[\Omega] \hat{\star} = [\Omega] \tau$ By Lemma 29 (Substitution Monotonicity) to each side

- **Case** $\hat{\beta} \in \text{unsolved}(\Gamma[\hat{\star} : \kappa],[\hat{\beta} : \kappa])$

  \[
  \frac{
  \Gamma[\hat{\star} : \kappa],[\hat{\beta} : \kappa] \vdash \star := \hat{\beta} : \kappa \vdash \Gamma[\hat{\star} : \kappa],[\hat{\beta} : \kappa]
  }{
  \Gamma[\hat{\star} : \kappa],[\hat{\beta} : \kappa] \vdash \star := \hat{\beta} : \kappa \vdash \Delta
  } \text{InstReach}
  \]

  $[\Delta] \hat{\beta} = [\Delta] \hat{\star}$ By definition

  $[\Omega] [\Delta] \hat{\beta} = [\Omega] [\Delta] \hat{\star}$ Applying $\Omega$ to each side

  $[\Omega] [\Delta] \hat{\beta} = [\Omega] [\Delta] \hat{\star}$ By Lemma 29 (Substitution Monotonicity) to each side

- **Case** $\Gamma'$

  \[
  \frac{
  \Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \star := \hat{\alpha}_1 : \star \vdash \Theta \vdash \star := \Theta[\tau_2 : \star \vdash \Delta
  }{
  \Gamma_0[\hat{\alpha} : \star] \vdash \star := \hat{\alpha}_1 \oplus \tau_2 : \star \vdash \Delta
  } \text{InstBin}
  \]
Proof of Lemma 85 (Soundness of Instantiation)

Proof.

\[\Delta \rightarrow \Omega\]
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)

\[[\Omega] \hat{\alpha}_1 = [\Omega] \tau_1\]
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.

\[[\Omega] \hat{\alpha} := [\Omega] \tau_1 \otimes [\Omega] \tau_2\]
By Lemma 29 (Substitution Monotonicity)

\[( [\Omega] \tau_1 ) \otimes ( [\Omega] \tau_2 ) = ( [\Omega] \hat{\alpha}_1 ) \otimes ( [\Omega] \hat{\alpha}_2 )\]
By above equalities

\[[\Omega] ( [\hat{\alpha}_1 \otimes \hat{\alpha}_2 ] )\]
By definition of substitution

\[[\Omega] ( [\hat{\alpha}_1 \otimes \hat{\alpha}_2 ] ) = [\Omega] \hat{\alpha}\]
By definition of substitution

- Case

\[\Gamma_0[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{zero } : N \vdash \Gamma_0[\hat{\alpha} : N = \text{zero}]\]
InstZero

Similar to the InstSolve case.

- Case

\[\Gamma_0[\hat{\alpha}_1 : N, \hat{\alpha} : N = \text{succ } (\hat{\alpha}_1 )] \vdash \hat{\alpha}_1 := \tau_1 : N \vdash \Delta\]
InstSucc

\[\Gamma_0[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{succ } (\tau_1 ) : N \vdash \Delta\]
Similar to the InstBin case, but simpler.

\[\Box\]

Lemma 86 (Soundness of Checkeq).
If \(\Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta\) where \(\Delta \rightarrow \Omega\) then \([\Omega] \sigma = [\Omega] t\).

Proof. By induction on the given derivation.

- Case

\[\Gamma \vdash u \equiv u : \kappa \vdash \Gamma\]
CheckeqVar

\[[\Omega] u = [\Omega] u\]
By reflexivity of equality

- Cases CheckeqUnit, CheckeqZero

Similar to the CheckeqVar case.

- Case

\[\Gamma \vdash \sigma_0 \equiv t_0 : N \vdash \Delta\]
CheckeqSucc

\[\Gamma \vdash \text{succ } (\sigma_0 ) \equiv \text{succ } (t_0 ) : N \vdash \Delta\]
Subderivation

\[[\Omega] \sigma_0 = [\Omega] t_0\]
By i.h.

\[\text{succ } ([\Omega] \sigma_0 ) = \text{succ } ([\Omega] t_0 )\]
By congruence

\[[\Omega] (\text{succ } (\sigma_0 ) ) = [\Omega] (\text{succ } (t_0 ) )\]
By definition of substitution
Proof of \textbf{Lemma 86} (Soundness of \textsc{Checkeq})

\begin{align*}
\text{Case} & \quad \Gamma \vdash \sigma_0 \triangleq t_0 : \star \rightarrow \Theta \quad \Theta \vdash [\Theta] \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \\
& \quad \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \tag{\text{CheckeqBin}}
\end{align*}

\begin{align*}
\Gamma \vdash \sigma_0 \triangleq t_0 : \mathbb{N} \rightarrow \Delta \\
\Theta \vdash [\Theta] \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \\
\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)} \\
[\Omega] \sigma_0 = [\Omega] t_0 & \quad \text{By i.h. on first subderivation} \\
[\Omega] [\Theta] \sigma_1 = [\Omega] [\Theta] t_1 & \quad \text{By i.h. on second subderivation} \\
[\Omega] [\Theta] \sigma_1 = [\Omega] \sigma_1 & \quad \text{By Lemma 29 (Substitution Monotonicity)} \\
[\Omega] \sigma_1 = [\Omega] t_1 & \quad \text{By Lemma 29 (Substitution Monotonicity)} \\
[\Omega] \sigma_0 \oplus [\Omega] \sigma_1 = [\Omega] (t_0 \oplus t_1) & \quad \text{By transitivity of equality} \\
\text{\texttt{\textit{\#}}} & \quad [\Omega] ([\sigma_0 \oplus \sigma_1]) = [\Omega] ([t_0 \oplus t_1]) & \quad \text{By definition of substitution}
\end{align*}

\begin{align*}
\text{Case} & \quad \Gamma[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta \quad \hat{\alpha} \notin \text{FV}(t) \\
& \quad \Gamma[\hat{\alpha}] \vdash \hat{\alpha} \triangleq t : \kappa \rightarrow \Delta \tag{\text{CheckeqInstL}}
\end{align*}

\begin{align*}
\Gamma[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta & \quad \text{Subderivation} \\
\hat{\alpha} \notin \text{FV}(t) & \quad \text{Premise} \\
\text{\texttt{\textit{\#}}} & \quad [\Omega][\hat{\alpha} \triangleq [\Omega] t] & \quad \text{By Lemma 85 (Soundness of Instantiation)}
\end{align*}

\begin{align*}
\text{Case} & \quad \Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \sigma : \kappa \rightarrow \Delta \quad \hat{\alpha} \notin \text{FV}(t) \\
& \quad \Gamma[\hat{\alpha} : \kappa] \vdash \sigma \triangleq \hat{\alpha} : \kappa \rightarrow \Delta \tag{\text{CheckeqInstR}}
\end{align*}

\begin{align*}
\sim & \quad \text{Similar to the CheckeqInstL case.} & \quad \Box
\end{align*}

\textbf{Lemma 87} (Soundness of Propositional Equivalence).
\begin{align*}
\text{If} \quad \Gamma \vdash P \equiv Q \rightarrow \Delta \quad \text{where} \quad \Delta \rightarrow \Omega \text{ then} \quad [\Omega] P = [\Omega] Q.
\end{align*}

\textbf{Proof.} \quad \text{By induction on the given derivation.}

\begin{align*}
\text{Case} & \quad \Gamma \vdash \sigma_1 \triangleq t_1 : \mathbb{N} \rightarrow \Theta \quad \Theta \vdash [\Theta] \sigma_2 \triangleq [\Theta] t_2 : \mathbb{N} \rightarrow \Delta \\
& \quad \Gamma \vdash (\sigma_1 = \sigma_2) \equiv (t_1 = t_2) \rightarrow \Delta \tag{\text{PropEq}}
\end{align*}

\begin{align*}
\Delta & \rightarrow \Omega & \quad \text{Given} \\
\Theta & \rightarrow \Delta & \quad \text{By Lemma 46 (Checkeq Extension) (on 2nd premise)} \\
\Theta & \rightarrow \Omega & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma \vdash \sigma_1 \triangleq t_1 : \mathbb{N} \rightarrow \Theta & \quad \text{Given} \\
[\Omega] \sigma_1 = [\Omega] t_1 & \quad \text{By Lemma 86 (Soundness of Checkeq)} \\
\Theta \vdash [\Theta] \sigma_2 \triangleq [\Theta] t_2 & \quad \text{By Lemma 29 (Substitution Monotonicity)} \\
[\Omega] [\Theta] \sigma_2 = [\Omega] [\Theta] t_2 & \quad \text{By Lemma 29 (Substitution Monotonicity)} \\
[\Omega] \sigma_2 = [\Omega] t_2 & \quad \text{By transitivity of equality} \\
([\Omega] \sigma_1 = [\Omega] \sigma_2) = ([\Omega] t_1 = [\Omega] t_2) & \quad \text{By congruence of equality} \\
\text{\texttt{\textit{\#}}} & \quad [\Omega] (\sigma_1 = \sigma_2) = [\Omega] (t_1 = t_2) & \quad \text{By definition of substitution} & \quad \Box
\end{align*}
Lemma 88 (Soundness of Algorithmic Equivalence).
If $\Gamma \vdash A \equiv B \vdash \Delta$ where $\Delta \rightarrow \Omega$ then $[\Omega]A = [\Omega]B$.

Proof. By induction on the given derivation.

- Case $\Gamma \vdash \alpha \equiv \alpha \vdash \Gamma$ 
  $\equiv \text{Var}$
  $[\Omega]\alpha = [\Omega]\alpha$ By reflexivity of equality

- Cases $\equiv \text{Exvar}, \equiv \text{Unit}$ Similar to the $\equiv \text{Var}$ case.

- Case $\Gamma \vdash A_1 \equiv B_1 \vdash \Theta$ $\Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta$
  $\Gamma \vdash A_1 \oplus A_2 \equiv B_1 \oplus B_2 \vdash \Delta$
  $\Delta \rightarrow \Omega$ Given
  $\Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta$ Subderivation
  $\Theta \rightarrow \Delta$ By Lemma 49 (Equivalence Extension)
  $\Theta \rightarrow \Omega$ By Lemma 33 (Extension Transitivity)
  $[\Omega]A_1 = [\Omega]B_1$ By i.h.
  $\Delta \rightarrow \Omega$ Given
  $[\Omega][\Theta]A_2 = [\Omega][\Theta]B_2$ By i.h.
  $[\Omega]A_2 = [\Omega]B_2$ By Lemma 29 (Substitution Monotonicity)
  $\equiv [\Omega]A_1 \oplus [\Omega]A_2 = ([\Omega]B_1) \oplus ([\Omega]B_2)$ By above equations

- Case $\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta'$ $\equiv \wedge$
  $\Gamma \vdash \forall \alpha : \kappa. \alpha \equiv \forall \alpha : \kappa. B_0 \vdash \Delta$ $\equiv \forall$
  $\Gamma, \alpha : \kappa, \alpha \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta'$ Subderivation
  $\Delta \rightarrow \Omega$ Given
  $\Gamma, \alpha : \kappa, \alpha \rightarrow \Delta, \alpha : \kappa, \Delta'$ By Lemma 49 (Equivalence Extension)
Proof of Lemma 88 (Soundness of Algorithmic Equivalence)

\[
\Gamma \vdash P \equiv Q \implies \Theta \vdash (\Theta)A_0 \equiv (\Theta)B_0 \vdash \Delta
\]

\[
\Delta \rightarrow \Omega
\]

By definition of substitution, since \(FV(A_0) \cap \text{dom}(\Omega_Z) = \emptyset\)

\[
\forall \alpha : \kappa, [\Omega]A_0 = [\Omega]B_0
\]

Adding quantifier to each side

\[
[\Omega](\forall \alpha : \kappa. A_0) = [\Omega](\forall \alpha : \kappa. B_0)
\]

By definition of substitution

\[
\Delta \text{ soft}
\]

Since \(\cdot\) is soft

\[
\Delta', \alpha : \kappa, \Delta' \rightarrow \Omega, \alpha : \kappa, \Omega \_
\]

By Lemma 24 (Soft Extension)

\[
\Gamma, \alpha : \kappa \vdash A_0 \text{ type}
\]

By validity on subderivation

\[
\Gamma, \alpha : \kappa \vdash B_0 \text{ type}
\]

By validity on subderivation

\[
FV(A_0) \subseteq \text{dom}(\Gamma, \alpha : \kappa)
\]

By well-typing of \(A_0\)

\[
FV(B_0) \subseteq \text{dom}(\Gamma, \alpha : \kappa)
\]

By well-typing of \(B_0\)

\[
\Gamma, \alpha : \kappa \rightarrow \Omega, \alpha : \kappa
\]

By \(\text{Uvar}\)

\[
FV(A_0) \subseteq \text{dom}(\Omega, \alpha : \kappa)
\]

By Lemma 20 (Declaration Order Preservation)

\[
FV(B_0) \subseteq \text{dom}(\Omega, \alpha : \kappa)
\]

By Lemma 20 (Declaration Order Preservation)

\[
[\Omega, \alpha : \kappa, \Omega_Z]A_0 = [\Omega, \alpha : \kappa]A_0
\]

By definition of substitution, since \(FV(A_0) \cap \text{dom}(\Omega_Z) = \emptyset\)

\[
[\Omega, \alpha : \kappa, \Omega_Z]B_0 = [\Omega, \alpha : \kappa]B_0
\]

By definition of substitution, since \(FV(B_0) \cap \text{dom}(\Omega_Z) = \emptyset\)

\[
[\Omega, \alpha : \kappa]A_0 = [\Omega, \alpha : \kappa]B_0
\]

By transitivity of equality

\[
[\Omega]A_0 = [\Omega]B_0
\]

From definition of substitution

\[
\forall \alpha : \kappa. [\Omega]A_0 = [\forall \alpha : \kappa. [\Omega]B_0
\]

Adding quantifier to each side

\[
[\Omega](\forall \alpha : \kappa. A_0) = [\Omega](\forall \alpha : \kappa. B_0)
\]

By definition of substitution

Case 1

\[
\Gamma \vdash P \equiv Q \implies \Theta \vdash (\Theta)A_0 \equiv (\Theta)B_0 \vdash \Delta
\]

\[
\Delta \rightarrow \Omega
\]

Given

\[
\Theta \vdash (\Theta)A_0 \equiv (\Theta)B_0 \vdash \Delta
\]

Subderivation

\[
\Theta \rightarrow \Delta
\]

By Lemma 49 (Equivalence Extension)

\[
\Theta \rightarrow \Omega
\]

By Lemma 33 (Extension Transitivity)

\[
\Gamma \vdash P \equiv Q \implies \Theta
\]

Subderivation

\[
[\Theta]P = [\Theta]Q
\]

By Lemma 87 (Soundness of Propositional Equivalence)

\[
\Theta \vdash (\Theta)A_0 \equiv (\Theta)B_0 \vdash \Delta
\]

Subderivation

\[
[\Omega](\Theta)A_0 = [\Omega](\Theta)B_0
\]

By i.h.

\[
[\Omega]A_0 = [\Omega]B_0
\]

By Lemma 29 (Substitution Monotonicity)

Case 2

\[
\Gamma \vdash P \equiv Q \implies \Theta \vdash (\Theta)A_0 \equiv (\Theta)B_0 \vdash \Delta
\]

\[
\Delta \rightarrow \Omega
\]

Similar to the \(\equiv\) case.

\[
\Gamma[\hat{\alpha}] \vdash \hat{\alpha} : \tau \vdash \Delta
\]

\[
\hat{\alpha} \notin FV(\tau)
\]

\[
\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \equiv \tau \vdash \Delta
\]

\[
\Gamma[\hat{\alpha}] \vdash \hat{\alpha} : \tau \vdash \Delta
\]

Subderivation

\[
[\Omega]\hat{\alpha} = [\Omega]\tau
\]

By Lemma 85 (Soundness of Instantiation)

Case 3

\[
\Gamma \vdash P \equiv Q \implies \Theta \vdash (\Theta)A_0 \equiv (\Theta)B_0 \vdash \Delta
\]

\[
\Delta \rightarrow \Omega
\]

Similar to the \(\equiv\) case.

\[
J'.2\quad\text{Soundness of Checkprop}
\]

Lemma 89 (Soundness of Checkprop).

\(\Gamma \vdash P \text{ true } \vdash \Delta \text{ and } \Delta \rightarrow \Omega \text{ then } \Psi \vdash [\Omega]P \text{ true.}\)
Proof of Lemma 89 (Soundness of Checkprop)

lem:checkprop-soundness

Proof. By induction on the derivation of $\Gamma \vdash P \text{ true } \dashv \Delta$.

- Case $\Gamma \vdash \sigma \triangleq t : N \vdash \Delta$ [CheckpropEq]
  
  $\Gamma \vdash \sigma \triangleq t \text{ true } \dashv \Delta$

  Subderivation
  
  $|\Omega| \sigma = |\Omega| t$

  By Lemma 86 (Soundness of Checkeq)
  
  $\Psi \vdash |\Omega| \sigma = |\Omega| t$ true

  By DeclCheckpropEq
  
  $\Psi \vdash |\Omega| (\sigma = t)$ true

  By def. of subst.

  $\Psi \vdash |\Omega| P$ true

  By $P = (\sigma = t)$

J.3 Soundness of Eliminations (Equality and Proposition)

Lemma 90 (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 $\Gamma \rightarrow \Theta$

   and all $\Gamma'$ such that $\Gamma \vdash \Gamma' : \kappa'$,

   it is the case that $[\Omega, \Theta] \Gamma' = [\emptyset][\Theta] \Gamma'$, where $\emptyset = \text{mgu}(\sigma, t)$.

2. If $\Gamma / \sigma \triangleq \bot$ 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 $\Gamma / \sigma \triangleq t : \kappa \vdash \Delta \downarrow$, proving both parts with a single induction.

- Case $\Gamma / \alpha \triangleq \alpha : \kappa \vdash \Gamma$ [ElimeqUvarRefl]

  Here we have $\Delta = \Gamma$, so we are in part (1).

  Let $\emptyset = \text{id}$ (which is $\text{mgu}(\sigma, \sigma)$).

  We can easily show $[\text{id})][\emptyset] \alpha = [\emptyset, \alpha] = [\emptyset, \cdot]$.

- Case $\Gamma / \text{zero} \triangleq \text{zero} : N \vdash \Gamma$ [ElimeqZero]

  Similar to the ElimeqUvarRefl case.
• Case \( \Gamma / t_1 \equiv t_2 : N \vdash \Delta \) \hspace{1cm} \text{ElimeqSucc}

\[
\Gamma / \text{succ}(t_1) \equiv \text{succ}(t_2) : N \vdash \Delta
\]

We distinguish two subcases:

– Case \( \Delta = \Delta \):

Since we have the same output context in the conclusion and premise, the “for all \( t' \ldots \)” 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 \( \Delta = \bot \):

\[
\Gamma / t_1 \equiv t_2 : N \vdash \bot
\]

\( \text{Subderivation} \)

\[ mgu(t_1, t_2) = \bot \]

By i.h. (2)

\[ \text{By contrapositive of property (ii)} \]

[Case \( \alpha \notin \text{FV}(t) \) \hspace{1cm} (\alpha = -) \notin \Gamma \]

\[
\Gamma / \alpha \equiv t : \kappa \vdash \Gamma, \alpha = t
\]

\[ \text{ElimeqUvarL} \]

Here \( \Delta \neq \bot \), so we are in part (1).

\[ \left[ \Omega, \alpha = t \right] t' = \left[ \Omega \right] t / \alpha \left[ \Omega \right] t' \]

\[ = \left[ \Omega \right] [t / \alpha] [\Omega] t' \]

\[ = \left[ \Omega \right] [\theta] [\Omega] t' \]

\[ \text{By mgu(\alpha, t) = (\alpha / t)} \]

\[ = \left[ \theta \right] [\Omega] t' \]

\[ \text{By a property of substitution (\theta creates no evars)} \]

• Case \( \alpha \notin \text{FV}(t) \) \hspace{1cm} (\alpha = -) \notin \Gamma \]

\[
\Gamma / t \equiv \alpha : \kappa \vdash \Gamma, \alpha = t
\]

\[ \text{ElimeqUvarR} \]

Similar to the \text{ElimeqUvarL} case.

• Case

\[
\Gamma / 1 \equiv 1 : * \vdash \Gamma
\]

\[ \text{ElimeqUnit} \]

Similar to the \text{ElimeqUvarR} case.

• Case

\[
\Gamma / \tau_1 \equiv \tau_1' : * \vdash \Theta \quad \Theta / [\Theta] \tau_1 \equiv [\Theta] \tau_1' : * \vdash \Delta
\]

\[ \Gamma / \tau_1 \otimes \tau_2 \equiv \tau_1' \otimes \tau_2' : * \vdash \Delta \]

\[ \text{ElimeqBin} \]

Either \( \Delta \) is some \( \Delta \), or it is \( \bot \).

– Case \( \Delta = \Delta \):

[Proof of Lemma 90 (Soundness of Equality Elimination) lem:elimeq-soundness]
Proof of [Lemma 90] (Soundness of Equality Elimination) lem:elimeq-soundness

\[ \Gamma / \tau \vdash \tau' : \ast \vdash \Theta \]
\[ \Theta = (\Gamma, \Delta_1) \]
\[ (\text{IH-1st}) \]
\[ [\Omega, \Delta_1]u_1 = [\theta_1][\Omega]u_1 \]
\[ \theta_1 = \text{mgu}(\tau_1, \tau'_1) \]
\[ \Theta / \theta_1 \vdash [\Theta]\tau_1' : \ast \vdash \Delta_1 \]
\[ \Delta = (\Theta, \Delta_2) \]
\[ (\text{IH-2nd}) \]
\[ [\Omega, \Delta_1, \Delta_2]u_2 = [\theta_2][\Omega, \Delta_1]u_2 \]
\[ \theta_2 = \text{mgu}(\tau_2, \tau'_2) \]

Subderivation

Suppose \( \Omega \vdash u : \kappa' \).
\[ [\Omega, \Delta_1, \Delta_2]u = [\theta_2][\Omega, \Delta_1]u \]
\[ = [\theta_2][\theta_1][\Omega]u \]
\[ \text{By (IH-2nd), with } u_2 = u \]
\[ = [\Omega][\theta_2 \circ \theta_1]u \]
\[ \text{By (IH-1st), with } u_1 = u \]
\[ \theta_2 \circ \theta_1 = \text{mgu}((\tau_1 \oplus \tau_2), (\tau'_1 \oplus \tau'_2)) \]
\[ \text{By property (iii) of substitution} \]

\( \vdash \Delta \perp = \perp \):

Use the i.h. (2) on the second premise to show \( \text{mgu}(\tau_2, \tau'_2) = \perp \), then use property (v) of unification to show \( \text{mgu}((\tau_1 \oplus \tau_2), (\tau'_1 \oplus \tau'_2)) = \perp \).

- Case \( \Delta \perp = \perp \):

  Similar to the \( \perp \) subcase for ElimeqSucc, but using property (v) instead of property (ii).

- Case \( \sigma \# t \):

  Since \( \sigma \# t \), we know \( \sigma \) and \( t \) have different head constructors, and thus no unifier. \( \square \)
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 \(\Gamma \vdash A \leq^P B \vdash \Delta\) then \([\Omega]\Delta \vdash [\Omega]A \leq^P [\Omega]B\).

Proof. By induction on the given derivation.

- **Case** \(B\) not headed by \(\forall\).

Let \(\Omega' = (\Omega, \triangleright_{\mathbf{a}, \Theta})\).

\[
\begin{align*}
\Gamma, ▶_{\mathbf{a}, \Theta} \vdash \kappa \to [\mathbf{a}/\alpha]A_0 \leq^\omega B + \Delta, ▶_{\mathbf{a}, \Theta} & \quad \text{Subderivation} \\
\Gamma \vdash \forall \alpha : \kappa. A_0 \leq^\omega B + \Delta & \quad \text{Given} \\
\end{align*}
\]

By i.h.

- **Case** \(\kappa\) \(\vdash [\mathbf{a}/\alpha]A_0 \leq^\omega B + \Delta, ▶_{\mathbf{a}, \Theta}\)

By def. of substitution

- **Case** \(\kappa \to [\mathbf{a}/\alpha]A_0 \leq^\omega B + \Delta, ▶_{\mathbf{a}, \Theta}\)

By \(\forall\).

- **Case** \(\forall \alpha : \kappa. A_0 \leq^\omega B + \Delta\)

By Lemma \(25\) (Filling Completes)
Proof of Theorem 6 (Soundness of Algorithmic Subtyping)

\( \Gamma, \beta : \kappa \vdash A <: B_0 \rightarrow \Delta, \beta : \kappa, \Theta \)

Subderivation

\( (\Delta, \beta : \kappa, \Theta) \rightarrow \Omega' \)

By Lemma 25 (Filling Completes)

\( \Gamma \vdash A \text{ type} \)

Given

\( \Gamma, \beta : \kappa \vdash \forall \beta : \kappa. B_0 \text{ type} \)

By Lemma 35 (Suffix Weakening) Given

\( \Theta \) is soft

By Lemma 22 (Extension Inversion) (i)

\( \Gamma, \beta : \kappa \vdash \forall \beta : \kappa. B_0 \text{ type} \)

By Lemma 50 (Subtyping Extension)

\( \Theta \) is soft

By Lemma 22 (Extension Inversion) (i)

\( \Gamma \vdash A \leq B_0 \)

By \( \leq \forall \text{R} \)

Similar to the \( \leq \forall \text{L} \) case.

\( \Gamma \vdash A = B \rightarrow \Delta \)

Subderivation

\( \Delta \rightarrow \Omega \)

By Lemma 88 (Soundness of Algorithmic Equivalence)

\( \Gamma, \beta : \kappa \vdash \Delta, \beta : \kappa, \Theta \)

By Lemma 49 (Equivalence Extension)

\( \Gamma \vdash A \text{ type} \)

By Lemma 16 (Substitution for Type Well-Formedness)

\( \Omega, \beta : \kappa \vdash \forall \beta : \kappa. B_0 \text{ type} \)

By Lemma 54 (Completing Stability)

\( [\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega, \beta : \kappa]A \leq [\Omega, \beta : \kappa]B_0 \)

By \( \leq \forall \text{R} \)

\( \Gamma \vdash A <: B \)

By inversion

\( \Gamma \vdash A <: B \rightarrow \Delta \)

By inversion

\( \Gamma \vdash A <: B \)

By inversion

\( \Gamma \vdash A <: B \rightarrow \Delta \)

\( \Gamma \vdash A <: B \)

By \( \leq \text{L} \)

\( \Gamma \vdash A <: B \rightarrow \Delta \)

By \( \leq \text{R} \)

Similar to the \( \leq \text{L} \) case.
Proof of Theorem 6 (Soundness of Algorithmic Subtyping)

\[ \text{thm:subtyping-soundness} \]

1. Case
   \[ \Gamma \vdash A <: B \rightarrow \Delta \quad \text{pos}(A) \]
   \[ \Gamma \vdash A <: B \rightarrow \Delta \quad \text{nonneg}(B) \]
   \[ \Gamma \vdash A <: B \rightarrow \Delta \quad \llhd \]

Similar to the \( \llhd \) case.

2. Case
   \[ \Gamma \vdash A <: B \rightarrow \Delta \quad \text{nonneg}(A) \]
   \[ \Gamma \vdash A <: B \rightarrow \Delta \quad \text{pos}(B) \]
   \[ \Gamma \vdash A <: B \rightarrow \Delta \quad \llhd \]

Similar to the \( \llhd \) case.

\[ \square \]

### 7' Soundness of Typing

**Theorem 7** (Soundness of Match Coverage).

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

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

**Proof.** By mutual induction on the given algorithmic coverage derivation.

1. • Case
   \[ \cdot \Rightarrow e_1 \mid \ldots \mid \cdot \vdash \Gamma \quad \text{CoversEmpty} \]
   \[ [\Omega] \Gamma \vdash \cdot \Rightarrow e_1 \mid \ldots \vdash \cdot \quad \text{By DeclCoversEmpty} \]

   • Cases\( \text{CoversVar} \), \( \text{Covers1} \), \( \text{Covers} \times \), \( \text{Covers} + \), \( \text{Covers} \exists \), \( \text{Covers} \land \), \( \text{CoversVec} \), \( \text{Covers} \land \neg \), \( \text{CoversVec} \neg \)

   Use the i.h. and apply the corresponding declarative rule.

2. • Case
   \[ \Gamma \vdash [\Gamma]\bar{t}_1 \equiv [\Gamma]\bar{t}_2 : \kappa \rightarrow \Delta \quad [\Delta] \Pi \vdash [\Delta] \bar{A} \text{ covers } \Delta \quad \text{CoversEq} \]
   \[ \Gamma \vdash \bar{t}_1 = \bar{t}_2 \vdash \Pi \text{ covers } \bar{A} ! \]
   \[ \Gamma \vdash [\Gamma]\bar{t}_1 \equiv [\Gamma]\bar{t}_2 : \kappa \rightarrow \Delta \quad \text{Subderivation} \]
   \[ [\Delta] \Pi \vdash [\Delta] \bar{A} \quad \text{Subderivation} \]
   \[ [\Omega] [\Delta] \Pi \vdash [\Delta] \bar{A} \quad \text{By i.h.} \]
   \[ \Delta = (\Gamma, \Theta) \quad \text{By Lemma 90 (Soundness of Equality Elimination)} \]
   \[ \text{mgu}(t_1, t_2) = \theta \quad \text{"} \]
   \[ \ldots \quad \text{"} \]
   \[ [\Omega] [\Delta] = [\Theta][\Omega] \quad \text{By Lemma 95 (Substitution Upgrade) (iii)} \]
   \[ [\Delta] \Pi = [\Theta] \Pi \quad \text{By Lemma 95 (Substitution Upgrade) (iv)} \]
   \[ ([\Delta] \bar{A}) = ([\Theta] A_0, [\Theta] \bar{A}) \quad \text{By Lemma 95 (Substitution Upgrade) (i)} \]
   \[ [\Theta][\Omega] [\Delta] \Pi \vdash [\Theta] \Pi \text{ covers } [\Theta] \bar{A} \quad \text{By above equalities} \]
   \[ [\Omega] \Gamma \vdash \bar{t}_1 = \bar{t}_2 \vdash \Pi \text{ covers } \bar{A} \quad \text{By DeclCoversEq} \]
Proof of Theorem 7 (Soundness of Match Coverage)

• Case  \( \Gamma \vdash t_1 \equiv \Gamma t_2 : \kappa \vdash \bot \) 
  \[ \frac{}{\Gamma \vdash t_1 \vdash \Pi \text{ covers } \bar{A} !} \]  
  CoversEqBot

\[ \frac{}{\Gamma \vdash t_1 = t_2 \vdash \Pi \text{ covers } \bar{A}} \]  
  Subderivation

\[ \text{mgu}(\Gamma t_1, \Gamma t_2) = \bot \]  
  By Lemma 90 (Soundness of Equality Elimination) (2)

\[ \frac{}{\text{mgu}(t_1, t_2) = \bot} \]  
  By given equality

\[ \frac{}{[\Omega] \Gamma / t_1 = t_2 \vdash \Pi \text{ covers } \bar{A}} \]  
  By DeclCoversEqBot

Lemma 91 (Well-formedness of Algorithmic Typing).

Given \( \Gamma \) ctx:

(i) If \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \) then \( \Delta \vdash A \ p \) type.

(ii) If \( \Gamma \vdash s : A \ p \gg B \ q \vdash \Delta \) and \( \Gamma \vdash A \ p \) type then \( \Delta \vdash B \ q \) type.

Proof. 1. Suppose \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \):

• Case  \( x : A \ p \) \( \in \) \( \Gamma \) 
  \[ \frac{}{\Gamma \vdash x \Rightarrow [\Gamma]A \ p \vdash \Gamma} \]  
  Var

\( \Gamma = (\Gamma_0, x : A \ p, \Gamma_1) \)  
\( (x : A \ p) \in \Gamma \)
\( \Gamma \vdash A \ p \) type  
  Follows from \( \Gamma \) ctx

• Case  \( \Gamma \vdash A! \) type  
  \( \Gamma \vdash e \Leftarrow [\Gamma]A! \vdash \Delta \) 
  \[ \frac{}{\Gamma \vdash (e : A) \Rightarrow [\Delta]A! \vdash \Delta} \]  
  Anno

\( \Gamma \vdash A! \) type  
  By inversion
\( \Gamma \Rightarrow \Delta \)  
  By Lemma 51 (Typing Extension)
\( \Delta \vdash A! \) type  
  By Lemma 41 (Extension Weakening for Principal Typing)

\( \frac{}{\Delta \vdash [\Delta]A! \) type} \)  
  By Lemma 39 (Principal Agreement) (i)

• Case

\[ \frac{}{\Gamma \vdash e \Rightarrow A \ p \vdash \Theta} \]  
  \( \Theta \vdash s : [\Theta]A \ p \gg B \ q \vdash \Delta \)  
  \( p = ! \) or \( q = ! \)  
  or \( \text{FEV}(\Delta|C) \neq \emptyset \)  
  \[ \frac{}{\Gamma \vdash e \Rightarrow C \ q \vdash \Delta} \]  
  \( \rightarrow \Omega \)

\( \Gamma \vdash e \Rightarrow A \ p \vdash \Theta \)  
  By inversion
\( \Theta \vdash A \ p \) type  
  By induction
\( \Theta \vdash [\Theta]A \ p \) type  
  By Lemma 40 (Right-Hand Subst. for Principal Typing)
\( \Theta \) ctx  
  By implicit assumption
\( \Theta \vdash s : [\Theta]A \ p \gg C \ q \vdash \Delta \)  
  By inversion

\( \frac{}{\Delta \vdash C \ q \type} \)  
  By mutual induction

2. Suppose \( \Gamma \vdash s : A \ p \gg B \ q \vdash \Delta \) and \( \Gamma \vdash A \ p \) type:
Proof of Lemma 91 (Well-formedness of Algorithmic Typing).

• Case

\[ \Gamma \vdash : A \ p \gg A \ p \mid \Gamma \]

\[ \Gamma \vdash A \ p \mid \text{Given} \]

• Case

\[ \frac{\Gamma \vdash e \triangleq A \ p \mid \Theta \quad \Theta \vdash s : [\Theta]B \ p \gg C \ q \mid \Delta}{\Gamma \vdash e \ s : A \rightarrow B \ p \gg C \ q \mid \Delta} \]

\[ \rightarrow \text{Spine} \]

\[ \Gamma \vdash A \rightarrow B \ p \mid \text{Given} \]

\[ \Gamma \vdash B \ p \mid \text{By Lemma 42 (Inversion of Principal Typing)} \]

\[ \Theta \vdash B \ p \mid \text{By Lemma 41 (Extension Weakening for Principal Typing)} \]

\[ \Theta \vdash [\Theta]B \ p \mid \text{By Lemma 40 (Right-Hand Subst. for Principal Typing)} \]

\[ \Delta \vdash C \ q \mid \text{By induction} \]

• Case

\[ \frac{\Gamma, \hat{\alpha} : \kappa \vdash e \ s : [\hat{\alpha}/\alpha]A \gg C \ q \mid \Delta}{\Gamma \vdash e \ s : [\hat{\alpha}/\alpha]A \vgg C \ q \mid \Delta} \]

\[ \rightarrow \text{Spine} \]

\[ \Gamma \vdash \forall \alpha : \kappa. A \ p \gg C \ q \mid \Delta \]

\[ \Gamma \vdash \forall \alpha : \kappa. A \ p \mid \text{Given} \]

\[ \Gamma \vdash \forall \alpha : \kappa. A \ p \mid \text{By inversion} \]

\[ \Gamma, \alpha : \kappa \vdash A \ p \mid \text{By inversion} \]

\[ \Gamma, \hat{\alpha} : \kappa, \alpha : \kappa \vdash A \ p \mid \text{By weakening} \]

\[ \Gamma, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A \ p \mid \text{By substitution} \]

\[ \Delta \vdash C \ q \mid \text{By induction} \]

• Case

\[ \frac{\Gamma \vdash P \ true \triangleq \Theta \quad \Theta \vdash e \ s : [\Theta]A \ p \gg C \ q \mid \Delta}{\Gamma \vdash e \ s : P \true \ A \ p \gg C \ q \mid \Delta} \]

\[ \rightarrow \text{Spine} \]

\[ \Gamma \vdash P \ p \mid \text{By Lemma 42 (Inversion of Principal Typing)} \]

\[ \Gamma \vdash P \ prop \mid \text{"} \]

\[ \Gamma \rightarrow \Theta \mid \text{By Lemma 47 (Checkprop Extension)} \]

\[ \Theta \vdash A \ p \mid \text{By Lemma 41 (Extension Weakening for Principal Typing)} \]

\[ \Theta \vdash [\Theta]A \ p \mid \text{By Lemma 40 (Right-Hand Subst. for Principal Typing)} \]

\[ \Delta \vdash C \ q \mid \text{By induction} \]

• Case

\[ \frac{\Theta \vdash \hat{\alpha} \rightarrow \hat{\alpha} : \star \vdash e \ s : (\hat{\alpha} \rightarrow \hat{\alpha}) \gg C \mid \Delta}{\Theta \vdash \hat{\alpha} \rightarrow \hat{\alpha} \mid \text{By rules} \}

\[ \Delta \vdash C \ q \mid \text{By induction} \]

Theorem 8 (Eagerness of Types).

(i) If \( D \) derives \( \Gamma \vdash e \triangleq A \ p \mid \Delta \) and \( \Gamma \vdash A \ p \mid \text{type} \) and \( A = [\Gamma]A \) then \( D \) is eager.
(ii) If $D$ derives $\Gamma \vdash e : A \rightarrow \Delta$ then $D$ is eager.

(iii) If $D$ derives $\Gamma \vdash s : A \rightarrow B q \rightarrow \Delta$ and $\Gamma \vdash A \text{ type and } A = [\Gamma]A$ then $D$ is eager.

(iv) If $D$ derives $\Gamma \vdash s : A \rightarrow B [q] \rightarrow \Delta$ and $\Gamma \vdash A \text{ type and } A = [\Gamma]A$ then $D$ is eager.

(v) If $D$ derives $\Gamma \vdash \Pi : \vec{A} q \rightarrow \Delta$ and $\vec{A} q \text{ types and } [\Gamma]\vec{A} = \vec{A}$ and $\Gamma \vdash C p \rightarrow \Delta$ then $D$ is eager.

(vi) If $D$ derives $\Gamma \vdash \Pi : \vec{A} ! \rightarrow \Delta$ and $\vec{A} \text{ type}$ then $D$ is eager.

Proof. By induction on the given derivation.

Part (i), checking

- Case $\text{Rec}$: By i.h. (i).
- Case $\text{Sub}$: By i.h. (ii) and (i).
- Case $\text{∀I}$: By i.h. (ii).
- Case $\text{∧I}$: Substitution is idempotent, so in the last premise $[\Theta][\Theta]A_0 = [\Theta]A_0$ and we can use the i.h. (i).
- Case $\text{∃I}$: Similar to the $\text{∀I}$ case.
- Case $\text{⇒I}$: This rule has no subderivations of the relevant form, so the case is trivial.
- Case $\text{⊃I}$: By i.h. (i).
- Case $\text{→I}^\alpha$: In the premise, $[\Gamma_0]^\alpha_1, ^\alpha_2, ^\alpha = ^\alpha_1 \rightarrow ^\alpha_2, x : ^\alpha_1$ so we can use the i.h. (i).
- Case $\text{+I}_k$: By i.h. (i).
- Case $\text{+I}_k^\alpha$: Similar to the $\text{→I}^\alpha$ case.
- Case $\text{×I}$: By i.h. (i) on the first subderivation, then i.h. (i) on the second subderivation (using the fact that $[\Theta][\Theta]A_2 = [\Theta]A_2$).
- Case $\text{×I}^\alpha$: Similar to the $\text{→I}^\alpha$ case.
- Case $\text{Nil}$: This rule has no subderivations of the relevant form, so the case is trivial.
- Case $\text{Cons}$: By i.h. (i) on the subderivations typing $e_1$ and $e_2$, using $[\Gamma'][\Gamma']A_0 = [\Gamma']A_0$ and $[\Theta][\Theta]([\text{Vec } \alpha A_0]) = [\Theta]([\text{Vec } \alpha A_0])$.
- Case $\text{Case}$: Subderivation $[\Theta]B q \leftarrow [\Theta]A p \rightarrow \Delta$

By Definition 8, the given derivation is eager.

Proof of Theorem 8 (Eagerness of Types)
Part (ii), synthesis

- **Case** \( \text{Var} \)  
  Substitution is idempotent: \( [\Gamma][\Gamma] A_0 = [\Gamma] A_0 \).
  
  By inversion, \( \Delta = \Gamma \) and \( A = [\Gamma] A_0 \) where \( (x : A_0) \in \Gamma \).
  
  Using the above equations, we have
  
  \[
  [\Gamma][\Gamma] A_0 = [\Gamma] A_0 \\
  [\Gamma] A = A \\
  [\Delta] A = A
  \]
  
  This rule has no subderivations, so there is nothing else to show.

- **Case** \( \text{Anno} \)  
  By inversion, \( A = [\Delta] A_0 \).
  
  Substitution is idempotent, so \( [\Gamma][\Gamma] A_0 = [\Gamma] A_0 \) and we can use the i.h. (i) to show that the checking subderivation is eager.
  
  The type in the conclusion is \( [\Delta] A_0 \), which by idempotence is equal to \( [\Delta][\Delta] A_0 \). Since \( A = [\Delta] A_0 \), we have \( A = [\Delta] A \).

- **Case**
  
  \[
  \Gamma \vdash e : B \rightarrow \Theta \\
  \Theta \vdash s : B \rightarrow \nabla A \rightarrow \Delta
  \]
  
  \[
  \Gamma \vdash e \rightarrow \Delta \\
  \rightarrow E
  \]
  
  \( D_1 :: \Gamma \vdash e : B \rightarrow \Theta \)
  
  Subderivation
  
  \( B = [\Theta] B \) and \( D_1 \) eager
  
  By i.h. (ii) on \( D_1 \)
  
  \( D_2 :: \Theta \vdash s : B \rightarrow \nabla A \rightarrow \Delta \)
  
  Subderivation
  
  \( B = [\Theta] B \)
  
  Above
  
  \( A = [\Theta] A \) and \( D_2 \) eager
  
  By i.h. (iv) on \( D_2 \)
  
  \( \not\in \Theta \)
  
  \( D_1 \) eager
  
  Above
  
  \( \not\in \Theta \)
  
  \( D_2 \) eager
  
  Above

Parts (iii) and (iv), spines

- **Case**
  
  \[
  \Gamma, \hat{\alpha} : \kappa \vdash e : s_0 : [\hat{\alpha}/\alpha] A_0 \rightarrow \nabla C \rightarrow \Delta
  \]
  
  \[
  \Gamma \vdash e : [\hat{\alpha}/\alpha] A_0 \rightarrow \nabla C \rightarrow \Delta
  \]
  
  \( \nabla \text{Spine} \)
  
  It is given that \( [\Gamma](\forall \alpha : \kappa. A_0) = (\forall \alpha : \kappa. A_0) \).
  
  Therefore, \( [\Gamma] A_0 = A_0 \).
  
  Since \( \hat{\alpha} \) is not solved in \( \Gamma, \hat{\alpha} : \kappa \), we also have
  
  \[
  [\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha] A_0 = [\hat{\alpha}/\alpha] A_0
  \]
  
  By i.h., \( C = [\Delta] C \) and all subderivations are eager. Since the output type and output context of the conclusion are \( C \) and \( \Delta \), the same as the premise, we have \( C = [\Delta] C \).

- **Case**
  
  \[
  \Gamma \vdash P : \text{true} \rightarrow \Theta \\
  \Theta \vdash e : s_0 : [\Theta] A_0 \rightarrow \nabla C \rightarrow \Delta
  \]
  
  \[
  \Gamma \vdash e : [\Theta] A_0 \rightarrow \nabla C \rightarrow \Delta
  \]
  
  \( \nabla \text{Spine} \)
  
  Substitution is idempotent, so \( [\Theta][\Theta] A_0 = [\Theta] A_0 \), and we can apply the i.h. showing \( C = [\Delta] C \) and that all subderivations are eager. Since the output type and output context of the conclusion are \( C \) and \( \Delta \), the same as the premise, we have \( C = [\Delta] C \).
Proof of Theorem 9 (Soundness of Algorithmic Typing).

Given $\Delta \rightarrow \Omega$:

(i) If $\Gamma \vdash e \triangleleft A \ p \triangleleft A$ and $\Gamma \vdash A \ p$ type and $A = [\Gamma]A$, then $[\Omega]A \vdash [\Omega]e \triangleleft [\Omega]A \ p$.

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

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

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

(v) If $\Gamma \vdash \Pi : \mathcal{A} \ q \triangleleft C \ p \triangleleft C$ and $\Gamma \vdash \mathcal{A} \ ! \ types$ and $[\Gamma]\mathcal{A} = \mathcal{A}$ and $\Gamma \vdash C \ p$ type then $\mathcal{P} \vdash [\Omega]\Delta : [\Omega]\Pi : [\Omega]C$.

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

Proof of Theorem 8 (Eagerness of Types)
Proof of Theorem 9 (Soundness of Algorithmic Typing)\[\text{thm:typing-soundness}\]

Proof. By induction, using the measure in Definition[7]

Where the i.h. is used, we elide the reasoning establishing the condition \([\Gamma]A = A\) for parts (i), (iii), (iv), (v) and (vi): this condition follows from Theorem[8] which ensures that the appropriate condition holds for all subderivations.

- Case \(\var{x: A \ p} \in \Gamma\)
  \[\Gamma \vdash \var{x} \Rightarrow [\Gamma]A \ p \vdash_{\text{Var}}\]
  \((\var{x: A \ p}) \in \Gamma\) \hspace{1cm} \text{Premise}
  \((\var{x: A \ p}) \in \Delta\) \hspace{1cm} \text{Given}
  \[\Delta \rightarrow \Omega\] \hspace{1cm} \text{By Lemma 9 (Uvar Preservation) (ii)}
  \([\Omega] \Gamma \vdash [\Omega]A \ p\] \hspace{1cm} \text{By DeclVar}
  \[\Delta \rightarrow \Omega\] \hspace{1cm} \text{Given}
  \[\Gamma \rightarrow \Omega\] \hspace{1cm} \[\Gamma = \Delta\]
  \[\Omega]A = [\Omega][\Gamma]A\] \hspace{1cm} \text{By Lemma 29 (Substitution Monotonicity) (iii)}
  \[\Omega] \Gamma \vdash [\Omega][\Gamma]A \ p\] \hspace{1cm} \text{By above equality}

- Case \(\Gamma \vdash e \Rightarrow A q \vdash_{\Theta}\) \hspace{1cm} \(\Theta \vdash A \prec_p \Join (\text{pol}(B), \text{pol}(A)) \ B \vdash_{\Delta}\)
  \(\Gamma \vdash e \Leftarrow B p \vdash_{\Delta}\) \hspace{1cm} \text{Sub}
  \(\Gamma \vdash e \Rightarrow A q \vdash_{\Theta}\) \hspace{1cm} \text{Subderivation}
  \(\Theta \vdash A \prec_p \Join (\text{pol}(B), \text{pol}(A)) \ B \vdash_{\Delta}\) \hspace{1cm} \text{Subderivation}
  \(\Theta \rightarrow \Delta\) \hspace{1cm} \text{By Lemma 51 (Typing Extension)}
  \(\Delta \rightarrow \Omega\) \hspace{1cm} \text{Given}
  \[\Theta \rightarrow \Omega\] \hspace{1cm} \text{By Lemma 33 (Extension Transitivity)}
  \([\Omega] \Theta \vdash [\Omega]A q\] \hspace{1cm} \text{By i.h.}
  \([\Omega] \Theta = [\Omega]\Delta\] \hspace{1cm} \text{By Lemma 56 (Confluence of Completeness)}
  \[\Omega] \Delta \vdash [\Omega]A q\] \hspace{1cm} \text{By above equality}
  \[\Theta \vdash A \prec_p \Join (\text{pol}(B), \text{pol}(A)) \ B \vdash_{\Delta}\] \hspace{1cm} \text{Subderivation}
  \([\Omega] \Delta \vdash [\Omega]A \leq \Join (\text{pol}(B), \text{pol}(A)) \ [\Omega]B\] \hspace{1cm} \text{By Theorem 6}
  \([\Omega] \Delta \vdash [\Omega]e \Leftarrow [\Omega]B p\] \hspace{1cm} \text{By DeclSub}

- Case \(\Gamma \vdash A_0 \prec \text{type}\) \hspace{1cm} \(\Gamma \vdash e_0 \Leftarrow [\Gamma]A_0 \ ! \vdash_{\Delta}\)
  \(\Gamma \vdash (e_0 : A_0) \Rightarrow [\Delta]A_0 \ ! \vdash_{\text{Anno}}\)
  \(\Gamma \vdash e_0 \Leftarrow [\Gamma]A_0 \ ! \vdash_{\Delta}\) \hspace{1cm} \text{Subderivation}
  \([\Omega] \Delta \vdash [\Omega]e_0 \Leftarrow [\Omega][\Gamma]A_0 \ !\] \hspace{1cm} \text{By i.h.}
  \(\Gamma \vdash A_0 \prec \text{type}\) \hspace{1cm} \text{Subderivation}
  \(\Gamma \vdash A_0 \prec \text{type}\) \hspace{1cm} \text{By inversion}
  \(\text{FEV}(A_0) = \emptyset\)

Proof of Theorem 9 (Soundness of Algorithmic Typing)\[\text{thm:typing-soundness}\]
Proof of Theorem 9 (Soundness of Algorithmic Typing)

\[ \Gamma \rightarrow \Delta \]
\[ \Delta \rightarrow \Omega \]
\[ \Gamma \rightarrow \Omega \]
\[ \Omega \vdash A_0 \text{type} \]
\[ [\Omega] \Omega \vdash [\Omega] A_0 \text{type} \]
\[ [\Omega] = [\Omega] \Delta \]
\[ [\Omega] \Delta \vdash [\Omega] A_0 \text{type} \]

By Lemma 33 (Extension Transitivity)
By Lemma 36 (Extension Weakening (Sorts))
By Lemma 16 (Substitution for Type Well-Formedness)
By above equality

\[ [\Omega][\Gamma] A_0 = [\Omega] A_0 \]
\[ [\Omega] \Delta \vdash [\Omega] e_0 \leftarrow [\Omega] A_0 ! \]
\[ [\Omega] A_0 = A_0 \]
\[ [\Omega] \Delta \vdash [\Omega] (e_0 : A_0) \Rightarrow [\Omega] A_0 ! \]

By Lemma 29 (Substitution Monotonicity) (iii)
By above equality
By DeclAnno
From definition of substitution
By above equality

• Case
\[ \Gamma \vdash () \leftarrow 1 p \vdash \Gamma \]
\[ [\Omega] \Delta \vdash () \leftarrow 1 p \]
\[ [\Omega] \Delta \vdash () \leftarrow [\Omega] 1 p \]

By Decl11
By definition of substitution

• Case
\[ \Gamma_0[\& : \star] \vdash () \leftarrow \& y \vdash \Gamma_0[\& : \star ] = 1 \]
\[ \Gamma_0[\& : \star] = 1 \]
\[ [\Omega] \& = [\Omega] [\Delta] \&
\[ = [\Omega] 1
\[ = \Gamma \]
\[ [\Omega] \Delta \vdash () \leftarrow 1 y \]
\[ [\Omega] \Delta \vdash () \leftarrow [\Omega] \& y \]

By Decl11
By above equality

• Case
\[ \nu \text{chk-l} \quad \Gamma, \alpha : \kappa \vdash \nu \leftarrow A_0 p \vdash \Delta, \alpha : \kappa, \Theta \]
\[ \Gamma \vdash \nu \leftarrow \forall \alpha : \kappa. A_0 p \vdash \Delta \]
\[ \Delta \rightarrow \Omega \]
\[ \Delta, \alpha \rightarrow \Omega, \alpha \]
\[ \Gamma, \alpha \rightarrow \Delta, \alpha, \Theta \]
\[ \Theta \text{ soft} \]
\[ \Delta, \alpha, \Theta \rightarrow \Omega, \alpha, \Theta \]

By Lemma 51 (Typing Extension)
By Uvar
By Lemma 51 (Typing Extension) (i) (with \( \Gamma_r = - \), which is soft)
By Lemma 25 (Filling Completes)
Proof of Theorem 9 (Soundness of Algorithmic Typing)  thm:typing-soundness

\[ \Gamma, \alpha \vdash n \iff A_0 \quad p \vdash \Delta' \quad \text{Subderivation} \]

\[ [\Omega'] \vdash [\Omega] n \iff [\Omega'] A_0 \quad p \quad \text{By i.h.} \]

\[ [\Omega'] A_0 = [\Omega] A_0 \quad \text{By Lemma 17 (Substitution Stability)} \]

\[ [\Omega'] \vdash [\Omega] n \iff [\Omega] A_0 \quad p \quad \text{By above equality} \]

\[ \Delta, \alpha, \Theta \rightarrow \Omega, \alpha, [\Theta] \quad \text{Above} \]

\[ \Theta \text{ is soft} \quad \text{Above} \]

\[ [\Omega'] \Delta' = ([\Omega] \Delta, \alpha) \quad \text{By Lemma 53 (Softness Goes Away)} \]

\[ [\Omega] \Delta, \alpha \vdash [\Omega] n \iff [\Omega] A_0 \quad p \quad \text{By above equality} \]

\[ [\Omega] \Delta \vdash [\Omega] n \iff [\Omega] (\forall \alpha. A_0) \quad p \quad \text{By definition of substitution} \]

\[ \vdash e \vdash s_0 : [\alpha/\alpha] A_0 \quad f \gg C \quad q \quad \Delta \quad \text{Subderivation} \]

\[ \Gamma \vdash e \vdash s_0 : \forall \alpha : \kappa. A_0 \quad p \gg C \quad q \quad \Delta \quad \text{Subderivation} \]

\[ \Gamma, \alpha : \kappa \vdash e \vdash s_0 : [\alpha/\alpha] A_0 \quad f \gg C \quad q \quad \Delta \quad \text{Subderivation} \]

\[ [\Omega] \Delta \vdash [\Omega] (e \vdash s_0) : [\Omega][\alpha/\alpha] A_0 \quad f \gg [\Omega] C \quad q \quad \text{By i.h.} \]

\[ [\Omega] \Delta \vdash [\Omega] (e \vdash s_0) : [\Omega][\forall \alpha : \kappa. A_0] \gg [\Omega] C \quad q \quad \text{By property of substitution} \]

\[ [\Omega] \Delta \vdash [\Omega] (e \vdash s_0) : [\Omega][\forall \alpha : \kappa. A_0] \gg [\Omega] C \quad q \quad \text{By def. of substitution} \]

\[ \text{Case} \quad e \text{ chk-I} \quad \Gamma \vdash P \quad \text{true} \quad \Theta \vdash e \iff [\Theta] A_0 \quad p \vdash \Delta \\ \Gamma \vdash e \iff A_0 \land P \quad p \vdash \Delta \quad \text{Subderivation} \]

\[ \Delta \rightarrow \Omega \quad \text{Given} \]

\[ \Theta \rightarrow \Delta \quad \text{By Lemma 51 (Typing Extension)} \]

\[ \Theta \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)} \]

\[ [\Omega] \Theta \vdash [\Omega] P \quad \text{true} \quad \Theta \rightarrow \Omega \quad \text{By Lemma 89 (Soundness of Checkprop)} \]

\[ [\Omega] \Delta \vdash [\Omega] P \quad \text{true} \quad \text{By Lemma 56 (Confluence of Completeness)} \]

\[ \Theta \vdash e \iff [\Theta] A_0 \quad p \vdash \Delta \quad \text{Subderivation} \]

\[ [\Omega] \Delta \vdash [\Omega] e \iff ([\Omega] [\Theta] A_0) \quad p \quad \text{By i.h.} \]

\[ [\Omega] \Delta \vdash [\Omega] e \iff ([\Omega] [\Theta] A_0) \land [\Omega] P \quad p \quad \text{By Decl/\land} \]

\[ [\Omega] [\Theta] A_0 = [\Omega] A_0 \quad \text{By Lemma 29 (Substitution Monotonicity) (iii)} \]

\[ [\Omega] \Delta \vdash [\Omega] e \iff ([\Omega] A_0) \land [\Omega] P \quad p \quad \text{By above equality} \]

\[ [\Omega] \Delta \vdash [\Omega] e \iff ([\Omega] A_0) \land [\Omega] P \quad p \quad \text{By def. of substitution} \]
Proof of Theorem 9: Soundness of Algorithmic Typing

- Case \( \Gamma \vdash t = \text{zero} \; \text{true} \rightarrow \Delta \)
  \[
  \Gamma \vdash \emptyset \iff (\text{Vec} \; \text{t} \; \text{A}) \; p \rightarrow \Delta
  \]

  \( \Gamma \vdash t = \text{zero} \; \text{true} \rightarrow \Delta \)
  Subderivation

  \[\Delta \rightarrow \Omega\]  Given

  \[\Omega | \Delta \vdash \{t = \text{zero}\} \; \text{true}\]  By Lemma 89 (Soundness of Checkprop)

  \[\Omega | \Delta \vdash t = \text{zero} \; \text{true}\]  By def. of substitution

  \(\equiv\) 
  \(\Omega | \Delta \vdash \{\} \; p \iff (\text{Vec} \; \{t \; \text{A}\} \; p) \rightarrow \Delta\)
  By DecNil

- Case
  \[
  \Gamma, \triangleright_{\text{inc}}, \Delta : \mathbb{N} \vdash t = \text{succ}(\hat{\Delta}) \; \text{true} \rightarrow \Gamma' \]

  \[
  \Gamma' \vdash e_1 \iff [\Gamma']A_0 \; p \rightarrow \Theta
  \]

  \[
  \Gamma' \vdash e_2 \iff [\Theta](\text{Vec} \; \text{e} \; \text{A}) \; f \rightarrow \Delta, \triangleright_{\text{inc}}, \Delta'
  \]

  \[
  \Gamma' \vdash e_1 \iff (\text{Vec} \; t \; \text{A}_0) \; p \rightarrow \Delta
  \]

  Subderivation

  \[\Delta \rightarrow \Omega\]  Given

  \[\Gamma' \rightarrow \Theta\]  By Lemma 51 (Typing Extension)

  \[\Theta \rightarrow \Delta, \triangleright_{\text{inc}}, \Delta'\]  By Lemma 51 (Typing Extension)

  \[\Delta, \triangleright_{\text{inc}}, \Delta' \rightarrow \Omega'\]  By Lemma 25 (Filling Completes)

  \[\Gamma' \rightarrow \Omega'\]  By Lemma 33 (Extension Transitivity)

  \[\Omega' | \Gamma' \vdash [\Omega' | t \; \text{true}]\]  By Lemma 89 (Soundness of Checkprop)

  \[\Omega' | \Delta, \triangleright_{\text{inc}}, \Delta' \vdash [\Omega' | t \; \text{true}]\]  By Lemma 56 (Confluence of Completeness)

  \[\Omega' | \Delta \vdash [\Omega | t \; \text{true}]\]  By Lemma 17 (Substitution Stability)

  \[\equiv\]

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

  \[
  \Theta \vdash e_2 \iff [\Theta](\text{Vec} \; \text{e} \; \text{A}) \; f \rightarrow \Delta, \triangleright_{\text{inc}}, \Delta'
  \]

  Subderivation

  \[\equiv\]

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

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

  \[
  \text{By def. of substitution}
  \]
Proof of Theorem 9 (Soundness of Algorithmic Typing) thm:typing-soundness

- **Case** $e \text{ chk-I}$
  \[ \Gamma, \alpha : \kappa \vdash e \iff [\tilde{\alpha}/\alpha] A_0 \vdash \Delta \] 
  \[ \Gamma \vdash e \iff \exists \alpha : \kappa. A_0 \vdash \Delta \] 

  $\Gamma, \alpha : \kappa \vdash e \iff [\tilde{\alpha}/\alpha] A_0 \vdash \Delta$ 
  Subderivation

  $[\Omega] \Delta \vdash [\Omega] e \iff [\Omega][\tilde{\alpha}/\alpha] A_0$ 
  By i.h.

  $[\Omega] \Delta \vdash [\Omega] e \iff [\Omega][\tilde{\alpha}/\alpha] A_0$ 
  By a property of substitution

- **Case** $v \text{ chk-I}$
  \[ \Gamma, \alpha : \kappa \vdash e \iff [\tilde{\alpha}/\alpha] A_0 \vdash \Delta \] 
  \[ \Gamma \vdash e \iff \exists \alpha : \kappa. [\tilde{\alpha}/\alpha] A_0 \vdash \Delta \] 

  $\Gamma, \alpha : \kappa \vdash e \iff [\tilde{\alpha}/\alpha] A_0 \vdash \Delta$ 
  By $\text{VarSort}$

  $\Delta \vdash \tilde{\alpha} : \kappa$ 
  By Lemma 51 (Typing Extension)

  $\Delta \rightarrow \Omega$ 
  Given

  $[\Omega] \Delta \vdash [\Omega] \tilde{\alpha} : \kappa$ 
  By Lemma 58 (Bundled Substitution for Sorting)

  $[\Omega] \Delta \vdash [\Omega] e \iff \exists \alpha : \kappa. [\tilde{\alpha}/\alpha] A_0 \vdash \Delta$ 
  By $\text{Decl}$

  $[\Omega] \Delta \vdash [\Omega] e \iff [\Omega][\alpha : \kappa. A_0] \vdash \Delta$ 
  By def. of subst.

- **Case** $v \text{ chk-I}$
  \[ \Gamma, \alpha : \kappa \vdash e \iff [\tilde{\alpha}/\alpha] A_0 \vdash \Delta \] 
  \[ \Gamma \vdash e \iff \exists \alpha : \kappa. [\tilde{\alpha}/\alpha] A_0 \vdash \Delta \] 

  $\Gamma, \alpha : \kappa \vdash e \iff [\tilde{\alpha}/\alpha] A_0 \vdash \Delta$ 
  By $\text{VarSort}$

  $\Delta \vdash \tilde{\alpha} : \kappa$ 
  By Lemma 51 (Typing Extension)

  $\Delta \rightarrow \Omega$ 
  Given

  $[\Omega] \Delta \vdash [\Omega] \tilde{\alpha} : \kappa$ 
  By Lemma 58 (Bundled Substitution for Sorting)

  $[\Omega] \Delta \vdash [\Omega] e \iff \exists \alpha : \kappa. [\tilde{\alpha}/\alpha] A_0 \vdash \Delta$ 
  By $\text{Decl}$

  $[\Omega] \Delta \vdash [\Omega] e \iff [\Omega][\alpha : \kappa. A_0] \vdash \Delta$ 
  By def. of subst.
Proof of Theorem 9 (Soundness of Algorithmic Typing)

\[ \Theta^+ \vdash v \iff [\Theta^+]A_0 \] + \Delta, \Gamma

\[ [\Omega^+][\Delta, \Gamma] \vdash [\Omega]v \iff [\Omega^+]A_0 \]

Subderivation

By i.h.

\[ \Gamma, \Gamma \vdash \Omega, \Delta' \]

By Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash \Omega \]

By Lemma 22 (Extension Inversion)

\[ [\Omega^+][\Theta^+]A_0 = [\Omega^+][\Theta^+]A_0 \]

By Lemma 29 (Substitution Monotonicity)

\[ [\Delta, \Gamma] \]

Above, with \( \langle \Omega, \Gamma \rangle \) as \( \Omega' \) and \( A_0 \) as \( B \)

By def. of substitution

\[ [\Delta, \Gamma, \Theta] \]

By Lemma 95 (Substitution Upgrade) (iii)

\[ [\Theta][\Omega] \]

By above equalities

\[ [\Omega^+][\Delta, \Gamma] / (\sigma = t) \vdash [\Omega]v \iff [\Omega]A_0 \]

By DeclCheckUnify

\[ [\Omega][\Delta] \]

From def. of context application

\[ [\Omega][\Delta] / (\sigma = t) \vdash [\Omega]v \iff [\Omega]A_0 \]

By above equality

\[ [\Omega][\Delta] \vdash [\Omega]v \iff (\sigma = t) \supset [\Omega]A_0 \]

By Decl\(=\)

\[ [\Omega][\Delta] \vdash [\Omega]v \iff (\Omega)[\sigma = t] \supset [\Omega]A_0 \]

By FEV condition above

Case \( \nu \) \( \text{chk}.I \)

\[ \Gamma, \Gamma \vdash P \vdash \bot \]

Subderivation

\[ \Gamma \vdash v \iff P \supset A_0 \]

By inversion

\[ \Gamma, \Gamma \vdash P \vdash \bot \]

By above

\[ \text{FEV}(\Gamma) \cup \text{FEV}(\Gamma t) = \emptyset \]

As in \( \Rightarrow \) case (above)

\[ \text{mgu}(\sigma, t) = \bot \]

By Lemma 90 (Soundness of Equality Elimination)

\[ [\Omega][\Delta] / (\sigma = t) \vdash [\Omega]v \iff [\Omega]A_0 \]

By DeclCheck\(=\)

\[ [\Omega][\Delta] \vdash [\Omega]v \iff (\sigma = t) \supset [\Omega]A_0 \]

By Decl\(=\)

\[ [\Omega][\Delta] \vdash [\Omega]v \iff \Omega(P \supset A_0) \]

By def. of subst.

Let \( \Omega' = \Omega \).

\[ \Omega \rightarrow \Omega' \]

By Lemma 32 (Extension Reflexivity)

\[ \Delta \rightarrow \Omega' \]

Given

Case \( \Gamma \vdash \text{true} \vdash \Theta \)

\[ \Theta \vdash e_s_0 : [\Theta]A_0 \]

Subderivation

\[ \Theta \rightarrow \Delta \]

By Lemma 51 (Typing Extension)

\[ \Delta \rightarrow \Omega \]

Given

\[ \Theta \rightarrow \Omega \]

By Lemma 33 (Extension Transitivity)
\[\Omega\Delta \vdash [\Omega](e\ s_0) : [\Omega]|\Theta|A_0\ p \gg [\Omega]|C\ q\]

By i.h.

By Lemma 29 (Substitution Monotonicity) (iii)

By above equality

Subderivation

By Lemma 97 (Completeness of Checkprop)

By Lemma 56 (Confluence of Completeness)

By above equality

By def. of subst.
Proof of Theorem 9: (Soundness of Algorithmic Typing) thm:typing-soundness

• Case $\Gamma, x : A_1 \vdash e_0 \iff A_2 \vdash \Delta, x : A_1, \Theta$

\[ \Gamma \vdash \lambda x. e_0 \iff A_1 \rightarrow A_2 \vdash \Delta \]

\[ \Delta \rightarrow \Omega \]
\[ \Delta, x : A_1 \vdash \Omega, x : [\Omega]A_1 \]
\[ \Theta \text{ soft} \]
\[ \Delta, x : A_1 \vdash \Omega, x : [\Omega]A_1, \Theta \]

Given

By $\rightarrow \text{Var}$

By Lemma 51 (Typing Extension)

By Lemma 22 (Extension Inversion) (v)

(with $\Gamma_R = \emptyset$, which is soft)

By Lemma 25 (Filling Completes)

Subderivation

\[ \begin{array}{c}
\Delta, x : A_1 \vdash e_0 \iff A_2 \vdash \Delta' \\
\Delta' \vdash \Omega, x : [\Omega]A_1, \Theta \\
\end{array} \]

By i.h.

By Lemma 17 (Substitution Stability)

By above equality

\[ \begin{array}{c}
\Delta', x : [\Omega]A_1, \Theta \\
\end{array} \]

Above

\[ \begin{array}{c}
\Delta' \iff [\Omega', \Delta] = (\Omega', \Delta, x : [\Omega]A_1) \\
\end{array} \]

By Lemma 53 (Softness Goes Away)

By above equality

\[ \begin{array}{c}
\Omega \Delta \vdash \lambda x. [\Omega]e_0 \iff ([\Omega]A_1) \rightarrow ([\Omega]A_2) \end{array} \]

By Decl→

\[ \Omega \Delta \vdash [\Omega](\lambda x. e_0) \iff [\Omega](A_1 \rightarrow A_2) \]

By definition of substitution

• Case $v \text{ chk-I} \quad \begin{array}{c}
\Gamma, x : A \vdash v \iff A \vdash \Delta, x : A, \Theta \\
\end{array} \quad \text{Rec}

\[ \Gamma \vdash \text{rec} x. v \iff A \vdash \Delta \]

Similar to the $\rightarrow I$ case, applying DeclRec instead of Decl→I
Proof of Theorem 9

(Soundness of Algorithmic Typing)

Case

\[ \Gamma[\alpha_1;\ast,\alpha_2;\ast,\alpha : \ast = \alpha_1 \rightarrow \alpha_2], x : \alpha_1 \vdash e_0 \triangleq \alpha_2 \vdash \Delta, x : \alpha_1 \vdash, \Theta, \ominus \]

By Lemma 51 (Typing Extension)

By Lemma 22 (Extension Inversion) (v)

(with \( \Gamma_k = \ominus \), which is soft)

""

By Lemma 53 (Softness Goes Away)

\[ \Delta \rightarrow \Omega \]

Given

\[ \Delta, x : \alpha_1 \vdash \Omega, x : [\Omega]\alpha_1 \vdash \]

By \( \ominus \rightarrow \) Var

By Lemma 25 (Filling Complements)

\[ \Delta, x : \alpha_1 \vdash \Omega, x : [\Omega]\alpha_1 \vdash, \Theta \]

Subderivation

\[ [\Omega']\Delta' \vdash [\Omega']\alpha_0 \triangleq [\Omega']\alpha_2 \vdash \]

By i.h.

By Lemma 17 (Substitution Stability)

\[ [\Omega']\alpha_2 = [\Omega, x : [\Omega]\alpha_1 \vdash ]\alpha_2 \]

By definition of substitution

\[ [\Omega']\Delta' = [\Omega, x : [\Omega]\alpha_1 \vdash \Delta, x : \alpha_1 \vdash ] \]

By Lemma 53 (Softness Goes Away)

\[ [\Omega]\Delta, x : [\Omega]\alpha_1 \vdash \]

By definition of context substitution

\[ [\Omega]\Delta, x : [\Omega]\alpha_1 \vdash ] \]

By above equalities

\[ [\Omega]\Delta \vdash \lambda x. [\Omega]e_0 \triangleq [\Omega]\alpha_2 \vdash \]

By Decl \( \rightarrow \) I

By above and Lemma 33 (Extension Transitivity)

\[ [\Omega]\alpha = [\Omega]([\Gamma]\alpha) \]

By Lemma 29 (Substitution Monotonicity) (i)

\[ = [\Omega, ([\Gamma]\alpha_1) \rightarrow [\Gamma]\alpha_2) \]

By definition of substitution

\[ = ([\Omega]([\Gamma]\alpha_1) \rightarrow ([\Omega]([\Gamma]\alpha_2) \]

By definition of substitution

\[ = ([\Omega]([\Gamma]\alpha_1) \rightarrow ([\Omega]\alpha_2) \]

By Lemma 29 (Substitution Monotonicity) (i)

\[ [\Omega]\Delta \vdash [\Omega]\lambda x. e_0 \triangleq [\Omega]\alpha \vdash \]

By above equality

\[ \Gamma \vdash e_0 \Rightarrow A \ q \rightarrow \Theta \]

\[ \Theta \vdash s_0 : A \ q \Rightarrow C \ p \vdash \Delta \]

\[ \Gamma \vdash e_0 \Rightarrow C \ p \vdash \Delta \]

Subderivation

Subderivation

By Lemma 51 (Typing Extension)

By Lemma 53 (Confluence of Completeness)
Proof of \textit{Theorem 9 (Soundness of Algorithmic Typing)}

We have thus shown the above “for all C’. . .” statement.

\[
\begin{align*}
\Gamma \vdash s : [\Omega]A & \gg C \triangleright \Delta \\
\text{by i.h.} & \\
\end{align*}
\]

\[
\begin{align*}
\text{by rule } \text{Decl} \rightarrow E
\end{align*}
\]

\[
\begin{align*}
\text{Case } \Gamma \vdash s : A \gg C \triangleright \Delta & \text{ FEV}(C) = \emptyset \\
\Gamma \vdash s : A \gg C \triangleright \Delta & \text{ SpineRecover}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s : A \gg C \triangleright \Delta & \text{ Subderivation}
\end{align*}
\]

\[
\begin{align*}
[\Omega] \Gamma \vdash [\Omega] s : [\Omega]A & \gg [\Omega] C q \text{ by i.h.}
\end{align*}
\]

We show the quantified premise of \textit{DeclSpineRecover}, namely,

\[
\begin{align*}
\text{for all } C’, \\
\text{ if } [\Omega] \Gamma \vdash s : [\Omega]A & \gg C’ \triangleright \Delta \text{ then } C’ = [\Omega] C
\end{align*}
\]

Suppose we have C’ such that [\Omega] \Gamma \vdash s : [\Omega]A \gg C’ \triangleright \Delta. To apply \textit{DeclSpineRecover}, we need to show C’ = [\Omega] C.

\[
\begin{align*}
[\Omega] \Gamma \vdash [\Omega] s : [\Omega]A & \gg C’ \triangleright \Delta
\end{align*}
\]

\[
\begin{align*}
\text{by assumption}
\end{align*}
\]

\[
\begin{align*}
\Omega_{canon} \longrightarrow \Omega
\end{align*}
\]

\[
\begin{align*}
\text{by Lemma 59 (Canonical Completion)}
\end{align*}
\]

\[
\begin{align*}
dom(\Omega_{canon}) = \text{dom}(\Gamma)
\end{align*}
\]

\[
\begin{align*}
\Gamma \longrightarrow \Omega_{canon}
\end{align*}
\]

\[
\begin{align*}
[\Omega] \Gamma = [\Omega_{canon}] \Gamma
\end{align*}
\]

\[
\begin{align*}
[\Omega] A = [\Omega_{canon}] A
\end{align*}
\]

\[
\begin{align*}
[\Omega_{canon}] \Gamma \vdash [\Omega] s : [\Omega_{canon}] A & \gg C’ \triangleright \Delta
\end{align*}
\]

\[
\begin{align*}
\text{by above equalities}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s : [\Gamma] A & \gg C” q \triangleright \Delta”
\end{align*}
\]

\[
\begin{align*}
\text{by Theorem 12 (iii)}
\end{align*}
\]

\[
\begin{align*}
\Omega_{canon} \longrightarrow \Omega’
\end{align*}
\]

\[
\begin{align*}
\Delta” \longrightarrow \Omega’
\end{align*}
\]

\[
\begin{align*}
C’ = [\Omega’] C”
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s : [\Gamma] A & \gg C” q \triangleright \Delta”
\end{align*}
\]

\[
\begin{align*}
\text{above}
\end{align*}
\]

\[
\begin{align*}
[\Gamma] A = A
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s : A & \gg C” q \triangleright \Delta”
\end{align*}
\]

\[
\begin{align*}
\text{by above equality}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s : A & \gg C \triangleright \Delta
\end{align*}
\]

\[
\begin{align*}
\text{subderivation}
\end{align*}
\]

\[
\begin{align*}
C” = C \text{ and } q = \Delta’ = \Delta
\end{align*}
\]

\[
\begin{align*}
\text{by Theorem 5}
\end{align*}
\]

\[
\begin{align*}
C’ = [\Omega’] C”
\end{align*}
\]

\[
\begin{align*}
= [\Omega’] C
\end{align*}
\]

\[
\begin{align*}
= [\Omega_{canon}] C
\end{align*}
\]

\[
\begin{align*}
= [\Omega] C
\end{align*}
\]

\[
\begin{align*}
\text{by Lemma 55 (Completing Completeness) (ii)}
\end{align*}
\]

\[
\begin{align*}
\text{by Lemma 55 (Completing Completeness) (ii)}
\end{align*}
\]

We have thus shown the above “for all C’. . .” statement.

\[
\begin{align*}
[\Omega] \Gamma \vdash [\Omega] s : [\Omega] A & \gg [\Omega] C [\Gamma] \text{ by } \text{DeclSpineRecover}
\end{align*}
\]
Proof of Theorem 9 (Soundness of Algorithmic Typing)

\[\Gamma \vdash s : A \text{ p } \Rightarrow \text{ C } q \vdash \Delta \quad (p = \ell) \text{ or } (q = !) \text{ or } (\text{FEV}(C) \neq \emptyset)\]

Subderivation

\[\Gamma \vdash s : A \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

Subderivation

\[\Theta \vdash s_0 : [\Theta]A_2 \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

By i.h.

\[\Theta \vdash s_0 : [\Theta]A_2 \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

By above equality

\[\Theta \vdash s_0 : [\Theta]A_2 \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

By above equality

\[\Theta \vdash s_0 : [\Theta]A_2 \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

By above equality

\[\Theta \vdash s_0 : [\Theta]A_2 \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

By above equality

\[\Theta \vdash s_0 : [\Theta]A_2 \text{ p } \Rightarrow \text{ C } q \vdash \Delta\]

By above equality
\[ \Gamma[\ldots, \hat{\alpha} : \ast = \hat{\alpha}_1 + \hat{\alpha}_2] \vdash e_0 \leftarrow \hat{\alpha}_k \forall \Delta \quad \text{Subderivation} \]
\[ [\Omega] \Delta \vdash [\Omega] e_0 \leftarrow [\Omega] \hat{\alpha}_k \forall \Delta \]
\[ [\Omega] \Delta \vdash \text{inj}_k [\Omega] e_0 \Rightarrow ([\Omega] \hat{\alpha}_1) + ([\Omega] \hat{\alpha}_2) \forall \Delta \quad \text{By i.h.} \]
\[ ([\Omega] \hat{\alpha}_1) + ([\Omega] \hat{\alpha}_2) = [\Omega] \hat{\alpha} \]
\[ [\Omega] \Delta \vdash [\Omega] (\text{inj}_k e_0) \Rightarrow [\Omega] \hat{\alpha} \forall \Delta \quad \text{By Decl+I_k} \]
\[ \text{Similar to the } \lnot \llbracket \hat{\alpha} \rrbracket \text{ case (above)} \]
\[ \text{By above equality / def. of subst.} \]

\[ \text{By above equality / def. of subst.} \]

\[
\begin{align*}
\Gamma \vdash e_1 & \Leftarrow A_1 \text{ p } \llbracket \Theta \rrbracket \quad \Theta \vdash e_2 & \Leftarrow [\Theta] A_2 \text{ p } \llbracket \Delta \rrbracket \\
\Gamma \vdash \langle e_1, e_2 \rangle & \Leftarrow A_1 \times A_2 \text{ p } \llbracket \Delta \rrbracket
\end{align*}
\]
\[ \Theta \vdash e_2 \Leftarrow [\Theta] A_2 \text{ p } \llbracket \Delta \rrbracket \]
\[ \Theta \quad \Delta \quad \Theta \quad \Omega \]
\[ \begin{align*}
\Gamma \vdash e_1 & \Leftarrow A_1 \text{ p } \llbracket \Theta \rrbracket \\
[\Omega] \Theta \vdash [\Omega] e_1 & \Leftarrow [\Omega] A_1 \text{ p } \\
[\Omega] \Delta \vdash [\Omega] e_1 & \Leftarrow [\Omega] A_1 \text{ p } \\
\text{By i.h.} \quad \text{By Lemma 33 (Extension Transitivity)}
\end{align*}
\]
\[ \Theta \vdash e_2 \Leftarrow [\Theta] A_2 \text{ p } \llbracket \Delta \rrbracket \]
\[ \begin{align*}
\Gamma \vdash e_2 & \Leftarrow [\Theta] A_2 \text{ p } \llbracket \Delta \rrbracket \\
[\Omega] \Delta \vdash [\Omega] e_2 & \Leftarrow [\Omega] [\Theta] A_2 \text{ p } \\
\text{By i.h.} \quad \text{Given} \\
\Gamma \vdash A_1 \times A_2 \text{ type} \quad \Gamma \vdash A_2 \text{ type} \quad \Gamma \quad \Theta \quad A_2 \text{ type} \\
[\Omega] \Delta \vdash [\Omega] e_2 & \Leftarrow [\Omega] A_2 \text{ p } \\
\text{Subderivation} \quad \text{By Lemma 36 (Confluence of Completeness)}
\end{align*}
\]
\[ [\Omega] \Delta \vdash ([\Omega] e_1, [\Omega] e_2) \Leftarrow ([\Omega] A_1) \times [\Omega] A_2 \text{ p } \\
\text{By Decl×I} \quad \text{By det. of substitution} \]

\[ \Theta \quad \Delta \quad \Theta \quad \Omega \]
\[ \begin{align*}
\Delta & \quad \Omega \\
\Theta & \quad \Delta \\
\Theta & \quad \Omega \\
\Gamma[\ldots, \hat{\alpha} : \ast = \hat{\alpha}_1 \times \hat{\alpha}_2] & \vdash e_1 \Leftarrow \hat{\alpha}_1 \forall \Theta \quad \Theta \vdash e_2 \Leftarrow [\Theta] \hat{\alpha}_2 \forall \Delta \\
\Gamma[\hat{\alpha} : \ast] & \vdash \langle e_1, e_2 \rangle \Leftarrow \hat{\alpha} \forall \Delta
\end{align*}
\]
\[ \text{Given} \quad \text{By Lemma 51 (Typing Extension)} \quad \text{By Lemma 51 (Typing Extension)} \quad \text{By Lemma 33 (Extension Transitivity)} \quad \text{By Lemma 56 (Confluence of Completeness)} \]
\[ [\Omega] \Theta \vdash [\Omega] e_1 \Leftarrow [\Omega] \hat{\alpha}_1 \forall \Theta \\
[\Omega] \Delta = [\Omega] \Delta \\
[\Omega] \Delta \vdash [\Omega] e_1 \Leftarrow [\Omega] \hat{\alpha}_1 \forall \Delta \\
\text{By above equality} \quad \text{By above equality} \quad \text{By i.h.} \quad \text{By i.h.} \quad \text{By Lemma 29 (Substitution Monotonicity)}
\]
\[ \Theta \vdash e_2 \Leftarrow [\Theta] \hat{\alpha}_2 \forall \Delta \\
[\Omega] \Delta \vdash [\Omega] e_2 \Leftarrow [\Omega] \hat{\alpha}_2 \forall \Delta \\
\text{By i.h.} \quad \text{By above equality} \quad \text{By above equality} \quad \text{By above equality} \quad \text{By above equality} \\
\]
\[ ([\Omega] \hat{\alpha}_1) \times [\Omega] \hat{\alpha}_2 = [\Omega] \hat{\alpha} \\
\text{Similar to the } \lnot \llbracket \hat{\alpha} \rrbracket \text{ case (above)} \\
\text{By above equality} \]
\[ [\Omega] \Delta \vdash [\Omega] \langle e_1, e_2 \rangle \Leftarrow [\Omega] \hat{\alpha} \forall \Delta \]
Proof of Theorem 9 (Soundness of Algorithmic Typing)

Case

\[ \Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \hat{\alpha}_1 \rightarrow \hat{\alpha}_2] \vdash e_0 \text{ s}_0 : (\hat{\alpha}_1 \rightarrow \hat{\alpha}_2) \] \( \implies \) \( C \) \( \vdash \Delta \)

\( \Gamma[\hat{\alpha} : \star] \vdash e_0 \text{ s}_0 : \hat{\alpha} \) \( \Gamma \vdash \Delta \)

Subderivation

By Theorem 12, there exist \( D \) \( \text{DeclMatchBase} \) By Lemma 7 (Soundness of Match Coverage), \( \text{DeclCase} \)

\[ [\Omega] \Delta \vdash [\Omega](e_0 \text{ s}_0) : [\Omega](\hat{\alpha}_1 \rightarrow \hat{\alpha}_2) \] \( \Gamma \vdash \Delta \)

By i.h.

Similar to the \( \Delta \cdash \Delta \) case

\[ [\Omega] \Delta \vdash [\Omega] \alpha \] \( \Gamma \vdash \Delta \)

By above equality

Case

\[ \Gamma \vdash e_0 \Rightarrow B \ q \vdash \Theta \ \Theta \vdash \Pi \vdash: [\Theta]B \ q \leftarrow [\Theta]C \ p \vdash \Delta \]

\( \Delta \vdash \Pi \vdash [\Delta]B \ q \vdash \Delta \vdash \Pi \vdash [\Delta]B \ q \)

\( \Gamma \vdash \text{case}(e_0, \Pi) \vdash C \ p \vdash \Delta \)

Subderivation

By Lemma 51 (Typing Extension)

\( \Theta \rightarrow \Delta \)

By Lemma 53 (Extension Transitivity)

\( [\Omega] \Theta \vdash [\Omega]e_0 \Rightarrow [\Omega]B \ q \)

By i.h.

By Lemma 56 (Confluence of Completeness)

\( [\Omega] \Theta \vdash [\Omega]e_0 \Rightarrow [\Omega]B \ q \)

\( \Theta \vdash \Pi \vdash: [\Theta]B \ q \leftarrow [\Theta]C \ p \vdash \Delta \)

Subderivation

By Lemma 63 (Well-Formed Outputs of Typing) (Synthesis)

\( \Gamma \vdash C \ p \vdash \Delta \)

Given

By Lemma 51 (Typing Extension)

\( \Theta \vdash \Pi \vdash: [\Theta]B \ q \leftarrow [\Theta]C \ p \vdash \Delta \)

By Lemma 41 (Extension Weakening for Principal Typing)

By Lemma 40 (Right-Hand Subst. for Principal Typing)

\[ [\Omega] \Delta \vdash [\Omega] \Pi \vdash [\Omega]B \ q \leftarrow [\Omega] \Pi \vdash [\Omega]C \ p \]

By i.h. (v)

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega] \Theta \vdash [\Omega] C \]

By above equalities

Assume \( \Omega \) such that \( \Delta \rightarrow \Omega \).

Assume \( D \) such that \( [\Omega] \Delta \vdash e \Rightarrow D \ q \).

Hence \( [\Omega] \Gamma \vdash e \Rightarrow D \ q \).

By Theorem 12, there exist \( B' \) and \( \Theta' \) such that \( \Gamma \vdash e_0 \Rightarrow B' \ q \vdash \Theta' \) and \( \Theta \rightarrow \Delta' \rightarrow \Theta' \) and \( D = [\Omega]'B' \) and \( B' = [\Theta]'B' \).

By Lemma 5 (Determinacy of Typing), we know \( \Theta' = \Theta \) and \( B' = B \), which means \( D = [\Omega] \Delta \vdash [\Omega] \Pi \vdash [\Omega]B \ q \).

Hence \( [\Omega] \Delta \vdash [\Omega] \Pi \vdash [\Omega]B \ q \).

By rule DeclCase, \( [\Omega] \Delta \vdash [\Omega] \Pi \vdash [\Omega] \Pi \vdash [\Omega]B \ q \).

Part (v):

Case \( \text{MatchEmpty} \) Apply rule DeclMatchEmpty

Case

\[ \Gamma \vdash e \Leftarrow C \ p \vdash \Delta \]

\( [\cdot \Rightarrow e] \vdash [\cdot \Rightarrow e] \vdash C \ p \Leftarrow \Delta \)

Apply the i.h. and DeclMatchBase

Case \( \text{MatchUnit} \) Apply the i.h. and DeclMatchUnit
Proof of Theorem 9 (Soundness of Algorithmic Typing)

• Case

\[
\Gamma \vdash \pi :: \tilde{A} \ q \leftrightarrow C \ p \vdash \Theta \quad \Theta \vdash \Pi' :: \tilde{A} \ q \leftarrow C \ p \vdash \Delta
\]

\[
\Gamma \vdash \pi \ | \ \Pi' :: \tilde{A} \ q \leftarrow C \ p \vdash \Delta
\]

Apply the i.h. to each premise, using lemmas for well-formedness under \( \Theta \); then apply \textbf{DeclMatchSeq}.

• Cases \textcolor{red}{\textbf{Match\textsuperscript{\textbullet}}} [Match\textsuperscript{\textbullet\textrightarrow}] MatchWild | MatchNil | MatchCons

Apply the i.h. and the corresponding declarative match rule.

• Cases \textcolor{red}{\textbf{Match\times}} [Match\textsuperscript{\textbullet\texttimes}]

We have \( \Gamma \vdash \tilde{A} ! \) types, so the first type in \( \tilde{A} \) has no free existential variables. Apply the i.h. and the corresponding declarative match rule.

• Case

\[
\text{A not headed by } \land \text{ or } \exists \quad \Gamma, z : A ! \vdash e' :: \tilde{A} \ q \leftarrow C \ p \vdash \Delta, z : A !, \Delta'
\]

\[
\Gamma \vdash \ z, \tilde{r} \Rightarrow e :: A, \tilde{A} \ q \leftarrow C \ p \vdash \Delta
\]

Construct \( \Omega' \) and show \( \Delta, z : A !, \Delta' \rightarrow \Omega' \) as in the \textcolor{red}{\textbf{\textbullet\textbullet\textbullet}} case. Use the i.h., then apply rule \textbf{DeclMatchNeg}.

Part (vi):

• Case

\[
\Gamma / \sigma \vdash \tau : \kappa \vdash \perp
\]

\[
\Gamma / \sigma \vdash \tau :: \kappa \vdash \perp
\]

\[
\Gamma / \sigma \vdash \tau : \kappa \vdash \perp
\]

Subderivation

\[
\text{Given}
\]

\[
\text{By Lemma 29 (Substitution Monotonicity) (i)}
\]

\[
\text{By Lemma 90 (Soundness of Equality Elimination)}
\]

\[
\text{By above equality}
\]

\[
\mathbf{\text{\textbullet\textbullet\textbullet}} \quad [\Omega] \Gamma / [\Omega](\sigma \vdash \tau) \vdash [\Omega](\tilde{r} \text{pe}) :: [\Omega] \tilde{A} \leftarrow [\Omega] C \ p \quad \text{By \textbf{DeclMatch\perp}}
\]

• Case

\[
\Gamma, \mathbf{\rightarrow} p / \sigma \vdash \tau : \kappa \vdash \Gamma'
\]

\[
\Gamma' \vdash \tilde{r} \Rightarrow e :: \tilde{A} \ q \leftarrow C \ p \vdash \Delta, \mathbf{\rightarrow} p, \Delta'
\]

\[
\Gamma / \sigma \vdash \tau : \kappa \vdash \perp
\]

\[
\Gamma / \sigma \vdash \tau :: \kappa \vdash \perp
\]

Subderivation

\[
\text{Given}
\]

\[
\text{By Lemma 29 (Substitution Monotonicity) (i)}
\]

\[
\text{By Lemma 90 (Soundness of Equality Elimination)}
\]

\[
\mathbf{\text{\textbullet\textbullet\textbullet}} \quad \text{for all } \Omega, \mathbf{\rightarrow} p \vdash t' : \kappa'
\]

\[
\Gamma, \mathbf{\rightarrow} p, \Theta \vdash \tilde{r} \Rightarrow e :: \tilde{A} \ q \leftarrow C \ p \vdash \Delta, \mathbf{\rightarrow} p, \Delta'
\]

Subderivation

\[
[\Omega, \mathbf{\rightarrow} p, \Theta][\Delta, \mathbf{\rightarrow} p, \Delta'] \vdash [\Omega, \mathbf{\rightarrow} p, \Theta][\tilde{r} \Rightarrow e] :: [\Omega, \mathbf{\rightarrow} p, \Theta] \tilde{A} \leftarrow [\Omega, \mathbf{\rightarrow} p, \Theta] C \ p \quad \text{By i.h.}
\]
Proof of Theorem 9 (Soundness of Algorithmic Typing)

\[ \theta([\Omega, \triangleright_p]\Theta) = [\theta(\Omega, \triangleright_p)] \]

(\Omega, \triangleright_p, \Theta) = [\emptyset](\Omega, \triangleright_p)

\[ [\Omega, \triangleright_p, \Theta] \therefore [\emptyset][\emptyset] \]

\[ [\Omega, \triangleright_p, \Theta]A = [\emptyset][\emptyset]A \]

\[ [\Omega, \triangleright_p, \Theta]C = [\emptyset][\emptyset]C \]

\[ [\Omega, \triangleright_p, \Theta](\varnothing \Rightarrow \varepsilon) = [\emptyset][\emptyset](\varnothing \Rightarrow \varepsilon) \]

By Lemma 95 (SubstitutionUpgrade) (iii)

By Lemma 95 (SubstitutionUpgrade) (i)

By Lemma 95 (SubstitutionUpgrade) (i)

By Lemma 95 (SubstitutionUpgrade) (iv)

\[ \theta([\Omega, \triangleright_p] \Gamma) \vdash [\emptyset][\emptyset] \varnothing \Rightarrow \varepsilon : \theta([\Omega, \triangleright_p] \Theta) \]

\[ \theta([\emptyset][\emptyset] \Gamma) \vdash [\emptyset][\emptyset] \varnothing \Rightarrow \varepsilon : \theta([\emptyset][\emptyset] \Theta) \]

\[ \theta([\emptyset][\emptyset] \Gamma) \vdash [\emptyset][\emptyset] \varnothing \Rightarrow \varepsilon : \theta([\emptyset][\emptyset] \Theta) \]

\[ \varnothing \Rightarrow \varepsilon : \theta([\emptyset][\emptyset] \Theta) \]

\[ \text{By above equalities} \]

\[ \text{Subst. not affected by } \triangleright_p \]

\[ [\emptyset][\emptyset] \Gamma / [\emptyset][\emptyset] (\sigma = \tau) \vdash [\emptyset][\emptyset] (\varnothing \Rightarrow \varepsilon) : [\emptyset][\emptyset] \Theta \]

\[ [\emptyset][\emptyset] \Gamma \]

By DecMatchUnified

\[ \Box \]

K’ Completeness

K’.1 Completeness of Auxiliary Judgments

Lemma 92 (Completeness of Instantiation).

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

If \( [\emptyset][\hat{\alpha}] \Gamma = [\emptyset][\hat{\alpha}] \tau \)

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

Proof. By induction on \( \tau \).

We have \( [\emptyset][\emptyset] \Gamma \vdash [\emptyset][\emptyset] \hat{\alpha} \leq^P [\emptyset][\emptyset] \Theta \). We now case-analyze the shape of \( \tau \).

- Case \( \tau = \hat{\beta} \):
  \( \hat{\alpha} \notin FV(\hat{\beta}) \)
  Given
  \( \hat{\alpha} \neq \hat{\beta} \)
  From definition of \( FV(-) \)
  \( \hat{\beta} \in \text{unsolved}(\Gamma) \)
  From \( \Gamma \hat{\beta} = \hat{\beta} \)
  Let \( \Omega' = \Omega \).
  \( [\emptyset][\emptyset] \Omega \rightarrow \Omega' \)
  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 \( \Delta = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa \leftarrow \hat{\alpha}] \).
  \( \Gamma \vdash \hat{\alpha} : \hat{\beta} \rightarrow \Delta \)
  By InstReach
  \( [\emptyset][\hat{\alpha}] = [\emptyset][\hat{\beta}] \)
  Given
  \( \Gamma \rightarrow \Omega \)
  Given
  \( [\emptyset][\Delta] = [\emptyset][\Omega] \rightarrow [\emptyset][\Omega'] \)
  By Lemma 27 (Parallel Extension Solution)
  \( \text{dom}(\Delta) = \text{dom}(\Omega') \)
  \( \text{dom}(\Omega) = \text{dom}(\Gamma) \) 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 \( [\emptyset][\hat{\alpha} = \alpha] \), 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, \( \Gamma = \Gamma_1[\hat{\alpha} : \kappa][\alpha : \kappa] \).
Proof of Lemma 92 (Completeness of Instantiation)

Let \( \Delta = \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa] \) and \( \Omega' = \Omega \).

By Lemma 27 (Parallel Extension Solution), \( \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \alpha : \kappa \rightarrow \Delta \).

We have \( \text{dom}(\Delta) = \text{dom}(\Gamma) \) 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** \( \tau = 0 \):
  
  Similar to the \( \tau = 1 \) case.

- **Case** \( \tau = \tau_1 \oplus \tau_2 \):
  
  \[
  \begin{align*}
  \Omega \hat{\alpha} &= \Omega(\tau_1 \oplus \tau_2) \\
  &= (\Omega(\tau_1) \oplus (\Omega(\tau_2)) \\
  \tau_1 \oplus \tau_2 &= \Gamma(\tau_1 \oplus \tau_2) \\
  \tau_1 &= \Gamma(\tau_1) \\
  \tau_2 &= \Gamma(\tau_2)
  \end{align*}
  \]

  By definition of substitution

  Given

  \[
  \begin{align*}
  \hat{\alpha} \notin \text{FV}(\tau_1 \oplus \tau_2) \\
  \hat{\alpha} \notin \text{FV}(\tau_1) \\
  \hat{\alpha} \notin \text{FV}(\tau_2)
  \end{align*}
  \]

  From definition of \( \text{FV}(\tau) \)

  Given

  Similarly

  By Lemma 23 (Deep Evar Introduction) (i) twice

  \[
  \begin{align*}
  \Gamma &= \Gamma_0[\hat{\alpha} : \kappa] \\
  \Gamma_0[\hat{\alpha} : \kappa] &\rightarrow \Gamma_0[\hat{\alpha} : \kappa, \hat{\alpha}_1 : \kappa, \hat{\alpha} : \kappa] \\
  \ldots, \hat{\alpha}_2, \hat{\alpha}_1 &\vdash \hat{\alpha}_1 \oplus \hat{\alpha}_2 : \kappa \\
  \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}] &\rightarrow \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \\
  \Gamma_0[\hat{\alpha}] &\rightarrow \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]
  \end{align*}
  \]

  By Lemma 23 (Deep Evar Introduction) (ii)

  By Lemma 33 (Extension Transitivity)

(In the last few lines above, and the rest of this case, we omit the “\( \star \)” annotations in contexts.)

Since \( \hat{\alpha} \in \text{unsolved}(\Gamma) \) and \( \Gamma \rightarrow \Omega \), we know that \( \Omega \) has the form \( \Omega_0[\hat{\alpha} = \tau_0] \).

To show that we can extend this context, we apply Lemma 23 (Deep Evar Introduction) (iii) twice to introduce \( \hat{\alpha}_2 = \tau_2 \) and \( \hat{\alpha}_1 = \tau_1 \), and then Lemma 28 (Parallel Variable Update) to overwrite \( \tau_0 \):

\[
\Omega_0[\hat{\alpha} = \tau_0] \, \rightarrow \, \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]
\]

We have \( \Gamma \rightarrow \Omega \), that is,

\[
\Gamma_0[\hat{\alpha}] \rightarrow \Omega_0[\hat{\alpha} = \tau_0]
\]

By Lemma 26 (Parallel Admissibility) (i) twice, inserting unsolved variables \( \hat{\alpha}_2 \) and \( \hat{\alpha}_1 \) on both contexts in the above extension preserves extension:

\[
\begin{align*}
\Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}] &\rightarrow \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \tau_0] \\
\Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2] &\rightarrow \Omega_0[\hat{\alpha} = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]
\end{align*}
\]

By Lemma 26 (Parallel Admissibility) (ii) twice

By Lemma 28 (Parallel Variable Update)

Since \( \hat{\alpha} \notin \text{FV}(\tau_1) \), it follows that \( \Gamma_1[\tau] = \Gamma[\tau] = \tau \).

Therefore \( \hat{\alpha}_1 \notin \text{FV}(\tau_1) \) and \( \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\tau_2) \).

By Lemma 55 (Completing Completeness) (i) and (iii), \( [\Omega_1] \Gamma_1 = [\Omega] \Gamma \) and \( [\Omega_1] \hat{\alpha}_1 = \tau_1 \).

By i.e., there are \( \Delta_2 \) and \( \Omega_2 \) such that \( \Gamma_1 \vdash \hat{\alpha}_1 := \tau_1 : \kappa \rightarrow \Delta_2 \) and \( \Delta_2 \rightarrow \Omega_2 \) and \( \Omega_1 \rightarrow \Omega_2 \).
Next, note that $[\Delta_2][\Delta_2]\tau_2 = [\Delta_2]\tau_2$.

By Lemma 64 (Left Unsolvedness Preservation), we know that $\Delta_2 \in \text{unsolved}(\Delta_2)$.
By Lemma 65 (Left Free Variable Preservation), we know that $\Delta_2 \notin \text{FV}([\Delta_2][\Delta_2]\tau_2)$.
By Lemma 33 (Extension Transitivity), $\Omega \rightarrow \Omega_2$.
We know $[\Omega_2]\Delta_2 = [\Omega]$$\Gamma$ because:

$$[\Omega_2]\Delta_2 = [\Omega_2]\Omega_2 \quad \text{By Lemma 54 (Completing Stability)}$$

$$= [\Omega]\Omega \quad \text{By Lemma 55 (Completing Completeness) (iii)}$$

$$= [\Omega]\Gamma \quad \text{By Lemma 54 (Completing Stability)}$$

By Lemma 55 (Completing Completeness) (i), we know that $[\Omega_2]\Delta_2 = [\Omega_1]\Delta_2 = \tau_2$.
By Lemma 55 (Completing Completeness) (i), we know that $[\Omega_2]\tau_2 = [\Omega]\tau_2$.
Hence we know that $[\Omega_2]\Delta_2 \leq [\Omega_2][\Delta_2][\Delta_2]\tau_2$.
By i.h., we have $\Delta$ and $\Omega'$ such that $\Delta_2 \vdash \Delta_2 := [\Delta_2][\Delta_2] : \kappa \rightarrow \Delta$ and $\Omega_2 \rightarrow \Omega'$ and $\Delta \rightarrow \Omega'$.
By rule $\text{InstBin}$, $\Gamma \vdash \Delta_2 := \tau : \kappa \rightarrow \Delta$.
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 93** (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 $\Gamma \vdash [\Gamma]|\sigma = [\Gamma]|\tau : \kappa \rightarrow \Delta$ 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** $[\Gamma]|\sigma = [\Gamma]|\tau = 1$:

  $$\Gamma \vdash 1 \equiv 1 : * \rightarrow \Delta$$

  By CheckeqUnit

  Let $\Omega' = \Omega$.

  $\Gamma \rightarrow \Omega$ Given

  $\Delta \rightarrow \Omega'$ $\Delta = \Gamma$ and $\Omega' = \Omega$ Given

  $\text{dom}(\Gamma) = \text{dom}(\Omega)$

  $\Omega \rightarrow \Omega'$ 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** $[\Gamma]|\sigma = \Delta$ and $[\Gamma]|t = \beta$:

  – If $\beta = \beta$: Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit

  – If $\beta \neq \beta$: 

Proof of Lemma 93 (Completeness of Checkeq) lem:checkeq-completeness
Proof of Lemma 93 (Completeness of Checkeq)

\[ \Gamma \rightarrow \Omega \] Given
\[ \hat{\alpha} \notin FV(\Gamma) \] By definition of \( FV(\cdot) \)

\[ [\Omega] \sigma = [\Omega] \Gamma t \] Given
\[ [\Omega][\Gamma] \sigma = [\Omega] \Gamma t \] By Lemma 29 (Substitution Monotonicity) \( \text{(i) twice} \)
\[ [\Omega] \hat{\alpha} = [\Omega] [\Gamma] t \] Given
\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \]
\[ \Gamma \vdash \hat{\alpha} := [\Gamma] t : \kappa \vdash \Delta \]
\[ \text{By CheckeqInstL} \]

**Case** \( [\Gamma] \sigma = \hat{\alpha} \) and \( [\Gamma] t = 1 \) or zero or \( \alpha \):
Similar to the previous case, except:
\[ \hat{\alpha} \notin FV(\Gamma) \] By definition of \( FV(\cdot) \)

and similarly for 1 and \( \alpha \).

**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 \( \hat{\alpha} \notin FV([\Gamma] t_0) \), then \( \hat{\alpha} \notin FV([\Gamma] t) \). Proceed as in the previous several cases.

The other case, \( \hat{\alpha} \in FV([\Gamma] t_0) \), is impossible:
We have \( \hat{\alpha} \prec [\Gamma] t_0 \).
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) \( \text{(i)}, [\Omega][\Gamma] t = [\Omega] t \), 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 \( [\Gamma] t = [\Gamma] t_1 \oplus [\Gamma] t_2 \):
\[ \Gamma \rightarrow \Omega \]
\[ \Gamma \vdash [\Gamma] \sigma_1 \oplus [\Gamma] t_1 : \star \vdash \Theta \]
\[ \Theta \rightarrow \Omega_0 \]
\[ [\Theta] [\Gamma] t_2 : \star \vdash \Delta \]
\[ \text{By i.h.} \]
\[ \text{By i.h.} \]
\[ \text{By i.h.} \]
\[ \text{By Lemma 33 (Extension Transitivity)} \]
\[ \text{By CheckeqBin} \]
Proof of Lemma 93 (Completeness of Checkeq).

- Case $[\Gamma]\sigma = \alpha$ and $[\Gamma]t = t_1 \oplus t_2$: Similar to the $\alpha$/\text{succ}(\_) case, showing the impossibility of $\alpha \in \text{FV}([\Gamma]t_k)$ for $k = 1$ and $k = 2$.

- Case $[\Gamma]t = \alpha$ and $[\Gamma]\sigma = \sigma_1 \oplus \sigma_2$: Symmetric to the previous case.

Lemma 94 (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 $\text{mgu}(\sigma, t) = \emptyset$ then $\Gamma / \sigma \not\vdash t : \kappa \rightarrow \bot$

   By properties of unification
   $\Gamma / \sigma \not\vdash t : N \not\vdash \Gamma$
   By rule \text{ElimeqSucc}
   $\rightarrow$
   $\Gamma / \sigma \not\vdash t : N \not\vdash \Gamma, \Delta$
   where $\Delta = \cdot$.
   $\rightarrow$
   Suppose $\Gamma \vdash u : \kappa'$.
   $[\Gamma, \Delta]u = [\Gamma]u$
   where $\Delta = \cdot$.
   $\rightarrow$
   $\Gamma / \sigma = \cdot$ for all $u$ such that $[\Gamma, \Delta]u = \emptyset([\Gamma]u)$.

2. If $\text{mgu}(\sigma, t) = \bot$ (that is, no most general unifier exists) then $\Gamma / \sigma \not\vdash t : \kappa \rightarrow \bot$.


- Case $[\Omega]\sigma = \cdot$:

  By properties of unification
  $\Gamma / \sigma \not\vdash t : \kappa \rightarrow \bot$.

  Suppose $\Gamma \vdash u : \kappa'$.
  $[\Gamma, \Delta]u = [\Gamma]u$
  where $\Delta = \cdot$.
  $\rightarrow$
  $\Gamma / \sigma = \cdot$ for all $u$ such that...

- Case $\sigma = \text{succ}(\sigma')$ and $t = \text{succ}(t')$:

  - Case $\text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \emptyset$:

    By properties of unification
    $\text{mgu}(\sigma', t') = \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \emptyset$
    Given
    $\text{succ}(\sigma') = [\Gamma]\text{succ}(\sigma')$
    $\rightarrow$
    $\text{succ}(\sigma') = [\Gamma]\sigma'$
    By definition of substitution
    $\sigma' = [\Gamma]\sigma'$
    By injectivity of successor
    $\text{succ}(t') = [\Gamma]\text{succ}(t')$
    Given
    $\rightarrow$
    $\text{succ}(t') = [\Gamma]t'$
    By definition of substitution
    $t' = [\Gamma]t'$
    By injectivity of successor
    $\Gamma / \sigma' \not\vdash t' : N \not\vdash \Gamma, \Delta$
    By i.h.
    $\rightarrow$
    $[\Gamma, \Delta]u = \emptyset([\Gamma]u)$ for all $u$ such that...
    $\rightarrow$
    $\Gamma / \sigma' \not\vdash t' : N \not\vdash \Gamma, \Delta$
    By rule \text{ElimeqSucc}

  - Case $\text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \bot$:

    By properties of unification
    $\text{mgu}(\sigma', t') = \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \bot$
    Given
    $\text{succ}(\sigma') = [\Gamma]\text{succ}(\sigma')$
    $\rightarrow$
    $\text{succ}(\sigma') = [\Gamma]\sigma'$
    By definition of substitution
    $\sigma' = [\Gamma]\sigma'$
    By injectivity of successor
    $\text{succ}(t') = [\Gamma]\text{succ}(t')$
    Given
    $\rightarrow$
    $\text{succ}(t') = [\Gamma]t'$
    By definition of substitution
    $t' = [\Gamma]t'$
    By injectivity of successor
    $\Gamma / \sigma' \not\vdash t' : N \not\vdash \bot$
    By i.h.
    $\rightarrow$
    $\Gamma / \sigma' \not\vdash t' : N \not\vdash \bot$
    By rule \text{ElimeqSucc}
Proof of Lemma 94 (Completeness of Elimeq)

Case $\sigma = \sigma_1 \oplus \sigma_2$ and $t = t_1 \oplus t_2$:

First we establish some properties of the subterms:

\[
\sigma_1 \oplus \sigma_2 = [\Gamma](\sigma_1 \oplus \sigma_2) \quad \text{Given}
\]
\[
= [\Gamma]\sigma_1 \oplus [\Gamma]\sigma_2 \quad \text{By definition of substitution}
\]
\[
= [\Gamma]\sigma_1 = \sigma_1 \quad \text{By injectivity of } \oplus
\]
\[
= [\Gamma]\sigma_2 = \sigma_2 \quad \text{By injectivity of } \oplus
\]
\[
t_1 \oplus t_2 = [\Gamma](t_1 \oplus t_2) \quad \text{Given}
\]
\[
= [\Gamma]t_1 \oplus [\Gamma]t_2 \quad \text{By definition of substitution}
\]
\[
= [\Gamma]t_1 = t_1 \quad \text{By injectivity of } \oplus
\]
\[
= [\Gamma]t_2 = t_2 \quad \text{By injectivity of } \oplus
\]

Subcase $\text{mgu}(\sigma, t) = \bot$:

* Subcase $\text{mgu}(\sigma_1, t_1) = \bot$:

\[
\Gamma / \sigma_1 \Downarrow t_1 : \kappa \vdash \bot \quad \text{By i.h.}
\]
\[
\Gamma / \sigma_1 \oplus \sigma_2 \Downarrow t_1 \oplus t_2 : \kappa \vdash \bot \quad \text{By rule ElimeqBin}
\]

* Subcase $\text{mgu}(\sigma_1, t_1) = \theta_1$ and $\text{mgu}(\theta_1(\sigma_2), \theta_1(t_2)) = \bot$:

\[
\Gamma / \sigma_1 \Downarrow t_1 : \kappa \vdash \Gamma, \Delta_1 \quad \text{By i.h.}
\]
\[
[\Gamma, \Delta_1]u = \theta_1([\Gamma]u) \text{ for all } u \text{ such that . . .} 
\]

\[
[\Gamma, \Delta_1] \sigma_2 = \theta_1([\Gamma]\sigma_2) \quad \text{Above line with } \sigma_2 \text{ as } u
\]
\[
= \theta_1(\sigma_2) \quad [\Gamma]\sigma_2 = \sigma_2
\]
\[
[\Gamma, \Delta_1] t_2 = \theta_1([\Gamma] t_2) \quad \text{Above line with } t_2 \text{ as } u
\]
\[
= \theta_1( t_2) \quad \text{Since } [\Gamma] \sigma_2 = \sigma_2
\]
\[
\text{mgu}( [\Gamma, \Delta_1] \sigma_2, [\Gamma, \Delta_1] t_2) = \theta_2 \quad \text{By transitivity of equality}
\]

\[
[\Gamma, \Delta_1] [\Gamma, \Delta_1] \sigma_2 = [\Gamma, \Delta_1] \sigma_2 \quad \text{By Lemma } 29 \text{ (Substitution Monotonicity)}
\]
\[
[\Gamma, \Delta_1] [\Gamma, \Delta_1] t_2 = [\Gamma, \Delta_1] t_2 \quad \text{By Lemma } 29 \text{ (Substitution Monotonicity)}
\]

\[
\Gamma, \Delta_1 / [\Gamma, \Delta_1] \sigma_2 \Downarrow [\Gamma, \Delta_1] t_2 : \kappa \vdash \bot \quad \text{By i.h.}
\]
\[
\text{By rule ElimeqBin}
\]

Subcase $\text{mgu}(\sigma, t) = \theta_2$:

\[
\text{mgu}(\sigma_1 \oplus \sigma_2, t_1 \oplus t_2) = \theta_2 \circ \theta_1 \quad \text{By properties of unifiers}
\]
\[
\text{mgu}(\sigma_1, t_1) = \theta_1 \quad 
\]
\[
\text{mgu}(\theta_1(\sigma_2), \theta_1(t_2)) = \theta_2 \quad 
\]
\[
\Gamma / \sigma_1 \Downarrow t_1 : \kappa \vdash \Gamma, \Delta_1 \quad \text{By i.h.}
\]

* \[
[\Gamma, \Delta_1]u = \theta_1([\Gamma]u) \text{ for all } u \text{ such that . . .} 
\]

\[
[\Gamma, \Delta_1] \sigma_2 = \theta_1([\Gamma]\sigma_2) \quad \text{Above line with } \sigma_2 \text{ as } u
\]
\[
= \theta_1(\sigma_2) \quad [\Gamma]\sigma_2 = \sigma_2
\]
\[
[\Gamma, \Delta_1] t_2 = \theta_1([\Gamma] t_2) \quad \text{Above line with } t_2 \text{ as } u
\]
\[
= \theta_1( t_2) \quad [\Gamma] \sigma_2 = \sigma_2
\]
\[
\text{mgu}( [\Gamma, \Delta_1] \sigma_2, [\Gamma, \Delta_1] t_2) = \theta_2 \quad \text{By transitivity of equality}
\]
Proof of Lemma 94 (Completeness of Elimeq).

Let \( \sigma \) be a substitution. Then:

\[
[\Gamma, \Delta_1 \cdot \Gamma, \Delta_1] \sigma_2 = [\Gamma, \Delta_1] \sigma_2
\]

By Lemma 29 (Substitution Monotonicity).

\[
[\Gamma, \Delta_1 \cdot \Gamma, \Delta_1] t_2 = [\Gamma, \Delta_1] t_2
\]

By Lemma 29 (Substitution Monotonicity).

\[
\Gamma, \Delta_1 / [\Gamma, \Delta_1] \sigma_2 \vdash [\Gamma, \Delta_1] t_2 : \kappa \vdash \Gamma, \Delta_2
\]

By i.h.

\[
[\Gamma, \Delta_1, \Delta_2] u' = \theta_2([\Gamma, \Delta_1] u') \text{ for all } u' \text{ such that } . . .
\]

By Lemma 29 (Substitution Monotonicity).

\[
\Gamma / \sigma _1 \oplus \sigma _2 \vdash t_1 \oplus t_2 : \kappa \vdash \Gamma, \Delta_1, \Delta_2
\]

By rule ElimeqBin.

** Suppose \( \Gamma \vdash u : \kappa' \).

\[
[\Gamma, \Delta_1, \Delta_2] u = \theta_2([\Gamma, \Delta_1] u) \]

By **

\[
= \theta_2(\theta_1([\Gamma] u)) \]

By *

\[
= \theta([\Gamma] u) \theta = \theta_2 \circ \theta_1
\]

Case \( \sigma = \alpha \):

– Subcase \( \alpha \in FV(t) \):

\[
\text{mgu}(\alpha, t) = \bot
\]

By properties of unification

\[
\Gamma / \alpha \vdash t : \kappa \vdash \bot
\]

By rule ElimeqUvarL ⊥.

– Subcase \( \alpha \notin FV(t) \):

\[
\text{mgu}(\alpha, t) = [t/\alpha]
\]

By properties of unification

\[
(\alpha = t') \notin \Gamma
\]

By *

\[
\Gamma / \alpha \vdash t : \kappa \vdash \Gamma, \alpha = t
\]

By rule ElimeqUvarL.

** Suppose \( \Gamma \vdash u : \kappa' \).

\[
[\Gamma, \alpha = t] u = [\Gamma](t/\alpha)[u]
\]

By definition of substitution

\[
= [\Gamma] t/\alpha[\Gamma] u
\]

By properties of substitution

\[
= [t/\alpha][\Gamma] u
\]

\[
[\Gamma] t = t
\]

Case \( t = \alpha \): Similar to previous case.

Lemma 95 (Substitution Upgrade).

If \( \Delta \) has the form \( \alpha_1 = t_1, . . . , \alpha_n = t_n \)

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 \( \Gamma \longrightarrow \Omega \) then \( [\Omega] \Gamma = \theta([\Omega] \Gamma) \).

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

(iv) If \( \Gamma \longrightarrow \Omega \) 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 \( \Gamma \longrightarrow \Gamma \) case.

Part (iii): By induction on \( \Delta \). In the base case (\( \Delta = \emptyset \)), 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.
Lemma 96 (Completeness of Propequiv).

Given $\Gamma \rightarrow \Omega$

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

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

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

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

Proof. By induction on the given derivations. There is only one possible case:

- **Case** $\Gamma \vdash \sigma_1 : N \quad \Gamma \vdash \sigma_2 : N$
  
  $\Gamma \vdash \sigma_1 = \sigma_2$ prop

  $[\Omega](\sigma_1 = \sigma_2) = [\Omega](\tau_1 = \tau_2)$

  $[\Omega]\sigma_1 = [\Omega]\tau_1$

  $[\Omega]\sigma_2 = [\Omega]\tau_2$

  $\Gamma \vdash \sigma_1 : N$

  $\Gamma \vdash \tau_1 : N$

  $\Gamma \vdash [\Gamma]\sigma_1 \equiv [\Gamma]\sigma_2 : N \vdash \Theta$

  $\Theta \rightarrow \Omega_0$

  $\Omega \rightarrow \Omega_0$

  $\Gamma \vdash \sigma_2 : N$

  $\Theta \vdash \sigma_2 : N$

  $\Theta \vdash \tau_2 : N$

  $\Theta \vdash \Theta[\tau_1 \equiv \Theta[\tau_2] : N \vdash \Delta$

  $\Theta \vdash \Theta[\tau_1 \equiv \Theta[\tau_2] : N \vdash \Delta$

  $\Delta \rightarrow \Omega_0$

  $\Omega_0 \rightarrow \Omega'$

  $\Theta[\tau_1] = \Theta[\Gamma]\tau_1$

  $\Theta[\tau_2] = \Theta[\Gamma]\tau_2$

  $\Theta \vdash \Theta[\tau_1 \equiv \Theta[\tau_2] : N \vdash \Delta$

  By above equalities

  By Lemma 93 (Completeness of Checkprop)

  By Lemma 93 (Completeness of Checkeq)

  By above equalities

  By Lemma 93 (Completeness of Checkprop)

  By above equalities

  By Lemma 93 (Completeness of Checkeq)

  By above equalities

  By Lemma 93 (Completeness of Checkprop)

  By above equalities

  By above equalities

Lemma 97 (Completeness of Checkprop).

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

Proof. Only one rule, $\text{DeclCheckpropEq}$ can derive $[\Omega]\Gamma \vdash [\Omega]P$ true, so by inversion, $P$ has the form $\{t_1 = t_2\}$ where $[\Omega]t_1 = [\Omega]t_2$.

By inversion on $\Gamma \vdash \{t_1 = t_2\}$ prop, we have $\Gamma \vdash t_1 : N$ and $\Gamma \vdash t_2 : N$.

Then by Lemma 93 (Completeness of Checkprop), $\Gamma \vdash \{t_1 \equiv t_2 : N \vdash \Delta$ where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$.

By $\text{CheckpropEq}$ $\Gamma \vdash \{t_1 = t_2\}$ true $\vdash \Delta$.

\[ \square \]
K'.2 Completeness of Equivalence and Subtyping

Lemma 98 (Completeness of Equiv).

If \( \Gamma \rightarrow \Omega \) and \( \Gamma \vdash A \) type and \( \Gamma \vdash B \) type and \( \Omega \models A = \Omega B \)
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 \( \text{ImpliesWF} \), \( \text{WithWF} \), \( \text{ForallWF} \), \( \text{ExistsWF} \), it follows from \( \Omega \models A = \Omega B \) 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 \( \text{ImpliesWF} \), \( \text{WithWF} \), \( \text{ForallWF} \) or \( \text{ExistsWF} \) 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 \( \text{imposs.} \) (impossible), \( \text{fully written out} \), and \( \text{similar to fully-written-out cases} \). For the lower-right case, where both \( [\Gamma]A \) and \( [\Gamma]B \) have a binary connective \( \oplus \), it must be the same connective.

The Vec type is omitted from the table, but can be treated similarly to \( \lor \) and \( \land \).

\[
\begin{array}{cccccccc}
\text{[\Gamma]B} & \text{\lor} & \land & \forall \beta. B' & \exists \beta. B' & 1 & \alpha & \beta & B_1 \oplus B_2 \\
\hline
\text{\lor} & \text{Implies} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \\
\land & \text{imposs.} & \text{With} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\forall \alpha. A' & \text{imposs.} & \text{imposs.} & \text{Forall} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\exists \alpha. A' & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{Exists} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
1 & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
\alpha & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{2.Uvars} & \text{2.BEx.Unit} \\
\& & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} \\
A_1 \oplus A_2 & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{imposs.} & \text{2.BEx.Bin} & \text{2.Bins} \\
\end{array}
\]

• Case \( \frac{\Gamma \vdash P \text{ prop} \quad \Gamma \vdash A_0 \text{ type}}{\Gamma \vdash P \lor A_0 \text{ type}} \text{ImpliesWF} \)

This case of the rule concluding the first derivation coincides with the \text{Implies} entry in the table.

We have \( \Omega A = \Omega B \), that is, \( [\Omega]P \vdash A_0 \) = \( [\Omega]B \).

Because \( \Omega \) is predicative, \( B \) must have the form \( Q \lor B_0 \), where \( [\Omega]P = [\Omega]Q \) and \( [\Omega]A_0 = [\Omega]B_0 \).
Proof of Lemma 98 (Completeness of Equiv)

\[ \Gamma \vdash P \quad \text{prop} \]
\[ \Gamma \vdash A_0 \quad \text{type} \]
\[ \Gamma \vdash Q \supset B_0 \quad \text{type} \]
\[ \Gamma \vdash Q \quad \text{prop} \]
\[ \Gamma \vdash \Theta \quad \text{By Lemma 96 (Completeness of Propequiv)} \]

\[ \Theta \quad \rightarrow \quad \Omega_0 \]
\[ \Omega \quad \rightarrow \quad \Omega_0 \]

\[ \Gamma \quad \rightarrow \quad \Theta \quad \text{By Lemma 48 (Prop Equivalence Extension)} \]
\[ \Gamma \quad \vdash \quad A_0 \quad \text{type} \]
\[ \Gamma \quad \vdash \quad B_0 \quad \text{type} \]
\[ \Gamma \vdash [\Gamma]A_0 \equiv [\Gamma]B_0 \quad \text{By i.h.} \]
\[ \equiv \quad \Delta \quad \rightarrow \quad \Omega' \]
\[ \Omega_0 \quad \rightarrow \quad \Omega' \]

\[ \equiv \quad \Omega \quad \rightarrow \quad \Omega' \quad \text{By Lemma 33 (Extension Transitivity)} \]
\[ \Gamma \vdash ([\Gamma]P \supset [\Gamma]A_0) \equiv ([\Gamma]Q \supset [\Gamma]B_0) \quad \text{By } \equiv \supset \]
\[ \equiv \quad \Gamma \vdash [\Gamma](P \supset A_0) \equiv [\Gamma](Q \supset B_0) \quad \text{By definition of substitution} \]

- Case WithWF: Similar to the ImpliesWF case, coinciding with the With entry in the table.

- Case \( \Gamma, \alpha : \kappa \vdash A_0 \quad \text{type} \)
  \[ \Gamma \vdash \forall \alpha : \kappa. A_0 \quad \text{type} \quad \text{ForallWF} \]

  This case coincides with the Forall entry in the table.

\[ \Gamma \quad \rightarrow \quad \Omega \]
\[ \Gamma, \alpha : \kappa \quad 
\[ \rightarrow \quad \Omega, \alpha : \kappa \quad \text{By } \rightarrow \rightarrow \text{Uvar} \]
\[ \Gamma, \alpha : \kappa \quad 
\[ \vdash \quad A_0 \quad \text{type} \quad \text{Subderivation} \]
\[ \Omega \quad \rightarrow \quad \Omega \quad \text{By } \rightarrow \rightarrow \text{Uvar} \]
\[ \Omega, \alpha : \kappa \quad \rightarrow \quad \Omega \quad \text{By } \rightarrow \rightarrow \text{Uvar} \]
\[ \equiv \quad \Delta \quad 
\[ \rightarrow \quad \Omega' \quad \text{By Lemma 22 (Extension Inversion) (i)} \]
\[ \equiv \quad \Delta_0 \quad 
\[ \rightarrow \quad \Omega_0 \quad \text{By Lemma 22 (Extension Inversion) (i)} \]
\[ \equiv \quad \Omega' \quad \text{By } \equiv \forall \]
\[ \equiv \quad \Gamma \vdash \forall \alpha : \kappa. [\Gamma]A_0 \equiv [\Gamma]B_0 \quad \text{By } \equiv \forall \]
\[ \equiv \quad \Gamma \vdash [\Gamma](\forall \alpha : \kappa. A_0) \equiv [\Gamma](\forall \alpha : \kappa. B_0) \quad \text{By definition of substitution} \]

- 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 96 (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:** $\Gamma A = \Gamma B = 1$

  $\Gamma \vdash 1 \equiv 1 \vdash \Gamma$ By \equivUnit

  Let $\Omega' = \Omega'$.

  $\Delta \rightarrow \Omega$ By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

  $\Omega \rightarrow \Omega'$

- **Case 2.Uvars:** $\Gamma A = \Gamma B = \alpha$

  $\Gamma \rightarrow \Omega$ Given

  Let $\Omega' = \Omega'$.

  $\Gamma \vdash \alpha \equiv \alpha \vdash \Gamma$ By \equivVar

  $\Delta \rightarrow \Omega$ By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

- **Case 2.AExUnit:** $\Gamma A = \hat{\alpha}$ and $\Gamma B = 1$

  $\Gamma \rightarrow \Omega$ Given

  $1 = [\Omega]1$ By definition of substitution

  $\hat{\alpha} \notin FV(1)$ By definition of $FV(-)$

  $[\Omega]\hat{\alpha} = [\Omega]1$ Given

  $\Gamma \vdash \hat{\alpha} := 1 : \star \vdash \Delta$ By Lemma 92 (Completeness of Instantiation) (1)

  $\Omega \rightarrow \Omega'$

  $\Delta \rightarrow \Omega'$

  $1 = [\Gamma]1$ By definition of substitution

  $\hat{\alpha} \notin FV(1)$ By definition of $FV(-)$

  $\Gamma \vdash \hat{\alpha} \equiv 1 \vdash \Delta$ By \equivInstantiateL

- **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 $\hat{\beta} = [\Omega]\hat{\beta} = [\Gamma]\beta$ and $\hat{\alpha} \notin FV(\hat{\beta})$.

- **Case 2.BExUvar:** $\Gamma A = 1$ and $\Gamma B = \hat{\beta}$

  Symmetric to the 2.AExUvar case.

- **Case 2.AEx.SameEx:** $\Gamma A = \hat{\alpha} = \hat{\beta} = [\Gamma]B$
Proof of Lemma 98

Theorem 10

Proof of Lemma 98

\[ \Gamma \vdash \alpha \equiv \alpha' \vdash \Gamma \]

By \(=\)Exvar\((\alpha = \beta)\)

\[ |\Gamma| \alpha = \alpha \]

\(\alpha\) unsolved in \(\Gamma\)

\[ \Gamma \vdash |\Gamma| \alpha \equiv |\Gamma| \beta \vdash \Gamma \]

By above equality \(+\) \(\alpha = \beta\)

\[ \Gamma \longrightarrow \Omega \]

Given

\[ \Delta \longrightarrow \Omega \]

\(\Delta = \Gamma\)

Let \(\Omega' = \Omega\).

\[ \Omega \longrightarrow \Omega' \]

By Lemma 32 [Extension Reflexivity] and \(\Omega' = \Omega\)

- **Case 2.AEx.OtherEx**: \([\Gamma] A = \alpha\) and \([\Gamma] B = \beta\) and \(\alpha \neq \beta\)

  Either \(\alpha \in \text{FV}(|\Gamma| \beta)\), or \(\alpha \notin \text{FV}(|\Gamma| \beta)\).

  - \(\alpha \in \text{FV}(|\Gamma| \beta)\):
    
    We have \(\alpha \leq |\Gamma| \beta\).
    Therefore \(\alpha = |\Gamma| \beta\), or \(\alpha \prec |\Gamma| \beta\).
    But we are in Case 2.AEx.OtherEx, so the former is impossible.
    Therefore, \(\alpha \prec |\Gamma| \beta\).
    By a property of substitutions, \([\Omega]\alpha \prec [\Omega]|\Gamma| \beta\).
    Since \(\Gamma \longrightarrow \Omega\), by Lemma 29 [Substitution Monotonicity (iii)], \([\Omega]|\Gamma| \beta = [\Omega] \beta\), so \([\Omega]\alpha \prec [\Omega] \beta\).
    But it is given that \([\Omega]\alpha = [\Omega] \beta\), a contradiction.

  - \(\alpha \notin \text{FV}(|\Gamma| \beta)\):
    
    \[ \Gamma \vdash \alpha := [\Gamma] \beta : \ast \vdash \Delta \]
    By Lemma 92 [Completeness of Instantiation]
    \[ \Gamma \vdash \alpha \equiv [\Gamma] \beta \vdash \Gamma \]
    By \(=\)InstantiateL
    \[ \Delta \longrightarrow \Omega' \]
    "
    \[ \Omega \longrightarrow \Omega' \]
    "

- **Case 2.AEx.Bin**: \([\Gamma] A = \alpha\) and \([\Gamma] B = B_1 \oplus B_2\)

  Since \([\Gamma] B\) is an arrow, it cannot be exactly \(\alpha\). By the same reasoning as in the previous case (2.AEx.OtherEx), \(\alpha \notin \text{FV}(|\Gamma| \beta)\).

  \[ \Gamma \vdash \alpha := [\Gamma] B : \ast \vdash \Delta \]
  By Lemma 92 [Completeness of Instantiation]
  \[ \Delta \longrightarrow \Omega' \]
  "
  \[ \Omega \longrightarrow \Omega' \]
  "
  \[ \Gamma \vdash [\Gamma] A \equiv [\Gamma] B \vdash \Delta \]
  By \(=\)InstantiateL

- **Case 2.BEx.Bin**: \([\Gamma] A = A_1 \oplus A_2\) and \([\Gamma] B = \beta\)

  Symmetric to the 2.AEx.Bin case, applying \(=\)InstantiateR instead of \(=\)InstantiateL.

\[ \Box \]

Theorem 10 (Completeness of Subtyping).

If \(\Gamma \longrightarrow \Omega\) and \(\text{dom}(\Gamma) = \text{dom}(\Omega)\) and \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type and \([\Omega]\Gamma \vdash [\Omega]A \leq^P [\Omega]B\) then there exist \(\Delta\) and \(\Omega'\) such that \(\Delta \longrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \longrightarrow \Omega'\) and \(\Gamma \vdash [\Gamma] A \leq^P [\Gamma] B \vdash \Delta\).

Proof. By induction on the number of \(\forall\) or \(\exists\) 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 12.

We have \([\Omega] \Gamma \vdash [\Omega] A \leq_{\text{join}([\text{pol}(\beta),\text{pol}(A)])} [\Omega] B\).
Proof of Theorem 10 (Completeness of Subtyping)

- Case \( \left[ \Omega \right] \Gamma \vdash \left[ \Omega \right] A \text{ type} \quad \text{nonpos} \left( \left[ \Omega \right] A \right) \)

\[ \begin{align*}
\left[ \Omega \right] \Gamma \vdash \left[ \Omega \right] A & \leq \left[ \Omega \right] B \\
\left[ \Omega \right] \Gamma & \vdash \left[ \Omega \right] A \\
\left[ \Omega \right] B & \leq \left[ \Omega \right] \left[ \Omega \right] B
\end{align*} \]

\[ \leq \text{Refl} \]

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 \( \left[ \Omega \right] A = (\forall \alpha : \kappa. \left[ \Omega \right] A_0) = (\forall \alpha : \kappa. \left[ \Omega \right] B_0) = \left[ \Omega \right] B \).

Let \( \Gamma_0 = (\Gamma, \alpha : \kappa, \triangleright \alpha, \triangleright \kappa) \).

Let \( \Omega_0 = (\Omega, \alpha : \kappa, \triangleright \alpha, \triangleright \kappa = \alpha) \).

* If \( \text{pol} \left( \left[ \Omega \right] A_0 \right) \in \{-, 0\} \), then:

  We elide the straightforward use of lemmas about context extension.

\[ \begin{align*}
\left[ \Omega_0 \right] \Gamma_0 & \vdash \left[ \Omega_0 \right] A_0 \\
\left[ \Omega_0 \right] \Gamma_0 & \vdash \left[ \Omega_0 \right] [\hat{\alpha}/\alpha] A_0 \leq A_0 \\
\Delta_0 & \rightarrow \Omega'_0 \\
\Omega_0 & \rightarrow \Omega'_0 \\
\Gamma_0 & \vdash \left[ \Omega_0 \right] [\hat{\alpha}/\alpha] A_0 < : \left[ \Omega \right] \left[ \Gamma \right] B_0 \vdash \Delta_0 \\
\Gamma_0 & \vdash [\hat{\alpha}/\alpha] \left[ \Gamma_0 \right] A_0 < : \left[ \Omega \right] \left[ \Gamma \right] B_0 \vdash \Delta_0 \\
\Gamma_0 & \vdash [\hat{\alpha}/\alpha] [\hat{\Gamma} \left[ \Gamma_0 \right] A_0] < : \left[ \Omega \right] \left[ \Gamma \right] B_0 \vdash \Delta_0 \\
\Gamma & \vdash [\hat{\alpha}/\alpha] [\hat{\Gamma} \left[ \Gamma \right] A_0] < : \left[ \Omega \right] \left[ \Gamma \right] B_0 \vdash \Delta
\end{align*} \]

\[ \begin{align*}
\Gamma, \alpha : \kappa \vdash \forall \alpha : \kappa. \left[ \forall \alpha : \kappa. \left[ \Gamma \right] A_0 < : \left[ \Gamma \right] B_0 \vdash \Delta, \alpha : \kappa, \Theta & \leq \forall \\text{L}
\end{align*} \]

\[ \begin{align*}
\Gamma & \vdash \forall \alpha : \kappa. \left[ \forall \alpha : \kappa. \left[ \Gamma \right] B_0 \vdash \Delta
\end{align*} \]

\[ \leq \forall \text{R} \]

\[ \begin{align*}
\left[ \forall \alpha : \kappa. \left[ \Gamma \right] A_0 < : \left[ \Gamma \right] B_0 \vdash \Delta & \leq \forall \text{Equiv}
\end{align*} \]

- If \( A \) is not headed by \( \forall \):

We have \( \text{nonneg} \left( \left[ \Omega \right] A \right) \). Therefore \( \text{nonneg} \left( A \right) \), 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.

\[ \begin{align*}
\Gamma & \rightarrow \Omega \quad \text{Given}
\Gamma & \vdash [\hat{\Gamma}] A \equiv [\hat{\Gamma}] B \vdash \Delta & \text{By Lemma 98 (Completeness of Equiv)}
\end{align*} \]

\[ \begin{align*}
[\hat{\Gamma}] & \rightarrow \Omega' \\
\Omega & \rightarrow \Omega'
\end{align*} \]

\[ \begin{align*}
\Gamma & \vdash [\hat{\Gamma}] A < : [\hat{\Gamma}] B \vdash \Delta & \leq \forall \text{Equiv}
\end{align*} \]

- Case \( \leq \text{Refl} \) Symmetric to the \( \leq \text{Refl} \) case, using \( \leq : \left[ \right] \) (or \( \leq : \left[ \right] \)) and \( \leq : \left[ \right] \leq : \left[ \right] \) instead of \( \leq : \left[ \right] \) (or \( \leq : \left[ \right] \)) and \( \leq : \left[ \right] \leq : \left[ \right] \) instead of \( \leq : \left[ \right] \) (or \( \leq : \left[ \right] \)) and \( \leq : \left[ \right] \leq : \left[ \right] \)

- Case \( \leq \forall \text{L} \)

\[ \begin{align*}
\left[ \Omega \right] \Gamma & \vdash \tau : \kappa \quad \left[ \Omega \right] \Gamma & \vdash \left[ \tau / \alpha \right] \left[ \Omega \right] A_0 \leq \left[ \Omega \right] B \\
\left[ \Omega \right] \Gamma & \vdash \forall \alpha : \kappa. \left[ \Omega \right] A_0 \leq \left[ \Omega \right] B & \leq \forall \text{L}
\end{align*} \]

We begin by considering whether or not \( \left[ \Omega \right] B \) is headed by a universal quantifier.

- \( \left[ \Omega \right] B = (\forall \beta : \kappa'. B') \):
Proof of Theorem 10 (Completeness of Subtyping) thm:subtyping-completeness

\[ [\Omega] \Gamma, \beta : \kappa \vdash [\Omega] A \leq B' \quad \text{By Lemma 5 (Subtyping Inversion)} \]

The remaining steps are similar to the \( \leq \forall \) case.

- \( [\Omega] B \) not headed by \( \forall \):

\[
\begin{align*}
[\Omega] \Gamma & \vdash \tau : \kappa & \text{Subderivation} \\
\Gamma & \rightarrow \Omega & \text{Given} \\
\Gamma, \Delta, \kappa & \rightarrow \Omega, \Delta, \kappa : \kappa & \text{By Marker} \quad \text{Solve}
\end{align*}
\]

\( \Omega = [\Omega_0] (\Gamma, \Delta, \kappa : \kappa) \quad \text{By definition of context application (lines 16, 13)} \)

\[
\begin{align*}
[\Omega] \Gamma & \vdash [\tau/\alpha] [\Omega] A_0 \leq [\Omega] B \quad \text{Subderivation} \\
[\Omega_0](\Gamma, \Delta, \kappa : \kappa) & \vdash [\tau/\alpha] [\Omega] A_0 \leq [\Omega] B & \text{By above equality} \\
[\Omega_0](\Gamma, \Delta, \kappa : \kappa) & \vdash [[\Omega_0] [\alpha/\alpha] [\Omega] A_0 \leq [\Omega] B & \text{By definition of substitution} \\
[\Omega_0](\Gamma, \Delta, \kappa : \kappa) & \vdash [\Omega_0] [\alpha/\alpha] [\Omega] A_0 \leq [\Omega_0] B & \text{By definition of substitution} \\
[\Omega_0](\Gamma, \Delta, \kappa : \kappa) & \vdash [\Omega_0] [\alpha/\alpha] A_0 \leq [\Omega_0] B & \text{By distributivity of substitution}
\end{align*}
\]

\( \Delta_0 \Rightarrow \Omega'' \quad \text{By i.h. (A lost a quantifier)} \)

\( \Omega_0 \Rightarrow \Omega'' \quad \text{"}
\]

\( \Gamma, \Delta, \kappa : \kappa \vdash [\Gamma][\alpha/\alpha] A_0 \leq [\Gamma] B \vdash \Delta_0 \quad \text{By definition of substitution} \)

\[
\begin{align*}
\Gamma, \Delta, \kappa : \kappa & \rightarrow \Delta_0 & \text{By Lemma 50 (Subtyping Extension)} \\
\Delta_0 & = (\Delta, [\alpha, \Theta]) & \text{By Lemma 22 (Subtyping Inversion) (ii)} \\
\Gamma & \rightarrow \Delta & \text{"} \\
\Omega'' & = (\Omega', [\alpha, \Omega_Z]) & \text{By Lemma 22 (Subtyping Inversion) (ii)} \\
\Delta & \rightarrow \Omega' & \text{"} \\
\Omega_0 & \rightarrow \Omega'' & \text{Above} \\
\Omega, \Delta, \kappa : \kappa & \rightarrow \Omega', \Delta, \Omega_Z & \text{By above equalities} \\
\Omega & \rightarrow \Omega' & \text{By Lemma 22 (Subtyping Inversion) (ii)}
\end{align*}
\]

\( \Gamma, \Delta, \kappa : \kappa \vdash [\Gamma][\alpha/\alpha] A_0 \leq [\Gamma] B \vdash \Delta, [\alpha, \Theta] \quad \text{By above equality } \Delta_0 = (\Delta, [\alpha, \Theta]) \\
\Gamma, \Delta, \kappa : \kappa \vdash [\alpha/\alpha] [\Gamma] A_0 \leq [\Gamma] B \vdash \Delta, [\alpha, \Theta] & \text{By def. of subst. } ([\Gamma][\alpha] = \Delta \text{ and } [\Gamma][\alpha] = \alpha) \\
[\Gamma] B \text{ not headed by } \forall & \text{From the case assumption} \\
\Gamma \vdash \forall \alpha : \kappa, [\Gamma] A_0 \leq [\Gamma] B \vdash \Delta & \text{By } \forall
\]

\( \Gamma \vdash [\Gamma](\forall \alpha : \kappa, A_0) \leq [\Gamma] B \vdash \Delta \quad \text{By definition of substitution} \)

\[
\begin{align*}
\text{Case} & \quad \frac{[\Omega] \Gamma, \beta : \kappa \vdash [\Omega] A \leq [\Omega] B_0}{[\Omega] \Gamma \vdash [\Omega] A \leq [\forall \beta : \kappa] [\Omega] B_0} & \text{By } \forall
\end{align*}
\]
Proof of Theorem 10 (Completeness of Subtyping)

\[ \forall \beta : \kappa. B_0 \]
\[ [\Omega] \Gamma \vdash [\Omega] A \leq [\Omega] B \]
\[ [\Omega] \Gamma \vdash [\Omega] A \leq \forall \beta. [\Omega] B_0 \]
\[ [\Omega, \beta : \kappa] \Gamma \vdash [\Omega] A \leq [\Omega] [\Omega] B_0 \]
\[ [\Omega, \beta : \kappa] \Gamma \vdash [\Omega] A \leq [\Omega] [\Gamma, \beta : \kappa] A \leq [\Omega, \beta : \kappa] B_0 \]
\[ \Gamma, \beta : \kappa \vdash [\Gamma, \beta : \kappa] A < : [\Gamma, \beta : \kappa] B_0 \vdash \Delta' \]
\[ \Delta' \rightarrow \Omega_0' \]
\[ \Gamma, \beta : \kappa \vdash [\Gamma] A < : [\Gamma] B_0 \vdash \Delta' \]

\[ \Gamma, \beta : \kappa \rightarrow \Delta' \]
\[ \Delta' = (\Delta, \beta : \kappa, \Theta) \]
\[ \Gamma \rightarrow \Delta \]
\[ \Delta, \beta : \kappa, \Theta \rightarrow \Omega_0' \]
\[ \Omega_0' = (\Omega', \beta : \kappa, \Omega_R) \]
\[ \Delta \rightarrow \Omega' \]

\[ \Gamma, \beta : \kappa \vdash [\Gamma] A < : [\Gamma] B_0 \vdash \Delta, \beta : \kappa, \Theta \]
\[ \Omega, \beta : \kappa \rightarrow \Omega', \beta : \kappa, \Omega_R \]
\[ \Omega \rightarrow \Omega' \]
\[ \Gamma \vdash [\Gamma] A < : [\Gamma] B_0 \vdash \Delta \]
\[ \Gamma \vdash [\Gamma] (\forall \beta : \kappa. B_0) \vdash \Delta \]

**Case**

\[ [\Omega, \alpha : \kappa] \vdash [\Omega] A_0 \leq [\Omega] B \]
\[ [\Omega, \alpha] \vdash \exists \alpha : \kappa. [\Omega] A_0 \leq [\Omega] B \]

\[ A = \exists \alpha : \kappa. A_0 \]
\[ [\Omega, \alpha] \vdash [\Omega] A \leq [\Omega] B \]
\[ [\Omega, \Gamma, \alpha : \kappa] \vdash \exists \alpha : \kappa. A_0 \leq [\Omega] B \]
\[ [\Omega, \alpha : \kappa] \vdash [\Omega] A_0 \leq [\Omega] B \]
\[ [\Omega, \alpha : \kappa] \vdash [\Omega, \alpha : \kappa] A_0 \leq [\Omega, \alpha : \kappa] B \]
\[ \Gamma, \alpha : \kappa \vdash [\Gamma, \beta : \kappa] A_0 \leq [\Gamma, \beta : \kappa] B_0 \vdash \Delta' \]
\[ \Delta' \rightarrow \Omega_0' \]
\[ \Gamma, \alpha : \kappa \vdash [\Gamma] A < : [\Gamma] B_0 \vdash \Delta' \]

\[ \Gamma, \alpha : \kappa \rightarrow \Delta' \]
\[ \Delta' = (\Delta, \alpha : \kappa, \Theta) \]
\[ \Gamma \rightarrow \Delta \]
\[ \Delta, \alpha : \kappa, \Theta \rightarrow \Omega_0' \]
\[ \Omega_0' = (\Omega', \alpha : \kappa, \Omega_R) \]
\[ \Delta \rightarrow \Omega' \]
Proof of Theorem 10 (Completeness of Subtyping) \( \text{thm:subtyping-completeness} \)

\[ \Gamma, \alpha : \kappa \vdash [\Gamma]A_0 <: + [\Gamma]B \vdash \Delta, \alpha : \kappa, \Theta \]  
\text{By above equality}

\[ \Omega, \alpha : \kappa \rightarrow \Omega', \alpha : \kappa, \Omega_R \]  
\text{By above equality}

\[ \Omega \rightarrow \Omega' \]  
\text{By Lemma 33 (Extension Transitivity)}

\[ \Gamma \vdash \exists \alpha : \kappa, [\Gamma]A_0 <: + [\Gamma]B \vdash \Delta \]  
\text{By } \text{\textless \vdash} \text{R}

\[ \Gamma \vdash [\Gamma](\exists \alpha : \kappa, A_0) <: + [\Gamma]B \vdash \Delta \]  
\text{By definition of substitution}

\[ \Psi \vdash \beta : \kappa \vdash [\Omega]A \leq + [\tau/\beta]B_0 \]  
\text{\textless \vdash} L

We consider whether \([\Omega]A\) is headed by an existential.

If \([\Omega]A = \exists \alpha : \kappa'. A'\):

\[ [\Omega] \Gamma, \alpha : \kappa' \vdash A' \leq + [\Omega]B \]  
\text{By Lemma 5 (Subtyping Inversion)}

The remaining steps are similar to the \textless \vdash L case.

If \([\Omega]A\) not headed by \(\exists\):

\[ [\Omega] \Gamma \vdash \tau : \kappa \]  
\text{Subderivation}

\[ \Gamma \rightarrow \Omega \]  
\text{Given}

\[ \Gamma, \triangleright_{\alpha} \rightarrow \Omega, \triangleright_{\alpha} \]  
\text{By } \text{\textless \vdash} \text{Marker}

\[ \Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa \rightarrow \Omega', \triangleright_{\alpha}, \hat{\alpha} : \kappa = \tau \]  
\text{By } \text{\textless \vdash} \text{Solve}

\[ [\Omega] \Gamma = [\Omega_0](\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa) \]  
\text{By definition of context application (lines 16, 13)}

\[ [\Omega] [\Gamma] \vdash [\Omega]A \leq + [\tau/\beta][\Omega]B_0 \]  
\text{Subderivation}

\[ [\Omega_0][\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa] \vdash [\Omega]A \leq + [\tau/\beta][\Omega]B_0 \]  
\text{By above equality}

\[ [\Omega_0][\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa] \vdash [\Omega]A \leq + [\Omega_0][\hat{\alpha}/\beta][\Omega]B_0 \]  
\text{By definition of substitution}

\[ [\Omega_0][\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa] \vdash [\Omega_0]A \leq + [\Omega_0][\hat{\alpha}/\beta][\Omega_0]B_0 \]  
\text{By definition of substitution}

\[ [\Omega_0][\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa] \vdash [\Omega_0]A \leq + [\Omega_0][\hat{\alpha}/\beta][\Omega_0]B_0 \]  
\text{By distributivity of substitution}

\[ \Gamma, \triangleright_{\alpha}, \hat{\alpha} \vdash [\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa]A <: + [\Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa][\hat{\alpha}/\beta]B_0 \vdash \Delta_0 \]  
\text{By i.h. (B lost a quantifier)}

\[ \Delta_0 \rightarrow \Omega'' \]  
\text{By definition of substitution}

\[ \Omega_0 \rightarrow \Omega'' \]  
\text{By definition of substitution}

\[ \Gamma, \triangleright_{\alpha}, \hat{\alpha} : \kappa \rightarrow \Delta_0 \]  
\text{By definition of substitution}

\[ \Delta_0 = (\Delta, \triangleright_{\alpha}, \Theta) \]  
\text{By Lemma 50 (Subtyping Extension)}

\[ \Gamma \rightarrow \Delta \]  
\text{By Lemma 22 (Extension Inversion) (ii)}

\[ \Omega'' = (\Omega', \triangleright_{\alpha}, \Omega_Z) \]  
\text{By Lemma 22 (Extension Inversion) (ii)}

\[ \Delta \rightarrow \Omega' \]  
\text{Above}

\[ \Omega_0 \rightarrow \Omega'' \]  
\text{By above equalities}

\[ \Omega, \triangleright_{\alpha}, \hat{\alpha} : \kappa = \tau \rightarrow \Omega', \triangleright_{\alpha}, \Omega_Z \]  
\text{By above equalities}

\[ \Omega \rightarrow \Omega' \]  
\text{By Lemma 22 (Extension Inversion) (ii)}
Proof of [Completeness of Subtyping] (thm:subtyping-completeness) 165

\[
\Gamma, \Delta, \alpha : \kappa \vdash [\Gamma] A \triangleright : [\Gamma][\alpha/\beta] B_0 \vdash \Delta, \Delta, \Theta
\]
By above equality \( \Delta_0 = (\Delta, \Delta, \Theta) \)
\[
\Gamma, \Delta, \alpha : \kappa \vdash [\Gamma] A \triangleright : [\Gamma][\alpha/\beta] B_0 \vdash \Delta, \Delta, \Theta
\]
By def. of subst. ([\Gamma][\alpha/\beta] and \( [\Gamma][\alpha/\beta] = \beta \))
\[
[\Gamma][\alpha/\beta] \not\vdash. \exists \beta : \kappa. [\Gamma][\beta] B_0 \vdash \Delta
\]
From the case hypothesis
\[
[\Gamma][\alpha/\beta] \not\vdash. \exists \beta : \kappa. [\Gamma][\beta] B_0 \vdash \Delta
\]
By \( \because \exists \zeta \)

K'3 Completeness of Typing

Lemma 99 (Variable Decomposition). If \( \Pi \vdash \var\Pi' \), then
1. if \( \Pi \not\vdash. \Pi'' \) then \( \Pi'' = \Pi' \).
2. if \( \Pi \not\vdash. \Pi'' \) then there exists \( \Pi'' \) such that \( \Pi'' \vdash \var\Pi' \) and \( \Pi'' \var\Pi' \).
3. if \( \Pi \not\vdash. \Pi_l \} \Pi_r \vdash \var\Pi' \) and \( \Pi_r \var\Pi' \).
4. if \( \Pi \var\Pi_1 \} \Pi \vdash \Pi' \) then \( \Pi' = \Pi_1 \).

Proof. Each of these follows by induction on \( \Pi \) and decomposition of the two input derivations.

Lemma 100 (Pattern Decomposition and Substitution).
1. If \( \Pi \vdash \var\Pi' \) then \( [\Omega]\Pi \vdash \var[\Omega]\Pi' \).
2. If \( \Pi \not\vdash. \Pi' \) then \( [\Omega]\Pi \not\vdash. [\Omega]\Pi' \).
3. If \( \Pi \not\vdash. \Pi' \) then \( [\Omega]\Pi \not\vdash. [\Omega]\Pi' \).
4. If \( \Pi \not\vdash. \Pi_1 \} \Pi_2 \) then \( [\Omega]\Pi \not\vdash. [\Omega]\Pi_1 \} [\Omega]\Pi_2 \).
5. If \( \Pi \var\Pi_1 \} \Pi_2 \) then \( [\Omega]\Pi \var\Pi_1 \} [\Omega]\Pi_2 \).
6. If \( [\Omega]\Pi \vdash \var\Pi' \) then there is \( \Pi'' \) such that \( [\Omega]\Pi'' = \Pi' \) and \( \Pi \var\Pi'' \).
7. If \( [\Omega]\Pi \not\vdash. \Pi' \) then there is \( \Pi'' \) such that \( [\Omega]\Pi'' = \Pi' \) and \( \Pi \not\vdash. \Pi'' \).
8. If \( [\Omega]\Pi \not\vdash. \Pi' \) then there is \( \Pi'' \) such that \( [\Omega]\Pi'' = \Pi' \) and \( \Pi \not\vdash. \Pi'' \).
9. If \( [\Omega]\Pi \not\vdash. \Pi_1 \} \Pi_2 \) then there are \( \Pi_1 \) and \( \Pi_2 \) such that \( [\Omega]\Pi_1 = \Pi_1 \) and \( [\Omega]\Pi_2 = \Pi_2 \) and \( \Pi \not\vdash. \Pi_1 \} \Pi_2 \).
10. If \( [\Omega]\Pi \var\Pi_1 \} \Pi_2 \) then there are \( \Pi_1 \) and \( \Pi_2 \) such that \( [\Omega]\Pi_1 = \Pi_1 \) and \( [\Omega]\Pi_2 = \Pi_2 \) and \( \Pi \var\Pi_1 \} \Pi_2 \).

Proof. Each case is proved by induction on the relevant derivation.

Lemma 101 (Pattern Decomposition Functionality).
1. If \( \Pi \vdash \var\Pi' \) and \( \Pi \var\Pi' \) then \( \Pi' = \Pi'' \).
2. If \( \Pi \not\vdash. \Pi' \) and \( \Pi \not\vdash. \Pi'' \) then \( \Pi' = \Pi'' \).
3. If \( \Pi \not\vdash. \Pi' \) and \( \Pi \not\vdash. \Pi'' \) then \( \Pi' = \Pi'' \).
4. If \( \Pi \not\vdash. \Pi_1 \} \Pi_2 \) and \( \Pi \not\vdash. \Pi_1 \} \Pi_2 \) then \( \Pi_1 = \Pi_1 \) and \( \Pi_2 = \Pi_2 \).
5. If \( \Pi \var\Pi_1 \} \Pi_2 \) and \( \Pi \var\Pi_1 \} \Pi_2 \) then \( \Pi_1 = \Pi_1 \) and \( \Pi_2 = \Pi_2 \).

Proof. By induction on the derivation of \( \Pi \var\Pi' \).
Lemma 102 (Decidability of Variable Removal). For all $\Pi$, either there exists a $\Pi'$ such that $\Pi \vartrianglerighteq \Pi'$ or there does not.

Proof. This follows from an induction on the structure of $\Pi$. $\square$

Lemma 103 (Variable Inversion).

1. If $\Pi \vartrianglerighteq \Pi'$ and $\Psi \vdash \Pi$ covers $\tilde{A} \ q$ then $\Psi \vdash \Pi'$ covers $\tilde{A} \ q$.
2. If $\Pi \vartrianglerighteq \Pi'$ and $\Gamma \vdash \Pi$ covers $\tilde{A} \ q$ then $\Gamma \vdash \Pi'$ covers $\tilde{A} \ q$.

Proof. This follows by induction on the relevant derivations. $\square$

Theorem 11 (Completeness of Match Coverage).

1. If $\Gamma \vdash \tilde{A} \ q$ types and $\Gamma \tilde{A} = \tilde{A}$ and (for all $\Omega$ such that $\Gamma \rightarrow \Omega$, we have $\Omega | \Gamma \vdash [\Omega] \Pi$ covers $[\Omega] \tilde{A} \ q$) then $\Gamma \vdash \Pi$ covers $\tilde{A} \ q$.
2. If $\Gamma \tilde{A} = \tilde{A}$ and $\Gamma \bar{P} = P$ and $\Gamma \vdash \tilde{A} !$ types and (for all $\Omega$ such that $\Gamma \rightarrow \Omega$, we have $\Omega | \Gamma / [\Omega] \bar{P} \vdash [\Omega] \Pi$ covers $[\Omega] \tilde{A} !$) then $\Gamma / \bar{P} \vdash \Pi$ covers $\tilde{A} !$.

Proof. By mutual induction, with the induction metric lexicographically ordered on the number of pattern constructor symbols in the branches of $\Pi$, the number of connectives in $\tilde{A}$, and 1 if $P$ is present/0 if it is absent.

1. Assume $\Gamma \vdash \tilde{A} \ q$ types and $\Gamma \tilde{A} = \tilde{A}$ and (for all $\Omega$ such that $\Gamma \rightarrow \Omega$, we have $\Omega | \Gamma \vdash [\Omega] \Pi$ covers $[\Omega] \tilde{A} \ q$)

   - Case $\tilde{A} = \cdot$:
     Choose a completing substitution $\Omega$.
     Then we have $\Omega | \Gamma \vdash [\Omega] \Pi$ covers $\cdot \ q$.
     By inversion, we see that $\text{DeclCoversEmpty}$ was the last rule, and that we have $\Omega | \Gamma \vdash [\Omega] \cdot \Rightarrow e_1 \ldots \text{covers} \cdot \ q$.
     Hence by $\text{CoversEmpty}$, we have $\Gamma \vdash \cdot \Rightarrow e_1 \ldots \text{covers} \cdot \ q$.

   - Case $\tilde{A} = \Lambda, \tilde{B}$:
     By Lemma 102 (Decidability of Variable Removal) either
     
     Case $\Pi \vartrianglerighteq \Pi'$:
     Assume $\Omega$ such that $\Gamma \rightarrow \Omega$.
     By assumption, $\Omega | \Gamma \vdash [\Omega] \Pi$ covers $[\Omega] \Lambda, \tilde{B} \ q$.
     By Lemma 100 (Pattern Decomposition and Substitution), $\Omega | \Pi \vartrianglerighteq [\Omega] \Pi'$.
     By Lemma 103 (Variable Inversion), $\Omega | \Gamma \vdash [\Omega] \Pi'$ covers $[\Omega] \tilde{B} \ q$.
     So for all $\Omega$ such that $\Gamma \rightarrow \Omega$, $\Omega | \Gamma \vdash [\Omega] \Pi'$ covers $[\Omega] \tilde{B} \ q$.
     By induction, $\Gamma \vdash \Pi'$ covers $\tilde{B} \ q$.
     $\Rightarrow$ By rule $\text{CoversVar}$, $\Gamma \vdash \Pi$ covers $\Lambda, \tilde{B} \ q$.

     Case $\forall \Pi', \neg(\Pi \vartrianglerighteq \Pi')$:
     * Case $\tilde{\alpha}, \tilde{B}$:
       This case is impossible. Choose a completing substitution $\Omega$ such that $[\Omega] \tilde{\alpha} = 1 \rightarrow 1$, and then by assumption we have $\Omega | \Gamma \vdash [\Omega] \Pi$ covers $1 \rightarrow 1, [\Omega] \tilde{B} \ q$. By inversion we have that $[\Omega] | \Pi | \Pi \vartrianglerighteq \Pi'$. By Lemma 100 (Pattern Decomposition and Substitution), we have a $\Pi''$ such that $[\Omega] | \Pi'' = \Pi'$, and $\Pi \vartrianglerighteq \Pi''$. This yields the contradiction.
     * Case $C \rightarrow D, \tilde{B}$:
     * Case $\forall \alpha : \kappa, \Lambda, \tilde{B}$:
     * Case $\alpha, \tilde{B}$:
       Similar to the $\tilde{\alpha}$ case.
Proof of Theorem 11 (Completeness of Match Coverage)

* Case $\vec{A} = 1, \vec{B}$:
  Choose an arbitrary $\Omega$ such that $\Gamma \rightarrow \Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](1, \vec{B}) \ q$.
  By inversion, we know the rule DeclCovers1 applies (since the variable case has been ruled out).
  Hence $[\Omega] \Pi \vdash [\Omega] \Pi''$ and $[\Omega] \Gamma \vdash [\Omega] \vec{B}$ q.
  By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi'$ such that $[\Omega] \Pi' = \Pi''$ and $\Pi' \vdash [\Omega] \vec{B}$ q.

Assume $\Omega$ such that $\Gamma \rightarrow \Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](1, \vec{B}) \ q$.
  By inversion, we know the rule DeclCovers1 applies (since the variable case has been ruled out).
  Hence $[\Omega] \Pi \vdash [\Omega] \Pi''$ and $[\Omega] \Gamma \vdash [\Omega] \vec{B}$ q.
  By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi''$ such that $\Pi'' = [\Omega] \Pi''$ and $\Pi' \vdash \Omega$.
  By Lemma 101 (Pattern Decomposition Functionality), we know $\hat{\Pi}' = \Pi'$.

So for all $\Omega$ such that $\Gamma \rightarrow \Omega$, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega] \vec{B}$ q.
By induction, $\Gamma \vdash \Pi'$ covers $\vec{B}$ q.
By rule Covers1, $\Gamma \vdash \Pi'$ covers $\vec{A}, \vec{B}$ q.

* Case $C \times D, \vec{B}$:
  Choose an arbitrary $\Omega$ such that $\Gamma \rightarrow \Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](C \times D, \vec{B}) \ q$.
  By inversion, we know the rule DeclCoversx applies (since the variable case has been ruled out).
  Hence $[\Omega] \Pi \vdash [\Omega] \Pi''$ and $[\Omega] \Gamma \vdash [\Omega] \vec{B}$ q.
  By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi'$ such that $[\Omega] \Pi' = \Pi''$ and $\Pi' \vdash [\Omega] \vec{B}$ q.

Assume $\Omega$ such that $\Gamma \rightarrow \Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](C \times D, \vec{B}) \ q$.
  By inversion, we know the rule DeclCoversx applies (since the variable case has been ruled out).
  Hence $[\Omega] \Pi \vdash [\Omega] \Pi''$ and $[\Omega] \Gamma \vdash [\Omega] \vec{B}$ q.
  By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi''$ such that $\Pi'' = [\Omega] \Pi''$ and $\Pi' \vdash \Omega$.
  By Lemma 101 (Pattern Decomposition Functionality), we know $\hat{\Pi}' = \Pi'$.

So for all $\Omega$ such that $\Gamma \rightarrow \Omega$, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](C, D, \vec{B}) \ q$.
By induction, $\Gamma \vdash \Pi'$ covers $C, D, \vec{B}$ q.
By rule Coversx1, $\Gamma \vdash \Pi'$ covers $C \times D, \vec{B}$ q.

* Case $C + D, \vec{B}$:
  Choose an arbitrary $\Omega$ such that $\Gamma \rightarrow \Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](C \times D, \vec{B}) \ q$.
  By inversion, we know the rule DeclCovers+ applies (since the variable case has been ruled out).
  Hence $[\Omega] \Pi \vdash [\Omega] \Pi''$ and $[\Omega] \Gamma \vdash [\Omega] \vec{B}$ q.
  By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi'$ such that $[\Omega] \Pi' = \Pi''$ and $\Pi' \vdash [\Omega] \vec{B}$ q.

Assume $\Omega$ such that $\Gamma \rightarrow \Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega] \Pi$ covers $[\Omega](C \times D, \vec{B}) \ q$.
  By inversion, we know the rule DeclCovers+ applies (since the variable case has been ruled out).
  Hence $[\Omega] \Pi \vdash [\Omega] \Pi''$ and $[\Omega] \Gamma \vdash [\Omega] \vec{B}$ q.
  By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi'$ such that $[\Omega] \Pi' = \Pi''$ and $\Pi' \vdash [\Omega] \vec{B}$ q.
Proof of Theorem 11 (Completeness of Match Coverage) thm:coverage-completeness

By Lemma 100 (Pattern Decomposition and Substitution), there is a $\Pi_1'$ such that $\Pi_1' = (\Omega')\Pi_1$ and $\Pi_1' = (\Omega)\Pi_2$ and $\Pi \not\sim\prod_i \| \prod_j$.

By Lemma 101 (Pattern Decomposition Functionality), we know $\Pi_i = \Pi_i$. So for all $\Omega$ such that $\Gamma \rightarrow\Omega$, $(\Omega)\Gamma \vdash (\Omega)\Pi_1$ covers $(\Omega)(C, \vec{B}) q$.

So for all $\Omega$ such that $\Gamma \rightarrow\Omega$, $(\Omega)\Gamma \vdash (\Omega)\Pi_2$ covers $(\Omega)(D, \vec{B}) q$.

By induction, $\Gamma \vdash \Pi_1$ covers $C, \vec{B} q$.

By induction, $\Gamma \vdash \Pi_2$ covers $D, \vec{B} q$.

By rule $\text{Covers}^+$, $\Gamma \vdash \Pi$ covers $C + D, \vec{B} q$.

* Case $\text{Vec} \cap \text{P}, \vec{B}$: Similar to the previous case.

* Case $\exists \alpha : \kappa, C, \vec{B}$:
  Assume $\Omega$ such that $\Gamma \rightarrow\Omega$.
  By assumption, $[\Omega] \Gamma \vdash [\Omega]\Pi$ covers $[\Omega]([\exists \alpha : \kappa, C, \vec{B}) q$.
  By inversion, we know the rule DeclCovers$^\exists$ applies.
  Hence $[\Omega] \Gamma, \alpha : \kappa \vdash [\Omega]\Pi$ covers $[\Omega](C, \vec{B}) q$.
  So for all $\Omega$ such that $\Gamma \rightarrow\Omega$, $(\Omega)\Gamma, \alpha : \kappa \vdash (\Omega)\Pi$ covers $(\Omega)(C, \vec{B}) q$.
  By induction, $\Gamma, \alpha : \kappa \vdash \Pi$ covers $C, \vec{B} q$.
  By rule $\text{Covers}^\exists$, $\Gamma \vdash \Pi$ covers $[\exists \alpha : \kappa, C, \vec{B} q$.

* Case $C \land P, \vec{B}$: 
  · Case $q = \emptyset$: Similar to the previous case.
  · Case $q = \lnot$:
    Assume $\Omega$ such that $\Gamma \rightarrow\Omega$.
    By assumption, $[\Omega] \Gamma \vdash [\Omega]\Pi$ covers $[\Omega]([C \land P, \vec{B}) q$.
    By inversion, we know the rule DeclCovers$^\land$ applies.
    Hence $[\Omega] \Gamma / [\Omega]P \vdash [\Omega]\Pi$ covers $[\Omega](C, \vec{B}) !$
    So for all $\Omega$ such that $\Gamma \rightarrow\Omega$, $(\Omega)\Gamma / [\Omega]P \vdash (\Omega)\Pi$ covers $(\Omega)(C, \vec{B}) !$
    By mutual induction, $\Gamma / P \vdash \Pi$ covers $C, \vec{B} !$
    By rule $\text{Covers}^\land$, $\Gamma \vdash \Pi$ covers $C \land P, \vec{B} !$

2. Assume $[\Gamma] \vec{A} = \vec{A}$ and $[\Gamma]P = \text{P}$ and $\Gamma \vdash \vec{A} !$ types and (for all $\Omega$ such that $\Gamma \rightarrow\Omega$, we have $[\Omega] \Gamma / [\Omega]P \vdash [\Omega]\Pi$ covers $(\Omega)(\vec{A} !)$.

Let $(t_1 = t_2) = \text{P}$.

Consider whether the $\text{mgu}(t_1, t_2)$ exists

- Case $\emptyset = \text{mgu}(t_1, t_2)$:
  
  \[
  \begin{align*}
  \text{mgu}(t_1, t_2) &= \emptyset \quad \text{Premise} \\
  \Gamma / t_1 &\leadsto t_2 : \kappa \vdash \Gamma, \Theta \quad \text{By Lemma 94 (Completeness of Eliminate)} (1) \\
  \Gamma / [\Gamma]t_1 &\leadsto [\Gamma]t_2 : \kappa \vdash \Gamma, \Theta \quad \text{Follows from given assumption}
  \end{align*}
  \]

Assume $\Omega$ such that $\Gamma, \Theta \rightarrow\Omega$.

By Lemma 59 (Canonical Completion), there is a $\Omega'$ such that $[\Omega] \Gamma = [\Omega'] \Gamma$ and $\text{dom}(\Gamma) = \text{dom}(\Gamma')$.

Moreover, by Lemma 22 (Extension Inversion), we can construct a $\Omega''$ such that $\Omega' = \Omega'', \Theta$ and $\Gamma \rightarrow\Omega''$.

By assumption, $[\Omega''] \Gamma / [\Omega''](t_1 = t_2) \vdash [\Omega'']\Pi$ covers $\vec{A} !$.

There is only one way this derivation could be constructed:
Proof of Theorem 11 (Completeness of Match Coverage).

Theorem 11 (Completeness of Match Coverage). Given $\Gamma \rightarrow \Omega$ such that $\text{dom}(\Gamma) = \text{dom}(\Omega)$:

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

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

(iii) If $\Gamma \vdash A \ p \ \text{type}$ and $[\Omega] \vdash [\Omega] s : [\Omega] A \ p \gg B \ q \ \text{and} \ p' \sqsubseteq p$
then there exist $\Delta$, $\Omega'$, $B'$, and $q' \sqsubseteq q$
such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ 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 $\Gamma \vdash A \ ! \ \text{types}$ and $\Gamma \vdash C \ p \ \text{type}$ and $[\Omega] \vdash [\Omega] ! \ q \leftrightarrow [\Omega] C \ p \ \text{and} \ p' \sqsubseteq p$
then there exist $\Delta$, $\Omega'$, and $C$
such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$ and $\Gamma \vdash ! \ ! \ A \ q \leftrightarrow ! \ ! \ C \ p' \vdash \Delta$.

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

Case $\theta = \text{mgu}(t_1, t_2)$

\[ [\theta][\Omega'] \Delta \vdash [\theta][\Omega'] \! \ A ! \]
\[ [\Omega'] \Delta / \theta t_1 = t_2 \vdash [\Omega'] \! \ A ! \]

Subderivation

By Lemma 95 (Substitution Upgrade) (iii)
By Lemma 95 (Substitution Upgrade) (iv)
By Lemma 95 (Substitution Upgrade) (i)
By above equalities
By above equalities
By above equalities

So we know by induction that $\Gamma, \Theta \vdash [\Gamma, \Theta] \! \ A !$.

Hence by \textbf{CoversEq} we have $\Gamma / t_1 = t_2 \vdash \Pi \ \text{covers} \ A !$.

Case $\text{mgu}(t_1, t_2) = \bot$:

\textbf{Premise}
\[ \Gamma / t_1 \triangleright t_2 : \kappa \vdash \bot \]
\[ \Gamma / [\Gamma] t_1 \triangleright [\Gamma] t_2 : \kappa \vdash \bot \]
Follows from given assumption

$\textbf{DeclCoversEq}$

$\square$
(vi) If \( \Gamma \vdash A ! \) types and \( \Gamma \vdash P \) prop and \( \text{FEV}(P) = \emptyset \) and \( \Gamma \vdash C \) p type 
and \( [\Omega] \Gamma \vdash [\Omega] \Pi : [\Omega] A ! \iff [\Omega] C p \)
and \( p \not\triangleleft p \)
then there exist \( \Delta, \Omega', \) and \( C \)
such that \( \Delta \to \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Omega \to \Omega' \)
and \( \Gamma / [\Gamma] P \vdash [\Gamma] A ! \iff [\Gamma] C p \not\vdash \Delta. \)

Proof. By induction, using the measure in Definition 7.

\[ \begin{align*}
\text{Case} \quad & (x : A \ p) \in [\Omega] \Gamma \\
\hspace{2cm} [\Omega] \Gamma \vdash x \Rightarrow A \ p & \quad \text{DeclVar} \\
& \hspace{2cm} (x : A \ p) \in [\Omega] \Gamma \\
& \hspace{4cm} \Gamma \to \Omega & \quad \text{Premise} \\
& \hspace{4cm} \Omega \to \Omega & \quad \text{Given} \\
& \hspace{4cm} \Gamma \vdash x \Rightarrow [\Gamma] A \ p \vdash \Gamma & \quad \text{By Var} \\
& \hspace{4cm} [\Gamma] A \ p = [\Gamma] A' & \quad \text{By idempotence of substitution} \\
& \hspace{4cm} \text{dom}(\Gamma) = \text{dom}(\Omega) & \quad \text{Given} \\
& \hspace{4cm} \Gamma \to \Omega & \quad \text{Given} \\
& \hspace{4cm} [\Omega]/[\Gamma] A' = [\Omega] A' & \quad \text{By Lemma 29 (Substitution Monotonicity)} (\text{iii}) \\
& \hspace{4cm} A & \quad \text{By above equality} \\
\end{align*} \]

\[ \begin{align*}
\text{Case} \quad & [\Omega] \Gamma \vdash [\Omega] e \Rightarrow B \ q \\
& \hspace{2cm} [\Omega] \Gamma \vdash B \leq [\text{join}(A, B)] [\Omega] A & \quad \text{DeclSub} \\
& \hspace{2cm} [\Omega] \Gamma \vdash A \ p \iff [\Omega] A \ p & \quad \text{DeclSub} \\
& \hspace{4cm} [\Omega] \Gamma \vdash [\Omega] e \Rightarrow B \ q \vdash \Theta & \quad \text{Subderivation} \\
& \hspace{6cm} B = [\Omega] B' & \quad \text{By i.h.} \\
& \hspace{6cm} \Theta \to \Omega_0 & \quad "" \\
& \hspace{6cm} \Omega \to \Omega_0 & \quad "" \\
& \hspace{6cm} \text{dom}(\Theta) = \text{dom}(\Omega_0) & \quad "" \\
& \hspace{4cm} \Gamma \to \Omega & \quad \text{By Lemma 33 (Extension Transition)} \\
& \hspace{4cm} \Gamma \to \Omega_0 & \quad \text{Subderivation} \\
& \hspace{4cm} [\Omega] \Gamma \vdash B \leq [\text{join}(A, B)] [\Omega] A & \quad \text{By Lemma 56 (Confluence of Completeness)} \\
& \hspace{4cm} [\Omega] \Theta \vdash B \leq [\text{join}(A, B)] [\Omega] A & \quad \text{By above equalities} \\
& \hspace{4cm} \Theta \to \Omega_0 & \quad \text{Above} \\
& \hspace{4cm} \Theta \vdash B' \leq [\text{join}(A, B)] A \vdash \Delta & \quad \text{By Theorem 10} (\text{iv}) \\
& \hspace{4cm} \Omega_0 \to \Omega' & \quad "" \\
& \hspace{4cm} \text{dom}(\Delta) = \text{dom}(\Omega') & \quad "" \\
& \hspace{4cm} \Delta \to \Omega' & \quad \text{By Lemma 33 (Extension Transition)} \\
& \hspace{4cm} \Omega \to \Omega' & \quad \text{By Lemma 33 (Extension Transition)} \\
& \hspace{4cm} \Gamma \vdash e \not\vdash A \ p \vdash \Delta & \quad \text{By Sub} \\
\end{align*} \]
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\[ \Gamma \vdash [\Omega] \alpha : \kappa \vdash [\Omega] \nu \equiv A_0 \ p \]

• Case

\[
\begin{align*}
[\Omega] \Gamma &\vdash [\Omega] \nu \equiv A_0 \\
[\Omega] \Gamma &\vdash [\Omega] \nu \equiv \forall \alpha : \kappa. A_0 \\
&\Rightarrow A_0 \ p
\end{align*}
\]

\[ \text{DeclvI} \]

We have \([\Omega] \Lambda = 1\). Either \([\Gamma] \Lambda = 1\), or \([\Gamma] \Lambda = \hat{\alpha}\) where \(\hat{\alpha} \in \text{unsolved}(\Gamma)\).

In the former case:

- Let \(\Delta = \Gamma\).
- Let \(\Omega' = \Omega\).

\[
\begin{align*}
\Gamma &\rightarrow \Omega \\
\Omega &\rightarrow \Omega' \quad \text{By Lemma 32 (Extension Reflexivity)}
\end{align*}
\]

- \(\text{dom}(\Gamma) = \text{dom}(\Omega)\) Given

\[
\begin{align*}
\Gamma &\vdash \emptyset \equiv 1 \ p \vdash \Gamma \quad \text{By 1I}
\end{align*}
\]

In the latter case, since \(\Lambda = \hat{\alpha}\) and \(\Gamma \vdash \hat{\alpha} \ p \ \text{type}\) is given, it must be the case that \(p = \lambda\).

\[
\begin{align*}
\Gamma_0[\hat{\alpha} : \star] &\vdash \emptyset \equiv \hat{\alpha} \vdash \Gamma_0[\hat{\alpha} : \star] \equiv 1, \quad \text{By 1I}\hat{\alpha}
\end{align*}
\]

- \(\Gamma_0[\hat{\alpha} : \star] \vdash \emptyset \equiv [[\Gamma_0[\hat{\alpha} : \star]] \hat{\alpha} \vdash \Gamma_0[\hat{\alpha} : \star] \equiv 1]\) By def. of subst.

\[
\begin{align*}
\Gamma_0[\hat{\alpha} : \star] &\rightarrow \Omega \\
\Gamma_0[\hat{\alpha} : \star = 1] &\rightarrow \Omega \quad \text{By Lemma 27 (Parallel Extension Solution)}
\end{align*}
\]

- \(\Omega \rightarrow \Omega \quad \text{By Lemma 32 (Extension Reflexivity)}

\[ \text{DeclvI} \]
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\[ [\Omega]A = \forall \alpha : \kappa. A_0 \]
\[ = \forall \alpha : \kappa. [\Omega]A' \]
\[ A_0 = [\Omega]A' \]

\[ \Gamma \vdash [\Omega]v \iff [\Omega]A' p \]

By definition of context substitution

\[ \Gamma, \alpha : \kappa \rightarrow \Omega, \alpha : \kappa \]

By definition of substitution

Given

Subderivation and above equality

By def. of subst. and predicativity of \( \Omega \)

By following from above equality

By definition of context substitution

By above equality

By definition of substitution

By definition of context application

By definition of substitution

By above equality

Given

By \( \text{Uvar} \)

By \( \text{DeclSpine} \)

Subderivation

By def. of subst.

By def. of subst.
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\[ [\Omega, \hat{\alpha} : \kappa = \tau] (\Gamma, \hat{\alpha} : \kappa) \vdash [\Omega](e \cdot s_0) : [\Omega, \hat{\alpha} : \kappa = \tau] [\hat{\alpha}/\alpha]A_0 \not\succ B \ q \] By above equalities

\[ \Gamma, \hat{\alpha} : \kappa \vdash e \cdot s_0 : [\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 \not\succ B' \ q \vdash \Delta \] By i.h.

\[ B = [\Omega, \hat{\alpha} : \kappa = \tau]B' \] "

\[ \Delta \rightarrow \Omega' \] "

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \] "

\[ \Omega \rightarrow \Omega' \] "

\[ B' \rightarrow [\Delta]B' \] "

\[ B \rightarrow [\Omega']B' \] "

\[ [\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 = [\Gamma][\hat{\alpha}/\alpha]A_0 \] By def. of context application

\[ = [\hat{\alpha}/\alpha][\tau]A_0 \] \(\Gamma\) does not subst. for \(\alpha\)

\[ \Gamma, \hat{\alpha} : \kappa \vdash e \cdot s_0 : [\hat{\alpha}/\alpha][\tau]A_0 \not\succ B' \ q \vdash \Delta \] By above equality

\[ \Gamma \vdash e \cdot s_0 : \forall \alpha : \kappa. [\Gamma]A_0 \not\succ B' \ q \vdash \Delta \] By \(\forall\)Spine

\[ \Gamma \vdash e \cdot s_0 : [\Gamma](\forall \alpha : \kappa. A_0) \not\succ B' \ q \vdash \Delta \] By def. of subst.

\[ \text{Case} \ \forall \text{chk-I} \quad [\Omega]\Gamma / [\Omega]\not\vdash [\Omega]\not\leftarrow [\Omega]A_0 \] ! **DeclCheck\!

\[ [\Omega]\Gamma / [\Omega]\not\vdash [\Omega]\not\leftarrow [\Omega]A_0 \] ! Subderivation

The concluding rule in this subderivation must be \textbf{DeclCheck\!

In either case, \([\Omega]\not\vdash [\Omega]A_0 \) has the form \(\sigma' = \tau\) where \(\sigma' = [\Omega]\sigma\) and \(\tau' = [\Omega]\tau\).

\[ \text{Case} \ mgu([\Omega]\sigma, [\Omega]\tau) = \bot \] ! **DeclCheck\!

\[ [\Omega]\Gamma / [\Omega]\not\vdash [\Omega]\not\leftarrow [\Omega]A_0 \] !

We have \(mgu([\Omega]\sigma, [\Omega]\tau) = \bot\). To apply Lemma 94 (Completeness of Elimeq) (2), we need to show conditions 1–5.

\[ *** \quad \Gamma \vdash (\sigma = \tau) \supset A_0 \] ! type

\[ [\Omega](\sigma = \tau) \supset A_0 = [\Gamma](\sigma = \tau) \supset A_0 \] By Lemma 39 (Principal Agreement) (i)

\[ [\Omega]\sigma = [\Gamma]\sigma \] ! By a property of subst.

\[ [\Omega]\tau = [\Gamma]\tau \] ! Similar

\[ \Gamma \vdash \sigma : \kappa \] By inversion

3 \[ \Gamma \vdash [\Gamma]\sigma : \kappa \] By Lemma 11 (Right-Hand Substitution for Sorting)

4 \[ \Gamma \vdash [\Gamma]\tau : \kappa \] Similar

\[ mgu([\Omega]\sigma, [\Omega]\tau) = \bot \] ! Given

\[ mgu([\Gamma]\sigma, [\Gamma]\tau) = \bot \] ! By above equalities

\[ \text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset \] ! By inversion on ***

\[ \text{FEV}([\Omega]\sigma) \cup \text{FEV}([\Omega]\tau) = \emptyset \] ! By a property of complete contexts

5 \[ \text{FEV}([\Gamma]\sigma) \cup \text{FEV}([\Gamma]\tau) = \emptyset \] ! By above equalities

1 \[ [\Gamma] [\Gamma]\sigma = [\Gamma]\sigma \] ! By idempotence of subst.

2 \[ [\Gamma] [\Gamma]\tau = [\Gamma]\tau \] ! By idempotence of subst.
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\[
\begin{align*}
\Gamma / [\Gamma] \sigma & \equiv [\Gamma] \tau : \kappa \vdash \bot & \text{By Lemma 94 (Completeness of Elimeq) (2)} \\
\Gamma, \triangleright_p / [\Gamma] \sigma & \equiv [\Gamma] \tau \vdash \bot & \text{By ElimpropEq}
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash v & \iff ([\Gamma] \sigma = [\Gamma] \tau) \triangleright [\Gamma] A_\emptyset \triangleleft \Gamma & \text{By \triangleright_1} \\
\therefore \Gamma \vdash v & \iff ([\Gamma] (\sigma = \tau) \triangleright A_\emptyset) \triangleright \Gamma & \text{By def. of subst.} \\
\therefore \Gamma & \longrightarrow \Omega & \text{Given} \\
\therefore \Omega & \longrightarrow \Omega & \text{By Lemma 32 (Extension Reflexivity)} \\
\therefore \text{dom}(\Gamma) = \text{dom}(\Omega) & \text{Given}
\end{align*}
\]

\[\text{Case} \quad \text{mgu}([\Omega] \sigma, [\Omega] \tau) = \emptyset \quad \therefore \emptyset ([\Omega] \Gamma) \vdash \emptyset ([\Omega] e) \iff \emptyset ([\Omega] A_\emptyset) ! \quad \text{(1)}\]

We have \text{mgu}([\Omega] \sigma, [\Omega] \tau) = \emptyset, and will need to apply Lemma 94 (Completeness of Elimeq) (1). That lemma has five side conditions, which can be shown exactly as in the case above.

\[
mgu(\sigma, \tau) = \emptyset \quad \text{Premise}
\]

\[
\begin{align*}
\Omega_0 & = ([\Omega], \triangleright_p). & \text{Let} \\
\Gamma & \longrightarrow \Omega & \text{Given} \\
\Gamma, \triangleright_p & \longrightarrow \Omega_0 & \text{By Marker} \\
\text{dom}(\Gamma) & = \text{dom}(\Omega) & \text{Given} \\
\text{dom}(\Gamma, \triangleright_p) & = \text{dom}(\Omega_0) & \text{By def. of dom(\_)}
\end{align*}
\]

\[
\begin{align*}
\Gamma, \triangleright_p / [\Gamma] \sigma & \equiv [\Gamma] \tau : \kappa \vdash \Gamma, \triangleright_p, \emptyset & \text{By Lemma 94 (Completeness of Elimeq) (1)}
\end{align*}
\]

\[
\begin{align*}
\Gamma, \triangleright_p / [\Gamma] \sigma & \equiv [\Gamma] \tau \vdash \Gamma, \triangleright_p, \emptyset & \text{By ElimpropEq}
\end{align*}
\]

\[
\begin{align*}
\text{EQ0} & \quad \text{for all } \Gamma, \triangleright_p \vdash u : \kappa. \quad \Gamma, \triangleright_p, \emptyset u = \emptyset ([\Gamma, \triangleright_p] u) \\
\Gamma & \vdash P \triangleright A_\emptyset \quad \text{type} & \text{Given} \\
\Gamma & \vdash A_\emptyset \quad \text{type} & \text{By inversion} \\
\Gamma & \longrightarrow \Omega & \text{Given} \\
\text{EQa} & \quad [\Gamma]A_\emptyset = [\Omega]A_\emptyset & \text{By Lemma 39 (Principal Agreement) (i)}
\end{align*}
\]

\[
\begin{align*}
\therefore \therefore \theta(([\Omega] \Gamma)) \vdash \theta(e) & \iff \theta(([\Omega] A_\emptyset)) ! & \text{Subderivation}
\end{align*}
\]

\[
\begin{align*}
\Gamma, \triangleright_p, \emptyset & \longrightarrow \Omega_1 & \text{By induction on } \emptyset \\
\theta([\Omega] A_\emptyset) & = \theta([\Gamma] A_\emptyset) & \text{By above equality EQa} \\
& = [\Gamma, \triangleright_p, \emptyset] A_\emptyset & \text{By Lemma 95 (Substitution Upgrade) (i) (with EQ0)} \\
& = [\Omega_1] A_\emptyset & \text{By Lemma 39 (Principal Agreement) (i)} \\
& = [\Omega_1] [\Gamma, \triangleright_p, \emptyset] A_\emptyset & \text{By Lemma 29 (Substitution Monotonicity) (iii)} \\
\theta([\Omega] \Gamma) & = [\Omega_1] [\Gamma, \triangleright_p, \emptyset] & \text{By Lemma 95 (Substitution Upgrade) (iii)} \\
\theta([\Omega] e) & = [\Omega_1] e & \text{By Lemma 95 (Substitution Upgrade) (iv)}
\end{align*}
\]

\[
\begin{align*}
[\Omega_1] [\Gamma, \triangleright_p, \emptyset] & \vdash [\Omega_1] e & \iff [\Omega_1] [\Gamma, \triangleright_p, \emptyset] A_\emptyset & \text{By above equalities} \\
\text{dom}(\Gamma, \triangleright_p, \emptyset) & = \text{dom}(\Omega_1) & \text{dom}(\Gamma) & = \text{dom}(\Omega)
\end{align*}
\]
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\r
\r
$\Gamma, \vdash p, \Theta \vdash e \iff [\Gamma, \vdash p, \Theta]A_0 \vdash \Delta'$

By i.h.

$\Delta' \rightarrow \Omega_2'$

$\Omega_1 \rightarrow \Omega_2'$

$\text{dom}(\Delta') = \text{dom}(\Omega_2')$

$\Delta' = (\Delta, \vdash p, \Delta'')$

$\Omega_2' = (\Omega', \vdash p, \Omega_Z)$

By Lemma \ref{lem:extension-inversion} (Extension Inversion) (ii)

$\Delta \rightarrow \Omega'$

$\Omega_0 \rightarrow \Omega_2'$

$\Omega, \vdash p \rightarrow \Omega', \vdash p, \Omega_Z$

By Lemma \ref{lem:extension-transitivity} (Extension Transitivity)

$\Omega \rightarrow \Omega'$

By above equalities

$\text{dom}(\Delta) = \text{dom}(\Omega')$

By Lemma \ref{lem:extension-inversion} (Extension Inversion) (ii)

$\Gamma, \vdash p, \Theta \vdash e \iff [\Gamma, \vdash p, \Theta]A_0 \vdash \Delta, \vdash p, \Delta''$

By above equality

$\Gamma \vdash e \iff ([\Gamma]s = [\Gamma]r) \vdash [\Gamma]A_0 \vdash \Delta$ By \ref{def:substitution}

$\Gamma \vdash e \iff [\Gamma](P \supset A_0) \vdash \Delta$ By def. of subst.

**Case**

$[\Omega] \Gamma \vdash [\Omega]P \text{ true}$

$[\Omega] \Gamma \vdash [\Omega](e \ s_0) : [\Omega]A_0 \vdash B \ q$

$[\Omega] \Gamma \vdash [\Omega](e \ s_0) : ([\Omega]P) \supset [\Omega]A_0 \vdash B \ q$ \hspace{1cm} \text{Decl} \rightarrow \text{Spine}

$[\Omega] \Gamma \vdash [\Omega]P \text{ true}$

Subderivation

$[\Omega] \Gamma \vdash [\Omega][\Gamma]P \text{ true}$

By Lemma \ref{lem:substitution-monotonicity} (Substitution Monotonicity) (ii)

$\Gamma \vdash [\Gamma]P \text{ true} \vdash \Theta$ By Lemma \ref{lem:completeness-of-checkprop} (Completeness of Checkprop)

$\Theta \rightarrow \Omega_1$

By Lemma \ref{lem:extension-inversion} (Extension Inversion) (ii)

$\Omega \rightarrow \Omega_1$

By above equality

$\text{dom}(\Theta) = \text{dom}(\Omega_1)$

By Lemma \ref{lem:extension-inversion} (Extension Inversion) (ii)

$\Gamma \rightarrow \Omega$

Given

$[\Omega] \Gamma = [\Omega_1] \Theta$

By Lemma \ref{lem:multi-confluence} (Multiple Confluence)

$[\Omega] A_0 = [\Omega_1] A_0$

By Lemma \ref{lem:completeness} (Completing Completeness) (ii)

$[\Omega] \Gamma \vdash [\Omega](e \ s_0) : [\Omega]A_0 \vdash B \ q$

Subderivation

$[\Omega_1] \Theta \vdash [\Omega](e \ s_0) : [\Omega_1]A_0 \vdash B \ q$

By above equalities

$\Theta \vdash e \ s_0 : [\Theta]A_0 \vdash B' \ q \vdash \Delta$ By i.h.

$B' = [\Delta]B'$

$\text{dom}(\Delta) = \text{dom}(\Omega')$

$\text{dom}(\Delta) = \text{dom}(\Omega')$

$B = [\Omega']B'$

$\Delta \rightarrow \Omega'$

$\Omega_1 \rightarrow \Omega'$

By Lemma \ref{lem:extension-transitivity} (Extension Transitivity)

$[\Theta] A_0 = [\Theta][\Gamma] A_0$

By Lemma \ref{lem:substitution-monotonicity} (Substitution Monotonicity) (iii)

$\Theta \vdash e \ s_0 : [\Theta][\Gamma]A_0 \vdash B' \ q \vdash \Delta$ By above equality

$\Gamma \vdash e \ s_0 : ([\Gamma]P) \supset [\Gamma]A_0 \vdash B' \ q \vdash \Delta$ By \ref{def:spine}

$\Gamma \vdash e \ s_0 : [\Gamma](P \supset A_0) \vdash B' \ q \vdash \Delta$ By def. of subst.
• Case \[\Gamma \vdash \Omega \vdash [\Omega]_0 \mapsto A' \_k \_p \mapsto (\text{Decl+I}_k)\]

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:

\[
\begin{align*}
\Gamma \vdash [\Omega]e_0 & \mapsto A'_k \_p \quad \text{Subderivation} \\
\Gamma \vdash [\Omega]e_0 & \mapsto [\Omega]A_k \_p \quad \text{[\Omega]A_k = A'_k} \\
\Gamma & \vdash e_0 \mapsto [\Gamma]A_k \_p \mapsto \Delta 
\end{align*}
\]

By i.h.

- \(\Delta \mapsto \Omega\)
- \(\text{dom}(\Delta) = \text{dom}(\Omega')\)
- \(\Omega \mapsto \Omega'\)

\[\Gamma \vdash \text{inj}_k e_0 \mapsto ([\Gamma]A_1) + ([\Gamma]A_2) \mapsto \Delta \quad \text{By +I}_k\]

By def. of subst.

In the latter case, \(A = \alpha\) and \([\Omega]A = [\Omega]\alpha = A'_1 + A'_2 = \tau'_1 + \tau'_2\).

By inversion on \(\Gamma \vdash \alpha \_p \_\text{type}\), it must be the case that \(p = \_f\).

\[\Gamma \mapsto \Omega\]

Given

\[\Gamma = \Gamma_0[\alpha : \_x] \quad \alpha \in \text{unsolved}(\Gamma)\]

\[\Omega = \Omega_0[\alpha : \_x = \tau_0] \quad \text{By Lemma 22 (Extension Inversion) (vi)}\]

Let \(\Omega_2 = \Omega_0[\alpha_1 : x = \tau'_1, \alpha_2 : x = \alpha_1 + \alpha_2]\).

Let \(\Gamma_2 = \Gamma_0[\alpha_1 : \_x, \alpha_2 : \_x, \alpha : \_x = \alpha_1 + \alpha_2]\).

- \(\Gamma \mapsto \Gamma_2\)
  - By Lemma 23 (Deep Evar Introduction) (iii) twice
  - and Lemma 26 (Parallel Admissibility) (ii)

- \(\Omega \mapsto \Omega_2\)
  - By Lemma 23 (Deep Evar Introduction) (iii) twice
  - and Lemma 26 (Parallel Admissibility) (iii)

- \(\Gamma_2 \mapsto \Omega_2\)
  - By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)

\[
\begin{align*}
\Gamma \vdash [\Omega]e_0 & \mapsto [\Omega]_2 \_\alpha_k \_\_y \quad \text{Subd. and } A'_k = \tau'_k = [\Omega]_2 \_\alpha_k \\
\Gamma & \vdash [\Omega] = [\Omega]_2 \_\Gamma_2 \\
[\Omega]_2 \_\Gamma_2 & \vdash e_0 \mapsto [\Omega]_2 \_\alpha_k \_y \\
\Gamma_2 & \vdash e_0 \mapsto [\Gamma]_2 \_\alpha_k \_y \mapsto \Delta 
\end{align*}
\]

By above equality

- \(\Delta \mapsto \Omega'\)
- \(\text{dom}(\Delta) = \text{dom}(\Omega')\)
- \(\Omega_2 \mapsto \Omega'\)

By Lemma 33 (Extension Transitivity)

\[\Gamma \vdash \text{inj}_k e_0 \mapsto \_\_y \mapsto \Delta \quad \text{By +I}_k\alpha_k\]

\[\Gamma \vdash \text{inj}_k e_0 \mapsto [\Gamma]_\alpha \_y \mapsto \Delta \quad \alpha \in \text{unsolved}(\Gamma)\]

• Case \[\Gamma, x : A'_1 \_p \vdash [\Omega]_0 \vdash A'_2 \_p \quad \text{Decl+I}_l\]

We have \([\Omega]A = A'_1 \rightarrow A'_2\). Either \([\Gamma]A = A_1 \rightarrow A_2\) where \(A'_1 = [\Omega]A_1\) and \(A'_2 = [\Omega]A_2\)—or \([\Gamma]A = \alpha\) and \([\Omega]\alpha = A'_1 \rightarrow A'_2\).
Proof of Theorem 12 \(\text{(Completeness of Algorithmic Typing)}\)

In the former case:

\[\Omega|\Gamma, x: A'_1 \vdash_p |\Omega)e_0 \iff A_2'\ p\]

Subderivation

\[A'_1 = |\Omega|A_1\]

Known in this subcase

\[= |\Omega||\Gamma|A_1\]

By Lemma 30 \(\text{(Substitution Invariance)}\)

\[|\Omega|A'_1 = |\Omega||\Omega||\Gamma|A_1\]

Applying \(\Omega\) on both sides

\[= |\Omega||\Gamma|A_1\]

By idempotence of substitution

\[|\Omega|\Gamma, x: A'_1 \vdash_p = |\Omega, x: A'_1 \vdash_p|(\Gamma, x: |\Gamma|A_1 \vdash_p)\]

By definition of context application

\[|\Omega, x: A'_1 \vdash_p|(\Gamma, x: |\Gamma|A_1 \vdash_p) \vdash_p |\Omega)e_0 \iff A_2'\ p\]

By above equality

\[\Gamma \rightarrow \Omega\]

Given

\[\Gamma, x: |\Gamma|A_1 \vdash_p \rightarrow \Omega, x: A'_1 \vdash_p\]

By \(\rightarrow\text{Var}\)

\[\text{dom}(\Gamma) = \text{dom}(\Omega)\]

Given

\[\text{dom}(\Gamma, x: |\Gamma|A_1 \vdash_p) = \Omega, x: A'_1 \vdash_p\]

By def. of \(\text{dom}(\vdash_p)\)

\[\Gamma, x: |\Gamma|A_1 \vdash_p \vdash_p e_0 \iff A_2' \vdash_p \Delta'\]

By i.h.

\[\Delta' \rightarrow \Omega'\]

By def. of context application

\[\Delta' = \Delta, x: \cdots, \Theta\]

By above equalities

\[\Delta \rightarrow \Omega'\]

By Lemma 22 \(\text{(Extension Inversion)}\) (v)

\[\text{dom}(\Delta) = \text{dom}(\Omega')\]

\[\Gamma, x: |\Gamma|A_1 \vdash_p \vdash_p e_0 \iff [\Gamma]A_2 \vdash_p \Delta, x: \cdots, p, \Theta\]

By above equality

\[\Gamma \vdash \lambda x. e_0 \iff [\Gamma]A_1 \rightarrow ([\Gamma]A_2) \vdash_p \Delta\]

By \(-I\)

\[\Gamma \vdash \lambda x. e_0 \iff [\Gamma](A_1 \rightarrow A_2) \vdash_p \Delta\]

By definition of substitution

In the latter case \((|\Gamma|A = \hat{\alpha} \in \text{unsolved}(\Gamma)\) and \([\Omega]\hat{\alpha} = A_1' \rightarrow A_2' = \tau_1' \rightarrow \tau_2')\):

By inversion on \(\Gamma \vdash \hat{\alpha} \text{ type}\), it must be the case that \(p = f\).

Since \(\hat{\alpha} \in \text{unsolved}(\Gamma)\), the context \(\Gamma\) must have the form \(\Gamma_0[\hat{\alpha} : \ast]\).

Let \(\Gamma_2 = \Gamma_0[\hat{\alpha}_1 : \ast, \hat{\alpha}_2 : \ast, \hat{\alpha} : \ast = \hat{\alpha}_1 \rightarrow \hat{\alpha}_2]\).

\[\Gamma \rightarrow \Gamma_2\]

By Lemma 23 \(\text{(Deep Evar Introduction)}\) (iii) twice

and Lemma 26 \(\text{(Parallel Admissibility)}\) (ii)

\[|\Omega]\hat{\alpha} = \tau_1' \rightarrow \tau_2'\]

Known in this subcase

\[\Gamma \rightarrow \Omega\]

Given

\[\Omega = \Omega_0[\hat{\alpha} : \ast = \tau_0]\]

By Lemma 22 \(\text{(Extension Inversion)}\) (vi)

Let \(\Omega_2 = \Omega_0[\hat{\alpha}_1 : \ast = \tau_1', \hat{\alpha}_1 : \ast = \tau_2', \hat{\alpha} : \ast = \hat{\alpha}_1 \rightarrow \hat{\alpha}_2]\).
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\( \Gamma \rightarrow \Gamma_2 \) 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)

\[ [\Omega] \Gamma, x: \tau'_1 \not\vdash [\Omega] e_0 \iff [\tau'_2] \not\vdash \] Subderivation
\[ [\Omega] \Gamma = [\Omega_2] \Gamma_2 \] By Lemma 57 (Multiple Confluence)
\[ \tau'_2 = [\Omega]\hat{\alpha}_2 \] From above equality
\[ \tau'_1 = [\Omega_2]\hat{\alpha}_1 \] Similar

\[ [\Omega_2] \Gamma_2, x: \tau'_1 \not\vdash [\Omega_2] e_0 \iff [\Omega_2]\hat{\alpha}_2 \not\vdash \] By def. of context application
\[ [\Omega_2, x: \tau'_1 \not\vdash [\Omega_2] e_0 \iff [\Omega_2]\hat{\alpha}_2 \not\vdash \] By above equalities
\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \] Given
\[ \text{dom}(\Gamma_2, x: \hat{\alpha}_1 \not\vdash) = \text{dom}(\Omega_2, x: \tau'_1 \not\vdash) \] By def. of \( \Gamma_2 \) and \( \Omega_2 \)
\[ \Gamma_2, x: \hat{\alpha}_1 \not\vdash e_0 \iff [\Gamma_2, x: \hat{\alpha}_1 \not\vdash] \hat{\alpha}_2 \not\vdash \Delta^+ \] By i.h.
\[ \Delta^+ \rightarrow \Omega^+ \] "
\[ \text{dom}(\Delta^+) = \text{dom}(\Omega^+) \] By def. of \( \Gamma_2 \) and \( \Omega_2 \)
\[ \Omega_2 \rightarrow \Omega^+ \] "

\[ \Gamma_2, x: \hat{\alpha}_1 \rightarrow \Delta^+ \not\vdash \] By Lemma 51 (Typing Extension)
\[ \Delta^+ = (\Delta, x: \hat{\alpha}_1 \not\vdash, \Delta_Z) \] By Lemma 22 (Extension Inversion) (v)
\[ \hat{\Omega}^+ = (\hat{\Omega}, x: \ldots \not\vdash, \Omega_Z) \] By Lemma 22 (Extension Inversion) (v)

\[ \Delta \rightarrow \hat{\Omega}' \] "
\[ \text{dom}(\Delta) = \text{dom}(\hat{\Omega}') \] "
\[ \Omega \rightarrow \Omega_2 \] Above
\[ \Omega \rightarrow \Omega^+ \] By Lemma 33 (Extension Transitivity)
\[ \Omega' \rightarrow \Omega' \] By Lemma 22 (Extension Inversion) (v)
\[ \Gamma \vdash \lambda x. e_0 \iff \hat{\alpha} \not\vdash \Delta \] By \(-I\hat{\alpha}\)
\[ \hat{\alpha} = [\Gamma]\hat{\alpha} \] \( \hat{\alpha} \in \text{unsolved}(\Gamma) \)
\[ \Gamma \vdash \lambda x. e_0 \iff [\Gamma]\hat{\alpha} \not\vdash \Delta \] By above equality
Proof of [Theorem 12](Completeness of Algorithmic Typing) thm:typing-completeness

- Case \[\Omega, x: [\Omega]A p \vdash [\Omega]v \iff [\Omega]A p\]
  \[\Omega, \Gamma \vdash \text{rec } x. [\Omega]v \iff [\Omega]A p\] DeqRec

  \[\Omega, x: [\Omega]A p \vdash [\Omega]v \iff [\Omega]A p\] Subderivation


  \[
  \begin{align*}
  \Gamma & \rightarrow \Omega \\
  \Gamma, x: [\Gamma]A p & \rightarrow \Omega, x: [\Omega]A p \\
  \text{dom}(\Gamma) & = \text{dom}(\Omega) \\
  \text{dom}(\Gamma, x: [\Gamma]A p) & = \Omega, x: [\Omega]A p \\
  \Gamma, x: [\Gamma]A p \vdash v & \iff [\Gamma]A p \vdash \Delta' \\
  \Delta' & \rightarrow \Omega_0' \\
  \text{dom}(\Delta') & = \text{dom}(\Omega_0') \\
  \Omega, x: [\Omega]A p & \rightarrow \Omega_0' \\
  \Omega_0' &= (\Omega', x: [\Omega]A p, \Theta) \\
  \Omega & \rightarrow \Omega'
  \end{align*}
  \]

  By Lemma 22 (Extension Inversion) (v)

  \[
  \begin{align*}
  \Gamma, x: [\Gamma]A p & \rightarrow \Delta' \\
  \Delta' &= (\Delta, x: \cdots, \Theta) \\
  \Delta, x: \cdots, \Theta & \rightarrow \Omega', x: [\Omega]A p, \Theta \\
  \Delta & \rightarrow \Omega' \\
  \text{dom}(\Delta) & = \text{dom}(\Omega')
  \end{align*}
  \]

  By Lemma 22 (Extension Inversion) (v)

  By above equalities

  By above equality

  By Rec

- Case \[\Omega \vdash [\Omega]e_0 \Rightarrow A \Theta \quad [\Omega] \vdash [\Omega]s_0 : A \Rightarrow C [p]\]
  \[\Omega \vdash [\Omega](e_0 s_0) \Rightarrow C p\] DeqE

  \[\Omega \vdash [\Omega]e_0 \Rightarrow A \Theta \] Subderivation

  \[\Gamma \vdash e_0 \Rightarrow A' \Theta \] 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 12](Completeness of Algorithmic Typing) thm:typing-completeness
Proof of Theorem 12 \textbf{(Completeness of Algorithmic Typing)}

\begin{proof}

\begin{align*}
\Gamma &\rightarrow \Omega \\
[\Omega] &\Gamma = [\Omega_{\Theta}] \Theta \\
[\Omega] &\Gamma \vdash [\Omega] s_0 : A \triangleright C [p] \\
[\Omega_{\Theta}] &\Theta \vdash [\Omega_{\Theta}] s_0 : [\Omega_{\Theta}] A' \triangleright C [p] \\
\Theta &\vdash s_0 : [\Theta] A' \triangleright C' [p] \dashv \Delta \\
\mu & C' = \Delta C' \\
\mu & \Delta \rightarrow \Omega' \\
\mu & \text{dom} (\Delta) = \text{dom} (\Omega') \\
\mu & \Omega_{\Theta} \rightarrow \Omega' \\
\mu & C = [\Omega'] C' \\
\Theta &\vdash s_0 : A' \triangleright C' [p] \dashv \Delta \\
\mu & \Omega \rightarrow \Omega' \\
\mu & \Gamma \vdash e_0 s_0 \Rightarrow C' p \dashv \Delta \\
\end{align*}

\end{proof}
Proof of **Theorem 12** *(Completeness of Algorithmic Typing)*

• Case

\[
[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A ! \gg C \ j \quad \text{for all } C_2.
\]

\[
[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A ! \gg C_2 \ j' \quad \text{then } C_2 = C
\]

\[
\Gamma \rightarrow \Omega \\
[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A ! \gg C \ j
\]

Given

\[
\Delta \rightarrow \Omega'
\]

Subderivation

\[
\Omega \rightarrow \Omega'
\]

By i.h.

\[
dom(\Delta) = dom(\Omega')
\]

\[
C = [\Omega']C'
\]

\[
C' = [\Delta]\Omega'
\]

Suppose, for a contradiction, that FEV([\Delta]C') \neq \emptyset.
That is, there exists some \(\hat{\alpha} \in \text{FEV}([\Delta]C')\).

\[
\Delta \rightarrow \Omega_2
\]

By Lemma 60 *(Split Solutions)*

\[
\Delta_1 \vdash [\hat{\alpha} : \kappa = t_1] \rightarrow \Omega'
\]

\[
[\Omega_2][\hat{\alpha}] = \Omega'_1 [\hat{\alpha} : \kappa = t_2]
\]

\[
t_2 \neq t_1
\]

\[
\{\text{NEQ}\} \quad [\Omega_2][\hat{\alpha}] \neq [\Omega'_2][\hat{\alpha}]
\]

\[
\{\text{EQ}\} \quad [\Omega_2][\hat{\beta}] = [\Omega'_2][\hat{\beta}] \quad \text{for all } \hat{\beta} \neq \hat{\alpha}
\]

By def. of subst. (\(t_2 \neq t_1\))

By construction of \(\Omega_2\) and \(\Omega_2\) canonical

Choose \(\hat{\alpha}_R\) such that \(\hat{\alpha}_R \in \text{FEV}(C')\) and either \(\hat{\alpha}_R = \hat{\alpha}\) or \(\hat{\alpha} \in \text{FEV}(\Delta[\hat{\alpha}_R])\).

Then either \(\hat{\alpha}_R = \hat{\alpha}\), or \(\hat{\alpha}_R\) is declared to the right of \(\hat{\alpha}\) in \(\Delta\).

\[
[\Omega_2]C' \neq [\Omega'_2]C'
\]

From (NEQ) and (EQ)

\[
\Gamma \vdash s : [\Gamma]A ! \gg C' \ j \rightarrow \Delta
\]

Above

\[
[\Omega_2]\Gamma \vdash [\Omega_2]s : [\Omega_2][\Gamma]A ! \gg [\Omega_2]C' \ j'
\]

By Theorem 9

\[
\Gamma \vdash s : [\Gamma]A ! \gg C' \ j \rightarrow \Delta
\]

Above

\[
\Gamma \vdash A ! \text{ type}
\]

By Lemma 13 *(Right-Hand Substitution for Typing)*

\[
\Gamma \vdash [\Gamma]A ! \text{ type}
\]

By inversion

\[
\text{FEV}([\Gamma]A) = \emptyset
\]

By property of \(\subseteq\)

\[
\text{FEV}([\Gamma]A) \subseteq \text{dom}(\cdot)
\]

\[
\Delta = (\Delta_L \cdot \Delta_R)
\]

By Lemma 72 *(Separation—Main)* (Spines)

\[
(\Delta \cdot \cdot) \rightarrow (\Delta_L \cdot \Delta_R)
\]

\[
\text{FEV}(C') \subseteq \text{dom}(\Delta_R)
\]

\[
\hat{\alpha}_L \in \text{FEV}(C')
\]

\[
\Delta_R \in \text{dom}(\Delta_R)
\]

\[
\text{dom}(\Delta_L) \cap \text{dom}(\Delta_R) = \emptyset
\]

\[
\hat{\alpha}_R \notin \text{dom}(\Delta_L)
\]

\[
\hat{\alpha}_R \notin \text{dom}(\Gamma)
\]

By Definition 5
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\[ \Omega_2 \vdash \Omega_2 s : [\Omega_2][\Gamma]A \gg [\Omega_2]C' \]

\[ FEV([\Gamma]A) = \emptyset \]
\[ FEV([\Gamma][\Gamma]A) = [\Omega_1][\Gamma]A \]
\[ \Gamma \vdash [\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 = \text{dom}(\Omega_1) \]
\[ \Omega_2 = \text{dom}(\Omega_2) \]
\[ \Omega_1 \vdash [\Gamma]A \text{ type} \]

\[ \Gamma \vdash [\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ [\Omega_1][\Gamma]A = [\Omega][\Gamma]A \]
\[ [\Omega_1][\Gamma]A = [\Omega][\Gamma][\Gamma]A \]
\[ [\Omega][\Gamma]A = [\Omega'][\Gamma]A = [\Omega][\Gamma]A \]
\[ \Omega_1 = \text{dom}(\Omega_1) \]
\[ \Omega_2 = \text{dom}(\Omega_2) \]
\[ \Omega_1 \vdash [\Gamma]A \text{ type} \]

\[ \Gamma \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ [\Omega_1][\Gamma]A = [\Omega][\Gamma][\Gamma]A \]
\[ [\Omega_1][\Gamma]A = [\Omega][\Gamma][\Gamma]A \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]

\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]

\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_2 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Omega_1 \vdash [\Gamma][\Gamma]A \text{ type} \]
\[ \Gamma \vdash \Delta : [\Gamma] A \ u \ [\Gamma] C \ u \ \gamma \Delta \quad \text{By}\ SpinePass \]

– Otherwise, \( p = ! \) and \( q = \gamma \). If \( \text{FEV}(C) \neq \emptyset \), we can apply \text{SpinePass} as above. If \( \text{FEV}(C) = \emptyset \), then we instead apply \text{SpineRecover}.

\[ \Gamma \vdash \Delta : [\Gamma] A \ u \ [\Gamma] C \ u \ \gamma \Delta \quad \text{By}\ SpineRecover \]

Here, \( q' = ! \) and \( q = \gamma \), so \( q' \sqsubseteq q \).

Case \( \Omega \Gamma \vdash \cdot : [\Omega] A \ p \ u [\Omega] A \ p \quad \text{DeclEmptySpine} \)

\[ \Gamma \vdash \Delta : [\Gamma] A \ p \ u [\Gamma] A \ p \quad \text{By}\ EmptySpine \]

\[ [\Gamma] A = [\Gamma][\Gamma] A \quad \text{By}\ idempotence\ of\ substitution \]

\[ \Gamma \rightarrow \Omega \quad \text{Given} \]

\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \quad \text{Given} \]

\[ [\Omega][\Gamma] A = [\Omega] A \quad \text{By}\ Lemma\ 29\ (Substitution\ Monotonicity)\ (iii) \]

\[ \Omega \rightarrow \Omega \quad \text{By}\ Lemma\ 32\ (Extension\ Reflexivity) \]

Case \( \Omega \Gamma \vdash B p : [\Omega] A \ q \ u [\Omega] A \ q \ u \ B p \quad \text{Decl}\ \rightarrow\ Spine \)

\[ \Omega \Gamma \vdash \cdot : [\Omega] A \ q \quad \text{Subderivation} \]

\[ \Theta \rightarrow \Omega \Theta \quad \text{By\ i.h.} \]

\[ \Omega \rightarrow \Omega \Theta \quad \text{"} \]

\[ A = [\Omega] A' \quad \text{"} \]

\[ A' = [\Theta] A' \quad \text{"} \]

\[ \Omega \Gamma \vdash [\Omega] s_0 : [\Omega] A_2 \ q \ u [\Omega] A_2 \ q \ u \ B p \quad \text{Subderivation} \]

\[ \Gamma \vdash s_0 : A_2 \ q \ u B p \ u \Delta \quad \text{By}\ i.h. \]

\[ \Delta \rightarrow \Omega' \quad \text{"} \]

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \quad \text{"} \]

\[ \Omega \rightarrow \Omega' \quad \text{"} \]

\[ B' = [\Delta] B' \quad \text{"} \]

\[ B = [\Omega'] B' \quad \text{"} \]

\[ \Gamma \vdash e_0 s_0 : A_1 \rightarrow A_2 \ q \ u B p \ u \Delta \quad \text{By}\ \rightarrow\ Spine \]
Proof of Theorem 12 \(\text{(Completeness of Algorithmic Typing)}\)

- **Case**

\[
\begin{align*}
[\Omega] \Gamma \vdash [\Omega] P & \text{ true} \\
[\Omega] \Gamma \vdash [\Omega] e & \equiv [\Omega] A_0 \ P \\
[\Omega] \Gamma \vdash [\Omega] e & \equiv ([\Omega] A_0) \land [\Omega] P
\end{align*}
\]

\(\text{Decl/}\triangledown\)

If \(e\) not a case, then:

- Subderivation
  - \(\Gamma \vdash P \rightarrow \neg \Theta\)
  - \(\Theta \rightarrow \Omega_0'\)
  - \(\Omega \rightarrow \Omega_0\)
  - \(\Gamma \rightarrow \Omega\)
  - \([\Omega] \Gamma = [\Omega] \Omega\)
  - \(\equiv [\Omega]_0 [\Omega]_0'\)
  - \(\equiv [\Omega]_0 [\Theta]\)
  - \(\Gamma \vdash A_0 \land P \ p \text{ type}\)
  - \(\Gamma \vdash A_0 \ p \text{ type}\)
  - \([\Omega] A_0 = [\Omega]_0 A_0\)

\(\text{By Lemma 33 (Extension Transitivity)}\)

\(\text{By Lemma 54 (Completing Stability)}\) (iii)

\(\text{By Lemma 56 (Confluence of Completeness)}\)

\(\text{Subderivation}\)

\(\Theta \vdash e \equiv [\Theta] A_0 \ p \vdash \Delta\)

\(\text{By i.h.}\)

\(\Delta \rightarrow \Omega'\)

\(\text{dom}(\Delta) = \text{dom}(\Omega')\)

\(\Omega_0' \rightarrow \Omega'\)

\(\Omega \rightarrow \Omega'\)

\(\Gamma \vdash e \equiv A_0 \land P \ p \vdash \Delta\)

\(\text{By } \land\)

Otherwise, we have \(e = \text{case}(e_0, \Pi)\). Let \(n\) be the height of the given derivation.

\(n - 1 \ [\Omega] \Gamma \vdash [\Omega](\text{case}(e_0, \Pi)) \equiv [\Omega] A_0 \ p\)

\(n - 2 \ [\Omega] \Gamma \vdash [\Omega] e_0 \Rightarrow B \ !\)

\(n - 2 \ [\Omega] \Gamma \vdash [\Omega] \Pi \vdash B \equiv [\Omega] A_0 \ p\)

\(n - 2 \ [\Omega] \Gamma \vdash [\Omega] \Pi \vdash \text{covers } B\)

\(n - 1 \ [\Omega] \Gamma \vdash [\Omega] P \text{ true}\)

\(n - 1 \ [\Omega] \Gamma \vdash [\Omega] \Pi \vdash B \equiv ([\Omega] A_0) \land ([\Omega] P) \ p\)

\(n - 1 \ [\Omega] \Gamma \vdash [\Omega] \Pi \vdash [\Omega] (A_0 \land P) \ p\)

\(\text{By def. of subst.}\)

\(\Gamma \vdash e_0 \Rightarrow B' \ ! \rightarrow \Theta\)

\(\Theta \rightarrow \Omega_0'\)

\(\Omega \rightarrow \Omega_0'\)

\(B = [\Omega]_0 B'\)

\(= [\Omega]_\theta [\Theta] B'\)

\(\text{By Lemma 30 (Substitution Invariance)}\)

\([\Omega] \Gamma = [\Omega]_0 [\Theta]\)

\([\Omega] (A_0 \land P) = [\Omega]_0 [(A_0 \land P)]\)

\(\text{By Lemma 57 (Multiple Confluence)}\)

\(\text{By Lemma 55 (Completing Completeness)}\) (ii)
Proof of [Theorem 12] (Completeness of Algorithmic Typing)

\[ n - 1 \quad [\Omega'_0] \Theta \vdash [\Omega'] \Pi \iff [\Omega'_0](A_0 \land P) p \quad \text{By above equalities} \]
\[ \Theta \vdash \Pi : [\Theta] B' \iff A_0 \land P \vdash \Delta \quad \text{By i.h.} \]
\[ \Delta \rightarrow \Omega' \]
\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]
\[ \Omega'_0 \rightarrow \Omega' \]
\[ \Theta \vdash \Pi \text{ covers } [\Theta] B' \quad \text{By Theorem 11} \]
\[ \Omega' \quad \text{By Lemma 33 (Extension Transitivity)} \]
\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff A_0 \land P \vdash \Delta \quad \text{By Case} \]

- **Case** \[ [\Omega] \Gamma \vdash \tau : \kappa \quad [\Omega] \Gamma \vdash e \iff [\tau/\alpha] [\Omega] A_0 \not\emptyset \]
  \[ [\Omega] \Gamma \vdash e \iff \exists \alpha : \kappa. [\Omega] A_0 \not\emptyset \]
  \[ \text{Subderivation} \]
  
  Let \( \Omega_0 = (\Omega, \& : * = \tau) \).
  
  \[ [\Omega] \Gamma = [\Omega_0](\Gamma, \& : *) \quad \text{By def. of context substitution} \]
  
  \[ [\Omega_0](\Gamma, \& : *) \vdash e \iff [\tau/\alpha] [\Omega] A_0 \not\emptyset \quad \text{By above equality} \]
  
  \[ [\tau/\alpha][\Omega] A_0 = [\Omega, \& : * = \tau][\&/\alpha] A_0 \quad \text{By a property of substitution} \]
  
  \[ [\Omega_0](\Gamma, \& : *) \vdash e \iff [\Omega_0][\&/\alpha] A_0 \not\emptyset \quad \text{By above equality} \]
  
  \[ \Gamma, \& : * \vdash e \iff [\&/\alpha] A_0 \not\emptyset \vdash \Delta \quad \text{By i.h.} \]
  
  \[ \Delta \rightarrow \Omega' \]
  
  \[ \text{dom}(\Delta) = \text{dom}(\Omega') \]
  
  \[ \Omega_0 \rightarrow \Omega' \]
  
  \[ \Omega \rightarrow \Omega_0 \quad \text{By AddSolved} \]
  
  \[ \Omega \rightarrow \Omega' \quad \text{By Lemma 33 (Extension Transitivity)} \]
  
  \[ \Gamma \vdash e \iff \exists \alpha : \kappa. A_0 \not\emptyset \vdash \Delta \quad \text{By [i]} \]

- **Case** \[ \text{DeclNil} \]
  Similar to the first part of the \[ \text{Decl\&l} \] case.

- **Case** \[ [\Omega] \Gamma \vdash (\{t\} \rightarrow \text{succ}(t_2)) \quad [\Omega] \Gamma \vdash e_1 \iff [\Omega] A_0 p \]
  
  \[ [\Omega] \Gamma \vdash (\{t\} \rightarrow \text{succ}(t_2)) \quad [\Omega] \Gamma \vdash e_2 \iff (\text{Vec } t_2 [\Omega] A_0) \not\emptyset \]
  
  \[ [\Omega] \Gamma \vdash (\{t\} e_1 : (\{t\} e_2) \iff (\text{Vec } \{t\}) [\Omega] A_0) p \quad \text{DeclCons} \]

Let \( \Omega^+ = (\Omega, \lt ; \& \& \& : N = t_2) \).

\[ [\Omega] \Gamma \vdash (\{t\} \rightarrow \text{succ}(t_2)) \quad \text{true} \quad \text{Subderivation} \]
\[ [\Omega^+](\Gamma, \lt ; \& \& \& : N) \vdash (\{t\} \rightarrow \text{succ}(\& \& \&)) \quad \text{true} \quad \text{Defs. of extension and subst.} \]
\[ \Gamma, \lt ; \& \& \& : N \vdash t = \text{succ}(\& \& \&) \quad \text{true} \quad \text{By Lemma 97 (Completeness of Checkprop)} \]
\[ \Gamma' \rightarrow \Omega'_0 \quad \text{true} \quad \text{By i.h.} \]
\[ \Omega' \rightarrow \Omega'_0 \quad \text{true} \quad \text{By i.h.} \]
Proof of Theorem 12 (Completeness of Algorithmic Typing).

Given \( \Gamma, \Delta, \alpha : \mathbb{N} \rightarrow \Omega' \),

\[
\Gamma \vdash_\alpha \text{Decl} \ \ \text{By Lemma 47 (Checkprop Extension)}
\]

\[
[\Omega] \Gamma = [\Omega_0] \Omega \\
= [\Omega_0^+] \Omega^+ \\
= [\Omega_0'] \Omega_0' \\
= [\Omega_0'] \Gamma' \\
\text{By def. of context application}
\]

\[
[\Omega] A_0 = [\Omega^+] A_0 \\
= [\Omega_0^+] A_0 \\
\text{By Lemma 55 (Completing Completeness) (ii)}
\]

\[
[\Omega] \Gamma \vdash [\Omega] e_1 \leftarrow [\Omega] A_0 \ p \\
\text{Subderivation}
\]

\[
[\Omega_0'] \Gamma' \vdash [\Omega_0'] e_2 \leftarrow [\Omega_0'] A_0 \ p \\
\text{By above equalities}
\]

\[
\begin{align*}
\Gamma' & \vdash e_1 \leftarrow [\Gamma'] A_0 \ p \vdash \Theta \\
\Theta & \rightarrow \Omega_0'' \\
\Omega_0' & \rightarrow \Omega_0''
\end{align*}
\]

\[
\begin{align*}
[\Omega] \Gamma & \vdash [\Omega] e_2 \leftarrow (\text{Vec } t_2 [\Omega] A_0) \ p \\
[\Omega] \Gamma & \vdash [\Omega] e_2 \leftarrow (\text{Vec } (\Omega^+) \alpha) [\Omega] A_0 \ p \\
[\Omega_0'] \Theta & \vdash [\Omega_0'] e_2 \leftarrow (\text{Vec } ([\Omega_0']^+) \alpha) [\Omega_0'] A_0 \ p \\
[\Omega_0'] \Theta & \vdash [\Omega_0'] e_2 \leftarrow ([\Omega_0']^+) (\text{Vec } \alpha A_0) \ p \\
\text{By def. of subst.}
\end{align*}
\]

\[
\begin{align*}
\Theta & \vdash e_2 \leftarrow [\Theta] A_0 \ p \vdash \Delta, \Gamma, \Delta' \\
\Theta & \rightarrow \Omega'' \\
\Omega_0'' & \rightarrow \Omega''
\end{align*}
\]

\[
\Delta, \Gamma, \Delta' \rightarrow \Omega'' \\
\text{By i.h.}
\]

\[
\begin{align*}
\Omega'' & = (\Omega^+, \alpha, \ldots) \\
\Delta & \rightarrow \Omega' \\
\text{By Lemma 22 (Extension Inversion) (ii)}
\end{align*}
\]

\[
\begin{align*}
dom(\Delta) & = dom(\Omega') \\
\text{By Lemma 33 (Extension Transitivity)}
\end{align*}
\]

\[
\begin{align*}
(\Gamma', \alpha, \ldots) & \rightarrow \Omega' \\
\text{By Lemma 33 (Extension Transitivity)}
\end{align*}
\]

\[
\begin{align*}
\Omega & \rightarrow \Omega' \\
\text{By Lemma 22 (Extension Inversion) (ii)}
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash e_1 :: e_2 \leftarrow (\text{Vec } t A_0) \ p \vdash \Delta \\
\text{By Cons}
\end{align*}
\]

\[
\text{Case} [\Omega] \Gamma \vdash [\Omega] e_1 \leftarrow A_1' \ p \\
[\Omega] \Gamma \vdash [\Omega] e_2 \leftarrow A_2' \ p
\]

\[
[\Omega] \Gamma \vdash ([\Omega] e_1, [\Omega] e_2) \leftarrow A_1' \times A_2' \ p \\
\text{Declx}
\]

Either \( [\Gamma] A = A_1 \times A_2 \) or \( [\Gamma] A = \alpha \in \text{unsolved}([\Gamma]) \).

- In the first case \( ([\Gamma] A = A_1 \times A_2) \), we have \( A_1' = [\Omega] A_1 \) and \( A_2' = [\Omega] A_2 \).
Proof of Theorem 12 (Completeness of Algorithmic Typing)

In the second case, where \( \Gamma A = \hat{\alpha} \), combine the corresponding subcase for \([\text{Decl} + l_k]\) with some straightforward additional reasoning about contexts (because here we have two subderivations, rather than one).

- Case

\[
\begin{align*}
\lbrack \Omega \rbrack \Gamma \vdash \lbrack \Omega \rbrack e_1 \leftarrow A_1 p & \quad \text{Subderivation} \\
\lbrack \Omega \rbrack \Gamma \vdash \lbrack \Omega \rbrack e_1 \leftarrow \lbrack \Omega \rbrack A_1 p & \quad \lbrack \Omega \rbrack A_1 = A_1' \\
\Gamma \vdash e_1 \leftarrow [\Gamma] A_1 p \vdash \Theta & \quad \text{By i.h.} \\
\Theta \rightarrow \Omega \Theta & \quad " \\
dom(\Theta) = \dom(\Omega_\Theta) & \quad " \\
\Omega \rightarrow \Omega \Theta & \quad " \\
\lbrack \Omega \rbrack \Gamma \vdash \lbrack \Omega \rbrack e_2 \leftarrow A_2 p & \quad \text{Subderivation} \\
\lbrack \Omega \rbrack \Gamma \vdash \lbrack \Omega \rbrack e_2 \leftarrow \lbrack \Omega \rbrack A_2 p & \quad \lbrack \Omega \rbrack A_2 = A_2' \\
\Gamma \rightarrow \Theta & \quad \text{By Lemma 51 (Typing Extension)} \\
\lbrack \Omega \rbrack \Gamma = [\Omega_\Theta] \Theta & \quad \text{By Lemma 57 (Multiple Confluence)} \\
\lbrack \Omega \rbrack A_2 = [\Omega_\Theta] A_2 & \quad \text{By Lemma 55 (Completing Completeness) (ii)} \\
\lbrack \Omega_\Theta \rbrack \Theta \vdash [\Omega \rbrack e_2 \leftarrow [\Omega_\Theta] A_2 p & \quad \text{By above equalities} \\
\Theta \vdash e_2 \leftarrow [\Gamma] A_2 p \vdash \Delta & \quad \text{By i.h.} \\
\Delta \rightarrow \Omega' & \quad " \\
\dom(\Delta) = \dom(\Omega') & \quad " \\
\Omega_\Theta \rightarrow \Omega' & \quad " \\
\Omega \rightarrow \Omega' & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma \vdash \langle e_1, e_2 \rangle \leftarrow (\lbrack \Gamma \rbrack A_1) \times (\lbrack \Gamma \rbrack A_2) p \vdash \Delta & \quad \text{By } \times \\
\Gamma \vdash \langle e_1, e_2 \rangle \leftarrow [\Gamma] (A_1 \times A_2) p \vdash \Delta & \quad \text{By def. of subst.}
\end{align*}
\]

- In the second case, where \( \lbrack \Gamma \rbrack A = \hat{\alpha} \), combine the corresponding subcase for \([\text{Decl} + l_k]\) with some straightforward additional reasoning about contexts (because here we have two subderivations, rather than one).

\[
\begin{align*}
\lbrack \Omega \rbrack \Gamma \vdash \lbrack \Omega \rbrack e_0 \Rightarrow C q & \quad \text{Subderivation} \\
\Gamma \vdash e_0 \Rightarrow C' q \vdash \Theta & \quad \text{By i.h.} \\
\Theta \rightarrow \Omega \Theta & \quad " \\
dom(\Theta) = \dom(\Omega \Theta) & \quad " \\
\Omega \rightarrow \Omega \Theta & \quad " \\
C = [\Omega_\Theta] C' & \quad " \\
\Theta \vdash C' q \text{ type} & \quad \text{By Lemma 63 (Well-Formed Outputs of Typing)} \\
\text{FEV}(C') = \emptyset & \quad \text{By inversion} \\
\lbrack \Omega_\Theta \rbrack C' = C' & \quad \text{By a property of substitution}
\end{align*}
\]
Proof of Theorem 12 (Completeness of Algorithmic Typing) thm:typing-completeness

Given

\[ \Gamma \longrightarrow \Theta \]
\[ \Delta \longrightarrow \Omega \]
\[ \Theta \longrightarrow \Omega \]
\[ [\Theta] \Gamma = [\Theta] \Delta \]
\[ \Omega \longrightarrow \Delta \]
\[ \Omega \longrightarrow \Theta \]
\[ \Omega \longrightarrow \Omega \]
\[ [\Omega] \Gamma = [\Omega] \Theta \]
\[ \Gamma \vdash A \text{ type} \]
\[ \Omega \vdash A \text{ type} \]
\[ [\Omega] \Delta \vdash [\Omega] \Pi \vdash C \leftarrow [\Omega] A \ p \]
\[ [\Omega] \Delta \vdash [\Omega] \Pi \vdash C' \leftarrow [\Omega] A \ p \]
\[ \theta \vdash \Pi \vdash C' \leftarrow [\Theta] A \ p \rightarrow \Delta \]

Case

\[ \text{dom} (\Delta) = \text{dom} (\Omega') \]
\[ \Omega \rightarrow \Omega' \]

\[ [\Omega] \Gamma \vdash [\Omega] \Pi \text{ covers } C \]

\[ [\Omega] \Gamma = [\Omega] \Delta \]
\[ = [\Omega'] \Delta \]
\[ [\Omega'] \Delta \vdash [\Omega] \Pi \text{ covers } C' \]
\[ \Delta \longrightarrow \Omega' \]

\[ \Gamma \vdash C' \text{ ! type} \]
\[ \Gamma \longrightarrow \Delta \]
\[ \Delta \vdash C' \text{! type} \]
\[ [\Delta] C' = C' \]
\[ \Delta \vdash \Pi \text{ covers } C' \]

\[ \Gamma \vdash \text{case}(e_0, \Pi) \leftarrow [\Gamma] A \ p \rightarrow \Delta \] By Case

Case

\[ [\Omega] \Gamma \vdash [\Omega] e_1 \leftarrow A_1 \ p \]
\[ [\Omega] \Gamma \vdash [\Omega] e_2 \leftarrow A_2 \ p \]
\[ [\Omega] \Gamma \vdash ([\Omega] e_1, [\Omega] e_2) \leftarrow A_1 \times A_2 \ p \]

Declx

Either \( A = \hat{\alpha} \) where \( [\Omega] \hat{\alpha} = A_1 \times A_2 \), or \( A = A_1^{\prime} \times A_2^{\prime} \) where \( A_1 = [\Omega] A_1^{\prime} \) and \( A_2 = [\Omega] A_2^{\prime} \). In the former case \((A = \hat{\alpha})\):

We have \( [\Omega] \hat{\alpha} = A_1 \times A_2 \). Therefore \( A_1 = [\Omega] A_1^{\prime} \) and \( A_2 = [\Omega] A_2^{\prime} \). Moreover, \( \Gamma = \Gamma_0 [\hat{\alpha} : \kappa] \).

\[ [\Omega] \Gamma \vdash [\Omega] e_1 \leftarrow [\Omega] A_1^{\prime} \ p \]

Subderivation

Let \( \Gamma' = \Gamma_0 [\hat{\alpha}_1 : \kappa, \hat{\alpha}_2 : \kappa, \hat{\alpha} : \kappa = \hat{\alpha}_1 + \hat{\alpha}_2] \).
Proof of Theorem 12 (Completeness of Algorithmic Typing)

Now we turn to parts (v) and (vi), completeness of matching.

In the latter case ($\Omega \rightarrow \Delta$):

\begin{align*}
[\Omega] \Gamma &\vdash [\Omega] e_1 \iff [\Omega] A'_1 \ p \quad \text{Subderivation} \\
[\Omega] \Gamma &\vdash [\Omega] e_1 \iff [\Omega] A'_1 \ p \quad \text{By above equality} \\
\Gamma &\vdash e_1 \iff [\Gamma] A'_1 \ p \vdash \Theta &\text{By i.h.} \\
\Theta &\rightarrow \Omega_1 &\text{"} \\
\Omega &\rightarrow \Omega_1 &\text{"} \\
\text{dom}(\Theta) &\rightarrow \text{dom}(\Omega_1) &\text{"} \\
\text{dom}(\Theta) &\rightarrow \text{dom}(\Omega_1) &\text{"} \\
\Gamma &\vdash (e_1, e_2) \iff \Theta \ p \vdash \Delta &\text{By \text{x} i.h.}
\end{align*}

By def. of context substitution

By Lemma 57 (Multiple Confluence)

By Lemma 55 (Completing Completeness) (ii)

By above equalities

By i.h.

Given ($A = A'_1 \times A'_2$):

\begin{align*}
[\Omega] \Gamma &\vdash [\Omega] e_2 \iff A_2 \ p \\
[\Omega] \Gamma &\vdash [\Omega] e_2 \iff [\Omega] A'_2 \ p \\
\Gamma &\vdash A'_1 \times A'_2 \ p \quad \text{type} &\text{Subderivation} \\
\Gamma &\vdash A'_2 \ p \quad \text{type} &\text{Given ($A = A'_1 \times A'_2$)} \\
\Gamma &\rightarrow \Omega &\text{By inversion} \\
\Gamma &\rightarrow \Omega &\text{By i.h.} \\
\Omega &\rightarrow \Omega &\text{By Lemma 33 (Extension Transitivity)} \\
\Omega_0 &\rightarrow A'_2 \ p &\text{type} \\
[\Omega] \Gamma &\vdash [\Omega] e_2 \iff [\Omega] A'_2 \ p &\text{By Lemma 29 (Substitution Monotonicity) (iii)} \\
[\Omega] \Gamma &\vdash [\Omega] e_2 \iff [\Omega] (\Theta A'_2) \ p &\text{By Lemma 57 (Multiple Confluence)} \\
\Theta &\vdash e_2 \iff [\Theta] A'_2 \ p \vdash \Delta &\text{By i.h.} \\
\text{dom}(\Delta) &\rightarrow \text{dom}(\Omega') &\text{"} \\
\text{dom}(\Delta) &\rightarrow \text{dom}(\Omega') &\text{"} \\
\Omega &\rightarrow \Omega &\text{By Lemma 33 (Extension Transitivity)} \\
\Gamma &\vdash (e_1, e_2) \iff ([\Omega] A_1) \times ([\Omega] A_2) \ p \vdash \Delta &\text{By \text{x} i.h.} \\
\Gamma &\vdash (e_1, e_2) \iff [\Omega] A_1 \times A_2 \ p \vdash \Delta &\text{By def. of substitution}
\end{align*}
Proof of Theorem 12  (Completeness of Algorithmic Typing)  thm:typing-completeness

- Case **DeclMatchEmpty**: Apply rule **MatchEmpty**

- Case **DeclMatchSeq**: Apply the i.h. twice, along with standard lemmas.

- Case **DeclMatchBase**: Apply the i.h. (i) and rule **MatchBase**

- Case **DeclMatchUnit**: Apply the i.h. and rule **MatchUnit**

- Case **DeclMatch×**: By i.h. and rule **Match×**

- Case **DeclMatch+**: By i.h. and rule **Match+**

- Case **DeclMatch∧**: By i.h. and rule **Match∧**

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) \( [\Omega]\Gamma / [\Omega]P \vdash \bar{p} \Rightarrow [\Omega]e :: [\Omega]\bar{A}! \Leftarrow [\Omega]C\ p\), and (6) \( p' \subseteq p\).

- Case **DeclMatchNeg**: By i.h. and rule **MatchNeg**

- Case **DeclMatchWild**: By i.h. and rule **MatchWild**

- Case **DeclMatchNil**: Similar to the **DeclMatch\^** case.

- Case **DeclMatchCons**: Similar to the **DeclMatch\^** and **DeclMatch\^** cases.

- Case **DeclMatch×**: By i.h. and rule **Match×**
Proof of Theorem 12 (Completeness of Algorithmic Typing)

\[\text{Case}\]
\[\Gamma \rightarrow \Omega \quad \text{Given}\]
\[\text{FEV}({\sigma} = {\tau}) = \emptyset \quad \text{Given}\]
\[|{\Omega}|{\sigma} = |{\Gamma}|{\sigma} \quad \text{By Lemma } 39 \text{ (Principal Agreement)} (i)\]
\[|{\Omega}|{\tau} = |{\Gamma}|{\tau} \quad \text{Similar}\]

\[\text{mg}u(|{\Omega}|{\sigma}, |{\Omega}|{\tau}) = \perp \quad \text{Given}\]
\[\text{mg}u(|{\Gamma}|{\sigma}, |{\Gamma}|{\tau}) = \perp \quad \text{By above equalities}\]
\[\Gamma / \sigma \Rightarrow \tau: \kappa \rightarrow \perp \quad \text{By Lemma } 94 \text{ (Completeness of Elimeq)} (2)\]
\[\Omega \rightarrow \Omega \quad \text{By Lemma } 32 \text{ (Extension Reflexivity)}\]
\[\text{dom}(\Gamma) = \text{dom}(\Omega) \quad \text{Given}\]

\[\text{Case}\]
\[\text{mg}u(|{\Omega}|{\sigma}, |{\Omega}|{\tau}) = \emptyset \quad \text{As in DeclMatch}\downarrow \text{ case}\]
\[\text{mg}u(|{\Gamma}|{\sigma}, |{\Gamma}|{\tau}) = \emptyset \quad \text{By above equalities}\]
\[\Gamma / \sigma \Rightarrow \tau: \kappa \rightarrow \perp (\Gamma, \Theta) \quad \text{By Lemma } 94 \text{ (Completeness of Elimeq)} (1)\]
\[\Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n) \quad \text{" for all } \Gamma \vdash : \kappa\]
\[\theta(|{\Omega}|{\Gamma}) \vdash \theta(\rho \Rightarrow |{\Omega}|e) :: \theta(|{\Omega}|A) \Leftrightarrow \theta(|{\Omega}|C) \quad \text{Subderivation}\]
\[\theta(|{\Omega}|{\Gamma}) = |{\Omega}|{\uparrow}_p, \Theta|{\Gamma}|{\uparrow}_p, \Theta \quad \text{By Lemma } 95 \text{ (Substitution Upgrade)} (iii)\]
\[\theta(|{\Omega}|A) = |{\Omega}|{\uparrow}_p, \Theta|A \quad \text{By Lemma } 95 \text{ (Substitution Upgrade)} (i) \text{ (over } A)\]
\[\theta(|{\Omega}|C) = |{\Omega}|{\uparrow}_p, \Theta|C \quad \text{By Lemma } 95 \text{ (Substitution Upgrade)} (i)\]
\[\theta(\rho \Rightarrow |{\Omega}|e) = |{\Omega}|{\uparrow}_p, \Theta|\rho \Rightarrow e) \quad \text{By Lemma } 95 \text{ (Substitution Upgrade)} (iv)\]

\[|{\Omega}|{\uparrow}_p, \Theta|{\Gamma}|{\uparrow}_p, \Theta \vdash |{\Omega}|{\uparrow}_p, \Theta|A \Leftrightarrow |{\Omega}|{\uparrow}_p, \Theta|C \quad \text{By above equalities}\]

\[\Gamma, {\uparrow}_p, \Theta \vdash (\rho \Rightarrow e) :: |{\Gamma}|{\uparrow}_p, \Theta|A \Leftrightarrow |{\Gamma}|{\uparrow}_p, \Theta|C \vdash \Delta, {\uparrow}_p, \Delta' \quad \text{By i.h.}\]
\[\Delta, {\uparrow}_p, \Delta' \rightarrow \Omega', {\uparrow}_p, \Omega'' \quad \text{"}\]
\[\Omega', {\uparrow}_p, \Theta \rightarrow \Omega', {\uparrow}_p, \Omega'' \quad \text{"}\]
\[\text{dom}(\Delta, {\uparrow}_p, \Delta') = \text{dom}(\Omega', {\uparrow}_p, \Omega'') \quad \text{"}\]

\[\Delta \rightarrow \Omega' \quad \text{By Lemma } 22 \text{ (Extension Inversion)} (ii)\]
\[\text{dom}(\Delta) = \text{dom}(\Omega') \quad \text{"}\]
\[\Omega \rightarrow \Omega' \quad \text{By Lemma } 22 \text{ (Extension Inversion)} (ii)\]
\[\Gamma / |{\Gamma}|{\sigma} = |{\Gamma}|{\tau} \vdash \rho \Rightarrow e :: |{\Gamma}|A \Leftrightarrow |{\Gamma}|C \vdash \Delta \quad \text{By MatchUnify}\]