Sound and Complete Bidirectional Typechecking for Higher-Rank Polymorphism with Existentials and Indexed Types (POPL 2016 submission #36): Full definitions, lemmas and proofs

Joshua Dunfield Neelakantan R. Krishnaswami

July 9, 2015

Contents

1 List of Judgments 7
2 Figures from paper, repeated 8
3 Figures omitted from paper 17
A Properties of the Declarative System 24
1 Lemma (Declarative Well-foundedness) ........................................... 24
2 Lemma (Declarative Weakening) ........................................... 24
3 Lemma (Declarative Term Substitution) ........................................... 24
4 Lemma (Reflexivity of Declarative Subtyping) ........................................... 24
5 Lemma (Subtyping Inversion) ........................................... 24
6 Lemma (Subtyping Polarity Flip) ........................................... 24
7 Lemma (Transitivity of Declarative Subtyping) ........................................... 24
B Substitution and Well-formedness Properties 25
8 Lemma (Substitution—Well-formedness) ........................................... 25
9 Lemma (Uvar Preservation) ........................................... 25
10 Lemma (Sorting Implies Typing) ........................................... 25
11 Lemma (Right-Hand Substitution for Sorting) ........................................... 25
12 Lemma (Right-Hand Substitution for Propositions) ........................................... 25
13 Lemma (Right-Hand Substitution for Typing) ........................................... 25
14 Lemma (Substitution for Sorting) ........................................... 25
15 Lemma (Substitution for Prop Well-Formedness) ........................................... 25
16 Lemma (Substitution for Type Well-Formedness) ........................................... 25
17 Lemma (Substitution Stability) ........................................... 25
18 Lemma (Equal Domains) ........................................... 25
C Properties of Extension 25
19 Lemma (Declaration Preservation) ........................................... 25
20 Lemma (Declaration Order Preservation) ........................................... 25
21 Lemma (Reverse Declaration Order Preservation) ........................................... 25
22 Lemma (Extension Inversion) ........................................... 25
23 Lemma (Deep Evar Introduction) ........................................... 26
24 Lemma (Soft Extension) ........................................... 26
26 Lemma (Parallel Admissibility) ........................................... 26
27 Lemma (Parallel Extension Solution) ........................................... 27
28 Lemma (Parallel Variable Update) ........................................... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lemma (Substitution Monotonicity)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Substitution Invariance)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Split Extension)</td>
<td>27</td>
</tr>
<tr>
<td>C.1</td>
<td>Reflexivity and Transitivity</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Extension Reflexivity)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Extension Transitivity)</td>
<td>27</td>
</tr>
<tr>
<td>C.2</td>
<td>Weakening</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Suffix Weakening)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Suffix Weakening)</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Extension Weakening (Sorts))</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Extension Weakening (Props))</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Lemma (Extension Weakening (Types))</td>
<td>27</td>
</tr>
<tr>
<td>C.3</td>
<td>Principal Typing Properties</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Principal Agreement)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Right-Hand Subst. for Principal Typing)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Extension Weakening for Principal Typing)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Inversion of Principal Typing)</td>
<td>28</td>
</tr>
<tr>
<td>C.4</td>
<td>Instantiation Extends</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Instantiation Extension)</td>
<td>28</td>
</tr>
<tr>
<td>C.5</td>
<td>Equivalence Extends</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Elimeq Extension)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Elimprop Extension)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Checkeq Extension)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Checkprop Extension)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Prop Equivalence Extension)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Equivalence Extension)</td>
<td>28</td>
</tr>
<tr>
<td>C.6</td>
<td>Subtyping Extends</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Subtyping Extension)</td>
<td>28</td>
</tr>
<tr>
<td>C.7</td>
<td>Typing Extends</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Lemma (Typing Extension)</td>
<td>28</td>
</tr>
<tr>
<td>C.8</td>
<td>Unfiled</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Context Partitioning)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Completing Stability)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Completing Completeness)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Confluence of Completeness)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Multiple Confluence)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Canonical Completion)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Split Solutions)</td>
<td>29</td>
</tr>
<tr>
<td>D</td>
<td>Internal Properties of the Declarative System</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Interpolating With and Exists)</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Lemma (Case Invertibility)</td>
<td>30</td>
</tr>
<tr>
<td>E</td>
<td>Miscellaneous Properties of the Algorithmic System</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Well-Formed Outputs of Typing)</td>
<td>30</td>
</tr>
<tr>
<td>F</td>
<td>Decidability of Instantiation</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Left Unsolvedness Preservation)</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Left Free Variable Preservation)</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Instantiation Size Preservation)</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Decidability of Instantiation)</td>
<td>30</td>
</tr>
<tr>
<td>G</td>
<td>Separation</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Transitivity of Separation)</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Lemma (Separation Truncation)</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Lemma (Separation for Auxiliary Judgments)</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Lemma (Separation for Subtyping)</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Lemma (Separation—Main)</td>
<td>31</td>
</tr>
</tbody>
</table>
#### CONTENTS

**H Decidability of Algorithmic Subtyping**

- H.1 Lemmas for Decidability of Subtyping
  - 73 Lemma (Substitution Isn’t Large)
  - 74 Lemma (Instantiation Solves)
  - 75 Lemma (Checkeq Solving)
  - 76 Lemma (Prop Equiv Solving)
  - 77 Lemma (Equiv Solving)
  - 78 Lemma (Decidability of Propositional Judgments)
  - 79 Lemma (Decidability of Equivalence)
- H.2 Decidability of Subtyping
  - 1 Theorem (Decidability of Subtyping)
- H.3 Decidability of Matching and Coverage
  - 80 Lemma (Decidability of Expansion Judgments)
  - 2 Theorem (Decidability of Coverage)
- H.4 Decidability of Typing
  - 8 Theorem (Decidability of Typing)

**I Determinacy**

- 81 Lemma (Determinacy of Auxiliary Judgments)
- 82 Lemma (Determinacy of Equivalence)
- 4 Theorem (Determinacy of Subtyping)
- 5 Theorem (Determinacy of Typing)

**J Properties of Algorithmic Subtyping**

**K Soundness**

- K.1 Soundness of Instantiation
  - 83 Lemma (Soundness of Instantiation)
- K.2 Soundness of Checkeq
  - 84 Lemma (Soundness of Checkeq)
- K.3 Soundness of Equivalence (Propositions and Types)
  - 85 Lemma (Soundness of Propositional Equivalence)
  - 86 Lemma (Soundness of Algorithmic Equivalence)
- K.4 Soundness of Checkprop
  - 87 Lemma (Soundness of Checkprop)
- K.5 Soundness of Eliminations (Equality and Proposition)
  - 88 Lemma (Soundness of Equality Elimination)
- K.6 Soundness of Subtyping
  - 6 Theorem (Soundness of Algorithmic Subtyping)
- K.7 Soundness of Typing
  - 7 Theorem (Soundness of Match Coverage)
  - 89 Lemma (Well-formedness of Algorithmic Typing)
  - 8 Theorem (Soundness of Algorithmic Typing)

**L Completeness**

- L.1 Completeness of Auxiliary Judgments
  - 90 Lemma (Completeness of Instantiation)
  - 91 Lemma (Completeness of Checkeq)
  - 92 Lemma (Completeness of Elimprop)
  - 93 Lemma (Substitution Upgrade)
  - 94 Lemma (Completeness of Propequiv)
  - 95 Lemma (Completeness of Checkprop)
- L.2 Completeness of Equivalence and Subtyping
  - 96 Lemma (Completeness of Equiv)
  - 9 Theorem (Completeness of Subtyping)
- L.3 Completeness of Typing
  - 10 Theorem (Completeness of Match Coverage)
  - 11 Theorem (Completeness of Algorithmic Typing)
## Proofs

### A'. Properties of the Declarative System

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Proof of Lemma (Declarative Well-foundedness)</td>
</tr>
<tr>
<td>2</td>
<td>Proof of Lemma (Declarative Weakening)</td>
</tr>
<tr>
<td>3</td>
<td>Proof of Lemma (Declarative Term Substitution)</td>
</tr>
<tr>
<td>4</td>
<td>Proof of Lemma (Reflexivity of Declarative Subtyping)</td>
</tr>
<tr>
<td>5</td>
<td>Proof of Lemma (Subtyping Inversion)</td>
</tr>
<tr>
<td>6</td>
<td>Proof of Lemma (Subtyping Polarity Flip)</td>
</tr>
<tr>
<td>7</td>
<td>Proof of Lemma (Transitivity of Declarative Subtyping)</td>
</tr>
</tbody>
</table>

### B'. Substitution and Well-formedness Properties

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Proof of Lemma (Substitution—Well-formedness)</td>
</tr>
<tr>
<td>9</td>
<td>Proof of Lemma (Uvar Preservation)</td>
</tr>
<tr>
<td>10</td>
<td>Proof of Lemma (Sorting Implies Typing)</td>
</tr>
<tr>
<td>11</td>
<td>Proof of Lemma (Right-Hand Substitution for Sorting)</td>
</tr>
<tr>
<td>12</td>
<td>Proof of Lemma (Right-Hand Substitution for Propositions)</td>
</tr>
<tr>
<td>13</td>
<td>Proof of Lemma (Right-Hand Substitution for Typing)</td>
</tr>
<tr>
<td>14</td>
<td>Proof of Lemma (Substitution for Sorting)</td>
</tr>
<tr>
<td>15</td>
<td>Proof of Lemma (Substitution for Prop Well-Formedness)</td>
</tr>
<tr>
<td>16</td>
<td>Proof of Lemma (Substitution for Type Well-Formedness)</td>
</tr>
<tr>
<td>17</td>
<td>Proof of Lemma (Substitution Stability)</td>
</tr>
<tr>
<td>18</td>
<td>Proof of Lemma (Equal Domains)</td>
</tr>
</tbody>
</table>

### C'. Properties of Extension

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Proof of Lemma (Declaration Preservation)</td>
</tr>
<tr>
<td>20</td>
<td>Proof of Lemma (Declaration Order Preservation)</td>
</tr>
<tr>
<td>21</td>
<td>Proof of Lemma (Reverse Declaration Order Preservation)</td>
</tr>
<tr>
<td>22</td>
<td>Proof of Lemma (Extension Inversion)</td>
</tr>
<tr>
<td>23</td>
<td>Proof of Lemma (Deep Evar Introduction)</td>
</tr>
<tr>
<td>24</td>
<td>Proof of Lemma (Parallel Admissibility)</td>
</tr>
<tr>
<td>25</td>
<td>Proof of Lemma (Parallel Extension Solution)</td>
</tr>
<tr>
<td>26</td>
<td>Proof of Lemma (Parallel Variable Update)</td>
</tr>
<tr>
<td>27</td>
<td>Proof of Lemma (Substitution Monotonicity)</td>
</tr>
<tr>
<td>28</td>
<td>Proof of Lemma (Substitution Invariance)</td>
</tr>
<tr>
<td>29</td>
<td>Proof of Lemma (Soft Extension)</td>
</tr>
<tr>
<td>30</td>
<td>Proof of Lemma (Split Extension)</td>
</tr>
</tbody>
</table>

#### C'.1 Reflexivity and Transitivity

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Proof of Lemma (Extension Reflexivity)</td>
</tr>
<tr>
<td>33</td>
<td>Proof of Lemma (Extension Transitivity)</td>
</tr>
</tbody>
</table>

#### C'.2 Weakening

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Proof of Lemma (Suffix Weakening)</td>
</tr>
<tr>
<td>35</td>
<td>Proof of Lemma (Suffix Weakening)</td>
</tr>
<tr>
<td>36</td>
<td>Proof of Lemma (Extension Weakening (Sorts))</td>
</tr>
<tr>
<td>37</td>
<td>Proof of Lemma (Extension Weakening (Props))</td>
</tr>
<tr>
<td>38</td>
<td>Proof of Lemma (Extension Weakening (Types))</td>
</tr>
</tbody>
</table>

#### C'.3 Principal Typing Properties

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>Proof of Lemma (Principal Agreement)</td>
</tr>
<tr>
<td>40</td>
<td>Proof of Lemma (Right-Hand Subst. for Principal Typing)</td>
</tr>
<tr>
<td>41</td>
<td>Proof of Lemma (Extension Weakening for Principal Typing)</td>
</tr>
<tr>
<td>42</td>
<td>Proof of Lemma (Inversion of Principal Typing)</td>
</tr>
</tbody>
</table>

#### C'.4 Instantiation Extends

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>Proof of Lemma (Instantiation Extension)</td>
</tr>
</tbody>
</table>

#### C'.5 Equivalence Extends

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>Proof of Lemma (Elimeq Extension)</td>
</tr>
<tr>
<td>45</td>
<td>Proof of Lemma (Elimprop Extension)</td>
</tr>
<tr>
<td>46</td>
<td>Proof of Lemma (Checkeq Extension)</td>
</tr>
<tr>
<td>47</td>
<td>Proof of Lemma (Checkprop Extension)</td>
</tr>
<tr>
<td>48</td>
<td>Proof of Lemma (Prop Equivalence Extension)</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>C'6.5</td>
<td>Proof of Lemma (Equivalence Extension)</td>
</tr>
<tr>
<td>C'6.6</td>
<td>Proof of Lemma (Subtyping Extension)</td>
</tr>
<tr>
<td>C'6.7</td>
<td>Proof of Lemma (Typing Extension)</td>
</tr>
<tr>
<td>C'8.1</td>
<td>Proof of Lemma (Context Partitioning)</td>
</tr>
<tr>
<td>C'8.2</td>
<td>Proof of Lemma (Completing Stability)</td>
</tr>
<tr>
<td>C'8.3</td>
<td>Proof of Lemma (Completing Completeness)</td>
</tr>
<tr>
<td>C'8.4</td>
<td>Proof of Lemma (Confluence of Completeness)</td>
</tr>
<tr>
<td>C'8.5</td>
<td>Proof of Lemma (Multiple Confluence)</td>
</tr>
<tr>
<td>C'8.6</td>
<td>Proof of Lemma (Canonical Completion)</td>
</tr>
<tr>
<td>C'8.7</td>
<td>Proof of Lemma (Split Solutions)</td>
</tr>
<tr>
<td>D'</td>
<td>Internal Properties of the Declarative System</td>
</tr>
<tr>
<td>D'1.1</td>
<td>Proof of Lemma (Interpolating With and Exists)</td>
</tr>
<tr>
<td>D'1.2</td>
<td>Proof of Lemma (Case Invertibility)</td>
</tr>
<tr>
<td>E'</td>
<td>Miscellaneous Properties of the Algorithmic System</td>
</tr>
<tr>
<td>E'1.1</td>
<td>Proof of Lemma (Well-Formed Outputs of Typing)</td>
</tr>
<tr>
<td>F'</td>
<td>Decidability of Instantiation</td>
</tr>
<tr>
<td>F'1.1</td>
<td>Proof of Lemma (Left Unsolvedness Preservation)</td>
</tr>
<tr>
<td>F'1.2</td>
<td>Proof of Lemma (Left Free Variable Preservation)</td>
</tr>
<tr>
<td>F'1.3</td>
<td>Proof of Lemma (Instantiation Size Preservation)</td>
</tr>
<tr>
<td>F'1.4</td>
<td>Proof of Lemma (Decidability of Instantiation)</td>
</tr>
<tr>
<td>G'</td>
<td>Separation</td>
</tr>
<tr>
<td>G'1.1</td>
<td>Proof of Lemma (Transitivity of Separation)</td>
</tr>
<tr>
<td>G'1.2</td>
<td>Proof of Lemma (Separation Truncation)</td>
</tr>
<tr>
<td>G'1.3</td>
<td>Proof of Lemma (Separation for Auxiliary Judgments)</td>
</tr>
<tr>
<td>G'1.4</td>
<td>Proof of Lemma (Separation for Subtyping)</td>
</tr>
<tr>
<td>G'1.5</td>
<td>Proof of Lemma (Separation—Main)</td>
</tr>
<tr>
<td>H'</td>
<td>Decidability of Algorithmic Subtyping</td>
</tr>
<tr>
<td>H'1.1</td>
<td>Lemmas for Decidability of Subtyping</td>
</tr>
<tr>
<td>H'1.2</td>
<td>Proof of Lemma (Substitution Isn't Large)</td>
</tr>
<tr>
<td>H'1.3</td>
<td>Proof of Lemma (Instantiation Solves)</td>
</tr>
<tr>
<td>H'1.4</td>
<td>Proof of Lemma (Checkeq Solving)</td>
</tr>
<tr>
<td>H'1.5</td>
<td>Proof of Lemma (Prop Equiv Solving)</td>
</tr>
<tr>
<td>H'1.6</td>
<td>Proof of Lemma (Equiv Solving)</td>
</tr>
<tr>
<td>H'1.7</td>
<td>Proof of Lemma (Decidability of Propositional Judgments)</td>
</tr>
<tr>
<td>H'1.8</td>
<td>Proof of Lemma (Decidability of Equivalence)</td>
</tr>
<tr>
<td>H'2.1</td>
<td>Decidability of Subtyping</td>
</tr>
<tr>
<td>H'2.2</td>
<td>Proof of Theorem (Decidability of Subtyping)</td>
</tr>
<tr>
<td>H'3.1</td>
<td>Decidability of Matching and Coverage</td>
</tr>
<tr>
<td>H'3.2</td>
<td>Proof of Lemma (Decidability of Expansion Judgments)</td>
</tr>
<tr>
<td>H'3.3</td>
<td>Proof of Theorem (Decidability of Coverage)</td>
</tr>
<tr>
<td>H'4.1</td>
<td>Decidability of Typing</td>
</tr>
<tr>
<td>H'4.2</td>
<td>Proof of Theorem (Decidability of Typing)</td>
</tr>
<tr>
<td>I'</td>
<td>Determinacy</td>
</tr>
<tr>
<td>I'1.1</td>
<td>Proof of Lemma (Determinacy of Auxiliary Judgments)</td>
</tr>
<tr>
<td>I'1.2</td>
<td>Proof of Lemma (Determinacy of Equivalence)</td>
</tr>
<tr>
<td>I'1.3</td>
<td>Proof of Theorem (Determinacy of Subtyping)</td>
</tr>
<tr>
<td>I'1.4</td>
<td>Proof of Theorem (Determinacy of Typing)</td>
</tr>
<tr>
<td>J'</td>
<td>Properties of Algorithmic Subtyping</td>
</tr>
</tbody>
</table>
## K. Soundness

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>K.1 Instantiation</td>
<td>105</td>
</tr>
<tr>
<td>83 Proof of Lemma (Soundness of Instantiation)</td>
<td>105</td>
</tr>
<tr>
<td>84 Proof of Lemma (Soundness of Checkeq)</td>
<td>105</td>
</tr>
<tr>
<td>85 Proof of Lemma (Soundness of Propositional Equivalence)</td>
<td>106</td>
</tr>
<tr>
<td>86 Proof of Lemma (Soundness of Algorithmic Equivalence)</td>
<td>107</td>
</tr>
<tr>
<td>K.2 Soundness of Checkprop</td>
<td>109</td>
</tr>
<tr>
<td>87 Proof of Lemma (Soundness of Checkprop)</td>
<td>109</td>
</tr>
<tr>
<td>K.3 Soundness of Eliminations (Equality and Proposition)</td>
<td>109</td>
</tr>
<tr>
<td>88 Proof of Lemma (Soundness of Equality Elimination)</td>
<td>109</td>
</tr>
<tr>
<td>90 Proof of Theorem (Soundness of Algorithmic Subtyping)</td>
<td>111</td>
</tr>
<tr>
<td>K.4 Soundness of Typing</td>
<td>113</td>
</tr>
<tr>
<td>91 Proof of Theorem (Soundness of Match Coverage)</td>
<td>113</td>
</tr>
<tr>
<td>92 Proof of Lemma (Well-formedness of Algorithmic Typing)</td>
<td>114</td>
</tr>
<tr>
<td>93 Proof of Theorem (Soundness of Algorithmic Typing)</td>
<td>116</td>
</tr>
</tbody>
</table>

## L. Completeness

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>L.1 Completeness of Auxiliary Judgments</td>
<td>126</td>
</tr>
<tr>
<td>90 Proof of Lemma (Completeness of Instantiation)</td>
<td>126</td>
</tr>
<tr>
<td>91 Proof of Lemma (Completeness of Checkeq)</td>
<td>128</td>
</tr>
<tr>
<td>92 Proof of Lemma (Completeness of Elimeq)</td>
<td>129</td>
</tr>
<tr>
<td>93 Proof of Lemma (Substitution Upgrade)</td>
<td>131</td>
</tr>
<tr>
<td>94 Proof of Lemma (Completeness of Propequiv)</td>
<td>131</td>
</tr>
<tr>
<td>95 Proof of Lemma (Completeness of Checkprop)</td>
<td>132</td>
</tr>
<tr>
<td>L.2 Completeness of Equivalence and Subtyping</td>
<td>132</td>
</tr>
<tr>
<td>96 Proof of Lemma (Completeness of Equiv)</td>
<td>132</td>
</tr>
<tr>
<td>97 Proof of Theorem (Completeness of Subtyping)</td>
<td>135</td>
</tr>
<tr>
<td>L.3 Completeness of Typing</td>
<td>139</td>
</tr>
<tr>
<td>98 Proof of Theorem (Completeness of Match Coverage)</td>
<td>139</td>
</tr>
<tr>
<td>99 Proof of Theorem (Completeness of Algorithmic Typing)</td>
<td>140</td>
</tr>
</tbody>
</table>
## 1 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 15</td>
</tr>
<tr>
<td>$\Psi \vdash P \ prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Psi \vdash A \ type$</td>
<td>Type is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Psi \vdash A \ types$</td>
<td>Type vector is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Psi \vDash \text{ctx}$</td>
<td>Declarative context is well-formed</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Psi \vdash A \leq_{\pm} B$</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>$\Psi \vdash e \ll A \ p$</td>
<td>Declarative checking</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash e \Rightarrow A \ p$</td>
<td>Declarative synthesis</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg C \ q$</td>
<td>Declarative spine typing</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg C \ lq$</td>
<td>Declarative spine typing, recovering principality</td>
<td>Figure 6</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi :: \bar{A} \ll C \ p$</td>
<td>Declarative pattern matching</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi \Pi :: \bar{A} \ll C \ p$</td>
<td>Declarative proposition assumption</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi \ covers \bar{A}$</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 16</td>
</tr>
<tr>
<td>$\Gamma \vdash P \ prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Gamma \vdash A \ type$</td>
<td>Polytype is well-formed</td>
<td>Figure 16</td>
</tr>
<tr>
<td>$\Gamma \vDash \text{ctx}$</td>
<td>Algorithmic context is well-formed</td>
<td>Figure 16</td>
</tr>
</tbody>
</table>

| [\Gamma] | Applying a context, as a substitution, to a type | Figure 11 |
| $\Gamma \vdash P \ true \vdash \Delta$ | Check proposition | Figure 17 |
| $\Gamma / P \vdash \Delta$ | Assume proposition | Figure 17 |
| $\Gamma \vdash s \ll t : \kappa \vdash \Delta$ | Check equation | Figure 18 |
| $s \# t$ | Head constructors clash | Figure 19 |
| $\Gamma / s \ll t : \kappa \vdash \Delta$ | Assume/eliminate equation | Figure 20 |
| $\Gamma \vdash A <_{\pm} B \vdash \Delta$ | Algorithmic subtyping | Figure 21 |
| $\Gamma / P \vdash A <_{\pm} B \vdash \Delta$ | Assume/eliminate proposition | Figure 21 |
| $\Gamma \vdash P \equiv Q \vdash \Delta$ | Equivalence of propositions | Figure 21 |
| $\Gamma \vdash A \equiv B \vdash \Delta$ | Equivalence of types | Figure 21 |
| $\Gamma \vdash \& := t : \kappa \vdash \Delta$ | Instantiate | Figure 22 |
| $e \ chk-I$ | Checking intro form | Figure 5 |
| $\Gamma \vdash e \ll A \ p \vdash \Delta$ | Algorithmic checking | Figure 14 |
| $\Gamma \vdash e \Rightarrow A \ p \vdash \Delta$ | Algorithmic synthesis | Figure 14 |
| $\Gamma \vdash s : A \ p \gg C \ q \vdash \Delta$ | Algorithmic spine typing | Figure 14 |
| $\Gamma \vdash s : A \ p \gg C \ lq \vdash \Delta$ | Algorithmic spine typing, recovering principality | Figure 14 |
| $\Gamma \vdash \Pi :: \bar{A} \ll C \ p \vdash \Delta$ | Algorithmic pattern matching | Figure 23 |
| $\Gamma / P \vdash \Pi :: \bar{A} \ll C \ p \vdash \Delta$ | Algorithmic pattern matching (assumption) | Figure 23 |
| $\Gamma \vdash \Pi \ covers \bar{A}$ | Algorithmic match coverage | Figure 24 |
| $\Gamma \vdash \Delta$ | Context extension | Figure 13 |
| $[\Omega] \Gamma$ | Apply complete context | Figure 12 |
Expressions
\[ e ::= x \mid \lambda x.e \mid e_1(e_2 \cdot s) \mid (e : A) \]
\[ \langle e_1, e_2 \rangle \mid \text{inj}_1 e \mid \text{inj}_2 e \mid \text{case}(e, \Pi) \]

Values
\[ v ::= x \mid \lambda x.e \mid \langle v : A \rangle \]
\[ \langle v_1, v_2 \rangle \mid \text{inj}_1 v \mid \text{inj}_2 v \]

Spines
\[ s ::= \cdot \mid e \cdot s \]

Patterns
\[ \rho ::= x \mid \langle \rho_1, \rho_2 \rangle \mid \text{inj}_1 \rho \mid \text{inj}_2 \rho \]

Branches
\[ \pi ::= \rho \Rightarrow e \]

Lists of branches
\[ \Pi ::= \cdot \mid (\pi \Pi) \]

Figure 1: Source syntax

Universal variables \( \alpha, \beta, \gamma \)

Sorts
\[ \kappa ::= * \mid \mathbb{N} \]

Types
\[ A, B, C ::= 1 \mid A \rightarrow B \mid A + B \mid A \times B \]
\[ \mid \alpha \mid \forall \alpha : \kappa. A \mid \exists \alpha : \kappa. A \]
\[ \mid P \supset A \mid A \land P \]

Terms/monotypes
\[ t, \tau, \sigma ::= \text{zero} \mid \text{succ}(t) \mid 1 \mid \alpha \]
\[ \mid \tau \Rightarrow \sigma \mid \tau + \sigma \mid \tau \times \sigma \]

Propositions
\[ P, Q ::= t = t' \]

Contexts
\[ \Psi ::= \cdot \mid \Psi, \alpha : \kappa \mid \Psi, x : A \rho \]

Polarities
\[ \pm ::= + \mid - \]

Binary connectives
\[ \oplus ::= \rightarrow \mid + \mid \times \]

Principalities
\[ p, q ::= ! \mid \dashv \]

sometimes omitted

Figure 2: Syntax of declarative types and contexts

## 2 Figures from paper, repeated

We repeat the figures from the main paper (for convenience, and to avoid numbering confusion).
checking, eq. elim.  \[ \Psi/P \vdash e \Leftarrow C_p \]
subtyping  \[ \Psi \vdash A \leq \pm B \]
coverage  \[ \Psi \vdash \Pi \text{ covers } A \]
spine typing  \[ \Psi \vdash s : A_p \gg B_q \]
type checking  \[ \Psi \vdash e \Leftarrow A_p \]
match, eq. elim.  \[ \Psi/P \vdash \Pi : \pm A \Leftarrow C_p \]
principality-recovering spine typing  \[ \Psi \vdash s : A_p \gg B \lceil q \rceil \]
pattern matching  \[ \Psi \vdash \Pi : \gg A \Leftarrow C_p \]
type synthesis  \[ \Psi \vdash e \Rightarrow B_p \]

Figure 3: Dependency structure of the declarative judgments

\[
\Psi \vdash A \leq \pm B \quad \text{Under context } \Psi, \text{ type } A \text{ is a subtype of } B, \text{ decomposing head connectives of polarity } \pm
\]

\[
\begin{array}{cccc}
\Psi \vdash A \text{ type} & \text{nonpos}(A) & \text{nonneg}(A) & \leq \text{Refl}_\pm \\
\hline
\Psi \vdash A \leq \pm A & \Psi \vdash A \leq B & \text{nonpos}(A) & \text{nonpos}(B) \leq \text{def}_o \\
\Psi \vdash A \leq \pm B & \Psi \vdash A \leq B & \text{nonneg}(A) & \text{nonneg}(B) \leq \text{def}_o \\
\Psi, \alpha : \kappa \vdash A \leq \pm B & \Psi, \beta : \kappa \vdash A \leq B & \leq \text{def}_l & \leq \text{def}_r \\
\Psi, \alpha : \kappa \vdash A \leq \pm B & \Psi, \beta : \kappa \vdash A \leq B & \leq \text{def}_l & \leq \text{def}_r \\
\Psi, \alpha : \kappa \vdash A \leq \pm B & \Psi, \beta : \kappa \vdash A \leq B & \leq \text{def}_l & \leq \text{def}_r \\
\Psi, \alpha : \kappa \vdash A \leq \pm B & \Psi, \beta : \kappa \vdash A \leq B & \leq \text{def}_l & \leq \text{def}_r \\
\end{array}
\]

Figure 4: Subtyping in the declarative system

\[
\text{e } \text{chk-I} \quad \text{Expression } e \text{ is a checked introduction form}
\]

\[
\begin{array}{cccc}
\lambda x. e \text{ chk-I} & \emptyset \text{ chk-I} & \langle e_1, e_2 \rangle \text{ chk-I} & \text{inj}_k e \text{ chk-I} \\
\end{array}
\]

Figure 5: “Checking intro form”
<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Psi \vdash P \ true$</td>
<td>Under context $\Psi$, check $P$</td>
</tr>
<tr>
<td>$\Psi \vdash \ (t = t) \ true$</td>
<td>$\text{DeclCheckpropEq}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>Under context $\Psi$, expression $e$ checks against input type $A$</td>
</tr>
<tr>
<td>$\Psi \vdash e \Rightarrow A$</td>
<td>Under context $\Psi$, expression $e$ synthesizes output type $A$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg C$</td>
<td>Under context $\Psi$, passing spine $s$ to a function of type $A$ synthesizes type $C$; in the $</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\boxed{\Psi \vdash P \ true}$</td>
<td>$\text{DeclSpineRecover}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \gg A$</td>
<td>$\text{DeclEmptySpine}$</td>
</tr>
<tr>
<td>$\Psi \vdash x : A \ p \gg A$</td>
<td>$\text{DeclVar}$</td>
</tr>
<tr>
<td>$\Psi \vdash x \Rightarrow A$</td>
<td>$\text{DeclSub}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclAnno}$</td>
</tr>
<tr>
<td>$\Psi \vdash (e : A) \Rightarrow A$</td>
<td>$\text{DeclIl}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \ chk-I$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \lambda x. \ e \leftarrow A \ p$</td>
<td>$\text{DeclIl}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \ chk-I$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$</td>
<td>$\text{DeclEmptySpine}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSub}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \lambda x. \ e \leftarrow A \ p$</td>
<td>$\text{DeclIl}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \ chk-I$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$</td>
<td>$\text{DeclEmptySpine}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSub}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \lambda x. \ e \leftarrow A \ p$</td>
<td>$\text{DeclIl}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \ chk-I$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$</td>
<td>$\text{DeclEmptySpine}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSub}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \lambda x. \ e \leftarrow A \ p$</td>
<td>$\text{DeclIl}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \ chk-I$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$ &amp; $\Psi \vdash \Psi \vdash s : A \ p \gg C[ q ]$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash s : A$</td>
<td>$\text{DeclEmptySpine}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSub}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \lambda x. \ e \leftarrow A \ p$</td>
<td>$\text{DeclIl}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \ chk-I$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi / \ p \vdash \psi \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash \psi \leftarrow \psi / (p \gg A)$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
<tr>
<td>$\Psi \vdash e \leftarrow A$</td>
<td>$\text{DeclSpinePass}$</td>
</tr>
</tbody>
</table>

Figure 6: Declarative typing
Under context $\Psi$, check branches $\Pi$ with patterns of type $\vec{A}$ and bodies of type $C$

\[
\begin{array}{ll}
\text{DeclMatchEmpty} & \Psi \vdash \Pi :: \vec{A} \leftarrow C p \\
\text{DeclMatchBase} & \Psi \vdash e :: \vec{C} p \\
\text{DeclMatchUnit} & \Psi \vdash \varrho \Rightarrow e :: \vec{C} p \\
\text{DeclMatchUnify} & \Psi \vdash \Pi \vdash \vec{A} \leftarrow C p \\
\text{DeclMatch\exists} & \Psi \vdash (\exists \alpha : k. A), \vec{A} \leftarrow C p \\
\text{DeclMatch\times} & \Psi \vdash \rho_1, \rho_2, \varrho \Rightarrow e :: A_1, A_2, \vec{A} \leftarrow C p \\
\text{DeclMatch\wedge} & \Psi \vdash \rho, \varrho \Rightarrow e :: A_k, \vec{A} \leftarrow C p \\
\text{DeclMatch\+} & \Psi \vdash \text{inj}_k \rho, \varrho \Rightarrow e :: A_1 + A_2, \vec{A} \leftarrow C p \\
\text{DeclMatchNeg} & \Psi \vdash x, \varrho \Rightarrow e :: A, \vec{A} \leftarrow C p \\
\text{DeclMatchWild} & \Psi \vdash \varphi, \varrho \Rightarrow e :: A, \vec{A} \leftarrow C p \\
\end{array}
\]

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

\[
\begin{array}{ll}
\text{DeclMatch\perp} & \Psi \vdash \Pi :: \vec{A} \leftarrow C p \\
\text{DeclMatchUnify} & \Psi \vdash \Pi :: \vec{A} \leftarrow C p \\
\end{array}
\]

Figure 7: Declarative pattern matching
**Figure 8: Match coverage**
Figure 9: Dependency structure of the algorithmic judgments
Universal variables  \( \alpha, \beta, \gamma \)

Existential variables  \( \hat{\alpha}, \hat{\beta}, \hat{\gamma} \)

Variables  
\[
\begin{align*}
\text{u} & := \alpha | \hat{\alpha} \\
\text{A, B, C} & := 1 | \alpha | \hat{\alpha} \\
| \forall \alpha : \kappa. A & | \exists \alpha : \kappa. A \\
| P \supset A & | A \land P \\
| A \rightarrow B & | A + B | A \times B
\end{align*}
\]

Types  
\[
\begin{align*}
P, Q & := t = t' \\
\odot & := \rightarrow | + | \times
\end{align*}
\]

Binary connectives  
\[
\begin{align*}
\text{t, \tau, \sigma} & := \text{zero | succ (t) | 1 | \alpha | \hat{\alpha}} \\
| \tau \rightarrow \sigma & | \tau + \sigma | \tau \times \sigma
\end{align*}
\]

Terms/monotypes  
\[
\begin{align*}
\text{t}, \tau, \sigma & := \text{zero | succ (t) | 1 | \alpha | \hat{\alpha}} \\
| \tau \rightarrow \sigma & | \tau + \sigma | \tau \times \sigma
\end{align*}
\]

Contexts  
\[
\begin{align*}
\Gamma, \Delta, \Theta & := \cdot | \Gamma, u : \kappa | \Gamma, x : Ap \\
| \Gamma, \hat{\alpha} : \kappa = \tau & | \Gamma, \alpha = t | \Gamma, u
\end{align*}
\]

Complete contexts  
\[
\begin{align*}
\Omega & := \cdot | \Omega, \alpha : \kappa | \Omega, x : Ap \\
| \Omega, \hat{\alpha} : \kappa = \tau & | \Omega, \alpha = t | \Omega, u
\end{align*}
\]

Possibly-inconsistent contexts  
\[
\Delta \bot := \Delta | \bot
\]

Figure 10: Syntax of types, contexts, and other objects in the algorithmic system

\[
\begin{align*}
[\Gamma]1 & = 1 \\
[\Gamma]\alpha & = \begin{cases} [\Gamma]\tau & \text{when } (\alpha = \tau) \in \Gamma \\ \alpha & \text{otherwise} \end{cases} \\
[\Gamma][\hat{\alpha} : \kappa = \tau] & = [\Gamma]\tau \\
[\Gamma][\hat{\alpha} : \kappa] & \otimes \hat{\alpha} = \hat{\alpha} \\
[\Gamma](A \supset B) & = ([\Gamma]A) \supset ([\Gamma]B) \\
[\Gamma](A \land B) & = ([\Gamma]A) \land ([\Gamma]B) \\
[\Gamma](A \oplus B) & = ([\Gamma]A) \oplus ([\Gamma]B) \\
[\Gamma]((\forall \alpha : \kappa). A) & = \forall \alpha : \kappa. [\Gamma]A \\
[\Gamma]((\exists \alpha : \kappa). A) & = \exists \alpha : \kappa. [\Gamma]A
\end{align*}
\]

Figure 11: Applying a context, as a substitution, to a type

\[
\begin{align*}
[\cdot] & = \cdot \\
[\Omega, \alpha : \kappa](\Gamma, \alpha : \kappa) & = [\Omega][\Gamma, \alpha : \kappa] \\
[\Omega, \hat{\alpha} : \kappa](\Gamma, \hat{\alpha} : \kappa) & = [\Omega][\Gamma] \\
[\Omega, \alpha = t](\Gamma, \alpha = t') & = [\Omega][\Gamma][\alpha / t] \text{ if } [\Omega]t = [\Omega]t' \\
[\Omega, \hat{\alpha} : \kappa = t][\Gamma] & = \begin{cases} [\Omega][\Gamma] \text{ when } \Gamma = (\Gamma', \hat{\alpha} : \kappa = t') \\
[\Omega][\Gamma] & \text{ when } \Gamma = (\Gamma', \hat{\alpha} : \kappa) \\
[\Omega][\Gamma] \text{ otherwise} 
\end{cases}
\end{align*}
\]

Figure 12: Applying a complete context \( \Omega \) to a context
Figure 13: Context extension
\[
\Gamma \vdash e \iff A \ p \vdash \Delta \\
\Gamma \vdash e \Rightarrow A \ p \vdash \Delta
\]
Under input context \(\Gamma\), expression \(e\) checks against input type \(A\), with output context \(\Delta\)

\[
\Gamma \vdash s : A \ p \Rightarrow C \ q \vdash \Delta \\
\Gamma \vdash s : A \ p \Rightarrow C \ [q] \vdash \Delta
\]
Under input context \(\Gamma\), expression \(e\) synthesizes output type \(A\), with output context \(\Delta\)

Under input context \(\Gamma\), passing spine \(s\) to a function of type \(A\) synthesizes type \(C\); in the \([q]\) form, recover principality in \(q\) if possible

\[
\frac{\{x : A\ p\} \in \Gamma}{\Gamma \vdash x \Rightarrow [\Gamma]A \ p \vdash \Gamma} \text{ Var} \\
\frac{\Gamma \vdash e \Rightarrow A \ q \vdash \Theta \quad \Theta \vdash A < : \text{pol}(B) \ B \vdash \Delta}{\Gamma \vdash e \Leftarrow B \ p \vdash \Delta} \text{ Sub} \\
\frac{\Gamma \vdash (e : A) \Rightarrow [\Delta]A \ ! \vdash \Delta}{\Gamma \vdash A \ ! \text{ type}} \\
\frac{\Gamma \vdash e \Leftarrow [\Theta]A \ p \vdash \Delta}{\Gamma \vdash (e : A) \Rightarrow [\Delta]A \ ! \vdash \Delta} \text{ Anno}
\]

\[
\frac{\Gamma \vdash \text{ not a case}}{\forall \vdash \text{Spine}}
\]

\[
\frac{\Gamma \vdash P \ true \ \Theta \vdash e \Leftarrow [\Theta]A \ p \vdash \Delta}{\Gamma \vdash P \ true \ \Theta \vdash e \Leftarrow [\Theta]A \ p \vdash \Delta} \text{ \&}
\]

\[
\frac{\Gamma \vdash \text{ SpineRecover}}{\Gamma \vdash \text{ SpinePass}}
\]

\[
\frac{\Gamma \vdash \text{ EmptySpine}}{\Gamma \vdash \text{ Spine}}
\]

\[
\frac{\Gamma \vdash e \Leftarrow A \ k \ p \vdash \Delta}{\Gamma \vdash \text{ inj}_{k\ e} \Leftarrow A_1 \vdash A_2 \ p \vdash \Delta} \text{ +k}
\]

\[
\frac{\Gamma \vdash e \Leftarrow A_1 \ p \vdash \Theta \quad \Theta \vdash e_2 \Leftarrow [\Theta]A_2 \ p \vdash \Delta}{\Gamma \vdash \langle e_1, e_2 \rangle \Leftarrow A_1 \times A_2 \ p \vdash \Delta} \text{ xI}
\]

\[
\Gamma \vdash \text{ \&Spin} \\
\frac{\Gamma \vdash \text{ \&Spin}}{\Gamma \vdash \text{ Case}}
\]

Figure 14: Algorithmic typing
3 Figures omitted from paper

This section contains figures omitted in the main paper for space reasons.

$$\Psi \vdash t : \kappa$$ Under context $\Psi$, term $t$ has sort $\kappa$

$$\frac{(\alpha : \kappa) \in \Psi}{\Psi \vdash \alpha : \kappa} \quad \text{UvarSort}$$

$$\frac{\Psi \vdash \kappa}{\Psi \vdash \kappa : \kappa} \quad \text{UnitSort}$$

$$\frac{\Psi \vdash \kappa}{\Psi \vdash \kappa : \kappa} \quad \text{BinSort}$$

$$\frac{\Psi \vdash \kappa}{\Psi \vdash \kappa : \kappa} \quad \text{SuccSort}$$

$$\Psi \vdash \text{zero} : \mathbb{N}$$

$$\Psi \vdash \text{succ}(t) : \mathbb{N}$$

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

$$\frac{\Psi \vdash t : \mathbb{N} \quad \Psi \vdash t' : \mathbb{N}}{\Psi \vdash t = t' \text{ prop}} \quad \text{EqDeclProp}$$

$$\Psi \vdash \Lambda \text{ type}$$ Under context $\Psi$, type $\Lambda$ is well-formed

$$\frac{(\alpha : \kappa) \in \Psi}{\Psi \vdash \alpha \text{ type}} \quad \text{DeclUvarWF}$$

$$\frac{\Psi \vdash \kappa}{\Psi \vdash \kappa \text{ type}} \quad \text{DeclUnitWF}$$

$$\frac{\Psi \vdash \Lambda \text{ type}}{\Psi \vdash \Lambda \oplus \text{ type}} \quad \text{DeclBinWF}$$

$$\frac{\Psi, \alpha : \kappa \vdash \Lambda \text{ type}}{\Psi \vdash (\forall \alpha : \kappa. \Lambda) \text{ type}} \quad \text{DeclAllWF}$$

$$\frac{\Psi, \alpha : \kappa \vdash \Lambda \text{ type}}{\Psi \vdash (\exists \alpha : \kappa. \Lambda) \text{ type}} \quad \text{DeclExistsWF}$$

$$\frac{\Psi \vdash P \text{ prop} \quad \Psi \vdash \Lambda \text{ type}}{\Psi \vdash P \implies \Lambda \text{ type}} \quad \text{DeclImpliesWF}$$

$$\frac{\Psi \vdash \Lambda \text{ type}}{\Psi \vdash \Lambda \land P \text{ type}} \quad \text{DeclWithWF}$$

$$\Psi \vdash \vec{\Lambda} \text{ types}$$ Under context $\Psi$, types in $\vec{\Lambda}$ are well-formed

$$\frac{\text{for all } \Lambda \in \vec{\Lambda},}{\Psi \vdash \Lambda \text{ type}} \quad \text{DeclTypevecWF}$$

$$\Psi \text{ ctx}$$ Declarative context $\Psi$ is well-formed

$$\frac{\Psi \text{ ctx} \quad x \notin \text{dom}(\Psi)}{\Psi, x : A \text{ ctx}} \quad \text{HypDeclCtx}$$

$$\frac{\Psi \text{ ctx} \quad \alpha \notin \text{dom}(\Psi)}{\Psi, \alpha : \kappa \text{ ctx}} \quad \text{VarDeclCtx}$$

Figure 15: Sorting; well-formedness of propositions, types, and contexts in the declarative system
\[ \Gamma \vdash \tau : \kappa \] Under context \( \Gamma \), term \( \tau \) has sort \( \kappa \)

\[ \begin{align*}
\frac{(u : \kappa) \in \Gamma}{\Gamma \vdash u : \kappa} & \quad \text{VarSort} \\
\frac{(\hat{\alpha} : \kappa = \tau) \in \Gamma}{\Gamma \vdash \hat{\alpha} : \kappa} & \quad \text{SolvedVarSort} \\
\frac{\Gamma \vdash 1 : \ast}{\Gamma \vdash 1} & \quad \text{UnitSort} \\
\frac{\Gamma \vdash \tau_1 : \ast}{\Gamma \vdash \tau_1} & \quad \text{BinSort} \\
\frac{\Gamma \vdash \tau_2 : \ast}{\Gamma \vdash \tau_2} & \quad \text{BinSort} \\
\frac{\Gamma \vdash \tau_1 \oplus \tau_2 : \ast}{\Gamma \vdash \tau_1 \oplus \tau_2} & \quad \text{ZeroSort}
\end{align*} \]

\[ \Gamma \vdash P \text{ prop} \] Under context \( \Gamma \), proposition \( P \) is well-formed

\[ \begin{align*}
\frac{\Gamma \vdash t : N}{\Gamma \vdash t' : N} & \quad \text{EqProp} \\
\frac{\Gamma \vdash t = t'}{\Gamma \vdash t' = N} & \quad \text{EqProp}
\end{align*} \]

\[ \Gamma \vdash A \text{ type} \] Under context \( \Gamma \), type \( A \) is well-formed

\[ \begin{align*}
\frac{(u : \ast) \in \Gamma}{\Gamma \vdash u} & \quad \text{VarType} \\
\frac{\hat{\alpha} : \ast = \tau}{\Gamma \vdash \hat{\alpha} : \ast} & \quad \text{SolvedVarType} \\
\frac{\Gamma \vdash A \in \ast}{\Gamma \vdash A} & \quad \text{UnitType} \\
\frac{\Gamma \vdash A \in \ast \oplus B}{\Gamma \vdash A \oplus B} & \quad \text{BinType}
\end{align*} \]

\[ \Gamma, \alpha : \kappa \vdash A \text{ type} \] Under context \( \Gamma \), \( \alpha \) is well-formed

\[ \begin{align*}
\frac{\Gamma, \alpha : \kappa \vdash A \text{ type}}{\Gamma \vdash \exists \alpha : \kappa. A} & \quad \text{ExistsType} \\
\frac{\Gamma \vdash P \text{ prop} \quad \Gamma \vdash A \text{ type}}{\Gamma \vdash P \supset A} & \quad \text{ImpliesType} \\
\frac{\Gamma \vdash P \text{ prop} \quad \Gamma \vdash A \text{ type}}{\Gamma \vdash A \land P} & \quad \text{WithType}
\end{align*} \]

\[ \Gamma \vdash \bar{A} \text{ [p] types} \] Under context \( \Gamma \), types in \( \bar{A} \) are well-formed (with principality \( p \))

\[ \begin{align*}
\text{for all } A \in \bar{A}, & \quad \frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \bar{A} \text{ types}} \quad \text{TypevecType} \\
\text{for all } A \in \bar{A}, & \quad \frac{\Gamma \vdash A \text{ p type}}{\Gamma \vdash \bar{A} \text{ p types}} \quad \text{PrincipalTypevecType}
\end{align*} \]

\[ \Gamma \text{ ctx} \] Algorithmic context \( \Gamma \) is well-formed

\[ \begin{align*}
\frac{\text{EmptyCtx}}{\Gamma \text{ ctx}} & \quad \text{EmptyCtx} \\
\frac{\Gamma \text{ ctx} \quad x \notin \text{dom}(\Gamma)}{\Gamma, x : A \text{ ctx} \quad \text{HypCtx}} & \quad \text{HypCtx} \\
\frac{\Gamma \text{ ctx}}{\Gamma, x : A \land \kappa \text{ ctx} \quad \text{VarCtx}} & \quad \text{VarCtx} \\
\frac{\Gamma \text{ ctx} \quad u \notin \text{dom}(\Gamma)}{\Gamma, u : \kappa \text{ ctx} \quad \text{SolvedCtx}} & \quad \text{SolvedCtx} \\
\frac{\alpha \in \kappa \quad \text{EqnCtx}}{\Gamma, \alpha = \tau \text{ ctx} \quad \text{EqnVarCtx}} & \quad \text{EqnVarCtx} \\
\frac{\Gamma \text{ ctx} \quad \kappa \notin \Gamma}{\Gamma, \kappa \text{ ctx} \quad \text{MarkerCtx}} & \quad \text{MarkerCtx}
\end{align*} \]

Figure 16: Well-formedness of types and contexts in the algorithmic system
\[ \Gamma 
vdash \text{P true} \not\vdash \Delta \]

Under context \( \Gamma \), check \( \text{P} \), with output context \( \Delta \)

\[
\begin{align*}
\Gamma &\vdash t_1 \doteq t_2 : \mathbb{N} \not\vdash \Delta \\
\Gamma &\vdash t_1 = t_2 \text{ true} \not\vdash \Delta
\end{align*}
\]

CheckpropEq

\[ \Gamma / \text{P} \not\vdash \Delta \]

Incorporate hypothesis \( \text{P} \) into \( \Gamma \), producing \( \Delta \) or inconsistency \( \bot \)

\[
\begin{align*}
\Gamma &/ t_1 \doteq t_2 : \mathbb{N} \not\vdash \Delta \\
\Gamma &/ t_1 = t_2 \not\vdash \Delta
\end{align*}
\]

ElimpropEq

Figure 17: Checking and assuming propositions

\[ \Gamma \vdash t_1 \doteq t_2 : \kappa \not\vdash \Delta \]

Check that \( t_1 \) equals \( t_2 \), taking \( \Gamma \) to \( \Delta \)

\[
\begin{align*}
\Gamma &\vdash u \doteq u : \kappa \not\vdash \Gamma \\
\Gamma &\vdash 1 \doteq 1 : \ast \not\vdash \Gamma
\end{align*}
\]

CheckeqVar

\[
\begin{align*}
\Gamma &\vdash \tau_1 \doteq \tau'_1 : \ast \not\Theta \\
\Theta &\vdash [\Theta]\tau_2 \doteq [\Theta]\tau'_2 : \ast \not\Delta
\end{align*}
\]

CheckeqBin

\[
\begin{align*}
\Gamma &\vdash \text{zero} \doteq \text{zero} : \mathbb{N} \not\vdash \Gamma \\
\Gamma &\vdash \text{succ}(t_1) \doteq \text{succ}(t_2) : \mathbb{N} \not\Delta
\end{align*}
\]

CheckeqZero

\[
\begin{align*}
\Gamma &\vdash t_1 \doteq t_2 : \mathbb{N} \not\Delta \\
\Gamma &\vdash \text{succ}(t_1) \doteq \text{succ}(t_2) : \mathbb{N} \not\Delta
\end{align*}
\]

CheckeqSucc

\[ \Gamma[\alpha : \kappa] \vdash \& := t : \kappa \not\Delta \quad \& \not\in \text{FV}(t) \]

\[
\Gamma[\alpha : \kappa] \vdash \& := t : \kappa \not\Delta
\]

CheckeqInstL

\[ \Gamma[\alpha : \kappa] \vdash \& := t : \kappa \not\Delta \quad \& 
ot\in \text{FV}(t) \]

\[
\Gamma[\alpha : \kappa] \vdash t \doteq \alpha : \kappa \not\Delta
\]

CheckeqInstR

Figure 18: Checking equations

\[ t_1 \neq t_2 \]

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

\[
\begin{align*}
\text{zero} \neq \text{succ}(t) & \quad \text{succ}(t) \neq \text{zero} \\
1 \neq \tau_1 \oplus \tau_2 & \quad \tau_1 \oplus \tau_2 \neq 1 \\
\oplus_1 \neq \oplus_2 & \quad \sigma_1 \oplus_1 \tau_1 \neq \sigma_2 \oplus_2 \tau_2
\end{align*}
\]

Figure 19: Head constructor clash
Unify $\sigma$ and $\tau$, taking $\Gamma$ to $\Delta$, or to inconsistency $\bot$.

$\Gamma / \sigma \doteq \tau : \kappa \rightarrow \Delta^\bot$  ElimeqUvarRefl

$\Gamma / \alpha \doteq \kappa : \kappa \rightarrow \Gamma$  ElimeqZero

$\Gamma / \text{zero} \doteq \text{zero} : \mathbb{N} \rightarrow \Gamma$  ElimeqZero

$\alpha \notin \text{FV}(\tau) \quad (\alpha = -) \notin \Gamma$  ElimeqUvarL

$\frac{\Gamma / \alpha \doteq \kappa \rightarrow \Gamma, \alpha = \tau} {\Gamma / \alpha \doteq \tau : \kappa \rightarrow \bot}$  ElimeqUvarLbot

$\Gamma / 1 \doteq \ast : \ast \rightarrow \Gamma$  ElimeqUnit

$\Gamma / \tau_1 \doteq \tau_1' : \ast \rightarrow \Theta \quad \Theta / \Theta / \tau_2 \doteq \Theta / \tau_2' : \ast \rightarrow \bot$  ElimeqBin

$\Gamma / \tau_1 \doteq \tau_1' : \ast \rightarrow \bot \quad \Gamma / \tau_2 \doteq \tau_2' : \ast \rightarrow \bot$  ElimeqBinBot

$\sigma \neq \tau$  ElimeqClash

Figure 20: Eliminating equations
Under input context \( \Gamma \), type \( A \) is a subtype of \( B \), with output context \( \Delta \)

\[
\Gamma \vdash A <^\pm B \vdash \Delta
\]

Under input context \( \Gamma \), type \( A \) is equivalent to \( B \), with output context \( \Delta \)

\[
\Gamma \vdash A \equiv B \vdash \Delta
\]

Figure 21: Algorithmic equivalence and subtyping
Under input context $\Gamma$, instantiate $\hat{\alpha}$ such that $\hat{\alpha} = t$ with output context $\Delta$

$$
\begin{align*}
\Gamma |\hat{\alpha} := t : \kappa &\vdash \Delta \\
\end{align*}
$$

Figure 22: Instantiation
\( \Gamma \vdash \Pi :: \bar{A} \iff C \ p \ : \ ! \Delta \) Under context \( \Gamma \), check branches \( \Pi \) with patterns of type \( \bar{A} \) and bodies of type \( C \)

\[
\begin{align*}
\Gamma & \vdash \Pi :: \bar{A} \iff C \ p \ : \ ! \Delta & \text{MatchEmpty} & \Gamma & \vdash \Pi' :: \bar{A} \iff C \ p \ : \ ! \Delta & \text{MatchSeq} \\
\Gamma & \vdash e :: C \ p \ : \ ! \Delta & \text{MatchBase} & \Gamma & \vdash \delta \Rightarrow e :: C \ p \ : \ ! \Delta & \text{MatchUnit} \\
\Gamma, \alpha : \kappa \vdash \rho \Rightarrow e :: A, \bar{A} \iff C \ p \ : \ ! \Delta, \alpha : \kappa, \Theta & \text{Match}\exists & \Gamma & \vdash \rho \Rightarrow e :: A, \bar{A} \iff C \ p \ : \ ! \Delta & \text{Match\wedge} \\
\Gamma & \vdash \rho_1, \rho_2 \Rightarrow e :: A_1, A_2, \bar{A} \iff C \ p \ : \ ! \Delta & \text{Match\times} & \Gamma & \vdash \langle \rho_1, \rho_2 \rangle \Rightarrow e :: A_1 \times A_2, \bar{A} \iff C \ p \ : \ ! \Delta & \text{Match+k} \\
\Gamma & \vdash \rho, \bar{A} \Rightarrow e :: \bar{A}, \bar{A} \iff C \ p \ : \ ! \Delta & \text{MatchNeg} & \Gamma & \vdash \rho \Rightarrow e :: \bar{A}, \bar{A} \iff C \ p \ : \ ! \Delta & \text{MatchWild} \\
\end{align*}
\]

\( \Gamma / P \vdash \Pi :: \bar{A} \iff C \ p \ : \ ! \Delta \) Under context \( \Gamma \), incorporate proposition \( P \) while checking branches \( \Pi \) with patterns of type \( \bar{A} \) and bodies of type \( C \)

\[
\begin{align*}
\Gamma & / \sigma = \tau : \kappa \vdash \rho \Rightarrow e :: \bar{A} \iff C \ p \ : \ ! \Delta & \text{Match\bot} \\
\Gamma, \Pi / P \delta = \tau : \kappa \vdash \rho \Rightarrow e :: \bar{A} \iff C \ p \ : \ ! \Delta, \Pi' \delta \vdash & \text{Match\Unify} \\
\Gamma & / \sigma = \tau : \kappa \vdash \rho \Rightarrow e :: \bar{A} \iff C \ p \ : \ ! \Delta & \text{Match\Unify} \\
\end{align*}
\]

Figure 23: Algorithmic pattern matching

\( \Gamma \vdash \Pi \text{ covers } \bar{A} \) Under context \( \Gamma \), patterns \( \Pi \) cover the types \( \bar{A} \)

\[
\begin{align*}
\Gamma & \vdash \Pi \text{ covers } \bar{A} & \text{CoversEmpty} & \Pi \text{ covers } A, \bar{A} & \text{CoversVar} \\
\Pi & \vdash \Pi' \text{ covers } \bar{A} & \text{Covers1} & \Pi \text{ covers } A_1, \bar{A} & \text{Covers}\times \\
\Pi & \vdash \Pi_1 \text{ covers } 1, \bar{A} & \text{Covers}\bot & \Pi \text{ covers } A_1 \times A_2, \bar{A} & \text{Covers\times} \\
\Pi & \vdash \Pi_1 \parallel \Pi_R \text{ covers } A_1, \bar{A} & \text{Covers\bot} & \Pi \text{ covers } A_1 + A_2, \bar{A} & \text{Covers\times} \\
\Pi & \vdash \Pi_1 \parallel \Pi \text{ covers } A_1, \bar{A} \quad \Pi & \vdash \Pi_1 \parallel \Pi_R \text{ covers } A_2, \bar{A} & \text{Covers\bot} \\
\Gamma & / \Pi \text{ covers } A_1 + A_2, \bar{A} & \text{Covers\bot} & \Gamma & / \Pi \text{ covers } A_0 \land \Pi \text{ covers } A_0 \land (t_1 = t_2), \bar{A} & \text{Covers\bot} \\
\Gamma & / \Pi \text{ covers } A_0 \land (t_1 = t_2), \bar{A} & \text{Covers\bot} & \Gamma & / \Pi \text{ covers } A_0 \land (t_1 = t_2), \bar{A} & \text{Covers\bot} \\
\end{align*}
\]

Figure 24: Algorithmic match coverage
A Properties of the Declarative System

Lemma 1 (Declarative Well-foundedness).\(^7\)
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 \(\Psi \vdash s : A P \Rightarrow B q\)
(v) recovery spine \(\Psi \vdash s : A P \Rightarrow B [q]\)
(vi) pattern matching \(\Psi \vdash \Pi : \tilde{A} \Leftarrow C P\)
(vii) pattern matching, equality elimination \(\Psi / P \vdash \Pi : \tilde{A} \Leftarrow C P\)

Lemma 2 (Declarative Weakening).\(^7\)

(i) If \(\Psi_0, \Psi_1 \vdash t : \kappa\) then \(\Psi_0, \Psi, \Psi_1 \vdash t : \kappa\).
(ii) If \(\Psi_0, \Psi_1 \vdash P prog\) then \(\Psi_0, \Psi, \Psi_1 \vdash P prog\).
(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).\(^7\)
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 \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash P prog\) then \(\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] P prog\).
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 \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \leq^\pm B\) then \(\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] A \leq^\pm [t/\alpha] B\).
5. If \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash P true\) then \(\Psi_0, [t/\alpha] \Psi_1 \vdash [t/\alpha] P true\).

Lemma 4 (Reflexivity of Declarative Subtyping).\(^7\)
Given \(\Psi \vdash A type\), we have that \(\Psi \vdash A \leq^\pm A\).

Lemma 5 (Subtyping Inversion).\(^7\)

- 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).\(^7\)

- 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 \(\text{nonneg}(B)\) and \(\text{nonpos}(A)\) and \(\text{nonneg}(B)\) and \(\Psi \vdash A \leq^+ B\) then \(A = B\).

Lemma 7 (Transitivity of Declarative Subtyping).\(^7\)
Given \(\Psi \vdash A type\) and \(\Psi \vdash B type\) and \(\Psi \vdash C type\):

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

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 Θ is soft iff it consists only of \( \alpha : \kappa \) and \( \hat{\alpha} : \kappa = \tau \) declarations.

Lemma 8 (Substitution—Well-formedness). \( \text{Go to proof} \)

(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 \( \text{FEV}(\Gamma|\tau) = \emptyset \) 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 \( \Gamma \vdash t : \kappa \) then \( \Gamma \vdash t \ \text{type} \).

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 \( \Gamma \vdash P \ \text{prop} \) then \( \Gamma \vdash [\Gamma]|P \ \text{prop} \).

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

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

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

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

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

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 \( \Theta \vdash A \ \text{type} \) and \( \Theta \longrightarrow \Gamma \) then \( [\Gamma]|\Theta = [\Gamma]|\Theta|A \) and \( [\Gamma]|\Lambda = [\Theta]|\Theta|A \).

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

Lemma 22 (Extension Inversion). \( \text{Go to proof} \)

(i) If \( D \vdash \Delta_0, \xi : \kappa, \Delta_1 \longrightarrow \Delta \)

then there exist unique \( \Delta_0 \) and \( \Delta_1 \)

such that \( \Delta = (\Delta_0, \xi : \kappa, \Delta_1) \) and \( D' \vdash \Delta_0 \longrightarrow \Delta_0 \) where \( D' < D \).

Moreover, if \( \Gamma_1 \) is soft, then \( \Delta_1 \) is soft.
(ii) If \( D \vdash \gamma_0, \alpha \cdot \Delta_1 \rightarrow \Delta \) then there exist unique \( \Delta_0 \) and \( \Delta_1 \) such that \( \Delta = (\Delta_0, \alpha \cdot \Delta_1) \) and \( D' \vdash \gamma_0 \rightarrow \Delta_0 \) where \( D' < D \).
Moreover, if \( \gamma_1 \) is soft, then \( \Delta_1 \) is soft.
Moreover, if \( \text{dom}(\gamma_0, \alpha \cdot \Delta_1) = \text{dom}(\Delta) \) 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 \( D \vdash \gamma_0, \alpha \cdot \kappa \cdot \Delta_1 \rightarrow \Delta \) then there exist unique \( \Delta_0, \tau', \) and \( \Delta_1 \) such that \( \Delta = (\Delta_0, \alpha \cdot \kappa \cdot \Delta_1) \) 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 \( \Delta_0, \tau' \) 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 \( D \vdash \gamma_0, \alpha \cdot \kappa, \gamma_1 \rightarrow \Delta \) then either

- there exist unique \( \Delta_0, \tau' \) and \( \Delta_1 \) such that \( \Delta = (\Delta_0, \alpha \cdot \kappa \cdot \Delta_1) \) and \( D' \vdash \gamma_0 \rightarrow \Delta_0 \) where \( D' < D \), or
- there exist unique \( \Delta_0 \) and \( \Delta_1 \) such that \( \Delta = (\Delta_0, \alpha \cdot \kappa, \Delta_1) \) and \( D' \vdash \gamma_0 \rightarrow \Delta_0 \) where \( D' < D \).

Lemma 23 (Deep Evar Introduction). [Go to proof]

(i) If \( \gamma_0, \gamma_1 \) is well-formed and \( \alpha \) is not declared in \( \gamma_0, \gamma_1 \) then \( \gamma_0, \gamma_1 \rightarrow \gamma_0, \alpha \cdot \kappa, \gamma_1 \).
(ii) If \( \gamma_0, \alpha \cdot \kappa, \gamma_1 \) is well-formed and \( \Gamma \vdash t : \kappa \) then \( \Gamma_0, \alpha \cdot \kappa, \gamma_1 \rightarrow \gamma_0, \alpha \cdot \kappa = t, \gamma_1 \).
(iii) If \( \gamma_0, \gamma_1 \) is well-formed and \( \Gamma \vdash t : \kappa \) then \( \Gamma_0, \gamma_1 \rightarrow \gamma_0, \alpha \cdot \kappa = t, \gamma_1 \).

Lemma 24 (Soft Extension). [Go to proof]

If \( \Gamma \rightarrow \Delta \) and \( \Gamma, \Theta \) is ct x and \( \Theta \) is soft, then there exists \( \Omega \) such that \( \text{dom}(\Theta) = \text{dom}(\Omega) \) and \( \Gamma, \Theta \rightarrow \Delta, \Omega \).

Definition 2 (Filling). The filling of a context \( [\Gamma] \) solves all unsolved variables:

\[
\begin{align*}
[\cdot] &= . \\
[\Gamma, \cdot : A] &= [\Gamma, \cdot : A] \\
[\Gamma, \alpha : \kappa] &= [\Gamma, \alpha : \kappa] \\
[\Gamma, \alpha = t] &= [\Gamma, \alpha = t] \\
[\Gamma, \alpha \cdot \kappa = t] &= [\Gamma, \alpha \cdot \kappa = t] \\
[\Gamma, \alpha \cdot \kappa] &= [\Gamma, \alpha \cdot \kappa] \\
[\Gamma, \alpha \cdot \kappa = \cdot] &= [\Gamma, \alpha \cdot \kappa = \cdot] \\
[\Gamma, \alpha \cdot : \kappa] &= [\Gamma, \alpha \cdot : \kappa] \\
[\Gamma, \alpha \cdot : \kappa = \cdot] &= [\Gamma, \alpha \cdot : \kappa = \cdot] \\
[\Gamma, \alpha \cdot : \kappa] &= [\Gamma, \alpha \cdot : \kappa = \cdot] \\
[\Gamma, \alpha \cdot : \kappa] &= [\Gamma, \alpha \cdot : \kappa = \cdot] \\
\end{align*}
\]

Lemma 25 (Filling Completes). If \( \Gamma \rightarrow \Omega \) and \( \Gamma, \Theta \) is well-formed, then \( \Gamma, \Theta \rightarrow \Omega, [\Theta] \).

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

Lemma 26 (Parallel Admissibility). [Go to proof]

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

(i) \( \Gamma_L, \alpha \cdot \kappa, \Gamma_R \rightarrow \Delta_L, \alpha \cdot \kappa, \Delta_R \)

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

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

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 $\Gamma_L, \hat{\alpha} : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_2, \Delta_R$.

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

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

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

Lemma 31 (Split Extension). If $\Delta \rightarrow \Omega$
and $\hat{\alpha} \in \text{unsolved}(\Delta)$
and $\Omega = \Omega_1[\hat{\alpha} : \kappa = t_1]$
and $\Omega$ is canonical (Definition 3)
and $\Omega \vdash t_2 : \kappa$
thenn $\Delta \rightarrow \Omega_1[\hat{\alpha} : \kappa = t_2]$.

C.1 Reflexivity and Transitivity

Lemma 32 (Extension Reflexivity). If $\Gamma$ ctx then $\Gamma \rightarrow \Gamma$.

Lemma 33 (Extension Transitivity). If $\Delta :: \Gamma \rightarrow \Theta$ and $\Delta' :: \Theta \rightarrow \Delta$ 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). If $\Gamma \vdash t : \kappa$ then $\Gamma, \Theta \vdash t : \kappa$.

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

The following proposed lemma is false.

"Extension Weakening (Truth)"

If $\Gamma \vdash P$ true $\vdash A$ and $\Gamma \rightarrow \Gamma'$ then there exists $\Delta'$ such that $\Delta \rightarrow A'$ and $\Gamma' \vdash P$ true $\rightarrow A'$.

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

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

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

Lemma 38 (Extension Weakening (Types)). If $\Gamma \vdash A$ type and $\Gamma \rightarrow \Delta$ then $\Delta \vdash A$ type.
C.3 Principal Typing Properties

Lemma 39 (Principal Agreement). \( \text{Go to proof} \)

(i) If \( \Gamma \vdash A ! \text{ type} \) and \( \Gamma \rightarrow \Delta \) then \( [\Delta]A = [\Gamma]A \).

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

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

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

Lemma 42 (Inversion of Principal Typing). \( \text{Go to proof} \)

(1) If \( \Gamma \vdash (A \rightarrow B) \text{ p type} \) then \( \Gamma \vdash A \text{ p type} \) and \( \Gamma \vdash B \text{ p type} \).

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

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

C.4 Instantiation Extends

Lemma 43 (Instantiation Extension). \( \text{Go to proof} \) If \( \Gamma \& \Delta := \tau : \kappa \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).

C.5 Equivalence Extends

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

Lemma 45 (Elimprop Extension). \( \text{Go to proof} \) If \( \Gamma / P \equiv Q : \kappa \rightarrow \Delta \) then there exists \( \Theta \) such that \( \Gamma, \Theta \rightarrow \Delta \).

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

Lemma 47 (Checkprop Extension). \( \text{Go to proof} \) If \( \Gamma \vdash P \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).

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

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

C.6 Subtyping Extends

Lemma 50 (Subtyping Extension). \( \text{Go to proof} \) If \( \Gamma \vdash A \triangleleft B \rightarrow \Delta \) then \( \Gamma \rightarrow \Delta \).

C.7 Typing Extends

Lemma 51 (Typing Extension). \( \text{Go to proof} \) If \( \Gamma \vdash e \leftarrow A \ p \rightarrow \Delta \)

or \( \Gamma \vdash e \rightarrow A \ p \rightarrow \Delta \)

or \( \Gamma \vdash s : A p \rightarrow B q \rightarrow \Delta \)

or \( \Gamma \vdash \Pi :: A \leftarrow C p \rightarrow \Delta \)

or \( \Gamma / P \vdash \Pi :: A \leftarrow C p \rightarrow \Delta \)

then \( \Gamma \rightarrow \Delta \).
C.8 Unfiled

Lemma 52 (Context Partitioning). [Go to proof]
If $\Delta, \triangledown, \Theta \rightarrow \Omega, \triangledown, \Omega_Z$ then there is a $\Psi$ such that $[\Omega, \triangledown, \Omega_Z](\Delta, \triangledown, \Theta) = [\Omega] \Delta, \Psi$.

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

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

Lemma 54 (Completing Stability). [Go to proof]
If $\Gamma \rightarrow \Omega$ then $[\Omega] \Gamma = [\Omega] \Omega$.

Lemma 55 (Completing Completeness). [Go to proof]
(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 $[\Omega] \Omega = [\Omega'] \Omega'$.

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

Lemma 57 (Multiple Confluence). [Go to proof]
If $\Delta \rightarrow \Omega$ and $\Omega \rightarrow \Omega'$ and $\Delta' \rightarrow \Omega'$ 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.

Lemma 59 (Canonical Completion). [Go to proof]
If $\Gamma \rightarrow \Omega$
then there exists $\Omega_{\text{canon}}$ such that $\Gamma \rightarrow \Omega_{\text{canon}}$ and $\Omega_{\text{canon}} \rightarrow \Omega$ and $\text{dom}(\Omega_{\text{canon}}) = \text{dom}(\Gamma)$, and for all $\kappa : \tau$ and $\alpha : \tau$ in $\Omega_{\text{canon}}$, we have $\text{FEV}(\tau) = \emptyset$.

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

Lemma 60 (Split Solutions). [Go to proof]
If $\Delta \rightarrow \Omega$ and $\kappa \in \text{unsolved}(\Delta)$
then there exists $\Omega_1 = \Omega_{\text{canon}}[\kappa : t_1]$ such that $\Omega_1 \rightarrow \Omega$ and $\Omega_2 = \Omega_{\text{canon}}[\kappa : t_2]$ 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). [Go to proof]

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

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

In both cases, the height of $D'$ is one greater than the height of $D$.
Moreover, similar properties hold for the eliminating judgment $\Psi \vdash \Pi :: \tilde{A} \iff C \ p$. 

July 9, 2015
Lemma 62 (Case Invertibility).  
If $\Psi \vdash e_0 \rightarrow A.1$ and $\Psi \vdash \Pi : A \leftrightarrow C.p$ and $\Psi \vdash \Pi$ covers $A$
then $\Psi \vdash e_0 \rightarrow A.1$ and $\Psi \vdash \Pi : A \leftrightarrow C.p$ and $\Psi \vdash \Pi$ covers $A$
where the height of each resulting derivation is strictly less than the height of the given derivation.

E Miscellaneous Properties of the Algorithmic System

Lemma 63 (Well-Formed Outputs of Typing).  
(Spines) If $\Gamma \vdash s : A \rightarrow C \rightarrow \Delta$ or $\Gamma \vdash s : A \rightarrow C \rightarrow \Delta$
and $\Gamma \vdash A \rightarrow q$ type
then $\Delta \vdash C \rightarrow p$ type.

(Synthesis) If $\Gamma \vdash e \rightarrow A.\Delta$
then $A \vdash p$ type.

F Decidability of Instantiation

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

Lemma 65 (Left Free Variable Preservation).  
If $\Gamma_0 \vdash \alpha / \beta$ and $\beta \in \text{unsolved}(\Gamma_0)$ and $\beta \notin \text{FV}(|\Gamma|)$, then $\beta \notin \text{FV}(|\Delta|)$.

Lemma 66 (Instantiation Size Preservation).  
If $\Gamma_0 \vdash \alpha / \beta$ and $|\Gamma_0| = |\Delta|$, then $|\Gamma| = |\Delta|$, where $|C|$ is the plain size of the term $C$.

Lemma 67 (Decidability of Instantiation).  
If $\Gamma = \Gamma_0[\alpha / \beta']$ and $\Gamma \vdash t : \kappa$ such that $|\Gamma|t = t$
and $\alpha \notin \text{FV}(t)$, then:
(1) Either there exists $\Delta$ such that $\Gamma_0[\alpha / \beta'] \vdash \alpha := t : \kappa \rightarrow \Delta$, 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, $\cdot * \Gamma$ and $\Gamma * .$

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

$$\Gamma_L * \Gamma_R \rightarrow (\Delta_L * \Delta_R)$$

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

Lemma 68 (Transitivity of Separation).  
If $(\Gamma_L * \Gamma_R) \rightarrow (\Theta_L * \Theta_R)$ and $(\Theta_L * \Theta_R) \rightarrow (\Delta_L * \Delta_R)$
then $(\Gamma_L * \Gamma_R) \rightarrow (\Delta_L * \Delta_R)$. 

July 9, 2015
Lemma 69 (Separation Truncation). \( \text{Go to proof} \)
If \( H \) has the form \( \alpha : \kappa \) or \( \triangleright \overline{\alpha} \) or \( \triangleright P \)
and \( (\Gamma_L \ast (\Gamma_R, H)) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \)
then \( (\Gamma_L \ast (\Gamma_R)) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \) where \( \Delta_R = (\Delta_0, H, \Theta) \).

Lemma 70 (Separation for Auxiliary Judgments). \( \text{Go to proof} \)

(i) If \( \Gamma_L \ast \Gamma_R \vdash \sigma \triangleq \tau : \kappa \vdash \Delta \)
and \( \text{FEV}(\sigma) \cup \text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

(ii) If \( \Gamma_L \ast \Gamma_R \vdash P \text{ true} \vdash \Delta \)
and \( \text{FEV}(P) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

(iii) If \( \Gamma_L \ast \Gamma_R / \sigma \triangleq \tau : \kappa \vdash \Delta \)
and \( \text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

(iv) If \( \Gamma_L \ast \Gamma_R / P \vdash \Delta \)
and \( \text{FEV}(P) = \emptyset \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

(v) If \( \Gamma_L \ast \Gamma_R \vdash \Delta \triangleq \tau : \kappa \vdash \Delta \)
and \( (\text{FEV}(\tau) \cup \{\triangle]\}) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

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

(vii) If \( \Gamma_L \ast \Gamma_R \vdash A \equiv B \vdash \Delta \)
and \( \text{FEV}(A) \cup \text{FEV}(B) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

Lemma 71 (Separation for Subtyping). \( \text{Go to proof} \)
If \( \Gamma_L \ast \Gamma_R \vdash A <_{\overline{\alpha}} B \vdash \Delta \)
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 \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

Lemma 72 (Separation—Main). \( \text{Go to proof} \)

(Spines) If \( \Gamma_L \ast \Gamma_R \vdash s : A <_{\overline{\alpha}} C \vdash \Delta \)
or \( \Gamma_L \ast \Gamma_R \vdash s : A <_{\overline{\alpha}} C \vdash \Delta \)
and \( \Gamma_L \ast \Gamma_R \vdash A \vdash \text{type} \)
and \( \text{FEV}(A) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \) and \( \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \).

(Checking) If \( \Gamma_L \ast \Gamma_R \vdash e \leftrightarrow C \vdash \Delta \)
and \( \Gamma_L \ast \Gamma_R \vdash C \vdash \text{type} \)
and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).

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

(Match) If \( \Gamma_L \ast \Gamma_R \vdash \Pi :: \overline{A} \leftarrow C \vdash \Delta \)
and \( \text{FEV}(\overline{A}) = \emptyset \)
and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash_{\overline{\alpha}} (\Delta_L \ast \Delta_R) \).
(Match Elim.) If \( \Gamma_L \ast \Gamma_R / P \vdash \Pi :: \vec{A} \leftarrow C \ p \rightarrow \Delta \)
and \( \text{FEV}(P) = \emptyset \)
and \( \text{FEV}(\vec{A}) = \emptyset \)
and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \rightarrow^* (\Delta_L \ast \Delta_R) \).

H Decidability of Algorithmic Subtyping

Definition 6. The following connectives are large:

\( \forall \ \supset \ \land \)

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 \( A \) is denoted by \( \#\text{large}(A) \).

H.1 Lemmas for Decidability of Subtyping

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

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

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

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

Lemma 77 (Equiv Solving). Go to proof
If \( \Gamma \vdash A \equiv B : \kappa \rightarrow \Delta \) 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 \( \Delta^\perp \) 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) \( \Gamma \vdash \sigma \equiv t : \kappa \rightarrow \Delta \)
(2) \( \Gamma \vdash P \text{ true} \rightarrow \Delta \)
(3) \( \Gamma \vdash P \equiv Q : \kappa \rightarrow \Delta \)
(4) \( \Gamma / \sigma \equiv t : \kappa \rightarrow \Delta^\perp \)
(5) \( \Gamma / P \rightarrow \Delta^\perp \)

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 \( \Gamma \vdash A \equiv B : \rightarrow \Delta \).

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 \( \Gamma \vdash A \rightarrow \Delta \).
H.3 Decidability of Matching and Coverage

Lemma 80 (Decidability of Expansion Judgments).

Given branches \( \Pi \), it is decidable whether:

1. there exists \( \Pi' \) such that \( \Pi \sim \Pi' \);
2. there exist \( \Pi_L \) and \( \Pi_R \) such that \( \Pi \sim \Pi_L \parallel \Pi_R \);
3. there exists \( \Pi' \) such that \( \Pi \sim \Pi' \);
4. there exists \( \Pi' \) such that \( \Pi \sim \Pi' \).

Theorem 2 (Decidability of Coverage).

Given a context \( \Gamma \), branches \( \Pi \) and types \( \Lambda \), it is decidable whether \( \Gamma \vdash \Pi \) covers \( \Lambda \) is derivable.

H.4 Decidability of Typing

Theorem 3 (Decidability of Typing).

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

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

(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
\[ \Gamma \vdash e \Leftarrow B \vdash \neg \Delta. \]

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

Also, if given a proposition \( P \) as well, it is decidable whether there exists \( \Delta \) such that \( \Gamma / P \vdash \Pi :: \vec{\Lambda} \Leftarrow C \vdash \neg \Delta. \)

I Determinacy

Lemma 81 (Determinacy of Auxiliary Judgments).

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

it is the case that \( \Delta_1 = \Delta_2 \).

2. Instantiation: Given \( \Gamma \), \( \hat{\alpha} \), \( t \), \( \kappa \) such that \( \hat{\alpha} \in \text{unsolved}(\Gamma) \) and \( \Gamma \vdash t : \kappa \) and \( \hat{\alpha} \notin \text{FV}(t) \)

and \( D_1 :: \Gamma \vdash \hat{\alpha} :: t : \kappa \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash \hat{\alpha} :: t : \kappa \vdash \Delta_2 \)

it is the case that \( \Delta_1 = \Delta_2 \).

3. Symmetric instantiation:

Given \( \Gamma \), \( \hat{\alpha} \), \( \hat{\beta} \), \( \kappa \) such that \( \hat{\alpha}, \hat{\beta} \in \text{unsolved}(\Gamma) \) and \( \hat{\alpha} \neq \hat{\beta} \)

and \( D_1 :: \Gamma \vdash \hat{\alpha} :: \hat{\beta} : \kappa \vdash \Delta_1 \) and \( D_2 :: \Gamma \vdash \hat{\beta} :: \hat{\alpha} : \kappa \vdash \Delta_2 \)

it is the case that \( \Delta_1 = \Delta_2 \).

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

it is the case that \( \Delta_1 = \Delta_2 \).

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

it is the case that \( \Delta_1 = \Delta_2 \).

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

it is the case that \( \Delta_1 = \Delta_2 \).
Lemma 82 (Determinacy of Equivalence). [Go to proof]

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

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

Theorem 4 (Determinacy of Subtyping). [Go to proof]

(1) Subtyping: Given $\Gamma, C, D$ such that $D_1 :: \Gamma \vdash C < : A \vdash \Delta_1$ and $D_2 :: \Gamma \vdash C < : A \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

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

(3) Spine judgments:

Given $\Gamma, C, D, \sigma$ such that $D_1 :: \Gamma \vdash C ; A : p \vdash \Delta_1$ and $D_2 :: \Gamma \vdash C ; A : p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

The same applies for derivations of the principality-recovering judgments $\Gamma \vdash e : A ; B \vdash \Delta_1$.

(4) Match judgments:

Given $\Gamma, C, D, \sigma$ such that $D_1 :: \Gamma \vdash C ; A : p \vdash \Delta_1$ and $D_2 :: \Gamma \vdash C ; A : p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

J Properties of Algorithmic Subtyping

K Soundness

K.1 Soundness of Instantiation

Lemma 83 (Soundness of Instantiation). [Go to proof]

If $\Gamma \vdash \Delta := \tau : \kappa \vdash \Delta$ and $\Delta \notin \text{FV}(\Gamma)\tau$ and $\Gamma )\tau = \tau$ and $\Delta \rightarrow \Omega$ then $|\Omega|\Delta = |\Delta|\tau$.

K.2 Soundness of Checkeq

Lemma 84 (Soundness of Checkeq). [Go to proof]

If $\Gamma \vdash \sigma = t : \kappa \vdash \Delta$ where $\Delta \rightarrow \Omega$ then $|\Omega|\sigma = |\Omega|t$.

K.3 Soundness of Equivalence (Propositions and Types)

Lemma 85 (Soundness of Propositional Equivalence). [Go to proof]

If $\Gamma \vdash P \equiv Q \vdash \Delta$ then $|\Omega|P = |\Omega|Q$.

Lemma 86 (Soundness of Algorithmic Equivalence). [Go to proof]

If $\Gamma \vdash A \equiv B \vdash \Delta$ then $|\Omega|A = |\Omega|B$.

K.4 Soundness of Checkprop

Lemma 87 (Soundness of Checkprop). [Go to proof]

If $\Gamma \vdash \psi$ true $\vdash \Delta$ and $\Delta \rightarrow \Omega$ then $\Psi \vdash |\Omega|\psi$ true.
K.5 Soundness of Eliminations (Equality and Proposition)

Lemma 88 (Soundness of Equality Elimination).

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

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

2. If $\Gamma / \sigma \not\vdash t : \kappa \not\vdash \bot$ then $\text{mgu}(\sigma, t) = \bot$ (that is, no most general unifier exists).

K.6 Soundness of Subtyping

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 $\Delta \not\rightarrow \Omega$ and $\Gamma \vdash A \leqsp \pm B$ then $[\Omega] \Delta \vdash [\Omega] A \leqsp \pm [\Omega] B$.

K.7 Soundness of Typing

Theorem 7 (Soundness of Match Coverage).

If $\Gamma \vdash \Pi$ covers $\vec{A}$ and $\Gamma \not\rightarrow \Omega$ and $\Gamma \vdash \vec{A}$ ! types and $\Gamma \vec{A} = \vec{A}$ then $[\Omega] \Gamma \vdash \Pi$ covers $\vec{A}$.

Lemma 89 (Well-formedness of Algorithmic Typing).

Given $\Gamma \text{ctx}$:

(i) If $\Gamma \vdash e \to A \ p \vdash \Delta$ then $\Delta \vdash A \ p$ type.

(ii) If $\Gamma \vdash s : A \ p \gg B \ q \vdash \Delta$ and $\Gamma \vdash A$ type then $\Delta \vdash B \ q$ type.

Definition 7 (Measure). Let measure $M$ 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.

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

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):

- $\text{Var} \| \Pi \| \Theta$ and $\text{EmptySpine}$ have no premises.
- $\text{Sub} \langle =, =, < \rangle$
- $\text{Anno} \langle <, -, - \rangle$
- $\Pi \| \text{Spine} \| \forall \langle =, =, < \rangle$
Theorem 8 (Soundness of Algorithmic Typing). \textit{Go to proof}

Given $\Delta \rightarrow \Omega$:

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

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

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

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

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

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

and $\Gamma \vdash \tilde{A} !$ types and $\Gamma \vdash C p$ type

then $[\Omega]\Delta / [\Omega]P \vdash [\Omega]\Pi :: [\Omega]\tilde{A} \leftarrow [\Omega]C p$.

\section{Completeness}

\subsection{Completeness of Auxiliary Judgments}

Lemma 90 (Completeness of Instantiation). \textit{Go to proof}

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

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

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

\textbf{Ref}
Lemma 91 (Completeness of Checkeq). *Go to proof*

Given $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$
and $\Gamma \vdash \sigma : \kappa$ and $\Gamma \vdash \tau : \kappa$
and $[\Omega] \sigma = [\Omega] \tau$
then $\Gamma \vdash [\Gamma] \sigma \Rightarrow [\Gamma] \tau : \kappa \rightarrow \Delta$
where $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$.

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

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

1. If $\text{mgu}(\sigma, t) = \emptyset$
   then $\Gamma / \sigma \rightarrow \Delta : \kappa \rightarrow \emptyset$
   where $\Delta$ has the form $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
   and for all $u$ such that $\Gamma \vdash u : \kappa$, it is the case that $[\Gamma, \Delta] u = \emptyset([\Gamma] u)$.

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

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

If $\Delta$ has the form $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
and, for all $u$ such that $\Gamma \vdash u : \kappa$, it is the case that $[\Gamma, \Delta] u = \emptyset([\Gamma] u)$, then:

1. If $\Gamma \vdash A$ type then $[\Gamma, \Delta] A = \emptyset([\Gamma] A)$.
2. If $\Gamma \rightarrow \Omega$ then $[\Omega] \Gamma = \emptyset([\Omega] \Gamma)$.
3. If $\Gamma \rightarrow \Omega$ then $[\Omega, \Delta] [\Gamma, \Delta] = \emptyset([\Omega] [\Gamma])$.
4. If $\Gamma \rightarrow \Omega$ then $[\Omega, \Delta] e = \emptyset([\Omega] e)$.

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

Given $\Gamma \rightarrow \Omega$
and $\Gamma \vdash P$ prop and $\Gamma \vdash Q$ prop
and $[\Omega] P = [\Omega] Q$
then $\Gamma \vdash [\Gamma] P \equiv [\Gamma] Q \rightarrow \Delta$
where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$.

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

If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$
and $\Gamma \vdash P$ prop
and $[\Gamma] P = P$
and $[\Omega] \Gamma \vdash [\Omega] P$ true
then $\Gamma \vdash P$ true $\rightarrow \Delta$
where $\Delta \rightarrow \Omega'$ and $\Omega \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$.

L.2 Completeness of Equivalence and Subtyping

Lemma 96 (Completeness of Equiv). *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 $\Gamma \vdash [\Gamma] A \equiv [\Gamma] B \rightarrow \Delta$.

Theorem 9 (Completeness of Subtyping). *Go to proof*

If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash A$ type and $\Gamma \vdash B$ type
and $[\Omega] \Gamma \vdash [\Omega] \Gamma A \leq [\Omega] B$
then there exist $\Delta$ and $\Omega'$ such that $\Delta \rightarrow \Omega'$
and $\text{dom}(\Delta) = \text{dom}(\Omega')$
and $\Omega \rightarrow \Omega'$
and $\Gamma \vdash [\Gamma] A <_{\pm} [\Gamma] B \rightarrow \Delta$. 

July 9, 2015
L.3 Completeness of Typing

**Theorem 10** (Completeness of Match Coverage). \[\text{Go to proof}\] If \([\Omega] \Gamma \vdash [\Omega] \Pi \text{ covers } [\Omega] \Lambda \] and \(\Gamma \twoheadrightarrow \Omega\) and \(\Gamma \vdash \Lambda ! \) types and \(\Gamma \Lambda = \tilde{\Lambda}\) then \(\Gamma \vdash \Pi \) covers \(\tilde{\Lambda}\).

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

(i) If \(\Gamma \vdash p\) type and \(\Gamma \vdash \Omega \odot \Gamma \odot \) such that \(\Delta \twoheadrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash e \odot [\Gamma] p \odot \Delta\).

(ii) If \(\Gamma \vdash A \ p \land \) and \(\Gamma \vdash \Omega \odot \Gamma \odot \) such that \(\Delta \twoheadrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash e \odot A' \ p' \odot \Delta\).

(iii) If \(\Gamma \vdash A \ p \land \) and \(\Gamma \vdash \Omega \odot \Gamma \odot \) such that \(\Delta \twoheadrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash s : [\Gamma] A \ p' \odot \Delta\).

(iv) If \(\Gamma \vdash A \ p \land \) and \(\Gamma \vdash \Omega \odot \Gamma \odot \) such that \(\Delta \twoheadrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash s : [\Gamma] A \ p' \odot \Delta\).

(v) If \(\Gamma \vdash \tilde{A} \ p \land \) and \(\Gamma \vdash C \ p \land \) and \(\Gamma \vdash \Omega \odot \Gamma \odot \) such that \(\Delta \twoheadrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash \Pi : [\Gamma] \tilde{A} \odot \Delta\).

(vi) If \(\Gamma \vdash \tilde{A} \ p \land \) and \(\Gamma \vdash P \ prop \land \) \(\text{FEV}(P) = \emptyset\) and \(\Gamma \vdash C \ p \land \) and \(\Gamma \vdash \Omega \odot \Gamma \odot \) such that \(\Delta \twoheadrightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash \Pi : [\Gamma] \tilde{A} \odot \Delta\).
Proofs

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

A’ Properties of the Declarative System

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

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

Proof. Let $|e|$ be the size of the expression $e$. Let $|s|$ be the size of the spine $s$. Let $|\Pi|$ be the size of the branch list $\Pi$. Let $\#large(A)$ 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, $\#large(\vec{A})$; 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 $\#large(A) = 1$
- checkelim judgment at $n$ with $\#large(A) = 1$
- checking judgment at $n$ with $\#large(A) = 2$
- checkelim judgment at $n$ with $\#large(A) = 2$
- ...)

- (match judgment at $n$ with $\#large(\vec{A}) = 1$ and $\vec{A}$ of size 1
- match judgment at $n$ with $\#large(\vec{A}) = 1$ and $\vec{A}$ of size 2
- matchelim judgment at $n$ with $\#large(\vec{A}) = 1$
- match judgment at $n$ with $\#large(\vec{A}) = 2$ and $\vec{A}$ of size 1
- match judgment at $n$ with $\#large(\vec{A}) = 2$ and $\vec{A}$ of size 2
- matchelim judgment at $n$ with $\#large(\vec{A}) = 2$
- ...)

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 $\text{Decl\lor Spine}$, $\text{Decl\supset Spine}$ and $\text{Decl\rightarrow Spine}$ 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 \texttt{DeclMatchEmpty} is applicable.

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

$$|e : A| = |e| + 1$$

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

We assume that coverage, which does not depend on any other typing judgments, is well-founded.

We likewise assume that subtyping, $\Psi \vdash A$ type, $\Psi \vdash \tau : \kappa$, and $\Psi \vdash P$ prop are well-founded.

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

- **Synthesis judgments**
  
  **Claim:** For all $n$, synthesis at $n$ depends only on judgments at $n - 1$ or less.

  **Proof.** Rule \texttt{DeclVar} has no premises.

  Rule \texttt{DeclAnno} depends on a premise at a strictly smaller term.

  Rule \texttt{Decl→E} 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 \texttt{DeclSub} depends on a synthesis judgment of size $n$. (1)

  Rule \texttt{Decl1I} has no nontrivial premises. (2)

  Rule \texttt{Decl∧I} depends on a checking judgment at $n$ with fewer large connectives. (3)

  Rule \texttt{Decl→I} depends on a checking judgment at $n$ with fewer large connectives. (3)

  Rules \texttt{Decl→I}, \texttt{Decl+I}$_k$, and \texttt{Decl×I} depend on checking judgments at size $< n$. (4)

  Rule \texttt{DeclCase} depends on:

  - a synthesis judgment at size $n$ (1),

  - a match judgment at size $< n$ (5), and

  - a coverage judgment.

- **Checkelim judgments**

  **Claim:** For all $n \geq 1$, the checkelim judgment $\Psi / P \vdash e \Leftarrow A \ p$ over terms of size $n$ depends only on checking judgments at size $n$, with a type $A'$ such that $\#\text{large}(A') = \#\text{large}(A)$.

  **Proof.** Rule \texttt{DeclCheck⊥} has no nontrivial premises.

  Rule \texttt{DeclCheckUnify} depends on a checking judgment: Since $|\theta (e)| = |e|$, this checking judgment is at $n$. Since the mgu $\theta$ is over monotypes, $\#\text{large}(\theta (A)) = \#\text{large}(A)$.

- **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 $\Psi \vdash s : A \ p \Rightarrow C \ q$ over spines of size $n$ depends only on

  (a) checking judgments at size $n - 1$ or smaller, and
Proof. Rule $\text{Decl\neg Spine}$ depends on an ordinary spine judgment of size $n$, with a type that has fewer large connectives. (c)
Rule $\text{Decl\neg Spine}$ depends on an ordinary spine judgment of size $n$, with a type that has fewer large connectives. (c)
Rule $\text{DeclEmptySpine}$ has no premises.
Rule $\text{DeclSpine}$ 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 $n$.

- Match judgments

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

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

- Matchelim judgments

Claim: For all $n \geq 1$, the matchelim judgment $\Psi / \Pi \vdash P \colon \vec{A} \leftarrow C \ p$ over $\Psi$ of size $n$ depends only on match judgments with the same number of large connectives in $\vec{A}$.

- Lemma 2 (Declarative Weakening).

(i) If $\Psi_0, \psi_1 \vdash \tau : \kappa$ then $\Psi_0, \psi, \psi_1 \vdash \tau : \kappa$.

(ii) If $\Psi_0, \psi_1 \vdash P \prop then \Psi_0, \psi, \psi_1 \vdash P \prop$.

(iii) If $\Psi_0, \psi_1 \vdash \true then \Psi_0, \psi, \psi_1 \vdash \true$.

(iv) If $\Psi_0, \psi_1 \vdash A \type then \Psi_0, \psi, \psi_1 \vdash A \type$.

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

1. If $\Psi_0, \alpha : \kappa, \psi_1 \vdash \tau : \kappa$ then $\Psi_0, [\tau/\alpha]\psi_1 \vdash [\tau/\alpha] \tau' : \kappa$.

2. If $\Psi_0, \alpha : \kappa, \psi_1 \vdash P \prop then \Psi_0, [\tau/\alpha]\psi_1 \vdash [\tau/\alpha] P \prop$.

3. If $\Psi_0, \alpha : \kappa, \psi_1 \vdash A \type then \Psi_0, [\tau/\alpha]\psi_1 \vdash [\tau/\alpha] A \type$.

4. If $\Psi_0, \alpha : \kappa, \psi_1 \vdash \leq B then \Psi_0, [\tau/\alpha]\psi_1 \vdash [\tau/\alpha] \leq \leq [\tau/\alpha] B$.

5. If $\Psi_0, \alpha : \kappa, \psi_1 \vdash \true then \Psi_0, [\tau/\alpha]\psi_1 \vdash [\tau/\alpha] \true$. 

- Lemma 4 (Reflexivity of Declarative Subtyping)
Proof. By induction on the derivation of the substitutee.

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

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

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

- **Case** $\text{nonpos}(A)$, $\text{nonneg}(A)$, $p = \pm$
  
  By rule $\leq_{\text{Ref} \pm}$

- **Case** $A = \exists b : \kappa, B, p = +$
  
  $\Gamma, b : \kappa \vdash B \leq^+ B$ \hspace{1cm} By i.h. (one less quantifier)
  
  $\Gamma, b : \kappa \vdash b : \kappa$ \hspace{1cm} By rule $\text{UvarSort}$
  
  $\Gamma, b : \kappa \vdash B \leq^+ \exists b : \kappa, B$ \hspace{1cm} By rule $\leq_{\exists \text{R}}$
  
  $\Gamma \vdash \exists b : \kappa, B \leq^+ \exists b : \kappa, B$ \hspace{1cm} By rule $\leq_{\exists \text{L}}$

- **Case** $A = \exists b : \kappa, B, p = -$:
  
  $\Gamma \vdash \exists b : \kappa, B \leq^- \exists b : \kappa, B$ \hspace{1cm} By i.h. (polarities match)
  
  $\Gamma \vdash \exists b : \kappa, B \leq^- \exists b : \kappa, B$ \hspace{1cm} By $\leq$

- **Case** $A = \forall b : \kappa, B, p = +$
  
  $\Gamma \vdash \forall b : \kappa, B \leq^+ \forall b : \kappa, B$ \hspace{1cm} By i.h. (polarities match)
  
  $\Gamma \vdash \forall b : \kappa, B \leq^+ \forall b : \kappa, B$ \hspace{1cm} By $\leq$

- **Case** $A = \forall b : \kappa, B, p = -$
  
  $\Gamma, b : \kappa \vdash B \leq^- B$ \hspace{1cm} By i.h. (one less quantifier)
  
  $\Gamma, b : \kappa \vdash b : \kappa$ \hspace{1cm} By rule $\text{UvarSort}$
  
  $\Gamma, b : \kappa \vdash \forall b : \kappa, B \leq^- \forall b : \kappa, B$ \hspace{1cm} By rule $\leq_{\forall \text{L}}$
  
  $\Gamma \vdash \forall b : \kappa, B \leq^- \forall b : \kappa, B$ \hspace{1cm} By rule $\leq_{\forall \text{R}}$

Lemma 5 (Subtyping Inversion).

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

Proof. By induction on the subtyping derivations.

Lemma 6 (Subtyping Polarity Flip).

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

Proof. By induction on the subtyping derivations.

Lemma 7 (Transitivity of Declarative Subtyping).

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

1. If $D_1 :: \Psi \vdash A \leq^\pm B$ and $D_2 :: \Psi \vdash B \leq^\pm C$ then $\Psi \vdash A \leq^\pm C$.

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

We begin by case analysis on the shape of $B$, and the polarity of subtyping:
• Case $B = \forall \beta : \kappa_2. B'$, polarity = $-:$

We case-analyze $D_1$:

- Case

  \[
  \begin{array}{c}
  \psi \vdash \tau : \kappa_1 \\
  \psi \vdash [\tau/\alpha]A' \leq - B \\
  \psi \vdash \forall \alpha : \kappa_1. A' \leq - B \\
  \end{array}
  \]

  $\leq \forall L$

  $\psi \vdash \tau : \kappa_1$

  Subderivation

  $\psi \vdash [\tau/\alpha]A' \leq - B$

  Subderivation

  $\psi \vdash B \leq - C$

  Given

  $\psi \vdash [\tau/\alpha]A' \leq - C$

  By i.h. ($A$ lost a quantifier)

  $\psi \vdash A \leq - C$

  By rule $\leq \forall L$

- Case

  \[
  \begin{array}{c}
  \psi, \beta : \kappa_2 \vdash A \leq - B' \\
  \psi \vdash A \leq - \forall \beta : \kappa_2. B' \\
  \end{array}
  \]

  $\leq \forall R$

  We case-analyze $D_2$:

* Case

  \[
  \begin{array}{c}
  \psi \vdash \tau : \kappa_2 \\
  \psi \vdash [\tau/\beta]B' \leq - C \\
  \psi \vdash \forall \beta : \kappa_2. B' \leq - C \\
  \end{array}
  \]

  $\leq \forall L$

  $\psi \vdash \tau : \kappa_2$

  Subderivation

  $\psi \vdash [\tau/\beta]B' \leq - C$

  Subderivation of $D_2$

  $\psi \vdash A \leq - [\tau/\beta]B'$

  By Lemma 3 (Declarative Term Substitution)

  $\psi \vdash A \leq - C$

  By i.h. ($B$ lost a quantifier)

* Case

  \[
  \begin{array}{c}
  \psi, c : \kappa_3 \vdash B \leq - C' \\
  \psi \vdash B \leq - \forall c : \kappa_3. C' \\
  \end{array}
  \]

  $\leq \forall R$

  $\psi \vdash B \leq - \forall c : \kappa_3. C'$

  By $\leq \forall R$

  $\psi, c : \kappa_3 \vdash A \leq - C'$

  By i.h. ($C$ lost a quantifier)

  $\psi, c : \kappa_3 \vdash A \leq - C'$

  By Lemma 2 (Declarative Weakening)

  $\psi, c : \kappa_3 \vdash B \leq - C'$

  By Lemma 2 (Declarative Weakening) ($D_2$)

  $\psi, c : \kappa_3 \vdash A \leq - C'$

  By i.h. ($A$ lost a quantifier)

  $\psi, c : \kappa_3 \vdash B \leq - \forall c : \kappa_3. C'$

  By $\leq \forall R$

• Case $\text{nonpos}(B)$, polarity = $+$:

Now we case-analyze $D_1$:

- Case

  \[
  \begin{array}{c}
  \psi, \alpha : \tau \vdash A' \leq + B \\
  \psi \vdash \exists \alpha : \kappa_1. A' \leq + B \\
  \end{array}
  \]

  $\leq \exists L$

  $\psi, \alpha : \tau \vdash A' \leq + B$

  Subderivation

  $\psi, \alpha : \tau \vdash B \leq + C$

  By Lemma 2 (Declarative Weakening) ($D_2$)

  $\psi, \alpha : \tau \vdash A' \leq + C$

  By i.h. ($A$ lost a quantifier)

  $\psi \vdash \exists \alpha : \kappa_1. A' \leq + C$

  By $\leq \exists L$

- Case

  \[
  \begin{array}{c}
  \psi \vdash A \leq - B \\
  \text{nonpos}(A) \\
  \text{nonpos}(B) \\
  \psi \vdash A \leq + B \\
  \end{array}
  \]

  $\leq \exists$

  Now we case-analyze $D_2$:

* Case

  \[
  \begin{array}{c}
  \psi \vdash \tau : \kappa_3 \\
  \psi \vdash B \leq [\tau/c]C' \\
  \psi \vdash B \leq \exists c : \kappa_3. C' \\
  \end{array}
  \]

  $\leq \exists R$
Proof of Lemma 7 (Transitivity of Declarative Subtyping)

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

* Case \( \vdash B \leq - C \) \( \text{nonpos}(B) \) \( \text{nonpos}(C) \)
\[ \Psi \vdash B \leq + C \] \( \leq \exists \R \)

\[ \Psi \vdash A \leq - B \] Subderivation of \( D_1 \)
\[ \Psi \vdash B \leq - C \] Subderivation of \( D_2 \)
\[ \Psi \vdash A \leq - C \] By i.h. (\( D_1 \) and \( D_2 \) smaller)
\[ \text{nonpos}(A) \] Subderivation of \( D_1 \)
\[ \text{nonpos}(C) \] Subderivation of \( D_2 \)
\[ \Psi \vdash A \leq - + C \] By \( \leq - \)

- Case \( \Psi \vdash \tau : \kappa_3 \)
\[ \Psi \vdash B \leq + [\tau/\alpha]C' \]
\[ \Psi \vdash B \leq + \exists \beta : \kappa_2, C' \]
\[ \Psi \vdash B \leq + C \] \( \leq \exists \L \)

Now we case-analyze \( D_2 \):

- Case \( \Psi, \beta : \kappa_2 \vdash B' \leq + C \)
\[ \Psi \vdash \exists \beta : \kappa_2, B' \leq + C \] \( \leq \exists \L \)

Now we case-analyze \( D_1 \):

* Case \( \Psi, \beta : \kappa_2 \vdash A \leq + [\tau/\beta]B' \)
\[ \Psi \vdash A \leq + \exists \beta : \kappa_2, B' \]
\[ \Psi, \beta : \kappa_2 \vdash B' \leq + C \] Subderivation of \( D_2 \)
\[ \Psi \vdash \tau : \kappa_2 \] Subderivation of \( D_1 \)
\[ \Psi \vdash A \leq + [\tau/\beta]B' \] Subderivation of \( D_1 \)
\[ \Psi \vdash [\tau/\beta]B' \leq + C \] By Lemma 3 (Declarative Term Substitution)
\[ \Psi \vdash A \leq + C \] By i.h. (\( B \) lost a quantifier)

* Case \( \Psi, \alpha : \kappa_1 \vdash A \leq + B \)
\[ \Psi \vdash \exists \alpha : \kappa_1, A' \leq + B \] \( \leq \exists \L \)
\[ \Psi \vdash B \leq + C \] Given
\[ \Psi, \alpha : \kappa_1 \vdash A' \leq + B \] Subderivation of \( D_1 \)
\[ \Psi, \alpha : \kappa_1 \vdash A' \leq + B \] By Lemma 2 (Declarative Weakening)
\[ \Psi, \alpha : \kappa_1 \vdash A' \leq + C \] By i.h. (\( A \) lost a quantifier)
\[ \Psi \vdash \exists \alpha : \kappa_1, A' \leq + C \] By \( \leq \exists \L \)

Case \( \text{nonneg}(B) \), polarity = -:

We case-analyze \( D_2 \):

Proof of Lemma 7 (Transitivity of Declarative Subtyping)
Proof of Lemma 7 (Transitivity of Declarative Subtyping)

We case-analyze $D_1$:

- **Case**
  \[
  \psi, c : \kappa_3 \vdash B \leq^+ C' \\
  \psi \vdash B \leq^+ \exists c : \kappa_3. C' \quad \leq_{VR}
  \]
  \[
  \psi, c : \kappa_3 \vdash B \leq^+ C' \\
  \psi, c : \kappa_3 \vdash A \leq^+ B \\
  \psi, c : \kappa_3 \vdash A \leq^+ C' \\
  \psi \vdash A \leq^+ \forall c : \kappa_3. C' \\
  \psi \vdash A \leq^+ B \leq C \\nonneg(B) \quad \nonneg(C) \\
  \psi \vdash B \leq C \\
  \leq
  \]

- **Case**
  \[
  \psi \vdash \tau : \kappa_1 \\
  \psi \vdash [\tau/\alpha]A' \leq B \\
  \psi \vdash \forall \alpha : \kappa_1. A' \leq B \quad \leq_{VL}
  \]
  \[
  \psi \vdash B \leq C \\
  \text{Given}
  \\
  \psi \vdash \tau : \kappa_1 \\
  \text{Subderivation of } D_1
  \\
  \psi \vdash [\tau/\alpha]A' \leq B \\
  \text{Subderivation of } D_1
  \\
  \psi \vdash [\tau/\alpha]A' \leq C \\
  \text{By i.h. (} \alpha \text{ lost a quantifier)}
  \\
  \psi \vdash \forall \alpha : \kappa_1. A' \leq C \\
  \text{By } \leq_{VL}
  \\
  \psi \vdash A \leq^+ B \\
  \text{Subderivation of } D_1
  \\
  \psi \vdash B \leq C \\
  \text{Subderivation of } D_2
  \\
  \psi \vdash A \leq^+ C \\
  \text{By i.h. (} D_1 \text{ and } D_2 \text{ smaller)}
  \\
  \text{nonneg} (A) \\
  \text{nonneg} (B) \\
  \psi \vdash A \leq C \\
  \leq^+
  \]

B’ Substitution and Well-formedness Properties

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

(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 $p = \top$ and $\text{FEV}(\Gamma) = \emptyset$ then $\text{FEV}(\Gamma)[\tau/\alpha]P = \emptyset$.

**Proof.** By induction on the derivations of $\Gamma \vdash A \ p \ type$ and $\Gamma \vdash P \ prop$.

**Lemma 9** (Uvar Preservation).

If $\Delta \rightarrow \Omega$ then:

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

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

**Proof.** By induction on $\Omega$, following the definition of context application (Figure 12).

**Lemma 10** (Sorting Implies Typing). If $\Gamma \vdash t : *$ then $\Gamma \vdash t \ type$.

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

**Lemma 11** (Right-Hand Substitution for Sorting). If $\Gamma \vdash t : \kappa$ then $\Gamma \vdash [\Gamma]t : \kappa$.
Proof. By induction on \( \Gamma \vdash t \) (the size of \( t \) under \( \Gamma \)).

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

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

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

- **Cases** VarSort: If \( t \) is a universal variable \( \alpha \) and the case for \( VarSort \).

- **Cases** SolvedVarSort: In this case \( t = \alpha \) and \( \Gamma = \Gamma_1[\alpha] \). Thus \( \Gamma \vdash \alpha = \Gamma_0[\alpha] \).

By the implicit assumption that \( \Gamma \) is well-formed, \( \Gamma_1, \alpha : \kappa, \Gamma_M \vdash \tau : \kappa \).

By Lemma 34 (Suffix Weakening), \( \Gamma \vdash \tau : \kappa \). Since \( |\Gamma| < |\Gamma_0| \), we can apply the i.h., giving

\[ \Gamma \vdash [\Gamma] \tau : \kappa \]

By the definition of substitution, \( |\Gamma| \tau = [\Gamma] \tau \), so we have \( \Gamma \vdash [\Gamma] \tau : \kappa \).

- **Case** SolvedVarSort: In this case \( t = t_1 \oplus t_2 \). By i.h., \( \Gamma \vdash [\Gamma] t_1 : \kappa \) and \( \Gamma \vdash [\Gamma] t_2 : \kappa \). By SolvedVarSort, \( \Gamma \vdash [\Gamma] (t_1 \oplus t_2) : \kappa \), which by the definition of substitution is \( \Gamma \vdash [\Gamma] (t_1 \oplus t_2) : \kappa \).

**Lemma 12** (Right-Hand Substitution for Propositions). \( \Gamma \vdash P \text{ prop then } \Gamma \vdash [\Gamma] P \text{ prop} \).

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

**Lemma 13** (Right-Hand Substitution for Typing). \( \Gamma \vdash A \text{ type then } \Gamma \vdash [\Gamma] A \text{ type} \).

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

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

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

Now, the new cases:

- **Case** ForallWF: In this case \( A = \forall \alpha : \kappa. A_0 \). By i.h., \( \Gamma, \alpha : \kappa \vdash [\Gamma] A \) type. By the definition of substitution, \( [\Gamma] A = [\Gamma] A_0 \) so by ForallWF, \( \Gamma \vdash \forall \alpha. [\Gamma] A \) 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.
Proof of Lemma 14 (Substitution for Sorting)

- **Case** \( u : \kappa \in \Omega \)
  \( \Omega \vdash u : \kappa \)
  \[ \text{VarSort} \]

  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 \[ \text{UvarSort} \].

  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** \[ \alpha : \kappa \in \Omega \]
  \[ \Omega \vdash \alpha : \kappa \]
  \[ \text{SolvedVarSort} \]

  \[ \begin{array}{l}
  \frac{\alpha : \kappa \in \Omega \quad \Omega \vdash \alpha : \kappa}{\Omega \vdash \alpha : \kappa} \quad \text{Subderivation} \\
  \alpha : \kappa \in \Omega \quad \Omega = (\Omega_L, \alpha : \kappa = \tau, \Omega_R) \quad \text{Decomposing } \Omega \\
  \Omega_L, \alpha : \kappa \in \Omega, \Omega_R \vdash \tau : \kappa \quad \text{By Lemma 34 (Suffix Weakening)} \\
  \Omega \vdash [\Omega] \tau : \kappa \quad \text{By Lemma 11 (Right-Hand Substitution for Sorting)} \\
  \end{array} \]

  \[ \frac{\Omega \vdash \tau : \kappa \quad [\Omega] \tau = [\Omega] \alpha}{\Omega \vdash [\Omega] \alpha : \kappa} \]

- **Case** \( \Omega \vdash 1 : \star \)
  \[ \text{UnitSort} \]

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

- **Case** \( \Omega \vdash \tau_1 : \star \quad \Omega \vdash \tau_2 : \star \)
  \[ \text{BinSort} \]

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

- **Case** \( \Omega \vdash \text{zero} : \mathbb{N} \)
  \[ \text{ZeroSort} \]

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

- **Case** \( \Omega \vdash t : \mathbb{N} \)
  \[ \text{SuccSort} \]

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

\[ \square \]

**Lemma 15** (Substitution for Prop Well-Formedness).

*If* \( \Omega \vdash P \ prop *then* \( [\Omega] \Omega \vdash [\Omega] P \ prop.\)*

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

- **Case** \( \Omega \vdash t : \mathbb{N} \quad \Omega \vdash t' : \mathbb{N} \)
  \[ \text{EqProp} \]

  \[ \begin{array}{l}
  \frac{\Omega \vdash t : \mathbb{N} \quad [\Omega] \Omega \vdash [\Omega] t : \mathbb{N} \quad \text{By Lemma 14 (Substitution for Sorting)} \\
  \Omega \vdash t : \mathbb{N} \quad [\Omega] \Omega \vdash [\Omega] t' : \mathbb{N} \quad \text{By Lemma 14 (Substitution for Sorting)} \\
  \Omega \vdash t = t' \ prop \quad [\Omega] \Omega \vdash [\Omega] (t = t') \ prop \quad \text{By EqProp} \\
  \end{array} \]

  \[ \frac{[\Omega] \Omega \vdash [\Omega] (t = t') \ prop \quad \text{By def. of subst.}}{\Omega \vdash t = t' \ prop} \]

\[ \square \]

**Lemma 16** (Substitution for Type Well-Formedness). *If* \( \Omega \vdash A \ type *then* \( [\Omega] \Omega \vdash [\Omega] A \ type.\)*
Proof of [Lemma 16](Substitution for Type Well-Formedness)

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

48

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

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

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

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

Now, the new cases:

- Case \(\Omega, \alpha : \kappa \vdash A_0\) type
  \[ \Omega \vdash \forall \alpha : \kappa. A_0 \]  
  Subderivation
  \[ [\Omega, \alpha : \kappa](\Omega, \alpha : \kappa) \vdash [\Omega]A_0 : \kappa' \]  
  By i.h.
  \[ [\Omega]\Omega \vdash \forall \alpha : \kappa. [\Omega]A_0 : \kappa' \]  
  By definition of completion
  \[ [\Omega]\Omega \vdash [\Omega]([\forall \alpha : \kappa. A_0]) : \kappa' \]  
  By \(\text{DeclAllWF}\)
  \[ [\Omega]\Omega \vdash [\Omega]([\forall \alpha : \kappa. A_0]) : \kappa' \]  
  By def. of subst.

- Case \(\exists x:\kappa\ A_0\) type
  Similar to the \(\text{ForallWF}\) case, using \(\text{DeclExistsWF}\) instead of \(\text{DeclAllWF}\).

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

- Case \(\Omega \vdash P\) prop
  \[ \Omega \vdash P \supset A_0\]  
  \(\text{ImpliesWF}\)
  Subderivation
  \[ [\Omega]\Omega \vdash [\Omega]P \supset ([\Omega]A_0) \]  
  By Lemma 15 (Substitution for Prop Well-Formedness)
  \[ [\Omega]\Omega \vdash [\Omega]P \supset ([\Omega]A_0) \]  
  By i.h.
  \[ [\Omega]\Omega \vdash ([\Omega]P) \supset ([\Omega]A_0) \]  
  By \(\text{DeclImpliesWF}\)
  \[ [\Omega]\Omega \vdash ([\Omega]P) \supset ([\Omega]A_0) \]  
  By def. of subst.

- Case \(\Omega \vdash P\) prop
  \[ \Omega \vdash A_0\wedge P\]  
  \(\text{WithWF}\)
  Similar to the \(\text{ImpliesWF}\) case.

\[ \square \]

**Lemma 17** (Substitution Stability).

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

**Proof.** By induction on \(\Omega_Z\).

Since \(\Omega_Z\) is soft, either (1) \(\Omega_Z = \cdot\) (and the result is immediate) or (2) \(\Omega_Z = (\Omega', \alpha : \kappa)\) or (3) \(\Omega_Z = (\Omega', \alpha : \kappa = 1)\). However, according to the grammar for complete contexts such as \(\Omega_Z\), (2) is impossible. Only case (3) remains.

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

\[ \square \]

**Lemma 18** (Equal Domains).

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

**Proof.** By induction on the given derivation.

\[ \square \]
Proof of Lemma 19 (Declaration Preservation)

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

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

- Case $\longrightarrow \text{Id}$
  
  This case is impossible, since by hypothesis $u$ is declared in $\Gamma$.

- Case $\Gamma \rightarrow \Delta$

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

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

- Case $\Gamma \rightarrow \Delta$

  $\frac{\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Unsolved}}{	ext{Similar to the } \rightarrow \text{Var} \text{ case.}}$

- Case $\Gamma \rightarrow \Delta$

  $\frac{\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Solved}}{	ext{Similar to the } \rightarrow \text{Var} \text{ case.}}$

- Case $\Gamma \rightarrow \Delta$

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

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

- Case $\Gamma \rightarrow \Delta$

  $\frac{\Gamma, \alpha \rightarrow \Delta, \alpha \rightarrow \text{Marker}}{	ext{Similar to the } \rightarrow \text{Eqn} \text{ case.}}$

- Case $\Gamma \rightarrow \Delta$

  $\frac{\Gamma, \beta : \kappa \rightarrow \Delta, \beta : \kappa \rightarrow \text{Solve}}{	ext{Similar to the } \rightarrow \text{Var} \text{ case.}}$

- Case $\Gamma \rightarrow \Delta$

  $\frac{\Gamma \rightarrow \Delta, \alpha : \kappa \rightarrow \text{Add}}{	ext{It is given that } u \text{ is declared in } \Gamma. \text{ By i.h., } u \text{ is declared in } \Delta, \text{ and therefore declared in } (\Delta, \alpha : \kappa).}$

- Case $\Gamma \rightarrow \Delta$

  $\frac{\Gamma \rightarrow \Delta, \alpha \rightarrow \Delta, \alpha \rightarrow \text{AddSolved}}{	ext{Similar to the } \rightarrow \text{Add} \text{ case.}}$
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  
  \[ \rightarrow \text{Id} \]
  This case is impossible, since by hypothesis $u$ and $v$ are declared in $\Gamma$.

- Case  
  \[ \Gamma \rightarrow \Delta \quad [\Delta]A = [\Delta]A' \]
  \[ \frac{\Gamma, x : A \rightarrow \Delta, x : A'}{\Delta, x : A' \rightarrow \text{Var}} \]
  Consider whether $v = x$:
  - Case $v = x$:
    It is given that $u$ is declared to the left of $v$ in $(\Gamma, x : A)$, so $u$ is declared in $\Gamma$.
    By Lemma 19 (Declaration Preservation), $u$ is declared in $\Delta$.
    Therefore $u$ is declared to the left of $v$ in $(\Delta, x : A')$.
  - Case $v \neq x$:
    Here, $v$ is declared in $\Gamma$. By i.h., $u$ is declared to the left of $v$ in $\Delta$.
    Therefore $u$ is declared to the left of $v$ in $(\Delta, x : A')$.

- Case  
  \[ \Gamma \rightarrow \Delta \quad \alpha : k \rightarrow \Delta, \alpha : k \rightarrow \text{Uvar} \]
  Similar to the $\rightarrow \text{Var}$ case.

- Case  
  \[ \Gamma \rightarrow \Delta \quad \alpha : k \rightarrow \Delta, \alpha : k \rightarrow \text{Unsolved} \]
  Similar to the $\rightarrow \text{Var}$ case.

- Case  
  \[ \Gamma \rightarrow \Delta \quad [\Delta]t = [\Delta]t' \]
  \[ \frac{\Gamma, \alpha : k \rightarrow \Delta, \alpha : k = t \rightarrow \Delta, \alpha : k = t'}{\rightarrow \text{Solved}} \]
  Similar to the $\rightarrow \text{Var}$ case.

- Case  
  \[ \Gamma \rightarrow \Delta \quad \beta : k' \rightarrow \Delta, \beta : k' = t \rightarrow \text{Solve} \]
  Similar to the $\rightarrow \text{Var}$ case.

- Case  
  \[ \Gamma \rightarrow \Delta \quad [\Delta]t = [\Delta]t' \]
  \[ \frac{\Gamma, \alpha = t \rightarrow \Delta, \alpha = t'}{\rightarrow \text{Eqn}} \]
  The equation $\hat{\alpha} = t$ 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 $(\Delta, \hat{\alpha} : k = t')$.

- Case  
  \[ \Gamma \rightarrow \Delta \quad \hat{\alpha} \rightarrow \Delta, \hat{\alpha} \rightarrow \text{Marker} \]
  Similar to the $\rightarrow \text{Eqn}$ case.
Proof of Lemma 20 (Declaration Order Preservation). If \( \Gamma \rightarrow \Delta \) and \( u \) and \( v \) are both declared in \( \Gamma \) and \( v \) is declared to the left of \( u \) 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 \( \Gamma \).

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. In each part, we proceed by induction on the derivation of \( \Gamma \rightarrow \Delta \).

- **Case** \( \Gamma \rightarrow \Delta \)
  
  By i.h., \( u \) is declared to the left of \( v \) in \( \Delta \).
  Therefore \( u \) is declared to the left of \( v \) in \( \{\Delta, \hat{\alpha} : \kappa\} \).

- **Case** \( \Gamma \rightarrow \Delta, \hat{\alpha} : \kappa \rightarrow Add \)
  
  Similar to the \( \rightarrow Add \) case.

\( \square \)

Lemma 22 (Extension Inversion).

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

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

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

(iv) If \( D \triangleright \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \rightarrow \Delta \)
then there exist unique \( \Delta_0, \tau \), and \( \Delta_1 \)
such that \( \Delta \triangleright (\Delta_0, \hat{\alpha} : \kappa = \tau, \Delta_1) \) and \( D' \triangleright \Gamma_0 \rightarrow \Delta_0 \) and \( |\Delta_0|\tau = |\Delta_0|\tau' \) where \( D' < D \).

(v) If \( D \triangleright \Gamma_0, x : A, \Gamma_1 \rightarrow \Delta \)
then there exist unique \( \Delta_0, A', \) and \( \Delta_1 \)
such that \( \Delta \triangleright (\Delta_0, x : A', \Delta_1) \) and \( D' \triangleright \Gamma_0 \rightarrow \Delta_0 \) 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 \( \text{dom}(\Gamma_0 : x : A, \Gamma_1) = \text{dom}(\Delta) \).

(vi) If \( D \triangleright \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \rightarrow \Delta \) then either

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

- there exist unique \( \Delta_0 \) and \( \Delta_1 \)
  such that \( \Delta \triangleright (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) \) and \( D' \triangleright \Gamma_0 \rightarrow \Delta_0 \) 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 \( \rightarrow Add \) case is impossible.

Throughout this proof, we shadow \( \Delta \) so that it refers to the largest proper prefix of \( \Delta \) in the statement of the lemma. For example, in the \( \rightarrow Var \) case of part (i), we really have \( \Delta = (\Delta'_{00}, x : A'), \) but we call \( \Delta'_{00} \) “\( \Delta \).”
Proof of Lemma 22 (Extension Inversion)

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

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

\[
\frac{[\Delta|A = [\Delta|A'] \quad \rightarrow \text{Var}}{\Gamma, x : A \rightarrow \Delta, x : A'}
\]

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

Given

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.

- **Case** \( \Gamma_0 \rightarrow \Delta_0 \)

if \( \Gamma'_1 \) soft then \( \Delta_1 \) soft

- **Case** \( \Delta_0, \alpha : \kappa, \Delta_1 \)

By congruence

- **Case** \( \Gamma_0, x : A \) soft then \( \Delta_1, x : A' \) soft

Since \( \Gamma'_1, x : A \) is not soft

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

\[
\frac{\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'}{\Gamma, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta, \beta : \kappa'}
\]

There are two cases:

- **Case** \( \alpha = \beta : \kappa' \):

  - Given \( \Gamma_0 = \Gamma \) and \( \Gamma_1 = \cdot \)

  - \( \Delta_0, \alpha : \kappa, \Delta_1 \)

  - Since \( \cdot \) is soft

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

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

  Given

  \( (\Gamma_0, \alpha : \kappa, \Gamma'_1, \beta : \kappa') \)

  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.

  - **Case** \( \Gamma_0 \rightarrow \Delta_0 \)

  if \( \Gamma'_1 \) soft then \( \Delta_1 \) soft

  - **Case** \( \Delta_0, \beta : \kappa', \Delta_1, \beta : \kappa' \)

  By congruence

  - **Case** \( \Gamma'_1, \beta : \kappa' \) soft then \( \Delta_1, \beta : \kappa' \) soft

  Since \( \Gamma'_1, \beta : \kappa' \) is not soft

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

\[
\frac{\Gamma, \hat{\alpha} : \kappa' \rightarrow \Delta, \hat{\alpha} : \kappa'}{\Gamma, \alpha : \kappa, \Gamma_1}
\]

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

\[ \{ \Gamma, \hat{\alpha} : \kappa' \} = (\Gamma_0, \alpha : \kappa, \Gamma_1) \]
\[ = (\Gamma_0, \alpha : \kappa, \Gamma'_1, \hat{\alpha} : \kappa') \]
\[ \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma'_1) \]

Given
Since the last element must be equal
By injectivity of syntax

\[ \Gamma \rightarrow \Delta \]
\[ \Gamma_0, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta \]
\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]

Subderivation
By equality
By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]
if \( \Gamma'_1 \) soft then \( \Delta_1 \) soft

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

By congruence

Case
\[ \Gamma \rightarrow \Delta \]
\[ [\Delta] \rightarrow [\Delta] \]
\[ \Gamma_0, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta, \hat{\alpha} : \kappa = t' \]

Similar to the \( \rightarrow \) case.

Case
\[ \Gamma \rightarrow \Delta \]
\[ [\Delta] \rightarrow [\Delta] \]
\[ \Gamma_0, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta, \hat{\alpha} : \kappa = t' \]

By congruence

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

Given
Since the last element must be equal
By injectivity of syntax

\[ \Gamma \rightarrow \Delta \]
\[ \Gamma_0, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta \]
\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]

Subderivation
By equality
By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]
if \( \Gamma'_1 \) soft then \( \Delta_1 \) soft

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

By congruence

\[ \Gamma'_1, \beta = t \rightarrow \Delta_1, \hat{\beta} = t' \]

Since \( \Gamma'_1, \beta = t \) is not soft

Case
\[ \Gamma \rightarrow \Delta \]
\[ \Gamma_0, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta, \hat{\alpha} \]

Given
Since the last element must be equal
By injectivity of syntax

\[ \Gamma \rightarrow \Delta \]
\[ \Gamma_0, \alpha : \kappa, \Gamma'_1 \rightarrow \Delta \]
\[ \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \]

Subderivation
By equality
By i.h.

\[ \Gamma_0 \rightarrow \Delta_0 \]
if \( \Gamma'_1 \) soft then \( \Delta_1 \) soft

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

By congruence

\[ \Gamma'_1, \hat{\alpha} \rightarrow \Delta_1, \hat{\alpha} \]

Since \( \Gamma'_1, \hat{\alpha} \) is not soft
Proof of Lemma 22 (Extension Inversion)

(a) We have $\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$ By i.h.

(ii) We have $\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$ By i.h.

Suppose $\Gamma_1$ soft.

(iii) We have $\Delta = (\Delta_0, \alpha : \kappa, \Delta_1, \bar{\alpha} : \kappa')$ By congruence of equality

Suppose $\Gamma_1$ soft.

• Case $\Gamma \rightarrow \Delta$

\[
\begin{array}{c}
\rightarrow \Delta, \bar{\alpha} : \kappa \\
\Gamma_2, \alpha : \kappa', \Gamma_1 \rightarrow \Delta, \bar{\alpha} : \kappa \\
\end{array}
\]

\[
\begin{array}{c}
\Rightarrow \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \\
\text{By i.h.} \\
\end{array}
\]

Suppose $\Gamma_1$ soft.

Case $•$

$\Gamma_0 \rightarrow \Delta_0$

if $\Gamma_1$ soft then $\Delta_1$ soft

By congruence of equality

Suppose $\Gamma_1$ soft.

Case $•$

$\Delta, \bar{\alpha} : \kappa' = (\Delta_0, \alpha : \kappa, \Delta_1, \bar{\alpha} : \kappa')$

By congruence of equality

Suppose $\Gamma_1$ soft.

Case $•$

if $\Gamma_1$ soft then $\Delta_1, \bar{\alpha} : \kappa'$ soft

Implication introduction

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

$\Rightarrow$ (ii) We have $\Gamma_0, \alpha : \tau, \Gamma_1 \rightarrow \Delta$. This part is similar to part (i) above, except for “if $\text{dom}(\Gamma_0, \Rightarrow u, \Gamma_1) = \text{dom}(\Delta)$ then $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$”, which follows by i.h. in most cases. In the case, either we have $\ldots, \Rightarrow u$ where $u' = u$—in which case the i.h. gives us what we need—or we have a matching $\Rightarrow u$. In this latter case, we have $\Gamma_1 = \emptyset$. We know that $\text{dom}(\Gamma_0, \Rightarrow u, \Gamma_1) = \text{dom}(\Delta)$ and $\Delta = (\Delta_0, \Rightarrow u)$. Since $\Gamma_1 = \emptyset$, we have $\text{dom}(\Gamma_0, \Rightarrow u) = \text{dom}(\Delta_0, \Rightarrow u)$. Therefore $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$.
Proof of Lemma 22 (Extension Inversion)

Case $\Gamma \rightarrow \Delta$

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

$\Gamma_0, \alpha = \tau, \Gamma_1$

$(\Gamma_0, \alpha = \tau, \Gamma_1, \beta : \kappa') = (\Gamma_0, \alpha = \tau, \Gamma_1')$

$\Gamma = (\Gamma_0, \alpha = \tau, \Gamma_1')$

$\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$

By i.h.

$\Rightarrow [\Delta_0] \tau = [\Delta_0] \tau'$

Given

$\Rightarrow \Gamma_0 \rightarrow \Delta_0$

Given

$\Rightarrow (\Delta, \beta : \kappa') = (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa')$

By congruence of equality

Case $\Gamma \rightarrow \Delta$

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

$\Gamma_0, \alpha = \tau, \Gamma_1$

$\Delta, \alpha : \kappa' \rightarrow \Delta, \alpha : \kappa' \rightarrow \text{Uvar}$

Similar to the $\text{Uvar}$ case.

Case $\Gamma \rightarrow \Delta$

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

Similar to the $\text{Uvar}$ case.

Case $\Gamma \rightarrow \Delta$

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

$\Gamma_0, \alpha = \tau, \Gamma_1$

$\Delta, \alpha : \kappa' = t \rightarrow \Delta, \alpha : \kappa' = t'$

Similar to the $\text{Uvar}$ case.

Case $\Gamma \rightarrow \Delta$

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

$\Gamma_0, \alpha = \tau, \Gamma_1$

Similar to the $\text{Uvar}$ case.

Case $\Gamma \rightarrow \Delta$

$[\Delta] t = [\Delta] t' \rightarrow \text{Eqn}$

$\Gamma_0, \alpha = \tau, \Gamma_1$

$\Gamma, \beta = t \rightarrow \Delta, \beta = t'$

There are two cases:

- Case $\alpha = \beta$:

  $\tau = t$ and $\Gamma_1 = \cdot$ and $\Gamma_0 = \Gamma$

  By injectivity of syntax

  $\Rightarrow \Gamma_0 \rightarrow \Delta_0$

  Subderivation ($\Gamma_0 = \Gamma$ and let $\Delta_0 = \Delta$)

  where $\Delta_1 = \cdot$

  $\Rightarrow [\Delta_0] t = [\Delta_0] t'$

  By premise $[\Delta] t = [\Delta] t'$

- Case $\alpha \neq \beta$:

  $(\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \beta = t)$

  Given

  Since the final elements must be equal

  $\Rightarrow \Gamma = (\Gamma_0, \alpha = \tau, \Gamma_1')$

  By injectivity of context syntax

  $\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$

  By i.h.

  $\Rightarrow [\Delta_0] \tau = [\Delta_0] \tau'$

  ""

  $\Rightarrow \Gamma_0 \rightarrow \Delta_0$

  ""

  $\Rightarrow (\Delta, \beta = t') = (\Delta_0, \alpha = \tau', \Delta_1, \beta = t')$

  By congruence of equality
Proof of Lemma 22 (Extension Inversion) \lem:extension-inversion

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma \rightarrow \Delta, \beta : \kappa' \rightarrow \text{Add}\]
  \[
  \Delta = (\Delta_0, \alpha = \tau', \Delta_1)
  \] By i.h.
  \[
  [\Delta_0] \tau = [\Delta_0] \tau'
  \]
  \[
  \Gamma_0 \rightarrow \Delta_0
  \]
  \[
  (\Delta, \beta : \kappa') = (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa')
  \] By congruence of equality

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma \rightarrow \Delta, \beta : \kappa' = \text{t} \rightarrow \text{AddSolved}\]
  \[
  \Delta = (\Delta_0, \alpha = \tau', \Delta_1)
  \] By i.h.
  \[
  [\Delta_0] \tau = [\Delta_0] \tau'
  \]
  \[
  \Gamma_0 \rightarrow \Delta_0
  \]
  \[
  (\Delta, \beta : \kappa' = \text{t}) = (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa' = \text{t})
  \] By congruence of equality

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

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma, \beta : \kappa' \rightarrow \text{Uvar}\]
  \[
  (\Gamma_0, \beta : \kappa = \tau, \Gamma_1) = (\Gamma, \beta : \kappa')
  \] Given
  \[
  (\Gamma_0, \beta : \kappa = \tau, \Gamma_1', \beta : \kappa')
  \] Since the final elements must be equal
  \[
  \Gamma = (\Gamma_0, \beta : \kappa = \tau, \Gamma_1')
  \] By injectivity of context syntax
  \[
  \Delta = (\Delta_0, \beta : \kappa = \tau', \Delta_1)
  \] By i.h.
  \[
  [\Delta_0] \tau = [\Delta_0] \tau'
  \]
  \[
  \Gamma_0 \rightarrow \Delta_0
  \]
  \[
  (\Delta, \beta : \kappa') = (\Delta_0, \beta : \kappa = \tau', \Delta_1, \beta : \kappa')
  \] By congruence of equality

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma, \beta : \kappa' \rightarrow \text{Var}\]
  \[
  [\Delta] A = [\Delta] A'
  \] Similar to the Uvar case.

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma, \beta : \kappa' \rightarrow \text{Marker}\]
  \[
  \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'
  \] Similar to the Uvar case.

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma, \beta : \kappa' \rightarrow \text{Unsolved}\]
  \[
  \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'
  \] Similar to the Uvar case.

- Case $\Gamma \rightarrow \Delta$
  \[\Gamma, \beta : \kappa' \rightarrow \text{Solved}\]
  \[
  [\Delta] t = [\Delta] t'
  \] There are two cases.
    - Case $\beta = \beta'$
      \[\kappa' = \kappa \text{ and } t = \tau \text{ and } \Gamma_1 = \cdot \text{ and } \Gamma = \Gamma_0 \]
      By injectivity of syntax
      \[
      (\Delta, \beta : \kappa' = \text{t'}) = (\Delta_0, \beta : \kappa' = \tau', \Delta_1)
      \] where $\tau' = \text{t'}$ and $\Delta_1 = \cdot$ and $\Delta = \Delta_0$
      \[
      \Gamma_0 \rightarrow \Delta_0
      \] From subderivation $\Gamma \rightarrow \Delta$
      \[
      [\Delta_0] \tau = [\Delta_0] \tau'
      \] From premise $[\Delta] t = [\Delta] t'$ and $x$
Proof of Lemma 22 (Extension Inversion) lem:extension-inversion

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

\[
\begin{align*}
\text{Case } \hat{\alpha} \neq \hat{\beta}: \\
(\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) = (\Gamma_0, \hat{\beta} : \kappa' = t) \\
= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \hat{\beta} : \kappa' = t) \\
\Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') \\
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \\
\text{By injectivity of context syntax}
\end{align*}
\]

\[
\begin{align*}
\hat{\alpha} \neq \hat{\beta} & \Rightarrow \Delta_0 \text{ and } \Delta_1 \\
\[\Delta_0] \tau = [\Delta_0] \tau' & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
\Gamma_0 \rightarrow \Delta_0 & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
(\Delta, \hat{\alpha} : \kappa' = \tau'') = (\Delta, \hat{\beta} : \kappa' = \tau''') & \Rightarrow \Delta \text{ by congruence of equality}
\end{align*}
\]

\[
\begin{align*}
\text{Case } \Gamma \rightarrow \Delta \\
\Gamma_0, \alpha : \kappa = \tau, \Gamma_1 & \Rightarrow \Delta, \hat{\alpha} : \kappa = \tau', \Delta_1 \\
[\Delta_0] \tau = [\Delta_0] \tau' & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
\Gamma_0 \rightarrow \Delta_0 & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
(\Delta, \hat{\alpha} : \kappa' = \tau'') = (\Delta, \hat{\beta} : \kappa' = \tau'''') & \Rightarrow \Delta \text{ by congruence of equality}
\end{align*}
\]

\[
\begin{align*}
\text{Case } \Gamma \rightarrow \Delta \\
\Gamma_0, \alpha : \kappa = \tau, \Gamma_1 & \Rightarrow \Delta, \hat{\alpha} : \kappa = \tau', \Delta_1 \\
\[\Delta_0] \tau = [\Delta_0] \tau' & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
\Gamma_0 \rightarrow \Delta_0 & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
(\Delta, \hat{\alpha} : \kappa' = \tau'') = (\Delta, \hat{\beta} : \kappa' = \tau'''') & \Rightarrow \Delta \text{ by congruence of equality}
\end{align*}
\]

\[
\begin{align*}
\text{Case } \Gamma \rightarrow \Delta \\
\Gamma_0, \alpha : \kappa = \tau, \Gamma_1 & \Rightarrow \Delta, \hat{\alpha} : \kappa = \tau', \Delta_1 \\
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) & \Rightarrow \Delta \text{ by injectivity of context syntax}
\end{align*}
\]

\[
\begin{align*}
[\Delta_0] \tau = [\Delta_0] \tau' & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
\Gamma_0 \rightarrow \Delta_0 & \Rightarrow \Delta_0 \rightarrow \Delta_1 \\
(\Delta, \hat{\alpha} : \kappa' = \tau'') = (\Delta, \hat{\beta} : \kappa' = \tau'''') & \Rightarrow \Delta \text{ by congruence of equality}
\end{align*}
\]
(vi) We have \( \Gamma_0, \check{\alpha} : \kappa, \Gamma_1 \rightarrow \Delta. \)

- Case \( \Gamma \rightarrow \Delta \)
  \[
  \frac{\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'}{\text{Uvar}}
  \]
  Given \((\Gamma_0, \check{\alpha} : \kappa, \Gamma_1) = (\Gamma, \beta : \kappa')\)
  Since the final elements must be equal
  \(\Gamma = (\Gamma_0, \check{\alpha} : \kappa, \Gamma_1')\)
  By injectivity of context syntax
  By induction, there are two possibilities:
  - \(\check{\alpha}\) is not solved:
    \[
    \Delta = (\Delta_0, \check{\alpha} : \kappa, \Delta_1) \quad \text{By i.h.}
    \]
    \[
    \Gamma_0 \rightarrow \Delta_0 \quad "\]
    \[
    (\Delta, \beta : \kappa') = (\Delta_0, \check{\alpha} : \kappa, \Delta_1, \beta : \kappa') \quad \text{By congruence of equality}
    \]
  - \(\check{\alpha}\) is solved:
    \[
    \Delta = (\Delta_0, \check{\alpha} : \kappa = \tau', \Delta_1) \quad \text{By i.h.}
    \]
    \[
    \Gamma_0 \rightarrow \Delta_0 \quad "\]
    \[
    (\Delta, \beta : \kappa') = (\Delta_0, \check{\alpha} : \kappa = \tau', \Delta_1, \beta : \kappa') \quad \text{By congruence of equality}
    \]

- Case \( \Gamma \rightarrow \Delta \)
  \[
  \frac{\Gamma, \alpha \downarrow A \rightarrow \Delta, x : A'}{\begin{array}{c}
  \text{Var} \\
  \tau : \alpha \rightarrow \alpha'
  \end{array}}
  \]
  Similar to the \(\text{Uvar}\) case.

- Case \( \Gamma \rightarrow \Delta \)
  \[
  \frac{\Gamma, \alpha \downarrow \rightarrow \Delta, \alpha \downarrow \beta}{\text{Marker}}
  \]
  Similar to the \(\text{Uvar}\) case.

- Case \( \Gamma \rightarrow \Delta \)
  \[
  \frac{\Gamma, \beta \equiv t \rightarrow \Delta, \beta \equiv t'}{\text{Eqn}}
  \]
  Similar to the \(\text{Uvar}\) case.

- Case \( \Gamma \rightarrow \Delta \)
  \[
  \frac{\Gamma, \beta : \kappa' \equiv t \rightarrow \Delta, \beta : \kappa' \equiv t'}{\text{Solved}}
  \]
  Similar to the \(\text{Uvar}\) case.

- Case \( \Gamma \rightarrow \Delta \)
  \[
  \frac{\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'}{\text{Unsolved}}
  \]
  \(\check{\alpha} \neq \check{\beta}:\)
  \[
  (\Gamma_0, \check{\alpha} : \kappa, \Gamma_1) = (\Gamma, \check{\beta} : \kappa') \quad \text{Given}
  \]
  \[
  = (\Gamma_0, \check{\alpha} : \kappa, \Gamma_1', \check{\beta} : \kappa') \quad \text{Since the final elements must be equal}
  \]
  \(\Gamma = (\Gamma_0, \check{\alpha} : \kappa, \Gamma_1')\)
  By injectivity of context syntax
  By induction, there are two possibilities:
  \* \(\check{\alpha}\) is not solved:
    \[
    \Delta = (\Delta_0, \check{\alpha} : \kappa, \Delta_1) \quad \text{By i.h.}
    \]
    \[
    \Gamma_0 \rightarrow \Delta_0 \quad "\]
    \[
    (\Delta, \beta : \kappa') = (\Delta_0, \check{\alpha} : \kappa, \Delta_1, \beta : \kappa') \quad \text{By congruence of equality}
    \]
  \* \(\check{\alpha}\) is solved:
    \[
    \Delta = (\Delta_0, \check{\alpha} : \kappa = \tau', \Delta_1) \quad \text{By i.h.}
    \]
    \[
    \Gamma_0 \rightarrow \Delta_0 \quad "\]
    \[
    (\Delta, \beta : \kappa') = (\Delta_0, \check{\alpha} : \kappa = \tau', \Delta_1, \beta : \kappa') \quad \text{By congruence of equality}
    \]
Proof of Lemma 22 (Extension Inversion)

- Case $\alpha = \beta$:
  \[
  \kappa' = \kappa \text{ and } \Gamma_0 = \Gamma \text{ and } \Gamma_1 = \cdot \quad \text{By injectivity of syntax}
  \]
  \[
  \begin{align*}
  &\Rightarrow (\Delta, \beta : \kappa') = (\Delta, \alpha : \kappa, \Delta_1) \quad \text{where } \Delta_0 = \Delta \text{ and } \Delta_1 = \cdot
  \\
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow \Gamma \rightarrow \Delta
  \\
  \\
  \end{align*}
  \]

- Case $\Gamma \rightarrow \Delta$
  \[
  \begin{align*}
  \Gamma \rightarrow \Delta
  \\
  \Gamma_0, \alpha : \kappa, \Gamma_1
  \\
  \Rightarrow \Delta, \beta : \kappa'
  \\
  \Rightarrow \text{Add}
  \\
  \end{align*}
  \]

By induction, there are two possibilities:

- $\alpha$ is not solved:
  \[
  \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \quad \text{By i.h.}
  \]
  \[
  \begin{align*}
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') \quad \text{By congruence of equality}
  \\
  \end{align*}
  \]

- $\alpha$ is solved:
  \[
  \Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \quad \text{By i.h.}
  \]
  \[
  \begin{align*}
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa') \quad \text{By congruence of equality}
  \\
  \end{align*}
  \]

- Case $\Gamma \rightarrow \Delta$
  \[
  \begin{align*}
  \Gamma \rightarrow \Delta
  \\
  \Gamma_2, \alpha : \kappa, \Gamma_1
  \\
  \Rightarrow \Delta, \beta : \kappa' = t
  \\
  \Rightarrow \text{AddSolved}
  \\
  \end{align*}
  \]

By induction, there are two possibilities:

- $\alpha$ is not solved:
  \[
  \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \quad \text{By i.h.}
  \]
  \[
  \begin{align*}
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa' = t) \quad \text{By congruence of equality}
  \\
  \end{align*}
  \]

- $\alpha$ is solved:
  \[
  \Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \quad \text{By i.h.}
  \]
  \[
  \begin{align*}
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa' = t) \quad \text{By congruence of equality}
  \\
  \end{align*}
  \]

- Case $\alpha \neq \beta$:
  \[
  (\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma_1, \beta : \kappa') \quad \text{Given}
  \]
  \[
  = (\Gamma_0, \alpha : \kappa, \Gamma_1, \beta : \kappa') \quad \text{Since the final elements must be equal}
  \]
  \[
  \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1) \quad \text{By injectivity of context syntax}
  \]

By induction, there are two possibilities:

* $\alpha$ is not solved:
  \[
  \Delta = (\Delta_0, \alpha : \kappa, \Delta_1) \quad \text{By i.h.}
  \]
  \[
  \begin{align*}
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow (\Delta, \beta : \kappa' = t) = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa' = t) \quad \text{By congruence of equality}
  \\
  \end{align*}
  \]

* $\alpha$ is solved:
  \[
  \Delta = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \quad \text{By i.h.}
  \]
  \[
  \begin{align*}
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow (\Delta, \beta : \kappa' = t) = (\Delta_0, \alpha : \kappa = \tau', \Delta_1, \beta : \kappa' = t) \quad \text{By congruence of equality}
  \\
  \end{align*}
  \]

- Case $\alpha = \beta$:
  \[
  \Gamma = \Gamma_0 \text{ and } \kappa = \kappa' \text{ and } \Gamma_1 = \cdot \quad \text{By injectivity of syntax}
  \]
  \[
  \begin{align*}
  &\Rightarrow (\Delta, \beta : \kappa' = t) = (\Delta_0, \alpha : \kappa = \tau', \Delta_1) \quad \text{where } \Delta_0 = \Delta \text{ and } \tau' = t \text{ and } \Delta_1 = \cdot
  \\
  &\Rightarrow \Gamma_0 \rightarrow \Delta_0
  \\
  &\Rightarrow \Gamma \rightarrow \Delta
  \\
  \end{align*}
  \]

\[\square\]
Lemma 23 (Deep Evar Introduction). (i) If $\Gamma_0, \Gamma_1$ is well-formed and $\hat{\alpha}$ is not declared in $\Gamma_0, \Gamma_1$ then $\Gamma_0, \Gamma_1 \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$.

(ii) If $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$ is well-formed and $\Gamma$ derivability $t : \kappa$ then $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$.

(iii) If $\Gamma_0, \Gamma_1$ is well-formed and $\Gamma$ derivability $t : \kappa$ then $\Gamma_0, \Gamma_1 \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$.

Proof.

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

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

- Case $\Gamma_1 = \Gamma'_1, x : A$:
  - $\Gamma_0, \Gamma'_1, x : A ctx \ derivationp$ Given
  - $\Gamma_0, \Gamma'_1 ctx \ derivationp$ By inversion
  - $x \ \notin \ dom(\Gamma_0, \Gamma'_1)$ By inversion (1)
  - $\Gamma_0, \Gamma'_1 \ derivationp$ $A type \ derivationp$ By inversion
  - $\hat{\alpha} \ \notin \ dom(\Gamma_0, \Gamma'_1, x : A)$ Given
  - $\hat{\alpha} \neq x \ derivationp$ By inversion (2)
  - $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1 ctx \ derivationp$ By i.h.
  - $\Gamma_0, \Gamma'_1 \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1$ \ derivationp By Lemma 36 (Extension Weakening (Sorts))
  - $x \ \notin \ dom(\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1)$ By (1) and (2)
  - $\Rightarrow \Gamma_0, \Gamma'_1, x : A \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1, x : A \ derivationp$ By $\Var$

- Case $\Gamma_1 = \Gamma'_1, \beta : \kappa'$:
  - $\Gamma_0, \Gamma'_1, \beta : \kappa' ctx \ derivationp$ Given
  - $\Gamma_0, \Gamma'_1 ctx \ derivationp$ By inversion
  - $\beta \ \notin \ dom(\Gamma_0, \Gamma'_1)$ By inversion (1)
  - $\hat{\alpha} \ \notin \ dom(\Gamma_0, \Gamma'_1, \beta : \kappa')$ Given
  - $\hat{\alpha} \neq \beta \ derivationp$ By inversion (2)
  - $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1 ctx \ derivationp$ By i.h.
  - $\Gamma_0, \Gamma'_1 \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1$ \ derivationp By (1) and (2)
  - $\Rightarrow \Gamma_0, \Gamma'_1, \beta : \kappa' \ derivationp$ $\Gamma_0, \hat{\alpha} : \kappa, \Gamma'_1, \beta : \kappa' \ derivationp$ By $\Uvar$

- Case $\Gamma_1 = (\Gamma'_1, \hat{\beta} : \kappa' = t)$:
Proof of Lemma 23 (Deep Evar Introduction)

\[
\Gamma_0, \Gamma'_1, \bar{\beta} : \kappa' = t \text{ cx} \\
\Gamma_0, \Gamma'_1 \text{ cx} \\
\beta \notin \text{dom}(\Gamma_0, \Gamma'_1) \\
\Gamma_0, \Gamma'_1 \vdash t : \kappa' \\
\bar{\alpha} \notin \text{dom}(\Gamma_0, \Gamma'_1, \bar{\beta} : \kappa' = t) \\
\bar{\alpha} \neq \bar{\beta} \\
\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \text{ cx} \\
\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \longrightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \\
\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \vdash t : \kappa' \\
\bar{\beta} \notin \text{dom}(\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1) \\
\]  

\[\Rightarrow \Gamma_0, \Gamma'_1, \bar{\beta} : \kappa' = t \longrightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1, \bar{\beta} : \kappa' = t \]

- Case \(\Gamma_1 = (\Gamma'_1, \bar{\beta} = t)\):
  \[
  \Gamma_0, \Gamma'_1, \bar{\beta} = t \text{ cx} \\
  \Gamma_0, \Gamma'_1 \text{ cx} \\
  \beta \notin \text{dom}(\Gamma_0, \Gamma'_1) \\
  \Gamma_0, \Gamma'_1 \vdash t : \mathbb{N} \\
  \bar{\alpha} \notin \text{dom}(\Gamma_0, \Gamma'_1, \bar{\beta} = t) \\
  \bar{\alpha} \neq \bar{\beta} \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \text{ cx} \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \longrightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \\
  \beta \notin \text{dom}(\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1) \\
  \]  

\[\Rightarrow \Gamma_0, \Gamma'_1, \bar{\beta} = t \longrightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1, \bar{\beta} = t \]

- Case \(\Gamma_1 = (\Gamma'_1, \bullet \bar{\beta})\):
  \[
  \Gamma_0, \Gamma'_1, \bullet \bar{\beta} \text{ cx} \\
  \Gamma_0, \Gamma'_1 \text{ cx} \\
  \beta \notin \text{dom}(\Gamma_0, \Gamma'_1) \\
  \bar{\alpha} \notin \text{dom}(\Gamma_0, \Gamma'_1, \bullet \bar{\beta}) \\
  \bar{\alpha} \neq \bar{\beta} \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \text{ cx} \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \longrightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \\
  \beta \notin \text{dom}(\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1) \\
  \]  

\[\Rightarrow \Gamma_0, \Gamma'_1, \bullet \bar{\beta} \longrightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1, \bullet \bar{\beta} \]

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

- Case \(\Gamma_1 = :\):
  \[
  \Gamma_0 \vdash t : \kappa \\
  \Gamma_0, \Gamma_1 \text{ cx} \\
  \Gamma_0 \text{ cx} \\
  \Gamma_0 \longrightarrow \Gamma_0 \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma_1 \longrightarrow \Gamma_0, \bar{\alpha} : \kappa = t, \Gamma_1 \\
  \]  

- Case \(\Gamma_1 = (\Gamma'_1, x : A)\):
  \[
  \Gamma_0 \vdash t : \kappa \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1, x : A \text{ cx} \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \text{ cx} \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \vdash A \text{ type} \\
  x \notin \text{dom}(\Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1) \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1 \longrightarrow \Gamma_0, \bar{\alpha} : \kappa = t, \Gamma_1 \\
  \Gamma_0, \bar{\alpha} : \kappa = t, \Gamma_1 \vdash A \text{ type} \\
  x \notin \text{dom}(\Gamma_0, \bar{\alpha} : \kappa = t, \Gamma'_1) \\
  \Gamma_0, \bar{\alpha} : \kappa, \Gamma'_1, x : A \longrightarrow \Gamma_0, \bar{\alpha} : \kappa = t, \Gamma_1, x : A \\
  \]  

Since this is the same domain as (1)
• Case $\Gamma_1 = (\Gamma_1', \beta : \kappa')$:
  
  \[ \Gamma_0 \vdash t : \kappa \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta : \kappa' \text{ctx} \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \text{ctx} \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa, \Gamma_1') \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \rightarrow \Gamma_0, \Delta : \kappa = t, \Gamma_1 \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa = t, \Gamma_1') \]
  since this is the same domain as (1)
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta : \kappa' \rightarrow \Gamma_0, \Delta : \kappa = t, \Gamma_1, \beta : \kappa' \]
  By rule $\rightarrow \text{Uvar}$

• Case $\Gamma_1 = (\Gamma_1', \beta : \kappa' = t')$:
  
  \[ \Gamma_0 \vdash t' : \kappa \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta : \kappa' = t' \text{ctx} \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \text{ctx} \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa, \Gamma_1') \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \rightarrow \Gamma_0, \Delta : \kappa = t, \Gamma_1 \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa = t, \Gamma_1') \]
  since this is the same domain as (1)
  \[ \Gamma_0, \Delta : \kappa = t, \Gamma_1 \vdash t' : \kappa' \]
  By Lemma 36 (Extension Weakening (Sorts))
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta : \kappa' = t' \rightarrow \Gamma_0, \Delta : \kappa = t', \Gamma_1, \beta : \kappa' = t' \]
  By rule $\rightarrow \text{Solved}$

• Case $\Gamma_1 = (\Gamma_1', \beta = t')$:
  
  \[ \Gamma_0 \vdash t' : \kappa \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta = t' \text{ctx} \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \text{ctx} \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa, \Gamma_1') \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \rightarrow \Gamma_0, \Delta : \kappa = t, \Gamma_1 \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa = t, \Gamma_1') \]
  since this is the same domain as (1)
  \[ \Gamma_0, \Delta : \kappa = t, \Gamma_1 \vdash t' : \kappa' \]
  By Lemma 36 (Extension Weakening (Sorts))
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta = t' \rightarrow \Gamma_0, \Delta : \kappa = t', \Gamma_1, \beta = t' \]
  By rule $\rightarrow \text{Eqn}$

• Case $\Gamma_1 = (\Gamma_1', \beta : \kappa')$:
  
  \[ \Gamma_0 \vdash t : \kappa \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta : \kappa' \text{ctx} \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \text{ctx} \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa, \Gamma_1') \]
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1' \rightarrow \Gamma_0, \Delta : \kappa = t, \Gamma_1 \]
  \[ \beta \notin \text{dom}(\Gamma_0, \Delta : \kappa = t, \Gamma_1') \]
  since this is the same domain as (1)
  \[ \Gamma_0, \Delta : \kappa, \Gamma_1', \beta : \kappa' \rightarrow \Gamma_0, \Delta : \kappa = t, \Gamma_1, \beta : \kappa' \]
  By rule $\rightarrow \text{Unsolved}$

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

\[ \square \]

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

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

(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 $\Delta_L$! It's easy to avoid decomposing $\Delta_L$: when $\Delta_R = \cdot$, we don't need to apply the i.h. anyway. To avoid decomposing $\Gamma_L$, we need to reason by contradiction, using Lemma 19 (Declaration Preservation).

- **Case $\Delta_R = \cdot$:** We have $\Gamma_L \rightarrow \Delta_L$. Applying $\rightarrow \text{Unsolved}$ to that derivation gives the result.

- **Case $\Delta_R = (\Delta_L', \beta):$** We have $\beta \neq \bar{\alpha}$ by the well-formedness assumption. The concluding rule of $\Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta_R$, $\beta$ must have been $\rightarrow \text{Unsolved}$ or $\rightarrow \text{Add}$ In both cases, the result follows by i.h. and applying $\rightarrow \text{Unsolved}$ or $\rightarrow \text{Add}$

Note: In $\rightarrow \text{Add}$, the left-hand context doesn't change, so we clearly maintain $\Gamma_L \rightarrow \Delta_L$. In $\rightarrow \text{Unsolved}$ we can correctly apply the i.h. because $\Gamma_R \neq \cdot$. Suppose, for a contradiction, that $\Gamma_R = \cdot$. Then $\Gamma_L = (\Gamma_L', \bar{\beta})$. It was given that $\Gamma_L \rightarrow \Delta_L$, that is, $\Gamma_L', \bar{\beta} \rightarrow \Delta_L$. By Lemma 19 (Declaration Preservation), $\Delta_L$ has a declaration of $\bar{\beta}$. But then $\Delta = (\Delta_L, \Delta_L', \bar{\beta})$ is not well-formed: contradiction. Therefore $\Gamma_R \neq \cdot$.

- **Case $\Delta_R = (\Delta_L', \bar{\alpha}):$** We have $\beta \neq \bar{\alpha}$ by the well-formedness assumption.

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

- **Case $\Delta_R = (\Delta_L', \bar{\alpha} = \tau):$** The concluding rule must have been $\rightarrow \text{Eqn}$. The result follows by i.h. and applying $\rightarrow \text{Eqn}$.

- **Case $\Delta_R = (\Delta_L', \bar{\tau}):$** Similar to the previous case, with rule $\rightarrow \text{Marker}$

- **Case $\Delta_R = (\Delta_L', \bar{x} : A):$** Similar to the previous case, with rule $\rightarrow \text{Var}$

(ii) Similar to part (i), except that when $\Delta_R = \cdot$, apply rule $\rightarrow \text{Solve}$

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

Lemma 27 (Parallel Extension Solution). If $\Gamma_L, \bar{x} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau', \Delta_R$ and $\Gamma_L \vdash \tau : \kappa$ and $[\Delta_L] \tau = [\Delta_L] \tau'$, then $\Gamma_L, \bar{x} : \kappa = \tau, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau', \Delta_R$.

Proof. By induction on $\Delta_R$.

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

Lemma 28 (Parallel Variable Update). If $\Gamma_L, \bar{x} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau_0, \Delta_R$ and $\Gamma_L \vdash \tau_1 : \kappa$ and $[\Delta_L] \tau_0 = [\Delta_L] \tau_1 = [\Delta_L] \tau_2$ then $\Gamma_L, \bar{x} : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau_2, \Delta_R$.

Proof. By induction on $\Delta_R$. Similar to the proof of Lemma 27 (Parallel Extension Solution), but applying $\rightarrow \text{Solved}$ at the end. □

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

(ii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash \mathsf{P}$ then $[\Delta][\Gamma]\mathsf{P} = [\Delta]\mathsf{P}$.

(iii) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash \mathsf{A}$ type then $[\Delta][\Gamma]\mathsf{A} = [\Delta]\mathsf{A}$.

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 $\mathcal{D} : \Gamma \rightarrow \Delta$ and $\Gamma \vdash t : \kappa$. We proceed by cases on the derivation of $\Gamma \vdash t : \kappa$.

  - **Case** $\hat{\alpha} : \kappa \in \Gamma$ 
    \[
    \Gamma \vdash \hat{\alpha} : \kappa
    \]
    \[\Gamma[\hat{\alpha}] = \hat{\alpha} \quad \text{Since } \hat{\alpha} \text{ is not solved in } \Gamma\]
    \[\Delta[\hat{\alpha}] = [\Delta]\hat{\alpha} \quad \text{Reflexivity}\]
    \[= [\Delta][\Gamma]\hat{\alpha} \quad \text{By above equality}\]

  - **Case** $(\alpha : \kappa) \in \Gamma$ 
    \[\Gamma \vdash \alpha : \kappa\] 
    \[\text{Consider whether or not there is a binding of the form } (\alpha = \tau) \in \Gamma.\]
    
    * **Case** $(\alpha = \tau) \in \Gamma$:
      \[
      \Delta = (\Delta_0, \alpha = \tau', \Delta_1)
      \]
      \[
      \mathcal{D}' :: \quad \Gamma_0 \rightarrow \Delta_0
      \]
      \[
      \mathcal{D}' < \mathcal{D}
      \]
      \[(1) \quad [\Delta_0][\tau'] = [\Delta_0][\tau]
      \]
      \[(2) \quad [\Delta_0][[\Gamma_0]] = [\Delta_0][\tau]
      \]
      \[
      [\Delta][\Gamma]\alpha = (\Delta_0, \alpha = \tau', \Delta_1)[[\Gamma_0]]\alpha
      \]
      \[
      = [\Delta_0, \alpha = \tau', \Delta_1][[\Gamma_0]]\alpha
      \]
      \[
      = [\Delta_0, \alpha = \tau', \Delta_1][[\Gamma_0]]\alpha
      \]
      \[
      = [\Delta_0][\tau']\alpha
      \]
      \[
      = [\Delta_0, \alpha = \tau', \Delta_1]\alpha
      \]
      \[
      = [\Delta]\alpha
      \]
      \[\text{By Lemma 22 (Extension Inversion) (i)}\]
      \[\text{By i.h.}\]
      \[\text{By definition of substitution}\]
      \[\text{By definition of substitution}\]
      \[\text{By definition of substitution}\]
      \[\text{By definition of } \Delta\]

    * **Case** $(\alpha = \tau) \notin \Gamma$:
      \[
      \Gamma[\hat{\alpha}] = \alpha \quad \text{By definition of substitution}\]
      \[\Delta[\Gamma]\alpha = [\Delta]\alpha \quad \text{Apply } [\Delta] \text{ to both sides}\]

  - **Case** $\Gamma \vdash 1 : *$
    \[
    \Gamma \vdash 1 : *
    \]
    \[\Delta_1 = 1 = [\Delta][\Gamma]1 \quad \text{Since } \text{FV(1)} = \emptyset\]

  - **Case** $\Gamma \vdash \tau_1 : *$ 
    \[
    \Gamma \vdash \tau_1 : *
    \]
    \[
    \Gamma \vdash \tau_1 \oplus \tau_2 : *
    \]
    \[\mathcal{D} \vdash \tau_1 \oplus \tau_2 : *
    \]
    \[
    [\Delta][\Gamma]\tau_1 = [\Delta][\Gamma]\tau_1
    \]
    \[\text{By i.h.}\]
    \[
    [\Delta][\Gamma]\tau_2 = [\Delta][\Gamma]\tau_2
    \]
    \[\text{By i.h.}\]
    \[
    [\Delta][\Gamma](\tau_1 \oplus \tau_2) = [\Delta](\tau_1 \oplus \tau_2)
    \]
    \[\text{By congruence of equality}\]
    \[\text{By definition of substitution}\]
Proof of Lemma 29 (Substitution Monotonicity)

- Case
  \[ \Gamma \vdash \text{zero} : \text{ZeroSort} \]
  \[ [\Delta]\text{zero} = \text{zero} = [\Delta][\Gamma]\text{zero} \quad \text{Since FV(\text{zero}) = } \emptyset \]
- Case
  \[ \Gamma \vdash \text{t : N} \]
  \[ \Gamma \vdash \text{succ(t) : SuccSort} \]
  \[ [\Delta][\Gamma]\text{t} = [\Delta]\text{t} \quad \text{By i.h.} \]
  \[ \text{succ([}\Delta][\Gamma]\text{t}) = \text{succ([}\Delta]\text{t}) \quad \text{By congruence of equality} \]
  \[ [\Delta][\Gamma]\text{succ(t)} = [\Delta]\text{succ(t)} \quad \text{By definition of substitution} \]

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

  - Case
    \[ \Gamma \vdash \text{t : N} \quad \Gamma \vdash \text{t' : N} \]
    \[ \Gamma \vdash \text{t = t'} \quad \text{prop} \quad \text{EqProp} \]
    \[ [\Delta][\Gamma]\text{t} = [\Delta]\text{t} \quad \text{By part (i)} \]
    \[ [\Delta][\Gamma]\text{t'} = [\Delta]\text{t'} \quad \text{By part (i)} \]
    \[ ([\Delta][\Gamma]\text{t}) = ([\Delta][\Gamma]\text{t'}) = ([\Delta]\text{t}) = ([\Delta]\text{t'}) \quad \text{By congruence of equality} \]
    \[ [\Delta][\Gamma]\text{(}t = t') = [\Delta]\text{(}t = t') \quad \text{Definition of substitution} \]

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

  - Case
    \[ (\text{u} : \ast) \in \Gamma \]
    \[ \Gamma \vdash \text{u type} \quad \text{VarWF} \]
    \[ \Gamma \vdash \text{u : \ast} \quad \text{By rule VarSort} \]
    \[ [\Delta][\Gamma]\text{u} = [\Delta]\text{u} \quad \text{By part (i)} \]

  - Case
    \[ (\& : \ast = \tau) \in \Gamma \]
    \[ \Gamma \vdash \& : \ast \quad \text{SolvedVarWF} \]
    \[ \Gamma \vdash \& : \ast \quad \text{By rule SolvedVarSort} \]
    \[ [\Delta][\Gamma]\& = [\Delta]\& \quad \text{By part (i)} \]

  - Case
    \[ 1 : \ast \quad \text{UnitSort} \]
    \[ \Gamma \vdash 1 \quad \text{UnitWF} \]
    \[ \Gamma \vdash 1 : \ast \quad \text{By rule UnitSort} \]
    \[ [\Delta][\Gamma]1 = [\Delta]1 \quad \text{By part (i)} \]

  - Case
    \[ \Gamma \vdash \ast : \text{type} \]
    \[ \Gamma \vdash \ast \quad \text{BinWF} \]
    \[ [\Delta][\Gamma] \ast \vdash [\Delta][\Gamma] \ast \quad \text{By i.h.} \]
    \[ [\Delta][\Gamma] \ast \vdash [\Delta][\Gamma] \ast \quad \text{By i.h.} \]
    \[ [\Delta][\Gamma] \ast \vdash [\Delta][\Gamma] \ast \quad \text{By congruence of equality} \]
    \[ [\Delta][\Gamma] \ast \vdash [\Delta][\Gamma] \ast \quad \text{Definition of substitution} \]

  - Case
    \[ \Gamma, \alpha : \kappa \vdash \ast : \text{type} \]
    \[ \Gamma \vdash \forall \alpha : \kappa. \ast : \text{type} \quad \text{ForallWF} \]
    \[ \Gamma \rightarrow \Delta \quad \text{Given} \]
    \[ \Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \quad \text{By rule } \rightarrow \text{Uvar} \]
    \[ [\Delta, \alpha : \kappa][\Gamma, \alpha : \kappa] \ast = [\Delta, \alpha : \kappa] \ast \quad \text{By i.h.} \]
    \[ [\Delta][\Gamma] \ast = [\Delta][\Gamma] \ast \quad \text{By definition of substitution} \]
    \[ \ast = \forall \alpha : \kappa. \ast \quad \text{By congruence of equality} \]
    \[ [\Delta][\Gamma] \ast = [\Delta][\Gamma] \ast \quad \text{Definition of substitution} \]
    \[ [\Delta][\Gamma] \ast = [\Delta][\Gamma] \ast \quad \text{Definition of substitution} \]
\[
\begin{align*}
\text{Case } \exists \text{ ExistsWF} & \quad \text{Similar to the } \exists \text{ ForallWF case.} \\
\text{Case } \Gamma \vdash P \text{ prop} & \quad \Gamma \vdash A_0 \text{ type} \\
\Gamma \vdash P \supset A_0 \text{ type} \quad \text{ImpliesWF} \\
[\Delta][\Gamma]P = [\Delta]P & \quad \text{By part (ii)} \\
[\Delta][\Gamma]A_0 = [\Delta]A_0 & \quad \text{By i.h.} \\
[\Delta][\Gamma](P \supset A_0) = [\Delta](P \supset A_0) & \quad \text{Definition of substitution} \\
\end{align*}
\]

\[\Gamma \vdash P \text{ prop} \quad \Gamma \vdash A_0 \text{ type} \quad \text{WithWF} \]

Similar to the ImpliesWF case.

\[\square\]

Lemma 30 (Substitution Invariance).

(i) If \( \Gamma \rightarrow \Delta \) and \( \Gamma \vdash t : \kappa \) and FEV(\([\Gamma]t\)) = 0 then \( [\Delta][\Gamma]t = [\Gamma]t \).

(ii) If \( \Gamma \rightarrow \Delta \) and \( \Gamma \vdash P \text{ prop} \) and FEV(\([\Gamma]P\)) = 0 then \( [\Delta][\Gamma]P = [\Gamma]P \).

(iii) If \( \Gamma \rightarrow \Delta \) and \( \Gamma \vdash A \text{ type} \) and FEV(\([\Gamma]A\)) = 0 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.

\[\square\]

Lemma 24 (Soft Extension).

If \( \Gamma \rightarrow \Delta \) and \( \Gamma, \Theta \text{ ctx} \) and \( \Theta \) is soft, then there exists \( \Omega \) such that dom(\( \Theta \)) = dom(\( \Omega \)) and \( \Gamma, \Theta \rightarrow \Delta, \Omega \).

Proof. By induction on \( \Theta \).

\[\begin{align*}
\text{Case } \Theta = : & \quad \text{We have } \Gamma \rightarrow \Delta. \text{ Let } \Omega =:. \text{ Then } \Gamma, \Theta \rightarrow \Delta, \Omega. \\
\text{Case } \Theta = (\Theta', \hat{\kappa} : \kappa = t): & \quad \Gamma, \Theta' \rightarrow \Gamma, \Omega' \\
\text{By i.h.} & \quad \text{By rule } \rightarrow \text{Solved} \\
\text{Case } \Theta = (\Theta', \hat{\kappa} : \kappa): & \quad \Gamma, \Theta' \rightarrow \Gamma, \Omega' \\
\text{By i.h.} & \quad \text{By rule } \rightarrow \text{Solve} \\
\end{align*}\]

\[\square\]

Lemma 31 (Split Extension).

If \( \Delta \rightarrow \Omega \)

and \( \hat{\alpha} \in \text{unsolved}(\Delta) \)

and \( \Omega = \Omega_1[\hat{\alpha} : \kappa = t_1] \)

and \( \Omega \) is canonical (Definition \[3\])

and \( \Omega \vdash t_2 : \kappa \)

then \( \Delta \rightarrow \Omega_1[\hat{\alpha} : \kappa = t_2] \).

Proof. By induction on the derivation of \( \Delta \rightarrow \Omega \). Use the fact that \( \Omega_1[\hat{\alpha} : \kappa = t_1] \) and \( \Omega_1[\hat{\alpha} : \kappa = t_2] \) agree on all solutions except the solution for \( \hat{\alpha} \). In the \( \rightarrow \text{Solve} \) case where the existential variable is \( \hat{\alpha} \), use \( \Omega \vdash t_2 : \kappa \).

\[\square\]
Proof of Lemma 32 (Extension Reflexivity)

Lemma 32 (Extension Reflexivity).

If $\Gamma \ctx$ then $\Gamma \rightarrow \Gamma$.

Proof. By induction on the derivation of $\Gamma \ctx$.

- **Case** $\ctx$ [EmptyCtx]
  
  $\Gamma \rightarrow \Gamma$  
  By rule $\rightarrow Id$

- **Case** $\Gamma \ctx \vdash x \notin \text{dom}(\Gamma) \quad \Gamma \vdash A \quad \Gamma, x : A \ctx$ [HypCtx]
  
  $\Gamma \rightarrow \Gamma$  
  By i.h.
  $\Gamma A = \Gamma A$  
  By reflexivity
  $\Gamma, x : A \rightarrow \Gamma, x : A$  
  By rule $\rightarrow Var$

- **Case** $\Gamma \ctx \vdash u : \kappa \notin \text{dom}(\Gamma)$ [VarCtx]
  
  $\Gamma \rightarrow \Gamma$  
  By i.h.
  $\Gamma, u : \kappa \rightarrow \Gamma, u : \kappa$  
  By rule $\rightarrow Uvar$ or $\rightarrow Unsolved$

- **Case** $\Gamma \ctx \vdash \& \notin \text{dom}(\Gamma) \quad \kappa \in \Gamma \quad \Gamma \vdash t : \kappa$ [SolvedCtx]
  
  $\Gamma \rightarrow \Gamma$  
  By i.h.
  $\Gamma t = \Gamma t$  
  By reflexivity
  $\Gamma, \& : \kappa = t \rightarrow \Gamma, \& : \kappa = t$  
  By rule $\rightarrow Solved$

- **Case** $\Gamma \ctx \vdash \alpha : \kappa \in \Gamma \quad (\alpha = -) \notin \Gamma \quad \Gamma \vdash \tau : \kappa$ [EqnVarCtx]
  
  $\Gamma \rightarrow \Gamma$  
  By i.h.
  $\Gamma t = \Gamma t$  
  By reflexivity
  $\Gamma, \alpha = t \rightarrow \Gamma, \alpha = t$  
  By rule $\rightarrow Eqn$

- **Case** $\Gamma \ctx \vdash u \notin \Gamma \quad \Gamma, u \ctx$ [MarkerCtx]
  
  $\Gamma \rightarrow \Gamma$  
  By i.h.
  $\Gamma, u \rightarrow \Gamma, u$  
  By rule $\rightarrow Marker$

Lemma 33 (Extension Transitivity).

If $\mathcal{D} : \Gamma \rightarrow \Theta$ and $\mathcal{D}' : \Theta \rightarrow \Delta$ then $\Gamma \rightarrow \Delta$.

Proof. By induction on $\mathcal{D}'$.

- **Case** $\Theta \rightarrow \Delta$ [Id]
  
  $\Gamma = \cdot$  
  By inversion on $\mathcal{D}$
  $\cdot \rightarrow \cdot$  
  By rule $\rightarrow Id$
  $\Gamma \rightarrow \Delta$  
  Since $\Gamma = \Delta = \cdot$
Proof of Lemma 33 (Extension Transitivity)

- Case \[ \Theta' \rightarrow \Delta' \]
  \[ \Theta', x : A \rightarrow \Delta', x : A' \] \[ \rightarrow \text{Var} \]

  \[ \Gamma = (\Gamma', x : A') \] By inversion on \( D \)
  \[ [\Theta]A'' = [\Theta]A \] By inversion on \( D \)
  \[ \Gamma' \rightarrow \Theta' \] By inversion on \( D \)
  \[ \Gamma' \rightarrow \Delta' \] By i.h.

- Case \[ \Theta' \rightarrow \Delta' \]
  \[ \Theta', \alpha : \kappa \rightarrow \Delta', \alpha : \kappa \] \[ \rightarrow \text{Uvar} \]

  \[ \Gamma = (\Gamma', \alpha : \kappa) \] By inversion on \( D \)
  \[ \Gamma' \rightarrow \Theta' \] By inversion on \( D \)
  \[ \Gamma' \rightarrow \Delta' \] By i.h.
  \[ \Gamma', \alpha : \kappa \rightarrow \Delta', \alpha : \kappa \] By \( \rightarrow \text{Uvar} \)

- Case \[ \Theta' \rightarrow \Delta' \]
  \[ \Theta', \hat{x} : \kappa \rightarrow \Delta', \hat{x} : \kappa \] \[ \rightarrow \text{Unsolved} \]

  Two rules could have concluded \( D : \Gamma \rightarrow (\Theta', \hat{x} : \kappa) \):

  - Case \[ \Gamma' \rightarrow \Theta' \]
    \[ \Gamma', \hat{x} : \kappa \rightarrow \Theta', \hat{x} : \kappa \] \[ \rightarrow \text{Unsolved} \]

      \[ \Gamma' \rightarrow \Delta' \] By i.h.
      \[ \Gamma', \hat{x} : \kappa \rightarrow \Delta', \hat{x} : \kappa \] By rule \( \rightarrow \text{Add} \)

  - Case \[ \Gamma \rightarrow \Theta' \]
    \[ \Gamma \rightarrow \Theta', \hat{x} : \kappa \] \[ \rightarrow \text{Add} \]

      \[ \Gamma \rightarrow \Delta' \] By i.h.
      \[ \Gamma \rightarrow \Delta', \hat{x} : \kappa \] By rule \( \rightarrow \text{Add} \)

- Case \[ \Theta' \rightarrow \Delta' \]
  \[ \Theta', \alpha : \kappa \rightarrow \Delta', \alpha : \kappa \] \[ \rightarrow \text{Solved} \]

  Two rules could have concluded \( D : \Gamma \rightarrow (\Theta', \hat{x} : \kappa = \alpha) \):

  - Case \[ \Gamma' \rightarrow \Theta' \]
    \[ \Gamma' \rightarrow [\Theta']t'' = [\Theta']t \] \[ \rightarrow \text{Solved} \]

      \[ \Gamma' \rightarrow \Delta' \] By i.h.
      \[ [\Theta']t'' = [\Theta']t \] Premise
      \[ [\Delta']t'' = [\Delta']t \] Applying \( \Delta' \) to both sides
      \[ [\Delta']t'' = [\Delta']t \] By Lemma \( \ref{lem:substitution-monotonicity} \) (Substitution Monotonicity)
      \[ \Gamma', \hat{x} : \kappa = t'' \rightarrow \Delta', \hat{x} : \kappa = t \] By \( \rightarrow \text{Solved} \)
Proof of Lemma 33 (Extension Transitivity) \(\text{lem:extension-transitivity}\)

- Case \(\Gamma \rightarrow \Theta'\) 
  \[ \Gamma \rightarrow \Theta', \bar{x} : \kappa = t \]  
  \(\rightarrow\text{AddSolved}\)

\(\Gamma \rightarrow \Delta'\) By i.h.
\(\Gamma' \rightarrow \Delta', \bar{x} : \kappa = t'\) By rule \(\rightarrow\text{AddSolved}\)

- Case \(\Theta' \rightarrow \Delta'\) 
  \[ [\Delta']t = [\Delta']t' \]  
  \(\rightarrow\text{Eqn}\)

\(\Theta', \bar{x} = t \rightarrow \Delta', \bar{x} = t'\)
\(\rightarrow\text{Eqn}\)

\(\Gamma = (\Gamma', \alpha = t'')\) By inversion on \(D\)
\(\Gamma' \rightarrow \Theta'\) By inversion on \(D\)
\([\Theta']t'' = [\Theta']t\) By inversion on \(D\)
\([\Delta']t'' = [\Delta']t]\) Applying \(\Delta'\) to both sides
\(\Gamma' \rightarrow \Delta'\) By i.h.
\([\Delta']t'' = [\Delta']t'\) By Lemma 29 (Substitution Monotonicity)
\(= [\Delta']t'\) By premise \([\Delta']t = [\Delta']t'\)
\(\Gamma', \alpha = t'' \rightarrow \Delta', \alpha = t'\) By rule \(\rightarrow\text{Eqn}\)

- Case \(\Theta \rightarrow \Delta'\) 
  \[ \Theta \rightarrow \Delta', \bar{x} : \kappa \rightarrow \Delta \]  
  \(\rightarrow\text{Add}\)

\(\Gamma \rightarrow \Delta'\) By i.h.
\(\Gamma' \rightarrow \Delta', \bar{x} : \kappa \rightarrow \Delta\) By rule \(\rightarrow\text{Add}\)

- Case \(\Theta \rightarrow \Delta'\) 
  \[ \Theta \rightarrow \Delta', \bar{x} : \kappa = t \rightarrow \Delta \]  
  \(\rightarrow\text{AddSolved}\)

\(\Gamma \rightarrow \Delta'\) By i.h.
\(\Gamma' \rightarrow \Delta', \bar{x} : \kappa = t \rightarrow \Delta\) By rule \(\rightarrow\text{AddSolved}\)

- Case \(\Theta' \rightarrow \Delta'\) 
  \[ \Theta' \rightarrow \Delta', \bar{x} : \kappa \rightarrow \Delta \]  
  \(\rightarrow\text{Marker}\)

\(\Gamma = \Gamma', \bar{x} \rightarrow \Delta', \bar{x} \rightarrow \Delta\)
\(\Gamma' \rightarrow \Theta'\) By inversion on \(D\)
\(\Gamma' \rightarrow \Delta'\) By i.h.
\(\Gamma', \bar{x} \rightarrow \Delta', \bar{x} \rightarrow \Delta\)
\(\rightarrow\text{Uvar}\)

C'.2 Weakening

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

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

Lemma 35 (Suffix Weakening). If \(\Gamma \vdash A \text{ type}\) then \(\Gamma, \Theta \vdash A \text{ type}\).

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.

Proof of Lemma 37 (Extension Weakening (Props)) \(\text{lem:extension-weakening-prop}\)
Lemma 37 (Extension Weakening (Props)). If \( \Gamma \vdash P \) prop and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash P \) prop.

Proof. By inversion on rule \texttt{EqProp} and Lemma 36 (Extension Weakening (Sorts)) twice. \( \square \)

Lemma 38 (Extension Weakening (Types)). If \( \Gamma \vdash \mathsf{A} \) type and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash \mathsf{A} \) type.

Proof. By a straightforward induction on \( \Gamma \vdash \mathsf{A} \) type.

In the \texttt{VarWF} case, use Lemma 22 (Extension Inversion) (i) or (v). In the \texttt{SolvedVarWF} case, use Lemma 22 (Extension Inversion) (iv).

In the other cases, apply the i.h. and/or (for \texttt{ImpliesWF} and \texttt{WithWF} Lemma 37 (Extension Weakening (Props)) to all subderivations, then apply the rule. \( \square \)

C.3 Principal Typing Properties

Lemma 39 (Principal Agreement).

(i) If \( \Gamma \vdash \mathsf{A} \) ! type and \( \Gamma \rightarrow \Delta \) then \( \Delta \vdash \mathsf{A} = [\Gamma] \mathsf{A} \).

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

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

Part (i):

- Case \( \Gamma_0 \rightarrow \Delta_0 \quad [\Delta_0]t = [\Delta_0]t' \)

If \( \alpha \notin \text{FV}(A) \), then:

\[ [\Gamma_0, \alpha = t]A = [\Gamma_0]A \quad \text{By def. of subst.} \]
\[ = [\Delta_0]A \quad \text{By i.h.} \]
\[ = [\Delta_0, \alpha = t']A \quad \text{By def. of subst.} \]

Otherwise, \( \alpha \in \text{FV}(A) \).

\( \Gamma_0 \vdash t \) type \quad \Gamma \text{ is well-formed} \]
\( \Gamma_0 \vdash [\Gamma_0]t \) 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.

\[ \text{FEV}([\Gamma_0]t) \neq \emptyset \quad \text{Assumption (for contradiction)} \]
\[ [\Gamma_0]t = [\Gamma]t \quad \text{By def. of subst.} \]
\[ \text{FEV}([\Gamma]t) \neq \emptyset \quad \text{By above equality} \]
\[ \alpha \in \text{FV}(A) \quad \text{Above} \]
\[ \text{FEV}([\Gamma]A) \neq \emptyset \quad \text{By a property of subst.} \]
\[ \Gamma \vdash \mathsf{A} \rightarrow \text{type} \quad \text{Given} \]
\[ \text{FEV}([\Gamma]A) = \emptyset \quad \text{By inversion} \]
\[ \Rightarrow \Leftarrow \]
\[ \text{FEV}([\Gamma_0]t) = \emptyset \quad \text{By contradiction} \]
\[ \Gamma_0 \vdash t \rightarrow \text{type} \quad \text{By PrincipalWF} \]
\[ [\Gamma_0]t = [\Delta_0]t \quad \text{By i.h.} \]
\[ \Gamma_0 \vdash [\Delta_0]t \text{ type} \quad \text{By above equality} \]
\[ \text{FEV}([\Delta_0]t) = \emptyset \quad \text{By above equality} \]
\[ \Gamma_0 \vdash [\Delta_0]t/\alpha \rightarrow \text{type} \quad \text{By Lemma 8 (Substitution—Well-formedness)} \]
\[ = [\Gamma_0]t/\alpha \text{ } A \quad \text{By i.h. (at } [\Delta_0]t/\alpha \text{ A)} \]
\[ [\Gamma_0, \alpha = t]A = [\Gamma_0]t/\alpha \text{ } A \quad \text{By def. of subst.} \]
\[ = [\Gamma_0]t/\alpha \text{ } A \quad \text{By above equality} \]
\[ = [\Delta_0]t/\alpha \text{ } A \quad \text{By above equality} \]
\[ = [\Delta_0]t'/\alpha \text{ } A \quad \text{By } [\Delta_0]t = [\Delta_0]t' \]
\[ = [\Delta]A \quad \text{By def. of subst.} \]
Proof of Lemma 39 (Principal Agreement)

- Case \[\text{Solved} \rightarrow \text{Solve} \rightarrow \text{Add} \rightarrow \text{Solved}\] Similar to the \[\text{Eqn}\] case.
- Case \[\text{Id} \rightarrow \text{Var} \rightarrow \text{Uvar} \rightarrow \text{Unsolved} \rightarrow \text{Marker}\]

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 \(\Gamma \vdash A p\ type\) then \(\Gamma \vdash [\Gamma] A p\ type\).

Proof. By cases of \(p\):
- Case \(p = !\):
  \[\Gamma \vdash A\ type\] By inversion
  \(\text{FEV}([\Gamma] A) = \emptyset\) By inversion
  \(\Gamma \vdash [\Gamma] A\ type\) By Lemma 13 (Right-Hand Substitution for Typing)
  \(\Gamma \rightarrow \Gamma\) By Lemma 32 (Extension Reflexivity)
  \([\Gamma][\Gamma] A = [\Gamma] A\) By Lemma 29 (Substitution Monotonicity)
  \(\text{FEV}([\Gamma][\Gamma] A) = \emptyset\) By inversion
  \(\Gamma \vdash [\Gamma] A !\ type\) By rule \(\text{PrincipalWF}\)
- Case \(p = \_\):
  \(\Gamma \vdash A\ type\) By inversion
  \(\Delta \vdash A\ type\) By Lemma 13 (Right-Hand Substitution for Typing)
  \(\Delta \vdash A \_\ type\) By rule \(\text{NonPrincipalWF}\)
  \(\Delta \vdash [\Delta] A \_\ type\) By rule \(\text{NonPrincipalWF}\)

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

Proof. By cases of \(p\):
- Case \(p = \_\):
  \(\Gamma \vdash A\ type\) By inversion
  \(\Delta \vdash A\ type\) By Lemma 38 (Extension Weakening (Types))
  \(\Delta \vdash A \_\ type\) By rule \(\text{NonPrincipalWF}\)
- Case \(p = !\):
  \(\Gamma \vdash A\ type\) By inversion
  \(\text{FEV}([\Gamma] A) = \emptyset\) By inversion
  \(\Delta \vdash A\ type\) By Lemma 38 (Extension Weakening (Types))
  \(\Delta \vdash [\Delta] A\ type\) By Lemma 13 (Right-Hand Substitution for Typing)
  \([\Delta] A = [\Gamma] A\) By Lemma 30 (Substitution Invariance)
  \(\text{FEV}([\Delta] A) = \emptyset\) By congruence of equality
  \(\Delta \vdash [\Delta] A !\ type\) By rule \(\text{PrincipalWF}\)

Lemma 42 (Inversion of Principal Typing).

1. If \(\Gamma \vdash (A \rightarrow B) p\ type\) then \(\Gamma \vdash A p\ type\) and \(\Gamma \vdash B p\ type\).
2. If \(\Gamma \vdash (P \supset A) p\ type\) then \(\Gamma \vdash P \ prop\) and \(\Gamma \vdash A p\ type\).
3. If \(\Gamma \vdash (A \land P) p\ type\) then \(\Gamma \vdash P \ prop\) and \(\Gamma \vdash A p\ type\).

Proof. Proof of part 1:
We have \(\Gamma \vdash A \rightarrow B p\ type\).
- Case \(p = \_\):
  \(1 \ \Gamma \vdash A \rightarrow B\ type\) By inversion
  \(\Gamma \vdash A\ type\) By inversion on 1
  \(\Gamma \vdash B\ type\) By inversion on 1
  \(\Gamma \vdash A \_\ type\) By rule \(\text{NonPrincipalWF}\)
  \(\Gamma \vdash B \_\ type\) By rule \(\text{NonPrincipalWF}\)
Proof of Lemma 42 (Inversion of Principal Typing).

**Lemma 42** (Inversion of Principal Typing).

If \( \Gamma \vdash A \rightarrow B \) then \( \Gamma \vdash \alpha : \kappa \rightarrow \Delta \). 

**Proof.** By induction on the given derivation.

- **Case** \( p = 1! \):
  \[ \Gamma \vdash A \rightarrow B \text{ type} \]
  \[ \emptyset = \text{FEV}(\Gamma(A \rightarrow B)) \]
  \[ = \text{FEV}(\Gamma A \rightarrow \Gamma B) \]
  \[ = \text{FEV}(\Gamma A \cup \text{FEV}(\Gamma B)) \]
  \[ \text{FEV}(\Gamma A) = \emptyset \]
  By properties of empty sets and unions
  \[ \Gamma \vdash A \text{ type} \]
  By inversion on \( 1! \)
  \[ \Gamma \vdash B \text{ type} \]
  By inversion on \( 1 \)
  \[ \Gamma \vdash A \text{ type} \]
  By rule \texttt{PrincipalWF}
  \[ \Gamma \vdash B \text{ type} \]
  By rule \texttt{PrincipalWF}

Part 2: We have \( \Gamma \vdash B \supset A \ A \ p \text{ type} \). Similar to Part 1.

Part 3: We have \( \Gamma \vdash A \wedge P \ p \text{ type} \). Similar to Part 2.

\[ \square \]

**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** \( \Gamma_L \vdash \tau : \kappa \)
  \[ \Gamma_L, \alpha : \kappa, \Gamma_R \vdash \alpha := \tau : \kappa \rightarrow \theta \]
  \[ \Gamma \]
  Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case** \( \hat{\beta} \in \text{unsolved}(\Gamma_0[\alpha : \kappa][\hat{\beta} : \kappa]) \)
  \[ \Gamma_0[\alpha : \kappa][\hat{\beta} : \kappa] \vdash \alpha := \hat{\beta} : \kappa \rightarrow \theta \]
  \[ \Gamma \]
  Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case** \( \Gamma_0[\alpha_2 : \times, \alpha_1 : \times, \alpha : \star = \alpha_1 \oplus \alpha_2] \vdash \alpha_1 := \tau_1 : \star \rightarrow \Theta \)
  \[ \Theta \vdash \alpha_2 := [\Theta]\tau_2 : \times \rightarrow \Delta \]
  \[ \Gamma_0[\alpha : \times] \vdash \alpha := \tau_1 \oplus \tau_2 : \star \rightarrow \Delta \]
  \[ \Gamma_0[\alpha_2 : \times, \alpha_1 : \times, \alpha : \star = \alpha_1 \oplus \alpha_2] \vdash \alpha_1 := \tau_1 : \star \rightarrow \Theta \]
  \[ \Theta \vdash \alpha_2 := [\Theta]\tau_2 : \times \rightarrow \Delta \]
  \[ \Theta \rightarrow \Delta \]
  \[ \Theta \rightarrow \Delta \]
  Subderivation
  By i.h.
  Subderivation
  By i.h.
  By Lemma 33 (Extension Transitivity)
  By Lemma 23 (Deep Evar Transitivity)
  (parts (i), (i), and (ii),
  using Lemma 33 (Extension Transitivity))
  \[ \Gamma_0[\alpha : \times] \rightarrow \Delta \]
  By Lemma 33 (Extension Transitivity)

- **Case** \( \Gamma_0[\alpha : N] \vdash \alpha := \text{zero} : N \rightarrow \Gamma_0[\alpha : N = \text{zero}] \)
  Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case** \( \Gamma[\alpha_1 : N, \alpha : N = \text{succ}(\alpha_1)] \vdash \alpha_1 := t_1 : N \rightarrow \Delta \)
  \[ \Gamma[\alpha : N] \vdash \alpha := \text{succ}(t_1) : N \rightarrow \Delta \]
  By reasoning similar to the \texttt{InstBin} case.
  \[ \square \]
C’.5 Equivalence Extends

Lemma 44 (Elimeq Extension).
If \( \Gamma / s \vdash t : \kappa \vdash \Delta \) then there exists \( \Theta \) such that \( \Gamma, \Theta \rightarrow \Delta \).

Proof. By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context \( \Delta \).

- Case \( \Gamma / \alpha \vdash \alpha : \kappa \vdash \Gamma \)
  Since \( \Delta = \Gamma \), applying Lemma 32 (Extension Reflexivity) suffices (let \( \Theta = \cdot \)).

- Case \( \Gamma / \text{zero} \vdash \text{zero} : \mathbb{N} \vdash \Gamma \)
  Similar to the ElimeqUvarRefl case.

- Case \( \Gamma / \sigma \vdash t : \mathbb{N} \vdash \Delta \)
  \( \Gamma / \text{succ}(\sigma) \vdash \text{succ}(t) : \mathbb{N} \vdash \Delta \)
  Follows by i.h.

- Case \( \Gamma[\alpha : \kappa] \vdash \alpha := t : \kappa \vdash \Delta \)
  Subderivation
  \( \Gamma \rightarrow \Delta \)
  By Lemma 43 (Instantiation Extension)
  Let \( \Theta = \cdot \).

- Case \( \alpha \notin \text{FV}(\Gamma t) \) (\( \alpha = - \) \( \notin \Gamma \))
  \( \Gamma / \alpha \vdash \alpha : \kappa \vdash \Gamma, \alpha = t \)
  ElimeqUvarL
  Let \( \Theta \) be \( (\alpha = t) \).

- Cases ElimeqInstR, ElimeqUvarR
  Similar to the respective L cases.

- Case \( \sigma \neq t \)
  \( \Gamma / \sigma \vdash t : \kappa \vdash \perp \)
  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 \( \Delta \).

- Case \( \Gamma / \sigma \vdash t : \mathbb{N} \vdash \Delta \)
  \( \Gamma / \sigma \vdash \sigma = t \vdash \Delta \)
  ElimeqClash
  The statement says that the output is a (consistent) context \( \Delta \), so this case is impossible.
Lemma 46 (Checkeq Extension).

If \( \Gamma \vdash A \equiv B \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).

Proof. By induction on the given derivation.

- **Case**
  \( \Gamma \vdash u \equiv u : \kappa \vdash \Gamma \) \( \text{CheckeqVar} \)

  Since \( \Delta = \Gamma \), applying Lemma 32 (Extension Reflexivity) suffices.

- **Cases** CheckeqUnit, CheckeqZero

- **Case**
  \( \Gamma \vdash \tau_1 \equiv \tau'_1 : \tau : \Theta \vdash (\Theta)\tau_2 \equiv (\Theta)\tau'_2 : \tau : \Delta \) \( \text{CheckeqBin} \)

  \( \Gamma \rightarrow \Theta \) By i.h.
  \( \Theta \rightarrow \Delta \) By i.h.

  \( \vDash \Gamma \rightarrow \Delta \) By Lemma 33 (Extension Transitivity)

- **Case**
  \( \Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta \) \( \text{CheckeqInstL} \)

  \( \Gamma \vdash \sigma = t \equiv t : t \vdash \Delta \) \( \text{CheckeqInstR} \)

  Similar to the CheckeqInstL case.

Lemma 47 (Checkprop Extension).

If \( \Gamma \vdash P \equiv \text{true} \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).

Proof. By induction on the given derivation.

- **Case**
  \( \Gamma \vdash \sigma \equiv t : \kappa \vdash \Delta \) \( \text{CheckpropEq} \)

  \( \Gamma \vdash \sigma \equiv \text{true} : \kappa \vdash \Delta \) Subderivation

  \( \Gamma \rightarrow \Delta \) By Lemma 46 (Checkeq Extension)

Lemma 48 (Prop Equivalence Extension).

If \( \Gamma \vdash P \equiv Q \vdash \Delta \) then \( \Gamma \rightarrow \Delta \).

Proof. By induction on the given derivation.

- **Case**
  \( \Gamma \vdash \sigma_1 \equiv \tau_1 : N : \Theta \vdash (\Theta)\sigma_2 \equiv (\Theta)\tau_2 : N : \Delta \) \( \equiv\text{PropEq} \)

  \( \Gamma \vdash (\Theta)\sigma_1 \equiv (\Theta)\sigma_2 : N : \Delta \) Subderivation

  \( \Gamma \rightarrow \Theta \) By Lemma 46 (Checkeq Extension)

  \( \Theta \rightarrow \Delta \) By Lemma 46 (Checkeq Extension)

  \( \vDash \Gamma \rightarrow \Delta \) By Lemma 33 (Extension Transitivity)
Lemma 49 (Equivalence Extension).
If $\Gamma \vdash A \equiv B \vdash \Delta$ then $\Gamma \vdash \Delta$.

Proof. By induction on the given derivation.

- Case $\Gamma \vdash \alpha \equiv \alpha \vdash \Gamma$
  Here $\Delta = \Gamma$, so Lemma 32 (Extension Reflexivity) suffices.

- Case $\Gamma \vdash \& \equiv \& \vdash \Gamma$
  Similar to the $\equiv \text{Var}$ case.

- Case $\Gamma \vdash 1 \equiv 1 \vdash \Gamma$
  Similar to the $\equiv \text{Var}$ case.

- Case
  $\Gamma \vdash A_1 \equiv B_1 \vdash \Theta \implies [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta$
  $\Gamma \vdash A_1 \equiv B_1 \vdash \Theta \implies [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta$
  Subderivation
  $\Gamma \implies \Theta$
  By i.h.
  $\Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \vdash \Delta$
  Subderivation
  $\Theta \implies \Delta$
  By i.h.
  $\Gamma \implies \Delta$
  By Lemma 33 (Extension Transitivity)

- Cases $\equiv \cup$, $\equiv \cap$
  Similar to the $\equiv \nabla$ case, but with Lemma 48 (Prop Equivalence Extension) on the first premise.

- Case
  $\Gamma, \alpha : \kappa \vdash A_0 \equiv B \vdash \Delta, \alpha : \kappa, \Delta'$
  $\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv B \vdash \Delta$
  $\equiv \forall$
  $\Gamma, \alpha : \kappa \vdash A_0 \equiv B \vdash \Delta, \alpha : \kappa, \Delta'$
  Subderivation
  $\Gamma, \alpha : \kappa \implies \Delta$
  By i.h.
  $\Gamma \implies \Delta$
  By Lemma 22 (Extension Inversion) (i)

- Case
  $\Gamma_0[\check{\alpha}] \vdash \check{\alpha} : \tau \equiv \Delta$
  $\& \notin \text{FV}([\Gamma_0[\check{\alpha}]]\tau)$
  $\equiv \text{InstantiateL}$
  $\Gamma_0[\check{\alpha}] \vdash \check{\alpha} : \tau \equiv \Delta$
  $\equiv \text{InstantiateL}$
  $\Gamma_0[\check{\alpha}] \rightarrow \Delta$
  By Lemma 43 (Instantiation Extension)

- Case $\equiv \text{InstantiateR}$
  Similar to the $\equiv \text{InstantiateL}$ case.

C.6 Subtyping Extends

Lemma 50 (Subtyping Extension). If $\Gamma \vdash A : T \vdash \Delta$ then $\Gamma \vdash \Delta$.

Proof. By induction on the given derivation.

- Case
  $\Gamma, \alpha : \kappa \vdash A : T \vdash \Delta, \alpha : \kappa, \Theta$
  $\Gamma \vdash \forall \alpha : \kappa. A \vdash \Delta$
  $\equiv \forall$
  $\Gamma, \alpha : \kappa \vdash A : T \vdash \Delta, \alpha : \kappa, \Theta$
  Subderivation
  $\Gamma, \alpha : \kappa \implies \Delta, \alpha : \kappa, \Theta$
  By i.h. (i)
  $\Gamma \implies \Delta$
  By Lemma 22 (Extension Inversion) (ii)
Proof of Lemma 50 (Subtyping Extension) lem:subtyping-extension

- Case $\langle < : R \rangle$ Similar to the $\langle < : \forall L \rangle$ case.
- Case $\Gamma, \alpha : \kappa \vdash A < : B \vdash \Delta, \alpha, \kappa, \Theta$
  $\Gamma \vdash A < : \forall \alpha : \kappa . B \vdash \Delta$ $\langle < : \forall R \rangle$
  Similar to the $\langle < : \forall L \rangle$ case, but using part (i) of Lemma 22 (Extension Inversion).
- Case $\langle < : \exists L \rangle$ Similar to the $\langle < : \forall R \rangle$ case.

$\Gamma \vdash A \equiv B \vdash \Delta$ Subderivation
  $\Gamma \rightarrow \Delta$ By Lemma 49 (Equivalence Extension)

C.7 Typing Extends

Lemma 51 (Typing Extension).
If $\Gamma \vdash e \leftarrow A p \vdash \Delta$

- Case $\forall I$ Use the i.h.
- Case $\forall Spine$ Use the i.h. and Lemma 22 (Extension Inversion) (v).
- Case $\forall !$ Immediate by Lemma 32 (Extension Reflexivity).
- Case $\forall$ Use the i.h.
- Case $\forall$ Use the i.h. and Lemma 22 (Extension Inversion) (v).
- Case $\forall$ Use the i.h.

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).
- MatchNeg Use the i.h.
- MatchA Use the i.h.
- MatchUnify
  $\Gamma, \Theta' \rightarrow \Theta$ By Lemma 44 (Elimeq Extension)
  $\Theta \rightarrow \Delta, \Theta'$ By i.h.
  $\Gamma, \Theta' \rightarrow \Delta, \Theta'$ By Lemma 33 (Extension Transitivity)
  $\Gamma \rightarrow \Delta$ By Lemma 22 (Extension Inversion) (ii)

- Synthesis, checking, and spine judgments: In rules Var, [], EmptySpine and $\exists !$ the output context $\Delta$ is exactly $\Gamma$, so the result follows by Lemma 32 (Extension Reflexivity).
  - Case $\forall !$ Use the i.h. and Lemma 33 (Extension Transitivity).
  - Case $\forall !$ Use Lemma 47 (Checkprop Extension), the i.h., and Lemma 33 (Extension Transitivity).
  - Case $\forall !$ Use the i.h.
    - Case $\forall !$ Use Lemma 45 (Elimprop Extension)
      $\Theta \rightarrow \Delta, \Theta'$ By i.h.
      $\Gamma, \Theta' \rightarrow \Delta, \Theta'$ By Lemma 33 (Extension Transitivity)
    - Case $\forall !$ Use the i.h.

Proof of **Lemma 54** *(Typing Extension)*

If \( \Gamma \rightarrow \Omega \), then there is a \( \Psi \) such that \([\Omega, \vartriangleright, \Omega Z](\Delta, \vartriangleright, \Theta) = [\Psi]\).

**Proof.** By induction on the derivation of \( \Gamma \rightarrow \Omega \).

- **Case** \( \rightarrow \text{Id} \): Impossible: \( \Delta, \vartriangleright, \Theta \) cannot have the form \( \cdot \).
- **Case** \( \rightarrow \text{Var} \): We have \( \Omega Z = (\Omega Z, x : A) \) and \( \Theta = (\Theta', x : A') \). By i.h., there is \( \Psi' \) such that \([\Omega, \vartriangleright, \Omega Z]|(\Delta, \vartriangleright, \Theta') = [\Psi'] \). Then by the definition of context application, \([\Omega, \vartriangleright, \Omega Z, x : A]|(\Delta, \vartriangleright, \Theta', x : A') = [\Psi']A \). Let \( \Psi = (\Psi', x : [\Psi']A) \).
- **Case** \( \rightarrow \text{Uvar} \): Similar to the \( \rightarrow \text{Var} \) case, with \( \Psi = (\Psi', \alpha : k) \).
- **Cases** \( \rightarrow \text{Eqn} \): Use the i.h. on the synthesis premise and the match premise, and then Lemma 33 (Extension Transitivity).

**Lemma 52** *(Context Partitioning)*.

If \( \Delta, \vartriangleright, \Theta \rightarrow \Omega, \vartriangleright, \Omega Z \) then there is a \( \Psi \) such that \([\Omega, \vartriangleright, \Omega Z](\Delta, \vartriangleright, \Theta) = [\Omega]\).

**Proof.** By induction on the given derivation.

- **Case** \( \rightarrow \text{Id} \): Impossible: \( \Delta, \vartriangleright, \Theta \) cannot have the form \( . \).
- **Case** \( \rightarrow \text{Var} \): We have \( \Omega Z = (\Omega Z, x : A) \) and \( \Theta = (\Theta', x : A') \). By i.h., there is \( \Psi' \) such that \([\Omega, \vartriangleright, \Omega Z]|(\Delta, \vartriangleright, \Theta') = [\Psi'] \). Then by the definition of context application, \([\Omega, \vartriangleright, \Omega Z, x : A]|(\Delta, \vartriangleright, \Theta', x : A') = [\Psi']A \). Let \( \Psi = (\Psi', x : [\Psi']A) \).
- **Case** \( \rightarrow \text{Uvar} \): Similar to the \( \rightarrow \text{Var} \) case, with \( \Psi = (\Psi', \alpha : k) \).
- **Cases** \( \rightarrow \text{Eqn} \): Use the i.h on the synthesis premise and the match premise, and then Lemma 33 (Extension Transitivity).

**Lemma 54** *(Completing Stability)*.

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

**Proof.** By induction on the derivation of \( \Gamma \rightarrow \Omega \).

- **Case** \( \rightarrow \text{Id} \): Immediate.
- **Case** \( \rightarrow \text{Var} \):

  \[
  \frac{\Gamma_0 \rightarrow \Omega_0}{\Gamma_0, x : A \rightarrow \Omega_0, x : A'}
  \]

  \( \rightarrow \text{Var} \)

  Subderivation

  \( \Gamma_0 \rightarrow \Omega_0 \)

  By i.h.

  \( [\Omega_0]\Gamma_0 = [\Omega_0]\Omega_0 \)

  Subderivation

  \( [\Omega_0]\Gamma_0 = [\Omega_0]\Omega_0 \)

  By congruence of equality

  \( [\Omega_0, x : A']|\Gamma_0, x : A) = [\Omega_0, x : A']|\Omega_0, x : A' \)

  By definition of substitution

  \( [\Omega_0, x : A']|\Gamma_0, x : A) = [\Omega_0, x : A']|\Omega_0, x : A' \)

  Similar to \( \rightarrow \text{Var} \)

- **Case** \( \rightarrow \text{Uvar} \):

  \[
  \frac{\Gamma_0 \rightarrow \Omega_0}{\Gamma_0, \alpha : k \rightarrow \Omega_0, \alpha : k}
  \]

  Similar to \( \rightarrow \text{Var} \)

  \( \rightarrow \text{Uvar} \)

  \( \rightarrow \text{Unsolved} \)

  \( \rightarrow \text{Solved} \)

  \( \rightarrow \text{Solve} \)

  \( \rightarrow \text{Add} \)

  \( \rightarrow \text{AddSolved} \)

  \( \rightarrow \text{Marker} \)

  Broadly similar to the \( \rightarrow \text{Uvar} \) case, but the rightmost context element disappears in context application, so we let \( \Psi = \Psi' \).

  \( \rightarrow \text{Marker} \)
Proof of Lemma 54 (Completing Stability).

(iii) If $\Omega \rightarrow \Omega'$, then $[\Omega]t = [\Omega']t'$.

Proof.

• **Case** $\Gamma_0 \rightarrow \Omega_0 \quad [\Omega_0]t = [\Omega_0]t'$
  $\Gamma_0, \hat{\alpha} : \kappa \rightarrow \Omega_0, \hat{\alpha} : \kappa \rightarrow t'$  
  Solved

  Similar to $\rightarrow \Var$

• **Case** $\Gamma_0 \rightarrow \Omega_0$
  $\Gamma_0, \hat{\alpha} : \kappa \rightarrow \Omega_0, \hat{\alpha} : \kappa$  
  Marker

  Similar to $\rightarrow \Var$

• **Case** $\Gamma_0 \rightarrow \Omega_0$
  $\Gamma_0, \hat{\beta} : \kappa' \rightarrow \Omega_0, \hat{\beta} : \kappa' = t'$  
  Solve

  Similar to $\rightarrow \Var$

• **Case** $\Gamma_0 \rightarrow \Omega_0$
  $[\Omega_0]t' = [\Omega_0]t$
  $\Gamma_0 \rightarrow \Omega_0$
  $\Rightarrow$ Subderivation

  By definition of context substitution

  $\Gamma_0 \rightarrow \Omega_0$
  $[\Omega_0]t = [\Omega_0]t'$
  $\Rightarrow$ Subderivation

  By congruence of equality

  $\Gamma_0 \rightarrow \Omega_0$
  $[\Omega_0]t/\alpha | (\Omega_0)_{\Gamma_0} = [\Omega_0]t/\alpha | (\Omega_0)_{\Omega_0}$
  $\Rightarrow$ By i.h.

  $\Omega_0, \hat{\alpha} : \kappa \rightarrow t'((\Gamma_0, \hat{\alpha} = t') = [\Omega_0, \hat{\alpha} = t']((\Omega_0, \hat{\alpha} = t)$
  $\Rightarrow$ By definition of context substitution

• **Case** $\Gamma \rightarrow \Omega_0$
  $\Gamma \rightarrow \Omega_0, \hat{\alpha} : \kappa$  
  Add

  Subderivation

  By definition of context substitution

  $\Gamma \rightarrow \Omega_0$
  $[\Omega_0]t = [\Omega_0]t'$
  $\Rightarrow$ Subderivation

  By i.h.

  $\Omega_0, \hat{\alpha} : \kappa | (\Gamma, \hat{\alpha} = \kappa) | (\Omega_0, \hat{\alpha} : \kappa)$
  $\Rightarrow$ By definition of context substitution

• **Case** $\Gamma \rightarrow \Omega_0$
  $\Gamma \rightarrow \Omega_0, \hat{\alpha} : \kappa$  
  AddSolved

  Similar to the $\rightarrow \Add$ case.

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) (i), $[\Omega']t = [\Omega']| [\Omega]t$.

  Now we need to show $[\Omega']| [\Omega]t = [\Omega]t$. Considered as a substitution, $\Omega'$ is the identity everywhere except existential variables $\hat{\alpha}$ and universal variables $\alpha$. First, since $\Omega$ is complete, $[\Omega]t$ has no free existentials. Second, universal variables free in $[\Omega]t$ have no equations in $\Omega$ (if they had, their occurrences would have been replaced). But if $\Omega$ has no equation for $\alpha$, it follows from $\Omega \rightarrow \Omega'$ and the definition of context extension in Figure 13 that $\Omega'$ also lacks an equation, so applying $\Omega'$ also leaves $\alpha$ alone.

  Transitivity of equality gives $[\Omega']t = [\Omega]t$.

• **Part (ii):** Similar to part (i), using Lemma 29 (Substitution Monotonicity) (iii) instead of (i).
• **Part (iii):** By induction on the given derivation of $\Omega \rightarrow \Omega'$.

Only cases $\rightarrow \text{id}$, $\rightarrow \text{Var}$, $\rightarrow \text{Uvar}$, $\rightarrow \text{Eqn}$, $\rightarrow \text{Solved}$, $\rightarrow \text{AddSolved}$, and $\rightarrow \text{Marker}$ are possible. In all of these cases, we use the i.h. and the definition of context application; in cases $\rightarrow \text{Var}$, $\rightarrow \text{Eqn}$, and $\rightarrow \text{Solved}$, we also use the equality in the premise of the respective rule.

**Lemma 56** (Confluence of Completeness).
If $\Delta_1 \rightarrow \Omega$ and $\Delta_2 \rightarrow \Omega$ then $[\Omega]\Delta_1 = [\Omega]\Delta_2$.

**Proof.**

$\Delta_1 \rightarrow \Omega$ Given

$[\Omega]\Delta_1 = [\Omega]\Omega$ By Lemma 54 (Completing Stability)

$\Delta_2 \rightarrow \Omega$ Given

$[\Omega]\Delta_2 = [\Omega]\Omega$ By Lemma 54 (Completing Stability)

$[\Omega]\Delta_1 = [\Omega]\Delta_2$ By transitivity of equality

**Lemma 57** (Multiple Confluence).
If $\Delta \rightarrow \Omega$ and $\Omega \rightarrow \Omega'$ and $\Delta' \rightarrow \Omega'$ then $[\Omega]\Delta = [\Omega']\Delta'$.

**Proof.**

$\Delta \rightarrow \Omega$ Given

$[\Omega]\Delta = [\Omega]\Omega$ By Lemma 54 (Completing Stability)

$\Omega \rightarrow \Omega'$ Given

$[\Omega]\Omega = [\Omega']\Omega'$ By Lemma 55 (Completing Completeness) (iii)

$= [\Omega']\Delta'$ By Lemma 54 (Completing Stability) ($\Delta' \rightarrow \Omega'$ given)

**Lemma 59** (Canonical Completion).

If $\Gamma \rightarrow \Omega$

then there exists $\Omega_{\text{canon}}$ such that $\Gamma \rightarrow \Omega_{\text{canon}}$ and $\Omega_{\text{canon}} \rightarrow \Omega$ and $\text{dom}(\Omega_{\text{canon}}) = \text{dom}(\Gamma)$ and, for all $\hat{\alpha} : \kappa = \tau$ 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 $\hat{\alpha} \in \text{unsolved}(\Delta)$

then there exists $\Omega_1 = \Omega_1'[\hat{\alpha} : \kappa = t_1]$ such that $\Omega_1 \rightarrow \Omega$ and $\Omega_2 = \Omega_2'[\hat{\alpha} : \kappa = t_2]$ 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 $\Omega_{\text{canon}} = \Omega_1'[\hat{\alpha} : \kappa = t_1]$, where $\text{FEV}(t_1) = \emptyset$. Therefore $\nexists \Omega_1'[\hat{\alpha} : \kappa = t_1] \rightarrow \Omega$.

Now choose $t_2$ as follows:

• If $\kappa = \ast$, let $t_2 = t_1$.
• If $\kappa = N$, let $t_2 = \text{succ}(t_1)$.

Thus, $\nexists \Omega_2 = \Omega_2'[\hat{\alpha} : \kappa = t_2]$.

$\nexists \Delta \rightarrow \Omega_2$ By Lemma 31 (Split Extension)

**D’ Internal Properties of the Declarative System**

**Lemma 61** (Interpolating With and Exists).

1. If $D : \Psi \vdash \Pi : \bar{A} \leftrightarrow C \text{ p and } \Psi \vdash P_0 \text{ true then } D' : \Psi \vdash \Pi : \bar{A} \leftrightarrow C \land P_0 \text{ p.}$

2. If $D : \Psi \vdash \Pi : \bar{A} \leftrightarrow [\tau/\alpha]C_0 \text{ p and } \Psi \vdash \tau : \kappa$ then $D' : \Psi \vdash \Pi : \bar{A} \leftrightarrow (\exists \alpha : \kappa. C_0) \text{ p.}$

---

Proof of **Lemma 61** (Interpolating With and Exists) lem:match-with-exists
Proof of Lemma 61 (Interpolating With and Exists)

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

Proof. By induction on the given match derivation.

In the $\text{DeclMatchBase}$ case, for part (1), apply rule $\land I$. For part (2), apply rule $\exists I$.

In the $\text{DeclMatchNeg}$ case, part (1), use Lemma 2 $\text{(Declarative Weakening)}$ (iii). In part (2), use Lemma 2 $\text{(Declarative Weakening)}$ (i).

Lemma 62 (Case Invertibility).
If $\Psi \vdash \text{case}(e_0, \Pi) :: \vec{A} \leftarrow C \ p$ then $\Psi \vdash e_0 \Rightarrow A !$ and $\Psi \vdash \Pi :: A \leftarrow C \ p$ and $\Psi \vdash \Pi$ covers $A$ where the height of each resulting derivation is strictly less than the height of the given derivation.

Proof. By induction on the given derivation.

- Case $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A q \leftarrow C_0 \ p$
  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C_0 \land \ p$ $\text{DeclSub}$
  Impossible, because $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A q$ is not derivable.

- Cases $\text{Decl\forall}, \text{Decl\exists}$ Impossible: these rules have a value restriction, but a case expression is not a value.

- Case $\Psi \vdash P \ true$
  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C_0 \ p$ $\text{Decl\land}$
  $< n - 1 \ \Psi \vdash e_0 \Rightarrow A ! \quad \text{By i.h.}$
  $< n - 1 \ \Psi \vdash \Pi :: A \leftarrow C_0 \ p \quad "$
  $\exists < n \ \Psi \vdash \Pi$ covers $A$
  $\leq n - 1 \ \Psi \vdash P \ true \quad \text{Subderivation}$
  $\leq n \ \Psi \vdash \Pi :: A \leftarrow C_0 \land \ p \ p$ By Lemma 61 $\text{(Interpolating With and Exists)}$ (1)

- Cases $\text{Decl\land}, \text{Decl\rightarrow}, \text{Decl\rightarrow\rightarrow}, \text{Decl\land\land}$ Impossible, because in these rules $e$ cannot have the form $\text{case}(e_0, \Pi)$.

- Case $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A !$
  $\Psi \vdash \Pi :: A \leftarrow C \ p$
  $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C \ p$ $\text{DeclCase}$
  Immediate.

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 $\Gamma \vdash A \ q \ type$
then $\Delta \vdash C \ p \ type$.

(Synthesis) If $\Gamma \vdash e \Rightarrow A p \vdash \Delta$
then $A \vdash p \ type$.

Proof. By induction on the given derivation.

- Case $\text{Anno}$ Use Lemma 51 $\text{(Typing Extension)}$ and Lemma 41 $\text{(Extension Weakening for Principal Typing)}$.

- Case $\text{\forall Spine}$ We have $\Gamma \vdash (\forall \alpha : \kappa. A_0) \ q \ type$.
  By inversion, $\Gamma, \alpha : \kappa \vdash A_0 \ q \ type$.
  By properties of substitution, $\Gamma, \alpha : \kappa \vdash [\alpha/a]A_0 \ q \ type$.
  Now apply the i.h.
Proof of Lemma 63 (Well-Formed Outputs of Typing)

Lemma 64 (Left Unsolvedness Preservation).
If \( \Gamma_0, \hat{\alpha}, \Gamma_1 \vdash \hat{\alpha} := \Lambda : \kappa \vdash A : \Delta \) and \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \) then \( \hat{\beta} \in \text{unsolved}(\Delta) \).

Proof. By induction on the given derivation.

- **Case Spine**: Use Lemma 42 (Inversion of Principal Typing), (2), Lemma 47 (Checkprop Extension), and Lemma 41 (Extension Weakening for Principal Typing).

- **Case SpineRecover**: By i.h., \( \Delta \vdash C \not\vdash \text{type} \).
We have as premise \( \text{FEV}(C) = \emptyset \).
Therefore \( \Delta \vdash C \vdash \text{type} \).

- **Case SpinePass**: Immediate.

- **Case EmptySpine**: Immediate.

- **Case \( \Rightarrow \text{Spine} \)**: Use Lemma 42 (Inversion of Principal Typing) (1), Lemma 51 (Typing Extension), and Lemma 41 (Extension Weakening for Principal Typing).

- **Case \( \alpha \Rightarrow \text{Spine} \)**: Show that \( \hat{\alpha}_1 \rightarrow \hat{\alpha}_2 \) is well-formed, then use the i.h. \( \square \)

\( F' \) Decidability of Instantiation

Lemma 64 (Left Unsolvedness Preservation).

If \( \Gamma_0, \hat{\alpha}, \Gamma_1 \vdash \hat{\alpha} := \Lambda : \kappa \vdash A : \Delta \) and \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \) then \( \hat{\beta} \in \text{unsolved}(\Delta) \).

Proof. By induction on the given derivation.

- **Case**

  \[
  \frac{\Gamma_0 \vdash \tau : \kappa}{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \vdash \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1}{\text{InstSolve}}
  \]

  Immediate, since to the left of \( \hat{\alpha} \), the contexts \( \Delta \) and \( \Gamma \) are the same.

- **Case**

  \[
  \frac{\hat{\beta} \in \text{unsolved}(\Gamma' [\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\Gamma' [\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} := \hat{\beta} : \kappa \vdash \Gamma' [\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} = \hat{\alpha}}{\text{InstReach}}
  \]

  Immediate, since to the left of \( \hat{\alpha} \), the contexts \( \Delta \) and \( \Gamma \) are the same.

- **Case**

  \[
  \frac{\Gamma_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \vdash \hat{\alpha}_1 := \tau_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := |\Theta| \tau_2 : \star \vdash \Delta}{\Gamma_0, \hat{\alpha} : \star, \Gamma_1 \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \vdash \Delta}{\text{InstBin}}
  \]

  We have \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \). Therefore \( \hat{\beta} \in \text{unsolved}(\Gamma_0, \hat{\alpha}_2 : \star) \).

  Clearly, \( \hat{\alpha}_2 \in \text{unsolved}(\Gamma_0, \hat{\alpha}_2 : \star) \).

  We have two subderivation:

  \[
  \frac{\Gamma_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \vdash \hat{\alpha}_1 := A_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := |\Theta| A_2 : \star \vdash \Delta}{\text{InstZero}}
  \]

  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 \( \Gamma_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} := \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \).

  Hence by Lemma 20 (Declaration Order Preservation), \( \hat{\beta} \) is declared to the left of \( \hat{\alpha}_2 \) in \( \Theta \). That is, \( \Theta = (\Theta_0, \hat{\alpha}_2 : \star, \Theta_1) \), where \( \hat{\beta} \in \text{unsolved}(\Theta_0) \).

  By induction on (2), \( \hat{\beta} \in \text{unsolved}(\Delta) \).

- **Case**

  \[
  \frac{\Gamma' [\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \text{zero} : \kappa \vdash \Gamma' [\hat{\alpha} : \kappa = \text{zero}]}{\text{InstZero}}
  \]

  Immediate, since to the left of \( \hat{\alpha} \), the contexts \( \Delta \) and \( \Gamma \) are the same.
Proof of Lemma 64 (Left Unsolvedness Preservation)

\[
\begin{align*}
\Gamma \vdash \alpha \colon \mathbb{N}, \alpha \colon \mathbb{N} = \text{succ}(\alpha_1) & \vdash \alpha_1 \colon \mathbb{N} \vdash \Delta \\
\Gamma \vdash \alpha \colon \mathbb{N} \vdash \alpha_1 \colon \mathbb{N} = \text{succ}(\alpha_1) \vdash \Delta
\end{align*}
\]

We have \( \hat{\beta} \in \text{unsolved}(\Gamma_0) \). Therefore \( \hat{\beta} \in \text{unsolved}(\Gamma_0, \alpha_1 : \mathbb{N}) \). By i.h., \( \hat{\beta} \in \text{unsolved}(\Delta) \).

\[\square\]

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

Proof. By induction on the given instantiation derivation.

\[\begin{align*}
\text{Case} \\
\Gamma_0 \vdash \tau : \kappa \\
\Gamma_0, \alpha : \kappa, \Gamma_1 \vdash \alpha : \kappa \vdash \tau = \kappa \vdash \Delta \\
\Gamma \vdash \alpha \colon \kappa \vdash \tau = \kappa \vdash \Delta
\end{align*}\]

We have \( \hat{\beta} \in \text{FV}(\Gamma_0) \). Since \( \Delta \) differs from \( \Gamma \) only in \( \alpha \), it must be the case that \( \Gamma \sigma = \Delta \sigma \). It is given that \( \hat{\beta} \not\in \text{FV}(\Gamma) \), so \( \hat{\beta} \not\in \text{FV}(\Delta) \).

\[\begin{align*}
\text{Case} \\
\hat{\gamma} \in \text{unsolved}(\Gamma_0) \vdash \hat{\gamma} : \kappa \\
\Gamma \vdash \hat{\gamma} : \kappa \vdash \Delta
\end{align*}\]

Since \( \Delta \) differs from \( \Gamma \) only in solving \( \hat{\gamma} \) to \( \alpha \), applying \( \Delta \) to a type will not introduce a \( \hat{\beta} \). We have \( \hat{\beta} \not\in \text{FV}(\Gamma_0) \), so \( \hat{\beta} \not\in \text{FV}(\Delta) \).

\[\begin{align*}
\text{Case} \\
\Gamma' \vdash \tau : \kappa \\
\Gamma' \vdash \tau = \kappa \vdash \Delta \\
\Gamma \vdash \alpha \colon \kappa \vdash \tau \vdash \Delta
\end{align*}\]

We have \( \hat{\gamma} \in \text{FV}(\Gamma_0) \) and \( \hat{\beta} \not\in \text{FV}(\Gamma) \). By weakening, we get \( \Gamma' \vdash \tau : \kappa' \); since \( \alpha \not\in \text{FV}(\Gamma) \) and \( \alpha \not\in \text{FV}(\Gamma) \), it follows that \( \Gamma' \sigma = \Gamma \sigma \).

Therefore \( \hat{\beta} \not\in \text{FV}(\Gamma) \) and \( \hat{\beta} \not\in \text{FV}(\Gamma) \), and \( \hat{\beta} \not\in \text{FV}(\Gamma) \).

Since we have \( \hat{\beta} \in \Gamma_0 \), we also have \( \hat{\beta} \in (\Gamma_0, \alpha_2 : \star) \).

By induction on the first premise, \( \hat{\beta} \not\in \text{FV}(\Theta) \).

Also by induction on the first premise, with \( \alpha_2 \) playing the role of \( \hat{\beta} \), we have \( \alpha_2 \not\in \text{FV}(\Theta) \).

Note that \( \hat{\beta} \in \text{unsolved}(\Gamma_0, \alpha_2 : \star) \).

By Lemma 20 (Left Unsolvedness Preservation), \( \alpha_2 \in \text{unsolved}(\Theta) \).

Therefore \( \Theta \) has the form \( (\Theta_0, \alpha_2 : \Theta_1) \).

Since \( \hat{\beta} \not\in \alpha_2 \), we know that \( \hat{\beta} \) is declared to the left of \( \alpha_2 \) in \( (\Gamma_0, \alpha_2 : \star) \), so by Lemma 20 (Declaration Order Preservation), \( \hat{\beta} \) is declared to the left of \( \alpha_2 \) in \( \Theta \). Hence \( \hat{\beta} \in \Theta_0 \).

Furthermore, by Lemma 43 (Instantiation Extension), we have \( \Gamma' \vdash \Theta \).

Then by Lemma 36 (Extension Weakening (Sorts)), we have \( \Delta \vdash \sigma : \kappa' \).

Using induction on the second premise, \( \hat{\beta} \not\in \text{FV}(\Delta) \).

\[\begin{align*}
\text{Case} \\
\Gamma' \vdash \alpha : \mathbb{N} \vdash \Delta \\
\Gamma' \vdash \alpha : \mathbb{N} \vdash \Delta
\end{align*}\]

We have \( \hat{\beta} \not\in \text{FV}(\Gamma) \). Since \( \Delta \) differs from \( \Gamma \) only in \( \alpha \), it must be the case that \( \Gamma \sigma = \Delta \sigma \). It is given that \( \hat{\beta} \not\in \text{FV}(\Gamma) \), so \( \hat{\beta} \not\in \text{FV}(\Delta) \).

\[\begin{align*}
\text{Case} \\
\Theta \vdash \alpha : \mathbb{N} \vdash \Delta \\
\Theta \vdash \alpha : \mathbb{N} \vdash \Delta
\end{align*}\]
Proof of **Lemma 65** *(Left Free Variable Preservation)*

\[
\Gamma \vdash \sigma : \kappa' \quad \text{Given}
\]
\[
\Theta \vdash \sigma : \kappa' \quad \text{By weakening}
\]
\[
\hat{\alpha} \notin \text{FV}(\Gamma[\sigma]) \quad \text{Given}
\]
\[
\hat{\alpha} \notin \text{FV}(\Theta[\sigma]) \quad \hat{\alpha} \notin \text{FV}(\Gamma[\sigma]) \text{ and } \Theta \text{ only solves } \hat{\alpha}
\]
\[
\Theta = (\Gamma_0, \hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \text{succ}(\hat{\alpha}_1), \Gamma_1) \quad \text{Given}
\]
\[
\hat{\beta} \notin \text{unsolved}(\Gamma_0) \quad \text{Given}
\]
\[
\hat{\beta} \notin \text{unsolved}(\Gamma_0, \hat{\alpha}_1 : \mathbb{N}) \quad \hat{\alpha}_1 \text{ fresh}
\]
\[
\hat{\beta} \notin \text{FV}(\Gamma[\sigma]) \quad \text{Given}
\]
\[
\hat{\beta} \notin \text{FV}(\Theta[\sigma]) \quad \hat{\alpha}_1 \text{ fresh}
\]
\[
\therefore \hat{\beta} \notin \text{FV}(\Delta[\sigma]) \quad \text{By i.h.}
\]

\[\square\]

**Lemma 66** *(Instantiation Size Preservation)*. If \(\Gamma_0, \hat{\alpha}_1 : \mathbb{N} \vdash \hat{\alpha} := \tau : \kappa \vdash \Delta \) and \(\Gamma \vdash s : \kappa' \) and \(\hat{\alpha} \notin \text{FV}(\Gamma[\sigma])\), then \(|\Delta|\sigma = |\Delta|\sigma\), where \(|C|\) is the plain size of the term \(C\).

**Proof**. By induction on the given derivation.

- Case

\[
\begin{align*}
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 & \vdash \hat{\alpha} := \tau : \kappa \vdash \Delta \Rightarrow \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \quad \text{InstSolve}
\end{align*}
\]

Since \(\Delta\) differs from \(\Gamma\) only in solving \(\hat{\alpha}\), and we know \(\hat{\alpha} \notin \text{FV}(\Gamma[\sigma])\), we have \(|\Delta|\sigma = |\Gamma|\sigma\); therefore \(|\Delta|\sigma = |\Gamma|\sigma|\).

- Case

\[
\begin{align*}
\Gamma', [\hat{\alpha} : \mathbb{N}] & \vdash \hat{\alpha} := \text{zero} : \mathbb{N} \vdash \Gamma', [\hat{\alpha} : \mathbb{N} = \text{zero}] \quad \text{InstZero}
\end{align*}
\]

Similar to the \text{InstSolve} case.

- Case

\[
\begin{align*}
\Gamma', [\hat{\alpha} : \kappa] & \vdash \hat{\alpha} := \hat{\beta} : \kappa \vdash \Gamma', [\hat{\alpha} : \kappa] \quad \text{InstReach}
\end{align*}
\]

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 \(\hat{\beta} \in \text{FV}(\Gamma[\sigma])\), we have \(|\Delta|\sigma = |\Gamma|\sigma|\).

- Case

\[
\begin{align*}
\Gamma' & \vdash \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2 \vdash \hat{\alpha}_1 : \tau_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \vdash \Delta \Rightarrow \Gamma'[\hat{\alpha} : \star] \vdash \hat{\alpha} := \tau_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \vdash \Delta \quad \text{InstBin}
\end{align*}
\]

We have \(\Gamma \vdash \sigma : \kappa'\) and \(\hat{\alpha} \notin \text{FV}(\Gamma[\sigma])\).

Since \(\hat{\alpha}_1, \hat{\alpha}_2 \notin \text{dom}(\Gamma)\), we have \(\hat{\alpha}, \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\Gamma[\sigma])\).

By Lemma \text{23} *(Deep Evar Introduction)*, \(\Gamma'[\hat{\alpha} : \star] \rightarrow \Gamma'\).

By Lemma \text{36} *(Extension Weakening (Sorts))* \(\Gamma' \vdash \tau_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \vdash \Delta\).

Since \(\hat{\alpha} \notin \text{FV}(\sigma)\), it follows that \(|\Gamma'|\sigma = |\Gamma|\sigma\), and so \(|\Gamma'|\sigma = |\Gamma|\sigma|\).

By induction on the first premise, \(|\Gamma'|\sigma = |\Theta|\sigma|\).

By Lemma \text{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 \text{64} *(Left Unsolvedness Preservation)*, since \(\hat{\alpha}_2 \in \text{unsolved}(\Gamma')\), it is unsolved in \(\Theta\): that is, \(\Theta = (\Theta_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \Theta_1)\).

By Lemma \text{43} *(Instantiation Extension)*, we have \(\Gamma' \rightarrow \Theta\).

By Lemma \text{36} *(Extension Weakening (Sorts))*, \(\Theta \vdash \sigma : \kappa'\).

Since \(\hat{\alpha}_2 \notin \text{FV}(\Omega[\sigma])\), Lemma \text{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, \(|\Gamma'|\sigma = |\Delta|\sigma|\).
Proof of Lemma 66 (Instantiation Size Preservation)

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

1. **Case**
   \[ \Gamma[\alpha : \star] \vdash \sigma : \kappa' \]
   - Given
   \[ \alpha \notin \Gamma[\alpha : \star] \sigma \]
   - Given
   \[ \Gamma[\alpha : \star] \Rightarrow \Gamma' \]
   - By Lemma 23 (Deep Evar Introduction)
   \[ \Gamma' \vdash \sigma : \kappa' \]
   - By Lemma 36 (Extension Weakening (Sorts))
   \[ ||\Gamma'||\sigma = ||\Gamma'[\alpha : \star]||\sigma \]
   - Since $\alpha \notin \text{FV}(||\Gamma'[\alpha : \star]||\sigma)$
   \[ ||\Gamma'||\sigma = ||\Gamma[\alpha : \star]||\sigma \]
   - By congruence of equality
   \[ \alpha_1 \notin \Gamma' \]
   - By i.h.
   \[ ||\Gamma'[\alpha : \star]||\sigma = ||\Theta||\sigma \]
   - By transitivity of equality

2. **Case**
   \[ \Gamma[\alpha : \star] \vdash \sigma : \kappa' \]
   - Given
   \[ \alpha \notin \Gamma[\alpha : \star] \sigma \]
   - Given
   \[ \Gamma[\alpha : \star] \Rightarrow \Gamma' \]
   - By Lemma 23 (Deep Evar Introduction)
   \[ \Gamma' \vdash \sigma : \kappa' \]
   - By Lemma 36 (Extension Weakening (Sorts))
   \[ ||\Gamma'||\sigma = ||\Gamma[\alpha : \star]||\sigma \]
   - Since $\alpha \notin \text{FV}(||\Gamma[\alpha : \star]||\sigma)$
   \[ ||\Gamma'||\sigma = ||\Theta||\sigma \]
   - By transitivity of equality

---

Lemma 67 (Decidability of Instantiation). If $\Gamma = \Gamma_0[\alpha : \kappa']$ and $\Gamma \vdash \tau : \kappa$ such that $||\Gamma||\tau = \tau$ and $\alpha \notin \text{FV}(\tau)$, then:

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

Proof. By induction on the derivation of $\Gamma \vdash \tau : \kappa$.

- **Case**
  \[ (u : \kappa) \in \Gamma \]
  - By rule $\text{VarSort}$
  If $\kappa \neq \kappa'$, no rule matches and no derivation exists.
  Otherwise:
    - If $(u : \kappa) \in \Gamma_L$, we can apply rule $\text{InstSolve}$
    - If $u$ is some unsolved existential variable $\beta$ and $(\hat{\beta} : \kappa) \in \Gamma_R$, then we can apply rule $\text{InstReach}$
    - Otherwise, $u$ is declared in $\Gamma_R$ and is a universal variable; no rule matches and no derivation exists.

- **Case**
  \[ (\hat{\beta} : \kappa = \tau) \in \Gamma \]
  - By rule $\text{SolvedVarSort}$
  By inversion, $(\hat{\beta} : \kappa = \tau) \in \Gamma$, but $||\Gamma||\hat{\beta} = \hat{\beta}$ is given, so this case is impossible.

- **Case**
  If $\kappa' = \star$, then apply rule $\text{InstSolve}$. Otherwise, no rule matches and no derivation exists.

- **Case**
  \[ \Gamma \vdash \tau_1 : \star \quad \Gamma \vdash \tau_2 : \star \]
  - By rule $\text{BinSort}$
  If $\kappa' \neq \star$, then no rule matches and no derivation exists. Otherwise:
    Given, $||\Gamma||([\tau_1 \oplus \tau_2] = \tau_1 \oplus \tau_2$ and $\alpha \notin \text{FV}(||\Gamma||([\tau_1 \oplus \tau_2]))$.
    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 $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2$, $\Gamma_R \vdash \hat{\alpha}_1 := \star : \dashv \Delta$ is decidable.

    By definition of substitution, $||\Gamma||([\tau_1 \oplus \tau_2] = ||\Gamma||[\tau_1] \oplus ||\Gamma||[\tau_2]$.
    Since $||\Gamma||([\tau_1 \oplus \tau_2] = \tau_1 \oplus \tau_2$, we have $||\Gamma||[\tau_1] = \tau_1$ and $||\Gamma||[\tau_2] = \tau_2$.

    By weakening, $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2$, $\Gamma_R \vdash \hat{\alpha}_1 \oplus \hat{\alpha}_2 : \star$.
    Since $\Gamma \vdash \tau_1 : \star$ and $\Gamma \vdash \tau_2 : \star$, we have $\hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\tau_1) \cup \text{FV}(\tau_2)$.
    Since $\hat{\alpha} \notin \text{FV}(t)$, it follows that $||\Gamma||[\tau_1] = \tau_1$.
    By i.h., either there exists $\Theta$ s.t. $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2$, $\Gamma_R \vdash \hat{\alpha}_1 := \tau_1 : \dashv \Theta$, or not.
    If not, then no derivation by $\text{InstBin}$ exists.
Proof.
By induction on \( \Gamma \).

By Lemma \textit{65} (Left Unsolvedness Preservation), we have \( \Delta_2 \in \text{unsolved}(\Theta) \).

Substitution is idempotent, so \( \Theta|\Theta|\tau_2 = \Theta|\tau_2 \).

By i.h., either there exists \( \Delta \) such that \( \Theta \vdash \Delta_2 := |\Theta|\tau_2 : \kappa \rightarrow \Delta \), or not.

If not, no derivation by \textit{InstBin} exists.

Otherwise, there exists such a \( \Delta \). By rule \textit{InstBin}, we have \( \Gamma \vdash \Delta := t : \kappa \rightarrow \Delta \).

- Case
  \( \Gamma \vdash \text{zero} : \text{ZeroSort} \)

If \( \kappa' \neq \text{N} \), then no rule matches and no derivation exists. Otherwise, apply rule \textit{InstSolve}.

- Case
  \( \Gamma \vdash \text{succ}\left(t_0\right) : \text{SuccSort} \)

If \( \kappa' \neq \text{N} \), then no rule matches and no derivation exists. Otherwise:

If \( \Gamma \vdash \text{succ}\left(t_0\right) : \text{N} \), then we have a derivation by \textit{InstSolve}.

If not, the only other rule whose conclusion matches \( \text{succ}\left(t_0\right) \) is \textit{InstSucc}.

The remainder of this case is similar to the \textit{BinSort} case, but shorter.

\( G' \) Separation

\textbf{Lemma 68} (Transitivity of Separation).
If \( (\Gamma_l \star \Gamma_r) \rightarrow (\Theta_l \star \Theta_r) \) and \( (\Theta_l \star \Theta_r) \rightarrow (\Delta_l \star \Delta_r) \)
then \( (\Gamma_l \star \Gamma_r) \rightarrow (\Delta_l \star \Delta_r) \).

\textbf{Proof}.
\( \begin{align*}
(\Gamma_l \star \Gamma_r) \rightarrow (\Theta_l \star \Theta_r) & \quad \text{Given} \\
(\Gamma_l, \Gamma_r) \rightarrow (\Theta_l, \Theta_r) & \quad \text{By Definition 5} \\
\Gamma_l \subseteq \Theta_l \text{ and } \Gamma_r \subseteq \Theta_r & \quad " \\
(\Theta_l \star \Theta_r) \rightarrow (\Delta_l \star \Delta_r) & \quad \text{Given} \\
(\Theta_l, \Theta_r) \rightarrow (\Delta_l, \Delta_r) & \quad \text{By Definition 5} \\
\Theta_l \subseteq \Delta_l \text{ and } \Theta_r \subseteq \Delta_r & \quad " \\
(\Gamma_l, \Gamma_r) \rightarrow (\Delta_l, \Delta_r) & \quad \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma_l \subseteq \Delta_l \text{ and } \Gamma_r \subseteq \Delta_r & \quad \text{By transitivity of } \subseteq \\
\Rightarrow (\Gamma_l \star \Gamma_r) \rightarrow (\Delta_l \star \Delta_r) & \quad \text{By Definition 5}
\end{align*} \)

\textbf{Lemma 69} (Separation Truncation).
If \( H \) has the form \( \alpha : \kappa \) or \( \downarrow \kappa \) or \( \uparrow \)
and \( (\Gamma_l \star \Gamma_r, H) \rightarrow (\Delta_l \star \Delta_r) \)
then \( (\Gamma_l \star \Gamma_r) \rightarrow (\Delta_l \star \Delta_r) \) where \( \Delta_r = (\Delta_0, H, \Theta) \).

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

If \( \Delta_r = (\ldots, H) \), we have \( (\Gamma_l \star \Gamma_r, H) \rightarrow (\Delta_l \star (\Delta, H)) \), and inversion on \( \rightarrow \text{Uvar} \) (if \( H \) is \( \alpha : \kappa \), or the corresponding rule for other forms) gives the result (with \( \Theta = \ldots \)).

Otherwise, proceed into the subderivation of \( \Gamma_l, \Gamma_r, \alpha : \kappa \rightarrow (\Delta_l, \Delta_r) \), with \( \Delta_r = (\Delta'_r, \Delta') \) where \( \Delta' \) is a single declaration. Use the i.h. on \( \Delta'_l \), producing some \( \Theta' \). Finally, let \( \Theta = (\Theta', \Delta') \).

\textbf{Lemma 70} (Separation for Auxiliary Judgments).
(i) If \( \Gamma_l \star \Gamma_r \vdash \sigma \models \tau : \kappa \rightarrow \Delta \) 
and FEV(\( \sigma \)) \cup FEV(\( \tau \)) \subseteq dom(\( \Gamma_r \))
then \( \Delta = (\Delta_l \star \Delta_r) \) and \( (\Gamma_l \star \Gamma_r) \rightarrow (\Delta_l \star \Delta_r) \).
(ii) If $\Gamma_L \vdash \Gamma_R \vdash P \rightarrow \perp$ and $\text{FEV}(P) \subseteq \text{dom}(\Gamma_R)$ then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \rightarrow^\perp (\Delta_L \ast \Delta_R)$. 

(iii) If $\Gamma_L \vdash \Gamma_R / \sigma \vdash \kappa \rightarrow \Delta$ and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset$ then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \rightarrow^\perp (\Delta_L \ast \Delta_R)$. 

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

(v) If $\Gamma_L \vdash \Gamma_R \vdash \Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \rightarrow^\perp (\Delta_L \ast \Delta_R)$. 

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

(vii) If $\Gamma_L \vdash \Gamma_R \vdash A \equiv B \rightarrow \Delta$ and $(\text{FEV}(A) \cup \text{FEV}(B)) \subseteq \text{dom}(\Gamma_R)$ then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \rightarrow^\perp (\Delta_L \ast \Delta_R)$. 

Proof. Part (i): By induction on the derivation of the given checkeq judgment. Cases CheckeqVar and CheckeqZero are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). For case CheckeqSucc apply the i.h. For cases CheckeqInl and CheckeqInR use the i.h. (v). For case CheckeqBin use reasoning similar to that in the $\Delta$ case (transitivity of separation, and applying $\Theta$ in the second premise).

Part (ii), checkprop: Use the i.h. (i).

Part (iii), elimeq: Cases ElimeqUvarRel, ElimeqUnit and CheckeqZero are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). Cases ElimeqUvarL, ElimeqUvarR, ElimeqBinBot and ElimeqClash 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 $\Delta = (\Gamma_L, \Gamma_R, \alpha = \tau)$ which, since $\text{FEV}(\tau) \subseteq \text{dom}(\Gamma_R)$, ensures that $(\Gamma_L \ast \Gamma_R) \rightarrow^\perp (\Delta_L \ast \Delta_R, \alpha = \tau)$.

Part (iv), elimpred: Use the i.h. (iii).

Part (v), instjudg:

- **Case InstSolve** Here, $\Gamma = (\Gamma_0, \beta : \kappa, \Gamma_1)$ and $\Delta = (\Gamma_0, \beta : \kappa, \Gamma_1)$. We have $\Delta \in \text{dom}(\Gamma_R)$, so the declaration $\beta : \kappa$ is in $\Gamma_R$. Since $\text{FEV}(\beta) \subseteq \text{dom}(\Gamma_R)$, the context $\Delta$ maintains the separation.

- **Case InstReach** Here, $\Gamma = \Gamma_0[\beta : \kappa] \subseteq \text{dom}(\Gamma_R)$ and $\Delta = \Gamma_0[\beta : \kappa]$. We have $\Delta \in \text{dom}(\Gamma_R)$, so the declaration $\beta : \kappa$ is in $\Gamma_R$. Since $\beta$ is declared to the right of $\alpha$, it too must be in $\Gamma_R$, which can also be shown from $\text{FEV}(\beta) \subseteq \text{dom}(\Gamma_R)$. Both declarations are in $\Gamma_R$, so the context $\Delta$ maintains the separation.

- **Case InstZero** In this rule, $\Delta$ is the same as $\Gamma$ except for a solution $\text{zero}$, which doesn’t violate separation.

- **Case InstSucc** The result follows by i.h., taking care to keep the declaration $\alpha : \kappa : \pi$ on the right when applying the i.h., even if $\alpha : \pi$ is the leftmost declaration in $\Gamma_R$, ensuring that $\text{succ}(\alpha_1)$ does not violate separation.

- **Case InstBin** As in the InstSucc case, the new declarations should be kept on the right-hand side of the separator. Otherwise the case is straightforward (using the i.h. twice and transitivity).

Part (vi), propequivicjudg: Similar to the CheckeqBin case of part (i), using the i.h. (i).

Part (vii), equivjudg:

- **Cases $\equiv \text{Var}$, $\equiv \text{Exvar}$, $\equiv \text{Unit}$** Immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$).

- **Case $\equiv \text{Bin}$** Similar to the case CheckeqBin in part (i).

- **Cases $\equiv \text{Clash}$** Similar to the case CheckeqBin in part (i).
Lemma 71 (Separation for Subtyping). If $\Gamma_L \ast \Gamma_R \vdash A \triangleleft\triangleq B \vdash \Delta$
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 $(\Gamma_L \ast \Gamma_R) \not	riangleleft\triangleq (\Delta_L \ast \Delta_R)$.

Proof. By induction on the given derivation. In the $\triangleleft\triangleq\equiv$ 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).

(Spine) If $\Gamma_L \ast \Gamma_R \vdash s : A \triangleright\triangleright C \triangleright\triangleright \Delta$
or $\Gamma_L \ast \Gamma_R \vdash s : A \triangleright\triangleright C \triangleright\triangleright \Delta$
and $\Gamma_L \ast \Gamma_R \vdash \Delta_L \ast \Delta_R$
and $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\triangleright\triangleright (\Delta_L \ast \Delta_R)$ and $\text{FEV}(C) \subseteq \text{dom}(\Delta_R)$.

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

(Synthesis) If $\Gamma_L \ast \Gamma_R \vdash e \Rightarrow A \triangleright\triangleright \Delta$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\triangleright\triangleright (\Delta_L \ast \Delta_R)$.

(Match) If $\Gamma_L \ast \Gamma_R \vdash P \vdash \Pi :: \bar{A} \triangleleft\triangleleft C \triangleright\triangleright \Delta$
and $\text{FEV}(\bar{A}) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \not\triangleright\triangleright (\Delta_L \ast \Delta_R)$.

(Match Elim.) If $\Gamma_L \ast \Gamma_R \vdash P \vdash \Pi :: \bar{A} \triangleleft\triangleleft C \triangleright\triangleright \Delta$
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 $(\Gamma_L \ast \Gamma_R) \not\triangleright\triangleright (\Delta_L \ast \Delta_R)$.

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 **MatchNeg** 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** [11] [51L] In all of these rules, the output context is the same as the input context, so just let $\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$.

Proof of Lemma 72 (Separation—Main)
Proof of Lemma 72 \textbf{(Separation—Main)}

\[ \Gamma_L, \Gamma_R \vdash e : A \quad \Gamma_L \vdash \alpha : \kappa \quad \rho \quad q \quad \Gamma_L * \Gamma_R \vdash \text{EmptySpine} \]

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 \( \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \).

\begin{itemize}
  \item Case \( \Gamma_L * \Gamma_R \vdash e \Rightarrow A, q \vdash \emptyset \quad \emptyset \vdash A <^{\alpha} B \vdash \Delta \)

  By i.h., \( \emptyset = (\Theta_L * \Theta_R) \) and \( (\Gamma_L * \Gamma_R) \vdash \Delta_L \Delta_R \).

  By Lemma 71 \textbf{(Separation for Subtyping)}, \( \Delta = (\Delta_L \Delta_R) \) and \( (\Theta_L * \Theta_R) \vdash (\Delta_L \Delta_R) \).

  By Lemma 68 \textbf{(Transitivity of Separation)}, \( (\Gamma_L * \Gamma_R) \vdash (\Delta_L \Delta_R) \).

  \end{itemize}

\begin{itemize}
  \item Case \( \Gamma \vdash \text{At type} \quad \Gamma \vdash e \Rightarrow [\Gamma]A ! \vdash \Delta \)

  By i.h.; since \( \text{FEV}(A) = \emptyset \), the condition on the \( (\text{Checking}) \) part is trivial.

  \end{itemize}

Adding a solution with a ground type cannot destroy separation.

\begin{itemize}
  \item Case \( \nu \text{chk-I} \quad \Gamma_L, \Gamma_R, \alpha : \kappa \vdash \nu \Leftarrow A_0 \quad \Gamma \vdash \nu \Leftarrow \forall \alpha : \kappa, A_0 \vdash \Delta \)

  \[ \text{FEV}(\forall \alpha : \kappa, A_0) \subseteq \text{dom}(\Gamma_R) \quad \text{Given} \]

  \[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R, \alpha : \kappa) \quad \text{From definition of FEV} \]

  \[ (\Delta, \alpha : \kappa, \emptyset) = (\Delta_L \Delta_R) \quad \text{By i.h.} \]

  \[ (\Gamma_L * (\Gamma_R, \alpha : \kappa)) \vdash (\Delta_L \Delta_R) \]

  \[ (\Gamma_L * \Gamma_R) \vdash (\Delta_L \Delta_R) \]

  \[ \Delta_L \Delta_R = (\Delta_L, \Delta_R) \quad \text{By Lemma 69 \textbf{(Separation Truncation)}} \]

  \[ (\Delta, \alpha : \kappa, \emptyset) = (\Delta_L, \Delta_R) \quad \text{Definition of *} \]

  \[ (\Delta_L, \Delta_R, \alpha : \kappa, \emptyset) \quad \text{By above equation} \]

  \[ \Delta = (\Delta_L, \Delta_R) \quad \alpha \text{ not multiply declared} \]

  \end{itemize}

\begin{itemize}
  \item Case \( \Gamma_L, \Gamma_R, \alpha : \kappa \vdash e \Rightarrow s : [\alpha/\alpha]A_0 \Rightarrow C \quad q \vdash \Delta \)

  \[ \text{FEV}(\forall \alpha : \kappa, A_0) \subseteq \text{dom}(\Gamma_R) \quad \text{Given} \]

  \[ \text{FEV}([\alpha/\alpha]A_0) \subseteq \text{dom}(\Gamma_R, \alpha : \kappa) \quad \text{From definition of FEV} \]

  \[ \Delta = (\Delta_L \Delta_R) \quad \text{By i.h.} \]

  \[ (\Gamma_L * (\Gamma_R, \alpha : \kappa)) \vdash (\Delta_L \Delta_R) \]

  \[ (\Gamma_L * \Gamma_R) \vdash (\Delta_L \Delta_R) \]

  \[ \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \quad \text{By Definition 5} \]

  \[ \text{dom}(\Gamma_L) \subseteq \text{dom}(\Delta_L) \quad \text{By Definition 5} \]

  \[ \text{dom}(\Gamma_R) \subseteq \text{dom}(\Delta_R) \quad \text{By Definition 5} \]

  \[ \text{dom}(\Gamma_R) \cup (\alpha) \subseteq \text{dom}(\Delta_R) \quad \text{By definition of \text{dom}(\cdots)} \]

  \[ \text{dom}(\Gamma_R) \subseteq \text{dom}(\Delta_R) \quad \text{Property of \subseteq} \]

  \[ (\Gamma_L, \Gamma_R) \Rightarrow (\Delta_L, \Delta_R) \quad \text{By Lemma 51 \textbf{(Typing Extension)}} \]

  \[ (\Gamma_L * \Gamma_R) \vdash (\Delta_L \Delta_R) \quad \text{By Definition 5} \]

  \end{itemize}
Proof of Lemma 72 (Separation—Main) lem:separation-main

\[ \text{Case} \quad \text{e not a case} \]
\[ \Gamma_L \vdash \Gamma_R \vdash P \text{ true } \quad \Theta \vdash e \iff [\Theta]A_0 \p \vdash \Delta \]
\[\Gamma_L \vdash \Gamma_R \vdash e \iff A_0 \wedge P \vdash \Delta \]

- **Case**
  - \( \Gamma_L \vdash \Gamma_R \vdash (A_0 \wedge P) \) \( p \) type
    - Given
  - \( \Gamma_L \vdash \Gamma_R \vdash P \) prop
    - By inversion
  - \( \Gamma_L \vdash \Gamma_R \vdash A_0 \) \( p \) type
    - By inversion
  - \( \text{FEV}(A_0 \wedge P) \subseteq \text{dom}(\Gamma_R) \)
    - Given
  - \( \text{FEV}(P) \subseteq \text{dom}(\Gamma_R) \)
    - By def. of FEV
  - \( \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \)
    - By previous line
  - \( \Theta = (\Theta_L \cdot \Theta_R) \)
    - By Lemma 70 (Separation for Auxiliary Judgments) (i)
  - \( (\Gamma_L \cdot \Gamma_R) \vdash (\Theta_L \cdot \Theta_R) \)
    - By earlier line
  - \( \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \)
    - Above
  - \( \text{dom}(\Gamma_R) \subseteq \text{dom}(\Theta_R) \)
    - By Definition 5
  - \( \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R) \)
    - By previous line
  - \( \text{FEV}(\Theta_A) \subseteq \text{dom}(\Theta_R) \)
    - Previous line and \( (\Gamma_L \cdot \Gamma_R) \vdash (\Theta_L \cdot \Theta_R) \)
  - \( \Gamma_L \cdot \Gamma_R \vdash (A_0 \wedge P) \) \( p \) type
    - Given
  - \( \Gamma_L \cdot \Gamma_R \vdash A_0 \) \( p \) type
    - By inversion
  - \( \Theta \vdash A_0 \) \( p \) type
    - By Lemma 41 (Extension Weakening for Principal Typing)
  - \( \Theta \vdash (\Theta)A_0 \) \( p \) type
    - By Lemma 13 (Right-Hand Substitution for Typing)
  - \( \Delta = (\Delta_L \cdot \Delta_R) \)
    - By i.h.
  - \( (\Theta_L \cdot \Theta_R) \vdash (\Delta_L \cdot \Delta_R) \)
    - By Lemma 68 (Transitivity of Separation)

\[ \text{Case} \quad \nu \text{ chk-I} \]
\[ \Gamma_L \vdash (\Gamma_R, \Gamma_R, p) \vdash P \vdash \Theta \quad \Theta \vdash \nu \iff [\Theta]A_0 \vdash \Delta, \Gamma_R, \nu, \Delta' \]
\[\Gamma_L \vdash (\Gamma_R, \Gamma_R, p) \vdash P \vdash \Theta \quad \Theta \vdash \nu \iff P \supset A_0 \vdash \Delta \]

- **Case**
  - \( \Gamma_L \cdot \Gamma_R \vdash P \supset A_0 ! \vdash \Delta \)
    - Given
  - \( \Gamma_L \cdot \Gamma_R \vdash P \supset A_0 \) prop
    - By inversion
  - \( \text{FEV}(P \supset A_0) = \emptyset \)
    - By def. of FEV
  - \( \text{FEV}(P) = \emptyset \)
    - By def. of FEV
  - \( \Gamma_L \vdash (\Gamma_R, \Gamma_R, p) \vdash P \vdash \Theta \)
    - By Lemma 70 (Separation for Auxiliary Judgments) (ii)
  - \( \Gamma_L, \Gamma_R, p \vdash A_0 ! \vdash \Delta \)
    - By Lemma 42 (Inversion of Principal Typing) (2)
  - \( \Theta \vdash [\Theta]A_0 ! \vdash \Delta \)
    - By Lemma 35 (Suffix Weakening)
  - \( \Theta \vdash (\Theta)A_0 ! \vdash \Delta \)
    - By Lemmas 41 and 40
  - \( \text{FEV}(A_0) = \emptyset \)
    - Above and def. of FEV
  - \( \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R) \)
    - Immediate
  - \( (\Delta, \Gamma_R, p, \Delta') = (\Delta_L \cdot \Delta_R') \)
    - By i.h.
  - \( (\Theta_L \cdot \Theta_R) \vdash (\Delta_L \cdot \Delta_R') \)
    - By Lemma 68 (Transitivity of Separation)
  - \( (\Gamma_L \cdot \Gamma_R) \vdash (\Delta_L \cdot \Delta_R') \)
    - By Lemma 68 (Transitivity of Separation)
  - \( \Delta_l = (\Delta_L, \Delta_R, \ldots) \)
    - By Lemma 68 (Transitivity of Separation)
  - \( \Delta = (\Delta_L, \Delta_R) \)
    - Similar to the \( \forall \) case

- **Case**
  - \( \Gamma_L \cdot \Gamma_R \vdash P \text{ true } \vdash \Theta \quad \Theta \vdash e \vdash s : [\Theta]A_0 \supset C \vdash q \vdash \Delta \)
  - \( \Gamma_L \cdot \Gamma_R \vdash e \vdash s : P \supset A_0 \supset C \vdash q \vdash \Delta \)
  - \( \supset \text{ Spine} \)
Proof of Lemma 72 (Separation—Main) 

\[ \Gamma_L, \Gamma_R \vdash (P \vdash \Lambda_0) \text{ p type} \quad \text{Given} \]
\[ \Gamma_L, \Gamma_R \vdash P \text{ prop} \quad \text{By inversion} \]
\[ \Gamma_L, \Gamma_R \vdash \Gamma \text{ true } \vdash \Theta \quad \text{Subderivation} \]
\[ \Theta = (\Theta_L \cdot \Theta_R) \quad \text{By Lemma 70 (Separation for Auxiliary Judgments) (i)} \]
\[ (\Gamma_L \ast \Gamma_R) \vdash \Theta \rightarrow (\Theta_L \cdot \Theta_R) \]
\[ (\Delta, \Theta, \Gamma) = (\Delta_L, \Delta_R) \quad \text{Subderivation} \]
\[ (\Theta_L \cdot \Theta_R) \vdash \Theta \rightarrow (\Delta_L \ast \Delta_R) \]
\[ \text{By i.h.} \]
\[ \text{By Lemma 68 (Transitivity of Separation)} \]

\[ \text{Case} \]
\[ \Gamma_L, \Gamma_R, x : A p \vdash e : B p \vdash \Delta, x : A p, \Theta \]
\[ \Gamma_L, \Gamma_R \vdash \lambda x. e \in A \rightarrow B p \vdash \Delta \]
\[ \text{By weakening and Definition 4} \]
\[ \text{Subderivation} \]
\[ \text{By i.h.} \]
\[ \text{By Lemma 69 (Separation Truncation)} \]
\[ \text{Similar to the \textcolor{red}{\textbf{\textbullet}}} \text{ case} \]

We have \( (\Gamma_L \ast \Gamma_R) = \Gamma_0[\hat{\alpha}: \ast] \). We also have \( \text{FEV}(\hat{\alpha}) \subseteq \text{dom}(\Gamma_R) \). Therefore \( \hat{\alpha} \in \text{dom}(\Gamma_R) \) and
\[ \Gamma_0[\hat{\alpha}: \ast] = \Gamma_L, \Gamma_2, \hat{\alpha}: \ast, \Gamma_3 \]

where \( \Gamma_R = (\Gamma_2, \hat{\alpha}: \ast, \Gamma_3) \).

Then the input context in the premise has the following form:
\[ \Gamma_0[\ast_1: \ast, \ast_2: \ast, \ast: \ast = \hat{\alpha} : \ast \rightarrow \hat{\alpha}_2], x : \hat{\alpha}_1 \vdash e_0 \in \hat{\alpha}_2 \vdash \Delta, x : \hat{\alpha}_1, \Delta' \]
\[ \Gamma_0[\ast: \ast] \vdash \lambda x. e_0 \in \hat{\alpha} \vdash \Delta \]

\[ \text{where} \quad \Gamma_0 \vdash \ast = \hat{\alpha} \rightarrow \hat{\alpha}_2, x : \hat{\alpha}_1 \]

Let us separate this context at the same point as \( \Gamma_0[\hat{\alpha}: \ast] \), that is, after \( \Gamma_L \) and before \( \Gamma_2 \), and call the resulting right-hand context \( \Gamma' \). That is,
\[ \Gamma_0[\ast_1: \ast, \ast_2: \ast, \ast: \ast = \hat{\alpha} : \ast \rightarrow \hat{\alpha}_2], x : \hat{\alpha}_1 \]
\[ \Gamma_L, \Gamma_2, \hat{\alpha} : \ast, \Gamma_3 \]

Then the input context in the premise has the following form:
\[ \Gamma_0[\hat{\alpha}: \ast] \subseteq \text{dom}(\Gamma_R) \]
\[ \Gamma_L, \Gamma_R \vdash e_0 \in \hat{\alpha}_2 \vdash \Delta, x : \hat{\alpha}_1, \Delta' \]
\[ \text{Subderivation} \]
\[ \text{Given} \]
\[ \text{By i.h.} \]
\[ \text{Similar to the \textcolor{red}{\textbf{\textbullet}}} \text{ case} \]

Proof of Lemma 72 (Separation—Main) lem:separation-main
Proof of Lemma 72 (Separation—Main)

---

**Proof of Lemma 72 (Separation—Main)**

**Case**
\[ \Gamma \vdash e \Rightarrow A \ p \vdash \Theta \]  
\[ \Theta \vdash s : \Theta A \ p \Rightarrow C \ [q] \vdash \Delta \]
\[ \Gamma \vdash e \cdot s \Rightarrow C \ q \vdash \Delta \]

Use the i.h. and Lemma 58 (Transitivity of Separation), with Lemma 89 (Well-formedness of Algorithmic Typing) and Lemma 13 (Right-Hand Substitution for Typing).

**Case**
\[ \Gamma \vdash s : A \Rightarrow C \ f \vdash \Delta \]
\[ \text{FEV}(\Delta C) = \emptyset \]
\[ \Gamma \vdash s : A \Rightarrow C \ f \vdash \Delta \]

Use the i.h.

**Case**
\[ \Gamma \vdash s : A \Rightarrow C \ q \vdash \Delta \]
\[ \text{(} p = f \text{ or } q = ! \text{ or } \text{FEV}(\Delta C) \neq \emptyset \text{)} \]
\[ \Gamma \vdash s : A \Rightarrow C \ q \vdash \Delta \]

Use the i.h.

**Case**
\[ \Gamma_1 + \Gamma_2 \vdash e \Leftarrow A_1 \ p \vdash \Theta \]
\[ \Theta \vdash s : \Theta A_2 \ p \Rightarrow C \ q \vdash \Delta \]
\[ \Gamma \vdash (A_1 \rightarrow A_2) \ p \text{ type} \]
\[ \Gamma \vdash A_1 \ p \text{ type} \]
\[ \Gamma \vdash \text{FEV}(A_1) \subseteq \text{dom}(\Gamma_R) \]
\[ \Theta = (\Theta_L, \Theta_R) \]
\[ \text{FEV}(A_1) \subseteq \text{dom}(\Gamma_R) \]
\[ \text{By def. of FEV} \]
\[ \text{by i.h.} \]
\[ \text{by i.h.} \]
\[ \Delta = (\Delta_L, \Delta_R) \]
\[ \text{by i.h.} \]
\[ \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \]
\[ \text{by i.h.} \]
\[ \text{FEV}(\Delta) \subseteq \Delta_R \]
\[ \text{by i.h.} \]
\[ \text{by Lemma 68 (Transitivity of Separation)} \]

**Case**
\[ \Gamma \vdash e \Leftarrow A_k \ p \vdash \Delta \]
\[ \Gamma \vdash \text{inj}_k e \Leftarrow A_1 + A_2 \ p \vdash \Delta \]

Use the i.h. (inverting \( \Gamma \vdash (A_1 + A_2) \ p \text{ type} \)).

**Case**
\[ \Gamma \vdash e_1 \Leftarrow A_1 \ p \vdash \Theta \]
\[ \Theta \vdash e_2 \Leftarrow \Theta A_2 \ p \vdash \Delta \]
\[ \Gamma \vdash (e_1, e_2) \Leftarrow A_1 \times A_2 \ p \vdash \Delta \]

\[ \Gamma \vdash A_1 \ p \text{ type} \]
\[ \Gamma \vdash A_1 \ p \text{ type} \]
\[ \Gamma \vdash e_1 \Leftarrow A_1 \ p \vdash \Theta \]
\[ \Theta = (\Theta_L, \Theta_R) \]
\[ \text{by i.h.} \]
\[ \text{by i.h.} \]
\[ \text{by i.h.} \]
\[ \Delta = (\Delta_L, \Delta_R) \]
\[ \text{by i.h.} \]
\[ \text{by Lemma 68 (Transitivity of Separation)} \]
• **Case**: 

\[
\Gamma[\tilde{\alpha}_2:*], \tilde{\alpha}_1:*; \tilde{\alpha}:* = \tilde{\alpha}_1 \times \tilde{\alpha}_2] \vdash e_1 \leftrightarrow \tilde{\alpha}_1 \dashv \Theta \quad \Theta \vdash e_2 \leftrightarrow [\Theta]\tilde{\alpha}_2 \dashv \Delta
\]

We have \((\Gamma_L \ast \Gamma_R) = \Gamma_0[\tilde{\alpha} : \star]\). We also have \(\text{FEV}(\tilde{\alpha}) \subseteq \text{dom}(\Gamma_R)\). Therefore \(\tilde{\alpha} \in \text{dom}(\Gamma_R)\) and

\[
\Gamma_0[\tilde{\alpha} : \star] = \Gamma_L, \Gamma_2, \tilde{\alpha} : \star, \Gamma_3
\]

where \(\Gamma_R = (\Gamma_2, \tilde{\alpha} : \star, \Gamma_3)\).

Then the input context in the premise has the following form:

\[
\Gamma_0[\tilde{\alpha}_1;\star, \tilde{\alpha}_2;\star, \tilde{\alpha}:* = \tilde{\alpha}_1 \times \tilde{\alpha}_2]
\]

Let us separate this context at the same point as \(\Gamma_0[\tilde{\alpha} : \star]\), that is, after \(\Gamma_L\) and before \(\Gamma_2\), and call the resulting right-hand context \(\Gamma_R'\):

\[
\Gamma_0[\tilde{\alpha}_1;\star, \tilde{\alpha}_2;\star, \tilde{\alpha}:* = \tilde{\alpha}_1 \times \tilde{\alpha}_2] = \Gamma_L \ast \left(\Gamma_2, \tilde{\alpha}_1;\star, \tilde{\alpha}_2;\star, \tilde{\alpha}:* = \tilde{\alpha}_1 \times \tilde{\alpha}_2, \Gamma_3\right)
\]

\[
\Gamma_R = (\Gamma_2, \tilde{\alpha} : \star, \Gamma_3)
\]

\[
\Gamma_R' = \left(\Gamma_2, \tilde{\alpha}_1;\star, \tilde{\alpha}_2;\star, \tilde{\alpha}:* = \tilde{\alpha}_1 \times \tilde{\alpha}_2, \Gamma_3\right)
\]

Above

By Lemma 23 [Deep Evar Introduction] (i), (ii) and the definition of separation, we can show

\[
(\Gamma_1 \ast (\Gamma_2, \tilde{\alpha} : \star, \Gamma_3)) \dashv \rightarrow (\Gamma_L \ast \left(\Gamma_2, \tilde{\alpha}_1;\star, \tilde{\alpha}_2;\star, \tilde{\alpha}:* = \tilde{\alpha}_1 \times \tilde{\alpha}_2, \Gamma_3\right))
\]

\[
(\Gamma_L \ast \Gamma_R) \dashv \rightarrow (\Gamma_L \ast \Gamma_R')
\]

By above equalities

\[
(\Delta_L \ast \Delta_R)
\]

By Lemma 68 [Transitivity of Separation] twice

• **Case**: 

\[
\Gamma[\tilde{\alpha}_1:*], \tilde{\alpha}_2:*; \tilde{\alpha}:* = \tilde{\alpha}_1+\tilde{\alpha}_2] \vdash e \leftrightarrow \tilde{\alpha}_k \dashv \Delta
\]

Similar to the \(\times\tilde{\alpha}_k\) case, but simpler.

• **Case**: 

\[
\Gamma[\tilde{\alpha}_2:*], \tilde{\alpha}_1:*; \tilde{\alpha}:* = \tilde{\alpha}_1 \rightarrow \tilde{\alpha}_2] \vdash e \rightarrow s_0 : (\tilde{\alpha}_1 \rightarrow \tilde{\alpha}_2) \Rightarrow C \dashv \Delta
\]

Similar to the \(\times \tilde{\alpha}_k\) and \(\times \tilde{\alpha}_k\) cases, except that (because we’re in the spine part of the lemma) we have to show that \(\text{FEV}(C) \subseteq \text{dom}(\Delta_R)\). But we have the same \(C\) in the premise and conclusion, so we get that by applying the i.h.

• **Case**: 

\[
\Gamma \vdash e \Rightarrow A ! \dashv \Theta \quad \Theta \vdash \Pi : A \leftarrow [\Theta]C p \dashv \Delta \quad \Delta \vdash \Pi \text{ covers } [\Delta]A
\]

Use the i.h. and Lemma 68 [Transitivity of Separation].
H' Decidability of Algorithmic Subtyping

H'.1 Lemmas for Decidability of Subtyping

Lemma 73 (Substitution Isn’t Large).
For all contexts \( \Theta \), we have \( \#\text{large}(\Theta \Delta) = \#\text{large}(\Delta) \).

Proof. By induction on \( \Delta \), following the definition of substitution.

Lemma 74 (Instantiation Solves).
If \( \Gamma \vdash \Delta := \tau : \kappa \vdash \Delta \) and \( \Gamma \vdash \Delta \) then \( \text{unsolved}(\Gamma) = \text{unsolved}(\Delta) + 1 \).

Proof. By induction on the given derivation.

- Case
  \[ \frac{\Gamma \vdash \Delta := \tau : \kappa \vdash \Delta \text{ InstSolve} \quad \Gamma, \Delta \vdash \Delta := \tau : \kappa \vdash \Delta \text{ InstSolve} \quad \Gamma \text{ InstReach} \quad \Gamma \text{ InstReach} }{\Gamma[\beta : \kappa] \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} } \]

It is evident that \( \text{unsolved}(\Gamma[\beta : \kappa]) = \text{unsolved}(\Gamma[\beta : \kappa]) + 1 \).

- Case
  \[ \frac{\Gamma[\beta : \kappa] \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} \quad \Gamma \text{ InstReach} }{\Gamma[\beta : \kappa] \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} } \]

Similar to the previous case.

- Case
  \[ \frac{\Gamma[\beta : \kappa] \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} }{\Gamma \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} } \]

Immediate

- Case
  \[ \frac{\Gamma[\beta : \kappa] \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} }{\Gamma \vdash \Delta := \beta : \kappa \vdash \Delta \text{ InstReach} } \]

By definition of unsolved.

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

Proof. By induction on the given derivation.

- Case
  \[ \frac{\Gamma \vdash u \doteq u : \kappa \vdash \Gamma \text{ CheckeqVar} }{\Delta } \]

Here \( \Delta = \Gamma \).

- Cases CheckeqUnit, CheckeqZero Similar to the CheckeqVar case.

- Case
  \[ \frac{\Gamma \vdash s \doteq t : \kappa \vdash \Delta \text{ CheckeqSucc} }{\Gamma \vdash \text{succ}(s) \doteq \text{succ}(t) : \kappa \vdash \Delta } \]

Follows by i.h.
Proof of Lemma 75 (Checkeq Solving)

If \( \Gamma \vdash Q \vdash \Delta \) then either \( \Delta = \Gamma \) or \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \).

Proof. Only one rule can derive the judgment:

- Case
  \[ \Gamma \vdash \sigma_1 : t_1 : \nu \vdash \Delta \quad \Theta \vdash [\Theta]t_2 : [\Theta][\nu] : \nu \vdash \Delta \]

  By Lemma 75 (Checkeq Solving) on the first premise, either \( \Theta = \Gamma \) or \( |\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)| \).

  In the former case, the result follows from Lemma 75 (Checkeq Solving) on the second premise.

  In the latter case, applying Lemma 75 (Checkeq Solving) to the second premise either gives \( \Delta = \Theta \), and therefore

  \[ |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \]

  or gives \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)| \), which also leads to \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \).

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. By induction on the given derivation.

- Case
  \[ \Gamma \vdash \alpha \equiv \alpha' \vdash \Gamma \]

  Here \( \Delta = \Gamma \).
Proof of Lemma 77 (Equiv Solving)

• Cases \(\equiv \text{Exvar} \equiv \text{Unit} \) Similar to the \(\equiv \text{Var} \) case.

• Case

\[
\begin{align*}
\Gamma \vdash A_1 &\equiv B_1 \vdash \Theta \\
\Theta \vdash [\Theta]A_2 &\equiv [\Theta]B_2 \vdash \Delta
\end{align*}
\]

\[\Gamma \vdash A_1 \uplus A_2 \equiv B_1 \uplus B_2 \vdash \Delta \]

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

\[
\begin{align*}
\Gamma_\alpha &\vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta' \\
\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \vdash \Delta \equiv \land
\end{align*}
\]

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 \(\Delta' = \cdot\). 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:

\[|\text{unsolved}(\Delta)| + 0 + |\text{unsolved}(\Delta')| < |\text{unsolved}(\Gamma)| + 0\]

Since \(|\text{unsolved}(\Delta')|\) cannot be negative, we have \(|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|\).

• Case

\[
\begin{align*}
\Gamma &\vdash P \equiv Q \vdash \Theta \\
\Theta &\vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta
\end{align*}
\]

Similar to the \(\equiv \Theta\) case, but using Lemma 76 (Prop Equiv Solving) on the first premise instead of the i.h.

• Case

\[
\begin{align*}
\Gamma &\vdash P \equiv Q \vdash \Theta \\
\Theta &\vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta
\end{align*}
\]

Similar to the \(\equiv \Delta\) case.

• Case

\[
\begin{align*}
\Gamma_0[\bar{\alpha}] &\vdash \& := \tau : \star \vdash \Delta \\
\& &\notin \text{FV}(\tau)
\end{align*}
\]

\[
\frac{\Gamma_0[\bar{\alpha}] \vdash \& \equiv \tau \vdash \Delta \quad \text{InstantiateL}}{\Gamma_0[\bar{\alpha}] \vdash \tau \equiv \Delta}
\]

By Lemma 74 (Instantiation Solves), \(|\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| - 1\).

• Case

\[
\begin{align*}
\Gamma_0[\bar{\alpha}] &\vdash \& := \tau : \star \vdash \Delta \\
\& &\notin \text{FV}(\tau)
\end{align*}
\]

\[
\frac{\Gamma_0[\bar{\alpha}] \vdash \tau \equiv \Delta \quad \text{InstantiateR}}{\Gamma_0[\bar{\alpha}] \vdash \& \equiv \tau \vdash \Delta}
\]

Similar to the \(\equiv \text{InstantiateL}\) case. \(\square\)

Lemma 78 (Decidability of Propositional Judgments).
The following judgments are decidable, with \(\Delta \) as output in (1)–(3), and \(\Delta^+ \) 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) \(\Gamma \vdash P \vdash \Delta\)

(3) \(\Gamma \vdash P \equiv Q \vdash \Delta\)

(4) \(\Gamma / \sigma \equiv t : \kappa \vdash \Delta^+\)
(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 $\text{ElimeqUvarRef}$, $\text{ElimeqZero}$: No premises, and the output is the same as the input.
- Case $\text{ElimeqBin}$: The only premise is the clash judgment, which is clearly decidable. There is no output.
- Case $\text{ElimeqBinBot}$: The premise is invoked on subterms, and does not yield an output context.
- Case $\text{ElimeqSucc}$: Both $\sigma$ and $t$ get smaller. By i.h., the output context has fewer unsolved variables, if it exists.
- Cases $\text{ElimeqInstL}$, $\text{ElimeqInstR}$: Follows from Lemma 67 (Decidability of Instantiation). Furthermore, by Lemma 74 (Instantiation Solves), instantiation solves a variable in the output.
- Cases $\text{ElimeqUvarL}$, $\text{ElimeqUvarR}$: These rules have no nontrivial premises, and $\alpha$ is solved in the output context.
- Cases $\text{ElimeqUvarL}$, $\text{ElimeqUvarR}$: These rules have no nontrivial premises, and produce the output context $\bot$.

(5) Decidability of $\Gamma / P \vdash \Delta^\perp$: By induction on $P$. But we have only one rule deriving this judgment form, $\text{ElimeqpropEq}$, 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**  
    \[ \text{Proof of Lemma 79 (Decidability of Equivalence)} \]
    \[ \Gamma \vdash A_1 \equiv B_1 \rightarrow \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \rightarrow \Delta \]
    \[ \Gamma \vdash A_1 \oplus A_2 \equiv B_1 \oplus B_2 \rightarrow \Delta \]
    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**  
    \[ \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \rightarrow \Delta, \alpha : \kappa, \Delta' \]
    \[ \Gamma \vdash \forall \alpha : \kappa, A_0 \equiv B_0 \rightarrow \Delta \]
    Since \(#\text{large}(A_0) + \text{large}(B_0) = \#\text{large}(A) + \text{large}(B) - 2\), the first part of the measure gets smaller.
  - **Case**  
    \[ \Gamma \vdash P \equiv Q \rightarrow \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \]
    \[ \Gamma \vdash P \equiv A_0 \equiv Q \equiv B_0 \rightarrow \Delta \]
    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**  
    \[ \Gamma \vdash P \equiv Q \rightarrow \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \]
    \[ \Gamma \vdash A_0 \land P \equiv B_0 \land Q \rightarrow \Delta \]
    Similar to the \(\equiv\) case.
  - **Case**  
    \[ \Gamma[\alpha] \vdash \alpha : \tau \rightarrow \Delta \quad \alpha \notin \text{FV}(\tau) \]
    \[ \Gamma[\alpha] \vdash \alpha \equiv \tau \rightarrow \Delta \]
    Follows from Lemma 67 (Decidability of Instantiation).
  - **Case**  
    Similar to the \(\equiv\) case.

### H’.2 Decidability of Subtyping

**Theorem 1** (Decidability of Subtyping).

*Given a context \(\Gamma\) and types \(A, B\) such that \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type and \([\Gamma]A = A\) and \([\Gamma]B = B\), it is decidable whether there exists \(\Delta\) such that \(\Gamma \vdash A \prec B \rightarrow \Delta\).*

*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 \(\Gamma \vdash A \prec B \rightarrow \Delta\).
Proof of Theorem 1

(Decidability of Subtyping)

\[ \text{thm:subtyping-decidable} \]

- **Case** A not headed by $\forall/\exists$
  \[ B \text{ not headed by } \forall/\exists \; \Gamma \vdash A \equiv B \vdash \Delta \]
  \[ \Gamma \vdash A < : - B \vdash \Delta \quad \text{\textsc{:equiv}} \]
  In this case, we appeal to Lemma 79 (Decidability of Equivalence).

- **Case** B not headed by $\forall$
  \[ \Gamma \vdash \Delta, \alpha : \kappa \vdash [\alpha/\alpha]A < : - B \vdash \Delta, \alpha, \Theta \]
  \[ \Gamma \vdash \forall\alpha : \kappa, A < : - B \vdash \Delta \quad \text{\textsc{:vl}} \]
  The premise has one fewer quantifier.

- **Case** $\Gamma, \alpha : \kappa \vdash A < : - B \vdash \Delta, \alpha : \kappa, \Theta$
  \[ \Gamma \vdash A < : + B \vdash \Delta \quad \text{\textsc{:el}} \]
  The premise has one fewer quantifier.

- **Case** A not headed by $\exists$
  \[ \Gamma \vdash \Delta, \beta : \kappa \vdash [\beta/\beta]B < : + \vdash \Delta, \beta, \Theta \]
  \[ \Gamma \vdash A < : + \exists \beta : \kappa, B \vdash \Delta \quad \text{\textsc{:el}} \]
  The premise has one fewer quantifier.

- **Case** neg(A)
  \[ \text{nonpos}(B) \]
  \[ \Gamma \vdash A < : - B \vdash \Delta \quad \text{\textsc{:<L}} \]
  Consider whether B is negative.

  - **Case** neg(B):
    \[ B = \forall\beta : \kappa, B' \quad \text{Definition of neg(B)} \]
    \[ \Gamma, \beta : \kappa \vdash A < : - B' \vdash \Delta, \beta : \kappa, \Theta \quad \text{Inversion on the premise} \]
    There is one fewer quantifier in the subderivation.

  - **Case** nonneg(B):
    In this case, B is not headed by a $\forall$.
    \[ A = \forall\alpha : \kappa, A' \quad \text{Definition of neg(A)} \]
    \[ \Gamma \vdash \Delta, \alpha : \kappa \vdash [\alpha/\alpha]A' < : - \vdash \Delta, \alpha, \Theta \quad \text{Inversion on the premise} \]
    There is one fewer quantifier in the subderivation.

- **Case** nonpos(A)
  \[ \text{neg}(B) \]
  \[ \Gamma \vdash A < : - B \vdash \Delta \quad \text{\textsc{:<R}} \]
  \[ B = \forall\beta : \kappa, B' \quad \text{Definition of neg(B)} \]
  \[ \Gamma, \beta : \kappa \vdash A < : - B' \vdash \Delta, \beta : \kappa, \Theta \quad \text{Inversion on the premise} \]
  There is one fewer quantifier in the subderivation.

- **Case** pos(A)
  \[ \text{nonneg}(B) \]
  \[ \Gamma \vdash A < : + B \vdash \Delta \quad \text{\textsc{:<L}} \]
  This case is similar to the \textsc{:<R} case.
Proof of Theorem 1 (Decidability of Subtyping)

\[ \text{Case } \Gamma \vdash A <_B \Gamma \vdash +B \vdash \Delta \quad \text{nonneg}(A) \]
\[ \Gamma \vdash A <_B \Gamma \vdash -B \vdash \Delta \quad \text{pos}(B) \]

This case is similar to the \(<_B \Gamma \text{ case.}\)

H’.3 Decidability of Matching and Coverage

Lemma 80 (Decidability of Expansion Judgments).

Given branches \(\Pi\), it is decidable whether:

1. there exists \(\Pi'\) such that \(\Pi \nsim \Pi'\);
2. there exist \(\Pi_L\) and \(\Pi_R\) such that \(\Pi \nsim \Pi_L \parallel \Pi_R\);
3. there exists \(\Pi'\) such that \(\Pi \nsim \Pi'\);
4. there exists \(\Pi'\) such that \(\Pi \nsim \Pi'\).

Proof. In each part, by induction on \(\Pi\): Every rule either has no premises, or breaks down \(\Pi\) in its nontrivial premise.

Theorem 2 (Decidability of Coverage).

Given a context \(\Gamma\), branches \(\Pi\) and types \(\vec{A}\), it is decidable whether \(\Gamma \vdash \Pi\) covers \(\vec{A}\) is derivable.

Proof. By induction on, lexicographically, (1) the number of \(\land\) connectives appearing in \(\vec{A}\), and then (2) the size of \(\vec{A}\), considered to be the sum of the sizes \(|A|\) of each type \(A\) in \(\vec{A}\).

(For \(\text{CoversVar}\), \(\text{Covers}_\times\) and \(\text{Covers}_+\) we also use the appropriate part of Lemma 80 (Decidability of Expansion Judgments).)

• Case \(\text{CoversEmpty}\): No premises.
• Case \(\text{CoversVar}\): The number of \(\land\) connectives does not grow, and \(\vec{A}\) gets smaller.
• Case \(\text{Covers}_1\): The number of \(\land\) connectives does not grow, and \(\vec{A}\) gets smaller.
• Case \(\text{Covers}_\times\): The number of \(\land\) connectives does not grow, and \(\vec{A}\) gets smaller, since \(|A_1| + |A_2| < |A_1 \times A_2|\).
• Case \(\text{Covers}_+\): Here we have \(\vec{A} = (A_1 + A_2, B)\). In the first premise, we have \((A_1, B)\), which is smaller than \(\vec{A}\), and in the second premise we have \((A_2, B)\), which is likewise smaller. (In both premises, the number of \(\land\) connectives does not grow.)
• Case \(\text{Covers}_\exists\): The number of \(\land\) connectives does not grow, and \(\vec{A}\) gets smaller.
• Case \(\text{CoversEq}\): The first premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (4). The number of \(\land\) connectives in \(\vec{A}\) gets smaller (note that applying \(\Delta\) as a substitution cannot add \(\land\) connectives).
• Case \(\text{CoversEqBot}\): Decidable by Lemma 78 (Decidability of Propositional Judgments) (4). □

H’.4 Decidability of Typing

Theorem 3 (Decidability of Typing).

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

(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
\[ \Gamma \vdash s : A p \gg B q \vdash \Delta. \]
(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

$\Gamma \vdash e \iff B \ p \vdash \Delta$.

(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 $\Gamma \vdash \Pi :: \vec{A} \iff C \ p \vdash \Delta$.

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

Proof. For rules deriving judgments of the form

\[
\begin{align*}
\Gamma \vdash e & \Rightarrow - - - - \dashv \dashv \\
\Gamma \vdash e & \iff B \ p \vdash - - \\
\Gamma \vdash s : B \ p & \Rightarrow - - - - \\
\Gamma \vdash \Pi :: \vec{A} & \iff C \ p \vdash -
\end{align*}
\]

(where we write “-” for parts of the judgments that are outputs), the following induction measure on such judgments is adequate to prove decidability:

\[
\langle e/s/\Pi, \Rightarrow \iff / \Rightarrow / \dashv, \ #\text{large}(B), \ B \ \text{Match}, \ \vec{A}, \ \text{match judgment form} \rangle
\]

where $(\ldots)$ denotes lexicographic order, and where (when comparing two judgments typing terms of the same size) the synthesis judgment (top line) is considered smaller than the checking judgment (second line). That is,

\[
\Rightarrow \prec \iff / \Rightarrow / \text{Match}
\]

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 $\text{Case}$ 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.

- **Case $\rightarrow$Spine**: In each premise, the expression/spine gets smaller (we have $e \cdot s$ 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 (Decidability of Subtyping) is decidable.

- **Case Anno**: It is easy to show that the judgment $\Gamma \vdash A$ type is decidable. The second premise types $e$, but the conclusion types $(e : A)$, so the first part of the measure gets smaller.

- **Cases 1I, 1I $^\alpha$**: No premises.

- **Case $\forall$I**: Both the premise and conclusion type $e$, and both are checking; however, $\#\text{large}(A_0) < \#\text{large}(\forall A : \kappa. A_0)$, so the premise is smaller.

- **Case $\top$Spine**: Both the premise and conclusion type $e \cdot s$, and both are spine judgments; however, $\#\text{large}(\cdot -)$ decreases.

- **Case $\forall$A**: By Lemma 78 (Decidability of Propositional Judgments) (2), the first premise is decidable. For the second premise, $\#\text{large}(\emptyset A_0) = \#\text{large}(A_0) < \#\text{large}(A_0 \land P)$. 

Proof of Theorem 3 (Decidability of Typing) thm:typing-decidable
• Case $\llbracket 1 \rrbracket$ For the first premise, use Lemma 78 (Decidability of Propositional Judgments) (5). In the second premise, $\# \text{large}(-)$ gets smaller (similar to the $\land$ case).

• Case $\llbracket 11 \rrbracket$ The premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (5).

• Case $\llbracket \text{Spine} \rrbracket$ Similar to the $\land$ case.

• Cases $\llbracket \bot \rrbracket$ In the premise, the term is smaller.

• Cases $\llbracket E \rightarrow E' \rrbracket$ In all premises, the term is smaller.

• Cases $\llbracket + 1 \rrbracket$ In all premises, the term is smaller.

• Case $\llbracket \llbracket \llbracket \rrbracket \rrbracket$ 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 (Decidability of Coverage).

We now consider the match rules:

• Case $\llbracket \text{MatchEmpty} \rrbracket$ No premises.

• Case $\llbracket \text{MatchSeg} \rrbracket$ 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 $\llbracket \text{MatchBase} \rrbracket$ The term $\epsilon$ in the premise is properly contained in $\Pi$.

• Cases $\llbracket \text{Match} \rrbracket = [\text{Match} \times, \text{Match} +, \text{MatchNeg}, \text{MatchWild}]$ Smaller by part (2) of the measure.

• Case $\llbracket \text{Match} \land \rrbracket$ 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 $\llbracket \text{Match} \llbracket \rrbracket$ For the premise, use Lemma 78 (Decidability of Propositional Judgments) (4).

• Case $\llbracket \text{MatchUnify} \rrbracket$

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 81 (Determinacy of Auxiliary Judgments).

(1) Elimeq: Given $\Gamma, \sigma, t, \kappa$ such that $\text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset$ and $D_1 : \Gamma \vdash \sigma = t : \kappa \vdash \Delta_1$ and $D_2 : \Gamma \vdash \sigma = t : \kappa \vdash \Delta_2^+$, it is the case that $\Delta_1^+ = \Delta_2^+$.

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

(3) Symmetric instantiation: Given $\Gamma, \bar{\alpha}, \bar{\beta}, \kappa$ such that $\bar{\alpha}, \bar{\beta} \in \text{unsolved}(\Gamma)$ and $\bar{\alpha} \neq \bar{\beta}$ and $D_1 : \Gamma \vdash \bar{\beta} := \bar{\alpha} : \kappa \vdash \Delta_1$ and $D_2 : \Gamma \vdash \bar{\beta} := \bar{\alpha} : \kappa \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

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

(5) Elimprop: Given $\Gamma, \text{P}$ such that $D_1 : \Gamma \vdash \text{P} \vdash \Delta_1^+$ and $D_2 : \Gamma \vdash \text{P} \vdash \Delta_2^+$, it is the case that $\Delta_1 = \Delta_2$.

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

Proof of Lemma 81 (Determinacy of Auxiliary Judgments) (lem:aux-det)
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 ElimeqUvarR applies if and only if \( t = \alpha \). (Rule ElimeqClash cannot apply; rules ElimeqUvarL and ElimeqUvarR have a free variable condition; rules ElimeqUvarL1 and ElimeqUvarR1 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 ElimeqUvarR1 applies. The conclusion matches if we have \( t = \beta \) for some \( \beta \neq \alpha \) (that is, \( \sigma = \alpha \) and \( t = \beta \)). But ElimeqUvarR1 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 \( \alpha \) in the given context:
- If \( \beta \) is declared to the left of \( \alpha \), then rule InstReach cannot be used, which leaves only InstSolve.
- If \( \beta \) is declared to the right of \( \alpha \), then InstSolve cannot be used because \( \beta \) is not well-formed under \( \Gamma_0 \) (the context to the left of \( \alpha \) in InstSolve). That leaves only InstReach.
- \( \alpha \) cannot be \( \beta \), because it is given that \( \alpha \notin \text{FV}(t) = \text{FV}(\beta) = (\bar{\beta}) \).

Proof of Part (3) (Symmetric instantiation).
Rule InstBin and InstSucc cannot have been used in either derivation.
Suppose that InstSolve concluded \( D_1 \). Then \( \Delta_1 \) is the same as \( \Gamma \) with \( \alpha \) solved to \( \beta \). Moreover, \( \beta \) is declared to the left of \( \alpha \) in \( \Gamma \). Thus, InstSolve cannot conclude \( D_2 \). However, InstReach can conclude \( D_2 \), but produces a context \( \Delta_2 \) which is the same as \( \Gamma \) but with \( \alpha \) solved to \( \beta \). Therefore \( \Delta_1 = \Delta_2 \).
The other possibility is that InstReach concluded \( D_1 \). Then \( \Delta_1 \) is the same as \( \Gamma \) with \( \beta \) solved to \( \alpha \), with \( \alpha \) declared to the left of \( \beta \) in \( \Gamma \). Thus, InstReach cannot conclude \( D_2 \). However, InstSolve can conclude \( D_2 \), producing a context \( \Delta_2 \) which is the same as \( \Gamma \) but with \( \beta \) solved to \( \alpha \). Therefore \( \Delta_1 = \Delta_2 \).

Proof of Part (4) (Checkeq).
Rule CheckeqVar applies if and only if \( \sigma = t = \bar{\alpha} \) or \( \sigma = t = \alpha \) (note the free variable conditions in CheckeqInstL and CheckeqInstR).
Rule CheckeqUnit applies if and only if \( \sigma = t = 1 \).
Rule CheckeqBin applies if and only if \( \sigma \) and \( t \) are both headed by the same binary connective.
Rule CheckeqZero applies if and only if \( \sigma = t = \text{zero} \).
Rule CheckeqSucc applies if and only if \( \sigma \) and \( t \) are headed by succ.
Now suppose \( \sigma = \bar{\alpha} \). If \( t \) is not an existential variable, then CheckeqInstR cannot be used, which leaves only CheckeqInstL. If \( t \) is an existential variable, that is, some \( \bar{\beta} \) (distinct from \( \bar{\alpha} \)), and is unsolved, then both CheckeqInstL and CheckeqInstR apply, but by part (3), we get the same output context from each.
The \( t = \bar{\alpha} \) 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 82 (Determinacy of Equivalence).

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

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

Proof. Proof of Part (1) (propositional equivalence). Only one rule derives judgments of this form; the result follows from Lemma 81 (Determinacy of Auxiliary Judgments) (4).

Proof of Part (2) (type equivalence). If neither $A$ nor $B$ is an existential variable, they must have the same head connectives, and the same rule must conclude both derivations.

If $A$ and $B$ are the same existential variable, then only $\equiv\text{Exvar}$ applies (due to the free variable conditions in $\equiv\text{Instantiate}_L$ and $\equiv\text{Instantiate}_R$).

If $A$ and $B$ are different unsolved existential variables, the judgment matches the conclusion of both $\equiv\text{Instantiate}_L$ and $\equiv\text{Instantiate}_R$ but by part (3) of Lemma 81 (Determinacy of Auxiliary Judgments), we get the same output context regardless of which rule we choose. □

Theorem 4 (Determinacy of Subtyping).

(1) Subtyping: Given $\Gamma, e, A, B$ such that $D_1 :: \Gamma \vdash A <: \pm B \vdash \Delta_1$ and $D_2 :: \Gamma \vdash A <: \pm B \vdash \Delta_2$, 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 $\Gamma \vdash A <: + B \vdash \Delta_1$ and $\Gamma \vdash A <: - B \vdash \Delta_2$, then we know the last rule ending the derivation of $D_1$ and $D_2$ must be:

\[ \begin{array}{c|c|c|c} 
 & \forall & \exists & \text{other} \\
\hline 
\forall & <: R & <: L & <: \text{Equiv} \\
\exists & <: L & <: L & <: L \\
\text{other} & <: R & <: \exists R & <: \text{Equiv} \\
\end{array} \]

The only case in which there are two possible final rules is in the $\forall/\forall$ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A <: - B \vdash \Delta_1$ and $\Gamma \vdash A <: - B \vdash \Delta_2$.

- If $\Gamma \vdash A <: - B \vdash \Delta_1$ and $\Gamma \vdash A <: + B \vdash \Delta_2$, then we know the last rule ending the derivation of $D_1$ and $D_2$ must be:

\[ \begin{array}{c|c|c|c} 
 & \forall & \exists & \text{other} \\
\hline 
\forall & <: R & <: L & <: \text{L} \\
\exists & <: \forall R & <: \forall L & <: \forall L \\
\text{other} & <: R & <: R & <: \text{Equiv} \\
\end{array} \]

The only case in which there are two possible final rules is in the $\forall/\forall$ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A <: + B \vdash \Delta_1$ and $\Gamma \vdash A <: + B \vdash \Delta_2$.

As a result, the result follows by a routine induction. □

Theorem 5 (Determinacy of Typing).
Proof of Part (1) (checking).

Proof.

Only four rules have a synthesis judgment in the conclusion:

\[ \begin{align*}
& \text{I} \\
& \text{⊃} \\
& \text{E} \\
& \text{α}
\end{align*} \]

and

\[ \begin{align*}
& \text{I} \\
& \text{⊃} \\
& \text{E} \\
& \text{α}
\end{align*} \]

\[ \text{∀} \]

\[ \text{ incur } \]

\[ \text{Ann} \]

\[ \text{E} \]

\[ \text{α} \]

\[ \text{Anno} \]

\[ \text{E} \]

\[ \text{α} \]

\[ \text{Var} \]

\[ \text{Anno} \]

\[ \text{E} \]

\[ \text{α} \]

Note 1:

The table below shows which rules apply for given e and A. The extra “chk-I?” column highlights the role of the “chk-I” ("check-intro") category of syntactic forms: we restrict the introduction rules for \( \lor \) and \( \to \) to type only these forms. For example, given \( e = x \) and \( A = \forall \alpha : k. A_o \), we need not choose between \( \text{Sub} \) and \( \text{Var} \) the latter is ruled out by its \( \text{chk-I} \) premise.

\[
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
\text{chk-I?} & \forall & \lor & \exists & \land & \to & \times & 1 & \vec{α} & α \\
\hline
\lambda x, e_0 & \text{chk-I} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
\in_{\alpha k}, e_0 & \text{chk-I} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
\langle e_1, e_2 \rangle & \text{chk-I} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
\text{case} (e_0, π) & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
x & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
(e_0 : A) & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
e_1 e_2 & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} & \text{Can} \\
\hline
\end{array}
\]

Notes:

- **Note 1**: The choice between \( \lor \) and \( \land \) is resolved by Lemma 81 (Determinacy of Auxiliary Judgments) (5).

- **Note 2**: Case expressions are a checking form, but not an introduction form. So if e is a case expression, we need not choose between an introduction rule for a large connective and the \( \text{Case} \) rule: only the \( \text{Case} \) rule is viable. Large connectives must, therefore, be introduced inside the branches.

Proof of Part (2) (synthesis). Only four rules have a synthesis judgment in the conclusion: \( \text{Var} \), \( \text{Anno} \), \( \text{E} \), \( \text{α} \). 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 \( \langle e_0 : A \rangle \).

Otherwise, the judgment can be derived only if e has the form \( e_1 e_2 \), by \( \text{E} \) or \( \text{E} \). If \( D_1 \) and \( D_2 \) both end in \( \text{E} \) or \( \text{E} \), we are done. Suppose \( D_1 \) ends in \( \text{E} \) and \( D_2 \) ends in \( \text{E} \). By i.h., the p in the first subderivation of \( \text{E} \) must be equal to the one in the first subderivation of \( \text{E} \) that is, \( p = 1 \). Thus the inputs to the respective second subderivations match, so by i.h. their outputs match; in particular, \( q = 1 \). However, from the condition in \( \text{E} \) it must be the case that FEV(\( \langle \Delta \rangle C \)).
Proof of Part (3) (spine judgments). For the ordinary spine judgment, rule 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: $→\overrightarrow{\text{Spine}}, \lor \overrightarrow{\text{Spine}}, \Rightarrow \overrightarrow{\text{Spine}}$, or $\& \overrightarrow{\text{Spine}}$.

For the principality-recovering spine judgment: If $p = f$, only rule SpinePass applies. If $p = 1$ and $q = 1$, only rule SpinePass applies. If $p = 1$ and $q = f$, then the rule is determined by FEV(C): if $\text{FEV}(C) = 0$ then only SpinePass applies; otherwise, $\text{FEV}(C) \neq 0$ and only SpinePass applies.

Proof of Part (4) (matching). First, the elimination judgment form $\Gamma \vdash \Delta \Rightarrow \ldots$. It cannot be the case that both $\Gamma \vdash \sigma \triangleq t : \kappa \cdot \perp$ and $\Gamma \vdash \sigma \triangleq t : \kappa \cdot \Theta$, so either $\text{Match}(\perp)$ concludes both $\Delta_1$ and $\Delta_2$ (and the result follows), or $\text{MatchUnify}$ concludes both $\Delta_1$ and $\Delta_2$ (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 $\Lambda$ has length zero. So in the rest of this part, we assume the length of $\Lambda$ is at least one.

Let $A$ be the first type in $\Lambda$. Inspection of the rules shows that given particular $A$ and $\rho$, where $\rho$ is the first pattern, only a single rule can apply, or no rule (“\(\emptyset\)”) can apply, as shown in the following table:

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>$\text{Match}_\setminus$</th>
<th>$\text{Match}^\setminus$</th>
<th>$\text{Match}_{+}$</th>
<th>$\text{Match}_\times$</th>
<th>$\text{other}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{inj}_k\rho_0$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\rho \mid (\rho_1, \rho_2)$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\emptyset$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$z$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\text{MatchNeg}$</td>
<td>$\text{MatchNeg}$</td>
<td>$\text{MatchWild}$</td>
</tr>
<tr>
<td>$-$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\text{Match}=\text{Match}^\setminus$</td>
<td>$\text{MatchWild}$</td>
<td>$\text{MatchWild}$</td>
<td>$\text{MatchWild}$</td>
</tr>
</tbody>
</table>

\[\boxed{}\]

J’ Properties of Algorithmic Subtyping

K’ Soundness

K’1 Instantiation

Lemma 83 (Soundness of Instantiation).

If $\Gamma \vdash \Delta := \tau : \kappa \cdot \perp$ and $\Delta \notin \text{FV}(\Delta)$ and $\Gamma \vdash \tau = \tau$ and $\Delta \longrightarrow \Omega$ then $|\Delta|\Delta = [\Omega]\Delta$.

Proof. By induction on the derivation of $\Gamma \vdash \Delta := \tau : \kappa \cdot \perp$.

- Case $\Gamma_0 \vdash \tau : \kappa$

  $\Gamma_0, \Delta : \kappa, \Gamma_1 \vdash \Delta := \tau : \kappa \not\vdash \Gamma_0, \Delta : \kappa = \tau, \Gamma_1$.

  $|\Delta|\Delta = |\Delta|\Delta$ By definition

  $\boxed{[\Omega]|\Delta|\Delta = [\Omega]|\Delta|\Delta}$ By Lemma 29 (Substitution Monotonicity) to each side

- Case $\Gamma_0[\Delta : k][\beta : \kappa] \vdash \Delta := [\beta : \kappa \not\vdash \Gamma_0[\Delta : k][\beta : \kappa]]$

  $|\Delta|\beta = |\Delta|\Delta$ By definition

  $|\Omega|\Delta|\beta = |\Omega|\Delta|\Delta$ Applying $\Omega$ to each side

  $\boxed{[\Omega]|\Delta|\beta = [\Omega]|\Delta|\Omega}$ By Lemma 29 (Substitution Monotonicity) to each side

- Case $\Gamma \vdash \Delta := \tau_1 : \tau_2 : \star \cdot \perp$

  $\Gamma_1[\Delta_2 : \star, \Delta_1 : \star, \Delta : \star := \Delta_1 \oplus \Delta_2] \vdash \Delta_1 := \tau_1 : \star \not\vdash \Theta \vdash \Delta_2 := [\Theta]|\tau_2 : \star \cdot \perp$

  $\Gamma_0[\Delta : \star] \vdash \Delta := \tau_1 \oplus \tau_2 : \star \cdot \perp$.

\[\boxed{}\]
Proof of Lemma 83 (Soundness of Instantiation)

\[ \Delta \rightarrow \Omega \]
\[ \Gamma' \vdash \alpha_1 := \tau_1 : \star \rightarrow \Theta \]
\[ \Theta \rightarrow \Delta \]
\[ \Theta \rightarrow \Omega \]
\[ [\Omega]\hat{\alpha}_1 = [\Omega]\tau_1 \]
\[ \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \rightarrow \Delta \]
\[ [\Omega]\hat{\alpha}_2 = [\Omega][\Theta]\tau_2 \]
\[ = [\Omega]\tau_2 \]
\[ ([\Omega]\tau_1) \oplus ([\Omega]\tau_2) = ([\Omega][\hat{\alpha}_1] \oplus ([\Omega][\hat{\alpha}_2]) \]
\[ = [\Omega][\hat{\alpha}_1 \oplus \hat{\alpha}_2]) \]
\[ = [\Omega][\Gamma']\hat{\alpha}) \]
\[ = [\Omega]\hat{\alpha} \]
\[ \Rightarrow [\Omega] (\tau_1 \oplus \tau_2) = [\Omega] \hat{\alpha} \]

- Case
  \[ \Gamma_0[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{zero} : N \rightarrow \Gamma_0[\hat{\alpha} : N = \text{zero}] \]
  Similar to the \text{InstZero} case.

- Case
  \[ \Gamma_0[\hat{\alpha}_1 : N, \hat{\alpha} : N = \text{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : N \rightarrow \Delta \]
  \[ \Gamma_0[\hat{\alpha} : N] \vdash \hat{\alpha} := \text{succ}(t_1) : N \rightarrow \Delta \]
  Similar to the \text{InstBin} case, but simpler.

\[ \text{Lemma 84 (Soundness of Checkeq)} \]

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

\[ \text{Proof.} \] By induction on the given derivation.

- Case
  \[ \Gamma \vdash u \triangleq u : \kappa \rightarrow \Gamma \]
  \[ [\Omega] u = [\Omega] u \]
  By reflexivity of equality

- Cases \text{CheckeqZero} \text{CheckeqUnit} \text{CheckeqSucc} \text{CheckeqBin} Similar to the \text{CheckeqVar} case.

- Case
  \[ \Gamma \vdash \sigma_0 \triangleq t_0 : N \rightarrow \Delta \]
  \[ \Gamma \vdash \text{succ}(\sigma_0) \triangleq \text{succ}(t_0) : N \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \triangleq t_0 : N \rightarrow \Delta \]
  By 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

- Case
  \[ \Gamma \vdash \sigma_0 \triangleq t_0 : \star \rightarrow \Theta \]
  \[ \Theta \vdash \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq [\Theta] t_1 : \star \rightarrow \Delta \]
  \[ \Gamma \vdash \sigma_0 \oplus \sigma_1 \triangleq t_0 \oplus t_1 : \star \rightarrow \Delta \]
Proof of Lemma 84 (Soundness of Checkeq)

\[ \Gamma \vdash \sigma_0 \equiv t_0 : \mathbb{N} \rightarrow \Delta \]

Subderivation

\[ \Theta \vdash [\Theta] \sigma_1 \equiv [\Theta] t_1 : \star \rightarrow \Delta \]

Subderivation

\[ \Delta \rightarrow \Gamma \]

Given

\[ \Theta \rightarrow \Delta \]

By Lemma 46 (Checkeq Extension)

\[ \Theta \rightarrow \Omega \]

By Lemma 33 (Extension Transitivity)

\[ [\Omega] \sigma_0 = [\Omega] t_0 \]

By i.h. on first subderivation

\[ [\Omega] [\Theta] \sigma_1 = [\Omega] [\Theta] t_1 \]

By i.h. on second subderivation

\[ [\Omega] [\Theta] \sigma_1 = [\Omega] \sigma_1 \]

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega] [\Theta] t_1 = [\Omega] t_1 \]

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega] \sigma_1 = [\Omega] t_1 \]

By transitivity of equality

\[ [\Omega] \sigma_0 \oplus [\Omega] \sigma_1 \equiv [\Omega] (t_0 \oplus t_1) \]

By congruence of equality

\[ \equiv [\Omega] (\sigma_0 + \sigma_1) \equiv [\Omega] (t_0 + t_1) \]

By definition of substitution

\[ \cdot \text{Case } \Gamma[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta \]

\[ \Gamma[\hat{\alpha}] \vdash \hat{\alpha} \equiv t : \kappa \rightarrow \Delta \]

Subderivation

\[ \hat{\alpha} \notin \text{FV}(t) \]

Premise

\[ \equiv [\Omega] \hat{\alpha} = [\Omega] t \]

By Lemma 83 (Soundness of Instantiation)

\[ \cdot \text{Case } \Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \sigma : \kappa \rightarrow \Delta \]

\[ \Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} \equiv \sigma : \kappa \rightarrow \Delta \]

Subderivation

\[ \hat{\alpha} \notin \text{FV}(t) \]

\[ \equiv [\Omega] \hat{\alpha} = [\Omega] t \]

By Lemma 83 (Soundness of Instantiation)

\[ \equiv \text{CheckeqInstL} \]

Similar to the CheckeqInstL case.

\[ \square \]

Lemma 85 (Soundness of Propositional Equivalence).

If \( \Gamma \vdash P \equiv Q \rightarrow \Delta \) where \( \Delta \rightarrow \Omega \) then \( [\Omega] P = [\Omega] Q \).

Proof. By induction on the given derivation.

\[ \cdot \text{Case } \Gamma \vdash \sigma_1 \equiv t_1 : \mathbb{N} \rightarrow \Theta \]

\[ \Theta \vdash [\Theta] \sigma_2 \equiv [\Theta] t_2 : \mathbb{N} \rightarrow \Delta \]

\[ \Gamma \vdash (\sigma_1 = \sigma_2) \equiv (t_1 = t_2) \rightarrow \Delta \]

\[ \equiv \text{PropEq} \]

\[ \Delta \rightarrow \Gamma \]

Given

\[ \Theta \rightarrow \Delta \]

By Lemma 46 (Checkeq Extension) (on 2nd premise)

\[ \Theta \rightarrow \Omega \]

By Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash \sigma_1 \equiv t_1 : \mathbb{N} \rightarrow \Theta \]

Given

\[ [\Omega] \sigma_1 \equiv [\Omega] t_1 \]

By Lemma 84 (Soundness of Checkeq)

\[ [\Omega] [\Theta] \sigma_2 \equiv [\Omega] [\Theta] t_2 \]

By Lemma 84 (Soundness of Checkeq)

\[ [\Omega] [\Theta] \sigma_2 \equiv [\Omega] \sigma_2 \]

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega] [\Theta] t_2 \equiv [\Omega] t_2 \]

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega] \sigma_2 \equiv [\Omega] t_2 \]

By transitivity of equality

\[ [\Omega] \sigma_1 \equiv [\Omega] \sigma_2 \equiv [\Omega] t_1 \equiv [\Omega] t_2 \]

By congruence of equality

\[ \equiv [\Omega] (\sigma_1 \oplus \sigma_2) \equiv [\Omega] (t_1 \oplus t_2) \]

By definition of substitution

\[ \square \]

Lemma 86 (Soundness of Algorithmic Equivalence).

If \( \Gamma \vdash \lambda \equiv \beta \rightarrow \Delta \) where \( \Delta \rightarrow \Omega \) then \( [\Omega] \lambda = [\Omega] \beta \).

Proof. By induction on the given derivation.

\[ \cdot \text{Case } \Gamma \vdash \alpha \equiv \alpha \rightarrow \Gamma \]

\[ \equiv \text{Var} \]

\[ \equiv [\Omega] \alpha = [\Omega] \alpha \]

By reflexivity of equality

\[ \equiv \text{Exvar} \equiv \text{Unit} \]

Similar to the \( \equiv \text{Var} \) case.
Proof of Lemma 86 (Soundness of Algorithmic Equivalence)

• Case

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

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

\[ \Delta \longrightarrow \Omega \]
Given

\[ \Theta \vdash [\Theta] A_2 \equiv [\Theta] B_2 \vdash \Delta \]
Subderivation

\[ \Theta \longrightarrow \Delta \]
By Lemma 49 (Equivalence Extension)

\[ \Theta \longrightarrow \Omega \]
By Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash A_1 \equiv B_1 \vdash \Theta \]
Subderivation

\[ [\Omega] A_1 = [\Omega] B_1 \]
By i.h.

\[ \Delta \longrightarrow \Omega \]
Given

\[ [\Omega] \Theta A_2 = [\Omega] [\Theta] B_2 \]
By i.h.

\[ [\Omega] A_2 = [\Omega] B_2 \]
By Lemma 29 (Substitution Monotonicity)

\[ ([\Omega] A_1) \oplus ([\Omega] A_2) = ([\Omega] B_1) \oplus ([\Omega] B_2) \]
By above equations

\[ [\Omega] \Delta \vdash [\Omega] (A_1 \oplus A_2) \leq [\Omega] (B_1 \oplus B_2) \]
By def. of substitution

• Case

\[ \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta' \]

\[ \Gamma \vdash \forall \alpha : \kappa . A_0 \equiv \forall \alpha : \kappa . B_0 \vdash \Delta \]

\[ \equiv \forall \]

\[ \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta' \]
Subderivation

\[ \Delta \longrightarrow \Omega \]
Given

\[ \Delta', \alpha : \kappa, \Delta' \longrightarrow \Omega, \alpha : \kappa, \Omega_Z \]
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 \( B_0 \)

\[ \Gamma, \alpha : \kappa \longrightarrow \Omega, \alpha : \kappa \]
By \( \longrightarrow \text{Uvar} \)

\[ FV(A_0) \subseteq \text{dom}(\Omega, \alpha : \kappa) \]
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

\[ \Gamma \vdash P \equiv Q \vdash \Theta \]
\[ \Theta \vdash [\Theta] A_0 \equiv [\Theta] B_0 \vdash \Delta \]

\[ \Gamma \vdash P \triangleright A_0 \equiv Q \triangleright B_0 \vdash \Delta \]

\[ \equiv \triangleright \]

\[ \Delta \longrightarrow \Omega \]
Given

\[ \Theta \vdash [\Theta] A_0 \equiv [\Theta] B_0 \vdash \Delta \]
Subderivation

\[ \Theta \longrightarrow \Delta \]
By Lemma 49 (Equivalence Extension)

\[ \Theta \longrightarrow \Omega \]
By Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash P \equiv Q \vdash \Theta \]
Subderivation

\[ [\Omega] P = [\Omega] Q \]
By Lemma 85 (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

\[ \Gamma \vdash P \equiv Q \vdash \Theta \]
\[ \Theta \vdash [\Theta] A_0 \equiv [\Theta] B_0 \vdash \Delta \]

\[ \Gamma \vdash A_0 \wedge P \equiv B_0 \wedge Q \vdash \Delta \]

\[ \equiv \wedge \]
Similar to the \( \equiv \) case.

- **Case** 
  \[
  \Gamma[\alpha] \vdash \alpha \implies \tau : \neg \Delta \quad \alpha \notin \text{FV}(\tau)
  \]
  \[
  \Gamma[\alpha] \vdash \alpha \equiv \tau \quad \Delta
  \]
  - **Instantiate**
  
  \[
  \Gamma[\alpha] \vdash \alpha \implies \tau : \neg \Delta
  \]
  - **Subderivation**

- **Case** 
  \[
  |\Omega|\alpha = |\Omega|\tau
  \]
  - **By Lemma 83 (Soundness of Instantiation)**

- **Case** 
  \[
  \Gamma / \sigma
  \]
  - **(Soundness of Algorithmic Equivalence)**

**K.2 Soundness of Checkprop**

**Lemma 87 (Soundness of Checkprop).**

If \( \Gamma \vdash P \) true \( \vdash \neg \Delta \) and \( \Delta \into \Omega \) then \( \Psi \vdash [\Omega]P \) true.

**Proof.** By induction on the derivation of \( \Gamma \vdash P \) true \( \vdash \neg \Delta \).

- **Case** 
  \[
  \Gamma \vdash \sigma \equiv t : \neg \Delta
  \]
  - **Checkprop**
  
  \[
  \Gamma \vdash \sigma = t \quad \Delta
  \]
  - **(Soundness of Checkprop)**

**K.3 Soundness of Eliminations (Equality and Proposition)**

**Lemma 88 (Soundness of Equality Elimination).**

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

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

2. If \( \Gamma / \sigma \equiv t : \kappa \vdash \neg \Delta \) and \( \neg \Delta \)
   then \( \text{mgu}(\sigma, t) = \emptyset \) (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 \( \sigma_2 = \text{mgu}(\sigma_2, t_2) \), then \( \sigma = \sigma_1 \circ \sigma_2 \).

(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 \( \Gamma / \sigma \equiv t : \kappa \vdash \neg \Delta \), proving both parts with a single induction.
Proof of Lemma 88: (Soundness of Equality Elimination)

\[ \text{lem:elimeq-soundness} \]

- Case
  \[ \Gamma \vdash \alpha \equiv \kappa : \kappa \vdash \Gamma \]
  \[ \text{ElimeqUvarRefl} \]
  Here we have \( \Delta = \Gamma \), so we are in part (1).
  Let \( \theta = \text{id} \) (which is \( \text{mgu}(\sigma, \sigma) \)).
  We can easily show \( \text{id}[\Omega] \alpha = [\Omega, \alpha] = [\Omega, \cdot] \alpha \).

- Case
  \[ \Gamma \vdash \text{zero} \equiv \text{zero} : \text{N} \vdash \Gamma \]
  Similar to the \[ \text{ElimeqUvarRefl} \] case.

- Case
  \[ \Gamma \vdash t_1 \equiv t_2 : \text{N} \vdash \Delta \]
  \[ \Gamma \vdash \text{succ}(t_1) \equiv \text{succ}(t_2) : \text{N} \vdash \Delta \]
  \[ \text{ElimeqSucc} \]
  We distinguish two subcases:
  - Case \( \Delta = \Delta \):
    Since we have the same output context in the conclusion and premise, the “for all \( t' \)…” part follows immediately from the i.h. (1).
    The i.h. also gives us \( \theta_0 = \text{mgu}(t_1, t_2) \).
    Let \( \theta = \theta_0 \). By property (ii), \( \text{mgu}(t_1, t_2) = \text{mgu}(\text{succ}(t_1), \text{succ}(t_2)) = \theta_0 \).
  - Case \( \Delta = \perp \):
    \[ \Gamma \vdash t_1 \equiv t_2 : \perp \]
    Subderivation
    \[ \text{mgu}(t_1, t_2) = \perp \]
    By i.h. (2)
    \[ \text{mgu}(\text{succ}(t_1), \text{succ}(t_2)) = \perp \]
    By contrapositive of property (ii)

- Case
  \[ \alpha \notin \text{FV}(t) \quad (\alpha = -) \notin \Gamma \]
  \[ \Gamma \vdash \alpha \equiv t : \kappa \vdash \Gamma, \alpha = t \]
  \[ \text{ElimeqUvarL} \]
  Here \( \Delta \neq \perp \), so we are in part (1).
  \[ [\Omega, \alpha = t] t' = [\Omega] t/\alpha] [\Omega] t' \]
  By a property of substitution
  \[ = [\Omega] t/\alpha] [\Omega] t' \]
  By a property of substitution
  \[ = [\Omega] [\theta] [\Omega] t' \]
  By \( \text{mgu}(\alpha, t) = (\alpha/t) \)
  \[ \theta \]
  By \( \text{mgu}(\alpha, t) = (\alpha/t) \)
  By a property of substitution (\( \theta \) creates no evars)

- Case
  \[ \alpha \notin \text{FV}(t) \quad (\alpha = -) \notin \Gamma \]
  \[ \Gamma \vdash t \equiv \alpha : \kappa \vdash \Gamma, \alpha = t \]
  \[ \text{ElimeqUvarR} \]
  Similar to the \[ \text{ElimeqUvarL} \] case.

- Case
  \[ \Gamma \vdash 1 \equiv 1 : * \vdash \Gamma \]
  \[ \text{ElimeqUnit} \]
  Similar to the \[ \text{ElimeqUvarRefl} \] case.

- Case
  \[ \Gamma \vdash \tau_1 \equiv \tau_1' : * \vdash \Theta \]
  \[ \Theta \vdash [\Theta] \tau_1 \equiv [\Theta] \tau_1' : * \vdash \Delta \]
  \[ \Gamma \vdash [\Theta] \tau_1 \equiv [\Theta] \tau_1': * \vdash \Delta \]
  \[ \text{ElimeqBin} \]
  Either \( \Delta \) is some \( \Delta \), or it is \( \perp \).
  - Case \( \Delta = \Delta \):

---

Proof of Lemma 88: (Soundness of Equality Elimination)
\textbf{Proof of Lemma 88 (Soundness of Equality Elimination)} \lem:elimeq-soundness

\begin{align*}
\Gamma & / \tau_1 \equiv \tau'_1 : \ast \vdash \Theta \quad \text{Subderivation} \\
\Theta &= (\Gamma, \Delta_1) \quad \text{By i.h. (1)} \\
(\text{IH-1st}) & [\Omega, \Delta_1]u_1 = [\theta_1][\Omega]u_1 \\
\theta_1 &= \text{mgu}(\tau_1, \tau'_1) \\
\Theta & / [\Theta]\tau_1 \equiv [\Theta]\tau'_1 : \ast \vdash \Delta \quad \text{Subderivation} \\
\Delta &= (\Theta, \Delta_2) \quad \text{By i.h. (1)} \\
(\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)
\end{align*}

Suppose \( \Omega \vdash u : \kappa' \).

\begin{align*}
[\Omega, \Delta_1, \Delta_2]u &= [\theta_2][\Omega, \Delta_1]u \\
&= [\theta_2][\theta_1][\Omega]u \\
&\triangleright \quad \text{By (IH-2nd), with } u_2 = u \\
\triangleright \theta_2 \circ \theta_1 &= \text{mgu}((\tau_1 \oplus \tau_2), (\tau'_1 \oplus \tau'_2)) \quad \text{By property (iii) of substitution}
\end{align*}

\begin{itemize}
  \item \textbf{Case} \( \Gamma / \tau_1 \equiv \tau'_1 : \ast \vdash \bot \)

    \begin{proof}
      \begin{align*}
      \Gamma & / \tau_1 \equiv \tau'_1 : \ast \vdash \bot \quad \text{ElimeqBinBot} \\
      \Gamma & / \tau_1 \oplus \tau_2 \equiv \tau'_1 \oplus \tau'_2 : \ast \vdash \bot
      \end{align*}
    \end{proof}

    Similar to the \( \bot \) subcase for \textit{ElimeqSucc}, but using property (v) instead of property (ii).

  \item \textbf{Case} \( \sigma \neq t \)

    \begin{proof}
      \begin{align*}
      \Gamma & / \sigma \equiv t : \kappa \vdash \bot \quad \text{ElimeqClash} \\
      \Gamma & / \sigma \neq t : \kappa \vdash \bot
      \end{align*}
    \end{proof}

    Since \( \sigma \neq t \), we know \( \sigma \) and \( t \) have different head constructors, and thus no unifier.
  \end{itemize}

\textbf{Theorem 6 (Soundness of Algorithmic Subtyping)}

If \( [\Gamma]A = A \) and \( [\Gamma]B = B \) and \( \Gamma \vdash A \text{ type and } \Gamma \vdash B \text{ type and } \Delta \rightarrow \Omega \) and \( \Gamma \vdash A : B \vdash \Delta \) then \( [\Omega]\Delta \vdash [\Omega]A \leq_{\pm} [\Omega]B \).

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

\begin{itemize}
  \item \textbf{Case} \( B \) not headed by \( \forall \)

    \begin{proof}
      \begin{align*}
      \Gamma & / \forall \alpha, \beta : \kappa \vdash [\alpha/\beta]A_0 : \leq \vdash \Delta, \forall \alpha, \beta, \Theta \\
      \Gamma & / \forall \alpha : \kappa, A_0 : \leq \vdash \Delta
      \end{align*}
    \end{proof}

    Let \( \Omega' = (\Omega, \forall \alpha, \Theta) \).
Proof of Theorem 6 (Soundness of Algorithmic Subtyping) thm:subtyping-soundness

\[ {\Gamma, \triangleright_{\alpha}, \xi : \kappa \vdash [\xi/\alpha]A_0 <_{\Theta} B \vdash_{\Delta, \triangleright_{\alpha}, \Theta} } \]

\[ \Delta \rightarrow \Omega \]
\[ (\Delta, \triangleright_{\alpha}, \Theta) \rightarrow \Omega' \]

\[ \Gamma \vdash \forall \alpha : \kappa. A_0 \text{ type} \]
\[ \Gamma, \alpha : \kappa \vdash A_0 \text{ type} \]
\[ \Gamma, \triangleright_{\alpha}, \xi : \kappa \vdash [\xi/\alpha]A_0 \text{ type} \]
\[ \Gamma \vdash B \text{ type} \]

\[ [\Omega'](\Delta, \triangleright_{\alpha}, \Theta) \vdash [\Omega'](\xi/\alpha)A_0 \leq^* [\Omega']B \]
\[ \Omega \vdash B \text{ type} \]
[\[\Omega']B = [\Omega]B \]

\[ [\Omega'](\Delta, \triangleright_{\alpha}, \Theta) \vdash [\Omega'](\xi/\alpha)A_0 \leq^* [\Omega']B \]

\[ [\Omega'](\Delta, \triangleright_{\alpha}, \Theta) \vdash [\Omega'](\xi/\alpha)\beta \leq^* [\Omega]B \]

\[ \Gamma, \triangleright_{\alpha}, \xi : \kappa \vdash \xi : \kappa \]
\[ \Gamma, \triangleright_{\alpha}, \xi : \kappa \rightarrow \Delta, \triangleright_{\alpha}, \Theta \]
\[ \Theta \text{ is soft} \]
\[ (\Delta, \triangleright_{\alpha}, \Theta) \rightarrow \Omega' \]

\[ [\Omega'\Omega'] \vdash [\Omega']\xi : \kappa \]

\[ [\Omega'](\Delta, \triangleright_{\alpha}, \Theta) \vdash [\Omega'](\forall \alpha : \kappa. A_0) \leq^* [\Omega]B \]

\[ [\Omega'(\Delta, \triangleright_{\alpha}, \Theta) \vdash [\Omega'(\forall \alpha : \kappa. A_0) \leq^* [\Omega]B \]

\[ \Gamma, \beta : \kappa \vdash A <_{\Theta} B \vdash_{\Delta, \beta : \kappa, \Theta} \]

\[ \Gamma \vdash \forall \beta : \kappa. B_0 \rightarrow \Delta \]

\[ \Gamma, \beta : \kappa \vdash A <_{\Theta} \forall \beta : \kappa. B_0 \rightarrow \Delta \]

\[ \Gamma \vdash A <_{\Theta} \forall \beta : \kappa. B_0 \rightarrow \Delta \]

\[ \Gamma \vdash \forall \alpha : \kappa. A_0 \leq^* [\Omega]B \]

\[\Gamma, \beta : \kappa \vdash \Delta, \beta : \kappa, \Theta \]
\[ \Theta \text{ is soft} \]

\[ [\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega, \beta : \kappa]A \leq^* [\Omega]B \]

\[ [\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega]A \leq^* [\Omega]B \]

\[ [\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega]A \leq^* \forall \beta : \kappa. [\Omega]B \]

\[ [\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega]A \leq^* [\Omega]B \]

\[\Gamma \vdash \forall \alpha : \kappa. A_0 \leq^* [\Omega]B \]

\[\Gamma, \beta : \kappa \vdash \Delta, \beta : \kappa, \Theta \]

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

\[\Gamma, \beta : \kappa \vdash A \rightarrow B_0 \rightarrow \Delta \]

\[\Gamma, \beta : \kappa \vdash A \rightarrow B_0 \rightarrow \Delta \]

\[\Gamma \vdash A \rightarrow B_0 \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \equiv \text{Equiv} \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]

\[\Gamma \vdash A <_{\Theta} B \rightarrow \Delta \]

\[\Gamma \vdash A \equiv B \rightarrow \Delta \]
Proof of Theorem 6 (Soundness of Algorithmic Subtyping)  

\[ \Gamma \vdash A \equiv B \vdash \Delta \quad \text{Subderivation} \]
\[ \Delta \rightarrow \Omega \quad \text{Given} \]
\[ [\Omega]A = [\Omega]B \quad \text{By Lemma 86 (Soundness of Algorithmic Equivalence)} \]
\[ \Gamma \vdash A \quad \text{type} \quad \text{Given} \]
\[ [\Omega]\Delta \vdash [\Omega]A \quad \text{By Lemma 54 (Completing Stability)} \]

• Case \( \Gamma \vdash A <: B \vdash \Delta \quad \text{nonpos}(B) \)
\[ \Gamma \vdash A <: B \vdash \Delta \quad \text{By inversion} \]
\[ \text{neg}(A) \quad \text{By inversion} \]
\[ \text{nonpos}(A) \quad \text{By inversion} \]
\[ [\Omega]\Gamma \vdash [\Omega]A \leq [\Omega]B \quad \text{By induction} \]

• Case \( \Gamma \vdash A <: B \vdash \Delta \quad \text{nonneg}(B) \)
\[ \Gamma \vdash A <: B \vdash \Delta \quad \text{By inversion} \]
\[ \text{pos}(A) \quad \text{By inversion} \]
\[ \text{nonneg}(A) \quad \text{By inversion} \]
\[ \text{pos}(B) \quad \text{By induction} \]

K’.4 Soundness of Typing

Theorem 7 (Soundness of Match Coverage).

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

Proof. By induction on the given algorithmic coverage derivation.

• Case \( \Gamma \vdash \cdot \Rightarrow e_1 | \ldots \text{ covers } \cdot \quad \text{CoversEmpty} \)
\[ [\Omega]\Gamma \vdash \cdot \Rightarrow e_1 | \ldots \text{ covers } \cdot \quad \text{By DeclCoversEmpty} \]

• Cases \text{CoversVar}, \text{Covers1}, \text{Covers}, \text{Covers+}, \text{Covers-} \)

Use the i.h. and apply the corresponding declarative rule.
Lemma 89 (Well-formedness of Algorithmic Typing).

Given \( \Gamma : \text{ctx} \):

(i) If \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \) then \( \Delta \vdash A \ p \) 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 \( (x : A \ p) \in \Gamma \)
  \[ \Gamma \vdash x \Rightarrow [\Gamma] A \ p \vdash \Gamma \]
  \[ \text{Var} \]
  \[ \Gamma = (\{\delta_0, x : A \ p, \Gamma_1\}) \quad (x : A \ p) \in \Gamma \]
  \[ \Gamma \vdash A \ p \] type
  \[ \text{Follows from } \Gamma : \text{ctx} \]

- Case \( \Gamma \vdash A \ ! \) type
  \[ \Gamma \vdash e \iff [\Gamma] A ! \vdash \Delta \]
  \[ \text{Anno} \]
  \[ \Gamma \vdash A \ ! \] type
  \[ \text{By inversion} \]
  \[ \Delta \Rightarrow \Delta \]
  \[ \text{By Lemma 51 (Typing Extension)} \]
  \[ \Delta \vdash A \ ! \] type
  \[ \text{By Lemma 41 (Extension Weakening for Principal Typing)} \]

- Case \( \Gamma \vdash e \Rightarrow A \ p \vdash \Omega \)
  \[ \Theta \vdash s : [\Theta] A \ p \gg C \ q \vdash \Delta \]
  \[ \text{or } \text{FEV}([\Delta] C) \neq \emptyset \]
  \[ \rightarrow E \]
  \[ \Gamma \vdash e \Rightarrow A \ p \vdash \Theta \]
  \[ \Theta \vdash A \ p \] type
  \[ \text{By inversion} \]

- Case \( \Theta \vdash A \ p \) type
  \[ \text{By induction} \]

- Case \( \Theta \vdash [\Theta] A \ p \) type
  \[ \text{By Lemma 40 (Right-Hand Subst. for Principal Typing)} \]

- Case \( \Theta : \text{ctx} \)
  \[ \text{By implicit assumption} \]

- Case \( \Theta \vdash s : [\Theta] A \ p \gg C \ q \vdash \Delta \)
  \[ \text{By inversion} \]

- Case \( \Delta \vdash C \ q \) type
  \[ \text{By mutual induction} \]
Proof of Lemma 89 (Well-formedness of Algorithmic Typing)

• Case \( \Gamma \vdash e : A \to \Theta \quad \Theta \vdash s : [\Theta]A \gg C \to \Delta \) \quad \text{FEV}(\Delta|C) = \emptyset
  
  \[ \Gamma \vdash e \Rightarrow A \to \Theta \]  
  By inversion

  \[ \Theta \vdash A \to \text{type} \]  
  By induction

  \[ \Theta \vdash \Theta|A \to \text{type} \]  
  By Lemma 40 (Right-Hand Subst. for Principal Typing)

  \[ \Theta|\text{ctx} \]  
  By implicit assumption

\[ \Theta \vdash\Theta|A \gg C \to \Delta \]  
By inversion

\[ \Delta \vdash C \to \text{type} \]  
By PrincipalWF

2. Suppose \( \Gamma \vdash s : A \gg B \qquad B \to \Delta \) and \( \Gamma \vdash A \to \text{type} \):

• Case \( \Gamma \vdash \cdot : A \gg A \to \Gamma \) \quad \text{EmptySpine}

  \[ \Gamma \vdash A \to \text{type} \]  
  Given

\[ \Gamma \vdash \vdash A \to \text{type} \]  
By Lemma 41 (Extension Weakening for Principal Typing)

\[ \Theta \vdash \vdash B \to \text{type} \]  
By Lemma 42 (Inversion of Principal Typing)

\[ \Delta \vdash C \to \text{type} \]  
By inversion

\[ \Delta \vdash \vdash \]  
By substitution

\[ \Delta \vdash \vdash \]  
By weakening

\[ \Delta \vdash \vdash \]  
By Lemma 47 (Checkprop Extension)

\[ \Delta \vdash \vdash \]  
By Lemma 41 (Extension Weakening for Principal Typing)

\[ \Delta \vdash \vdash \]  
By Lemma 40 (Right-Hand Subst. for Principal Typing)

\[ \Delta \vdash C \to \text{type} \]  
By induction

• Case \( \Gamma \vdash \text{true} \to \Theta \quad \Theta \vdash e \to [\Theta]A \gg C \to \Delta \) \quad \text{Spine}

  \[ \Gamma \vdash \vdash \]  
  By Lemma 42 (Inversion of Principal Typing)

\[ \Gamma \vdash \vdash \]  
By Lemma 47 (Checkprop Extension)

\[ \Theta \vdash \vdash \]  
By Lemma 41 (Extension Weakening for Principal Typing)

\[ \Theta \vdash \vdash \]  
By Lemma 40 (Right-Hand Subst. for Principal Typing)

\[ \Delta \vdash C \to \text{type} \]  
By induction

• Case

  \[ \theta[\alpha_1 : \alpha_1 : \alpha_1 : \cdots \to \alpha_1 \to \alpha_2] \vdash e \to [\alpha_1 \to \alpha_2] \gg C \to \Delta \]  
  \[ \theta[\alpha : \alpha] \vdash e \to [\alpha] \gg C \to \Delta \]  
  By rules

\[ \Delta \vdash C \to \text{type} \]  
By induction
Theorem 8 (Soundness of Algorithmic Typing).

Given $\Delta \rightarrow \Omega$:

(i) If $\Gamma \vdash e \leftarrow A \ p \vdash \Delta$ and $\Gamma \vdash A \ p$ type then $[\Omega]_{\Delta} \vdash [\Omega]_{\Delta} e \leftarrow [\Omega]_{\Delta} A \ p$.

(ii) If $\Gamma \vdash e \leftarrow A \ p \vdash \Delta$ then $[\Omega]_{\Delta} \vdash [\Omega]_{\Delta} e \rightarrow [\Omega]_{\Delta} A \ p$.

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

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

(v) If $\Gamma \vdash \Pi \Leftrightarrow \Lambda \leftarrow C \ p \vdash \Delta$ and $\Gamma \vdash \Lambda \ ! \ types$ and $\Gamma \vdash \Lambda \ ! \ types$ and $\Gamma \vdash C \ p$ type then $[\Omega]_{\Delta} / [\Omega]_{\Delta} [\Omega]_{\Delta} \Pi \Leftrightarrow [\Omega]_{\Delta} [\Omega]_{\Delta} \Lambda \Leftrightarrow [\Omega]_{\Delta} C \ p$.

(vi) If $\Gamma / P \vdash \Pi \Leftrightarrow \Lambda \leftarrow C \ p \vdash \Delta$ and $\Gamma \vdash P$ prop and $\text{FEV}(P) = \emptyset$ and $[\Gamma']_{P} = P$ and $\Gamma \vdash \Lambda \ ! \ types$ and $\Gamma \vdash C \ p$ type then $[\Omega]_{\Delta} / [\Omega]_{\Delta} [\Omega]_{\Delta} \Pi \Leftrightarrow [\Omega]_{\Delta} [\Omega]_{\Delta} \Lambda \Leftrightarrow [\Omega]_{\Delta} C \ p$.

Proof. By induction, using the measure in Definition 7.

- Case $\{x : A \ p\} \in \Gamma$
  $\Gamma \vdash x \Rightarrow [\Gamma]_{A \ p} \vdash \Gamma \text{Var}$
  
  (x : A \ p) \in \Gamma \quad \text{Premise}
  
  (x : A \ p) \in \Delta \quad \Gamma = \Delta
  
  \Delta \rightarrow \Omega \quad \text{Given}
  
  (x : [\Omega]_{A} A) \in [\Omega]_{\Gamma} \quad \text{By Lemma 9 (Uvar Preservation)} (ii)
  
  [\Omega]_{\Gamma} \vdash [\Omega]_{\Gamma} x \Rightarrow [\Omega]_{A} A \ p \quad \text{By DecvVar}
  
  \Delta \rightarrow \Omega \quad \text{Given}
  
  \Gamma \rightarrow \Omega \quad \Gamma = \Delta
  
  [\Omega]_{\Gamma} \vdash [\Omega]_{\Gamma} x \Rightarrow [\Omega]_{\Gamma} [\Omega]_{\Gamma} A \ p \quad \text{By above equality} \Rightarrow

- Case $\Gamma \vdash e \Rightarrow A \ q \vdash \Theta \vdash A \leftarrow \pm B \vdash \Delta$
  $\Gamma \vdash e \leftarrow B \ p \vdash \Delta \quad \text{Sub}$
  
  $\Gamma \vdash e \Rightarrow A \ q \vdash \Theta \quad \text{Subderivation}$
  
  $\Theta \vdash A \leftarrow \pm B \vdash \Delta \quad \text{Subderivation}$
  
  $\Theta \rightarrow \Delta \quad \text{By Lemma 51 (Typing Extension)}$
  
  $\Delta \rightarrow \Omega \quad \text{Given}$
  
  $\Theta \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)}$
  
  $[\Omega]_{\Theta} \vdash [\Omega]_{\Theta} e \Rightarrow [\Omega]_{\Theta} A \ q \quad \text{By i.h.}$
  
  $[\Omega]_{\Theta} = [\Omega]_{\Delta} \quad \text{By Lemma 56 (Confluence of Completeness)}$
  
  $[\Omega]_{\Delta} \vdash [\Omega]_{\Delta} e \Rightarrow [\Omega]_{\Delta} A \ q \quad \text{By above equality} \Rightarrow$
  
  $\Theta \vdash A \leftarrow \pm B \vdash \Delta \quad \text{Subderivation}$
  
  $[\Omega]_{\Delta} \vdash [\Omega]_{\Delta} A \leftarrow \pm [\Omega]_{\Delta} B \ p \quad \text{By DecvSub}$

- Case $\Gamma \vdash A_{0}! \ type \quad \Gamma \vdash e_{0} \leftarrow [\Gamma]_{A_{0}} ! \vdash \Delta$
  $\Gamma \vdash \{e_{0} : A_{0}\} \Rightarrow [\Delta]_{[A_{0} ! \ Delta]} \quad \text{Anno}$
  
  $\Gamma \vdash e_{0} \leftarrow [\Gamma]_{A_{0}} ! \vdash \Delta \quad \text{Subderivation}$
  
  $[\Omega]_{\Delta} \vdash [\Omega]_{\Delta} e_{0} \leftarrow [\Omega]_{\Gamma} [\Omega]_{A_{0}} ! \vdash \Delta \quad \text{By i.h.}$
  
  $\Gamma \vdash A_{0} ! \ type \quad \text{Subderivation}$
  
  $\Gamma \vdash A_{0} type \quad \text{By inversion}$
  
  $\text{FEV}(A_{0}) = \emptyset$
Proof of **Theorem 8** (**Soundness of Algorithmic Typing**)

\[ \Gamma \rightarrow \Delta \quad \text{By Lemma 51 (Typing Extension)} \]
\[ \Delta \rightarrow \Omega \quad \text{Given} \]
\[ \Gamma \rightarrow \Omega \quad \text{By Lemma 33 (Extension Transitivity)} \]
\[ \Omega \vdash A_0 \text{ type} \quad \text{By Lemma 36 (Extension Weakening (Sorts))} \]
\[ [\Omega] \Omega \vdash [\Omega] A_0 \text{ type} \quad \text{By Lemma 16 (Substitution for Type Well-Formedness)} \]
\[ [\Omega] \Omega = [\Omega] \Delta \quad \text{By Lemma 54 (Completing Stability)} \]
\[ [\Omega] \Delta \vdash [\Omega] A_0 \text{ type} \quad \text{By above equality} \]
\[ [\Omega] \Gamma A_0 = [\Omega] A_0 \quad \text{By Lemma 29 (Substitution Monotonicity) (iii)} \]
\[ [\Omega] \Delta \vdash [\Omega] e_0 \ll [\Omega] A_0 ! \quad \text{By above equality} \]
\[ [\Omega] \Delta \vdash [\Omega] e_0 : A_0 \quad \text{From definition of substitution} \]
\[ \mathsf{Case} \]
\[ \Gamma \vdash () \ll 1 p \quad \text{By Dec11} \]
\[ [\Omega] \Delta \vdash () \ll 1 p \quad \text{By definition of substitution} \]
\[ \mathsf{Case} \]
\[ \Gamma_0 [\& : * = 1] \vdash () \ll \& f \quad \text{By Dec11} \]
\[ \Gamma_0 [\& : * = 1] \rightarrow \Omega \quad \text{Given} \]
\[ [\Omega] \& \ll [\Omega] \Delta \& \quad \text{By Lemma 25 (Substitution Monotonicity) (i)} \]
\[ = [\Omega] 1 \quad \text{By definition of context application} \]
\[ = 1 \quad \text{By definition of context application} \]
\[ [\Omega] \Delta \vdash () \ll 1 f \quad \text{By Dec11} \]
\[ [\Omega] \Delta \vdash [\Omega] () \ll [\Omega] f \quad \text{By above equality} \]
\[ \mathsf{Case} \]
\[ \nu \ \mathsf{chk-I} \quad \Gamma, \alpha : \kappa \vdash \nu \ll A_0 \quad \text{From definition of substitution} \]
\[ \Gamma \vdash \nu \ll \forall \alpha : \kappa. A_0 \quad \text{From definition of substitution} \]
\[ \Delta \rightarrow \Omega \quad \text{Given} \]
\[ \Delta, \alpha \rightarrow \Omega, \alpha \quad \text{By Uvar} \]
\[ \Gamma, \alpha \vdash \Delta, \alpha, \Theta \quad \text{By Lemma 51 (Typing Extension)} \]
\[ \Theta \text{ soft} \quad \text{By Lemma 22 (Extension Inversion) (i) (with } \Gamma_R = \tau, \text{ which is soft)} \]
\[ \Delta, \alpha, \Theta \rightarrow \Omega, \alpha_j, [\Theta] \quad \text{By Lemma 25 (Filling Completes)} \]
\[ \Gamma, \alpha \vdash \nu \ll A_0 \quad \text{By above equality} \]
\[ [\Omega] \Gamma' \ll [\Omega] \nu \ll [\Omega] A_0 \quad \text{By Lemma 17 (Substitution Stability)} \]
\[ [\Omega] \Gamma' \ll [\Omega] \nu \ll [\Omega] A_0 \quad \text{By above equality} \]
\[ \Delta, \alpha, \Theta \rightarrow \Omega, \alpha_j, [\Theta] \quad \text{Above} \]
\[ \Theta \text{ is soft} \quad \text{Above} \]
\[ [\Omega] \Gamma' = ([\Omega] \Delta, \alpha) \quad \text{By Lemma 53 (Softness Goes Away)} \]
\[ [\Omega] \Delta, \alpha \vdash [\Omega] \nu \ll [\Omega] A_0 \quad \text{By above equality} \]
\[ \mathsf{Case} \]
\[ \Gamma, \alpha \vdash [\Omega] \nu \ll [\Omega] A_0 \quad \text{By Dec11} \]
\[ [\Omega] \Delta \vdash [\Omega] \nu \ll [\Omega] (\forall \alpha. A_0) \quad \text{By definition of substitution} \]

---

**Proof of Theorem 8 (Soundness of Algorithmic Typing)**

\[ \text{thm:typing-soundness} \]
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \Gamma, \alpha : \kappa \vdash e \cdot s_0 : [\alpha/\alpha]A_0 \gg C \qquad \Gamma \vdash e \cdot s_0 : \forall \alpha : \kappa. A_0 \gg C \]

- **Case** \( \Gamma, \alpha : \kappa \vdash e \cdot s_0 : [\alpha/\alpha]A_0 \gg C \)

\[ \begin{align*}
\Gamma &\vdash e \cdot s_0 : \forall \alpha : \kappa. A_0 \gg C \\
\end{align*} \]

\( \Gamma, \alpha : \kappa \vdash e \cdot s_0 : [\alpha/\alpha]A_0 \gg C \qquad \Gamma \vdash e \cdot s_0 : \forall \alpha : \kappa. A_0 \gg C \]

\( \begin{align*}
\text{Subderivation} \\
\end{align*} \)

\( \begin{align*}
\text{By Lemma 51 (Typing Extension)} \\
\end{align*} \)

\( \begin{align*}
\text{By Lemma 36 (Extension Weakening (Sorts))} \\
\end{align*} \)

\( \begin{align*}
\text{By def. of subst.} \\
\end{align*} \)

- **Case** \( e \text{ chk-I} \)

\[ \Gamma \vdash P \text{ true} \quad \Theta \vdash e \iff [\Theta]A_0 \gg \Delta \]

\[ \Gamma \vdash e \iff A_0 \gg p \gg \Delta \]

\( \begin{align*}
\text{Subderivation} \\
\text{Given} \\
\text{By Lemma 51 (Typing Extension)} \\
\text{By Lemma 35 (Extension Transitivity)} \\
\text{By Lemma 87 (Soundness of Checkprop)} \\
\text{By Lemma 56 (Confluence of Completeness)} \\
\end{align*} \)

\( \begin{align*}
\text{By def. of subst.} \\
\end{align*} \)
Proof of Theorem 8: (Soundness of Algorithmic Typing)

- Case \( \nu \) chk-I
  \[ \Gamma, p \vdash P \vdash \Theta^+ \quad \Theta^+ \vdash \nu \leftarrow [\Theta^+]A_0 \vdash i_\Delta, p, \Delta' \]

  \[ \Gamma \vdash A \vdash \text{type} \quad \text{Given} \]

  \[ \Theta^+ = (\Gamma, p, \Theta) \quad \text{By Lemma 88 (Soundness of Equality Elimination)} \]

  \[ [\Omega', \Theta](\nu') = [\theta][\Gamma, p](\nu') \quad \text{for all } \Omega' \text{ extending } (\Gamma, p) \text{ and } \nu' \text{ s.t. } \Omega' \vdash \nu' : \kappa' \]

  \[ \theta = \text{mgu}(\sigma, \tau) \quad \text{Subderivation} \]

  \[ \Delta \rightarrow \Omega \quad \text{Given} \]

  \[ \Theta^+ \rightarrow \Delta, p, \Delta' \quad \text{By Lemma 51 (Typing Extension)} \]

  \[ \Gamma, p, \Theta \rightarrow \Delta, p, \Delta' \quad \text{By above equalities} \]

  \[ \Delta, p, \Theta \rightarrow \Delta, p, \Delta' \quad \text{Let } \Omega^+ = (\Delta, p, \Delta'). \]

  \[ \Omega^+ = (\Omega, p, \Delta'). \]

  \[ [\Omega', \Theta]B = [\theta][\Gamma, p]B \quad \text{By Lemma 93 (Substitution Upgrade) (i)} \]

  \[ \text{for all } \Omega' \text{ extending } (\Gamma, p) \text{ and } B \text{ s.t. } \Omega' \vdash B : \kappa' \]

  \[ [\Omega^+](\Delta, p, \Delta') \vdash [\Omega](\nu) \leftarrow [\Theta^+](\nu)A_0 ! \quad \text{Subderivation} \]

  \[ \text{By i.h.} \]

  \[ [\Omega^+](\Delta, p, \Delta') = [\theta][\Gamma, p] \quad \text{By Lemma 33 (Extension Transitivity)} \]

  \[ \Gamma \rightarrow \Omega \quad \text{By Lemma 22 (Extension Inversion)} \]

  \[ [\Omega^+](\Delta, p, \Delta') = [\theta][\Gamma, p]A_0 \quad \text{By Lemma 29 (Substitution Monotonicity)} \]

  \[ \text{Above, with } (\Omega, p, \Delta') \text{ as } \Omega' \text{ and } A_0 \text{ as } B \]

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

  \[ [\Omega, p, \Theta](\Delta, p, \Delta') = [\theta][\Omega](\Delta) \quad \text{By Lemma 93 (Substitution Upgrade) (iii)} \]

  \[ [\theta][\Omega](\Delta) \vdash [\Omega][\theta](\nu) \leftarrow [\theta][\Omega]A_0 ! \quad \text{By above equalities} \]

  \[ [\Omega^+](\Delta, p, \Delta') / (\sigma = t) \vdash [\nu] \leftarrow [\Omega]A_0 ! \quad \text{By DeclCheckUnify} \]

  \[ [\Omega^+](\Delta, p, \Delta') = [\Omega](\Delta) \quad \text{From def. of context application} \]

  \[ [\Omega](\Delta) / (\sigma = t) \vdash [\nu] \leftarrow [\Omega]A_0 ! \quad \text{By above equality} \]

  \[ [\Omega](\Delta) \vdash [\nu] \leftarrow (\sigma = t) \vdash [\Omega]A_0 ! \quad \text{By Decl} \]

  \[ [\Omega](\Delta) \vdash [\nu] \leftarrow ([\nu] = [\Omega](\sigma = t) \vdash [\Omega]A_0 ! \quad \text{By FEV condition above} \]

- Case \( \nu \) chk-I
  \[ \Gamma, p \vdash P \vdash \perp \]

  \[ \Gamma \vdash \nu \leftarrow P \vdash A_0 ! \vdash i_\Delta \quad \text{Subderivation} \]

  \[ \Gamma, p \vdash P \vdash \perp \quad \text{Subderivation} \]

  \[ \Gamma, p \vdash \sigma \vdash t : \kappa \vdash \perp \quad \text{By inversion} \]

  \[ P = (\sigma = t) \quad \text{"} \]

  \[ \text{As in } (\text{Case above)} \]

  \[ \text{By Lemma 88 (Soundness of Equality Elimination)} \]

Proof of Theorem 8: (Soundness of Algorithmic Typing)
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \text{thm:typing-soundness} \]

\[ [\Omega][\Delta] / (\sigma = t) \vdash \llbracket \Omega \rrbracket A_0 ! \]
\[ [\Omega][\Delta] \vdash \llbracket \Omega \rrbracket v \iff (\sigma = t) \gg \llbracket \Omega \rrbracket A_0 ! \]
\[ [\Omega][\Delta] \vdash \llbracket \Omega \rrbracket v \iff ([\Omega](\sigma = t)) \gg \llbracket \Omega \rrbracket A_0 ! \]

\[ \Rightarrow \]
\[ [\Omega][\Delta] \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

Let \( \Omega' = \Omega \).

\[ \Rightarrow \]
\[ \Omega \longrightarrow \Omega' \]

By Lemma 32 (Extension Reflexivity)

Given

\[ \Rightarrow \]
\[ \Delta \longrightarrow \Omega' \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By above FEV condition

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By above equality

By above equality

By Lemma 33 (Extension Transitivity)

By Lemma 56 (Confluence of Completeness)

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By above equality

By above equality

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By above equality

By above equality

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By above equality

By above equality

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.

\[ \Rightarrow \]
\[ \llbracket \Omega \rrbracket \Delta \vdash \llbracket \Omega \rrbracket v \iff \llbracket \Omega \rrbracket (P \gg A_0) ! \]

By def. of subst.
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[
\Gamma[\alpha_1:*, \alpha_2:*, \alpha:* = \alpha_1 \rightarrow \alpha_2], x : \alpha_j \Rightarrow \Delta, x : \alpha_j \Theta
\]

\[
\Gamma[\alpha_1:*, \alpha_2:*, \alpha:* = \alpha_1 \rightarrow \alpha_2] \rightarrow \Delta
\]

\[
\Delta \rightarrow \Omega
\]

\[
\Delta, x : \alpha_1 \theta \Rightarrow \Omega, x : [\Omega]\alpha_1 \theta
\]

\[
\Delta, x : \alpha_1 \theta, \Theta \Rightarrow \Omega, x : [\Omega]\alpha_1 \theta, \Theta
\]

\[
\Gamma[\alpha_1:*, \alpha_2:*, \alpha:* = \alpha_1 \rightarrow \alpha_2] \Rightarrow e_0 \Leftrightarrow \alpha_2 \theta \Rightarrow \Delta, x : \alpha_j \Theta
\]

\[
\alpha \Rightarrow \text{Subderivation}
\]

\[
[\Omega]\alpha \Leftrightarrow [\Omega][\Gamma] \alpha
\]

\[
[\Omega] \Gamma \Leftrightarrow [\Omega] \Theta \Rightarrow [\Omega] \Delta
\]

\[
\text{By Lemma 17 (Substitution Stability)}
\]

\[
\text{By definition of substitution}
\]

\[
\text{By definition of context substitution}
\]

\[
\text{By above equalities}
\]

\[
\Gamma \vdash e_0 \Rightarrow A q \rightarrow \Theta \quad \quad \Theta \vdash s_0 : A q \Rightarrow C [p] \Rightarrow \Delta
\]

\[
\Gamma \vdash e_0 s_0 \Rightarrow C p \Rightarrow \Delta
\]

\[
\text{Subderivation}
\]

\[
\text{Subderivation}
\]

\[
\text{Given}
\]

\[
\text{By Lemma 33 (Extension Transitivity)}
\]

\[
\text{By definition of substitution}
\]

\[
\text{By definition of substitution}
\]

\[
\text{By definition of context substitution}
\]

\[
\text{By above equality}
\]

\[
\text{By above equality}
\]

\[
\text{By above equality}
\]

\[
\text{By above equality}
\]

\[
\text{By i.h.}
\]

\[
\text{By i.h.}
\]

\[
\text{By i.h.}
\]

\[
\text{By rule Decl→E}
\]

\[
\text{By rule Decl→E}
\]

\[
\text{Subderivation}
\]

\[
\text{Subderivation}
\]

\[
\text{By i.h.}
\]

\[
\text{We show the quantified premise of DeclSpineRecover namely,}
\]

\[
\text{for all C'}.
\]

\[
\text{if } [\Omega]\Theta \vdash s : [\Omega]A \Rightarrow C' \rightarrow \Delta \text{ then } C' = [\Omega]C
\]

\[
\text{Suppose we have C' such that } [\Omega] \Gamma \vdash s : [\Omega]A \Rightarrow C' \rightarrow \Delta. \text{ To apply DeclSpineRecover we need to show } C' = [\Omega]C.
\]
Proof of Theorem 8 (Soundness of Algorithmic Typing)  thm:typing-soundness

\[
\begin{align*}
\{\Omega\} \vdash [\{\Omega\} : [\{\Omega\} A ! \gg C \gg \Omega] & \quad \text{Assumption} \\
\Omega_{\text{canon}} \rightarrow \Omega & \\
\text{dom}(\Omega_{\text{canon}}) = \text{dom}(\Gamma) & \\
\Gamma \rightarrow \Omega_{\text{canon}} & \\
[\{\Omega\} \vdash [\{\Omega_{\text{canon}}\} \Gamma] & \\
[\{\Omega\} A : [\{\Omega_{\text{canon}}\} A] & \\
[\{\Omega_{\text{canon}}\} \Gamma \vdash [\{\Omega_{\text{canon}}\} A ! \gg C \gg \Delta] & \\
\Gamma \vdash s : [\{\Gamma\} A ! \gg C" \gg \Delta"] & \quad \text{By Theorem 11 (Completeness of Algorithmic Typing)} \\
\Omega_{\text{canon}} \rightarrow \Omega" & \\
\Delta" \rightarrow \Omega" & \\
C" = [\{\Omega"\} C" & \\
& = [\{\Omega"\} C & \\
& = [\{\Omega_{\text{canon}}\} C & \\
C \rightarrow C & & \text{By above equalities} & \\
q = \frac{\text{C"}}{\text{C}} & & \text{Above} & \\
\Delta" = \Delta & & \text{By Lemma 55 (Completing Completeness)} (ii) & \\
& \text{We have thus shown the above “for all C’…” statement.} & & \text{By Lemma 55 (Completing Completeness)} (ii) & \\
& \text{By above equalities} & & \text{Above} & \\
& \text{By Lemma 55 (Completing Completeness)} (ii) & & \text{By above equalities} & \\
& \text{By Lemma 55 (Completing Completeness)} (ii) & & \text{By above equalities} & \\
& \text{By above equalities} & & \text{By above equalities} & \\
& \text{By above equalities} & & \text{By above equalities} & \\
& \text{By above equalities} & & \text{By above equalities} & \\
\end{align*}
\]

\[\Box\]

\[
\begin{align*}
& [\{\Omega\} \vdash [\{\Omega\} A ! \gg [\{\Omega\} C ] & \\
& \text{By DeclSpineRecover} & \\
\bullet \text{ Case } & & \text{SpinePass} & \\
& \Gamma \vdash s : A \gg C \gg \Delta & \quad \text{(p = \{\Omega\}) or (q = \{\Omega\}) or (FEV(C) \neq \emptyset)} & \\
& \Gamma \vdash s : A \gg C [q \gg \Delta] & & \text{SpinePass} & \\
& \text{Subderivation} & \\
& [\{\Omega\} \vdash [\{\Omega\} A p \gg [\{\Omega\} C q & & \text{By i.h.} & \\
& & \text{By DeclSpinePass} & \\
& \Box & \\
\bullet \text{ Case } & & \text{EmptySpine} & \\
& \Gamma \vdash \vdash : A \gg A \gg \Gamma & & \text{DeclEmptySpine} & \\
& \Box & \\
& \Gamma \vdash e_0 \leftarrow A_1 p \gg \Theta & \quad \Theta \vdash s_0 : [\{\Theta\} A_2 p \gg [\{\Theta\} C q \gg \Delta & & \text{Spine} & \\
& \Gamma \vdash e_0 \cdot s_0 : A_1 \rightarrow A_2 p \gg [\{\Theta\} C q \gg \Delta & & \text{Spine} & \\
\Delta \rightarrow \Omega & & \text{Given} & \\
\Theta \rightarrow \Delta & & \text{By Lemma 53 (Extension Transitivity)} & \\
\Theta \rightarrow \Omega & & \text{By Lemma 53 (Extension Transitivity)} & \\
& \text{Subderivation} & \\
& [\{\Omega\} \Theta \vdash [\{\Omega\} e_0 \leftarrow [\{\Omega\} A_1 p & & \text{By i.h.} & \\
& & \text{By Lemma 56 (Confluence of Completeness)} & \\
& [\{\Omega\} \Theta = [\{\Omega\} \Delta & & \text{By above equality} & \\
& & \text{By above equality} & \\
& [\{\Omega\} \Delta \vdash [\{\Omega\} e_0 \leftarrow [\{\Omega\} A_1 p & & \text{Subderivation} & \\
& \Theta \vdash s_0 : [\{\Theta\} A_2 p \gg [\{\Theta\} C q & & \text{By i.h.} & \\
& \Theta \vdash s_0 : [\{\Theta\} A_2 p \gg [\{\Theta\} C q & & \text{By DeclSpine} & \\
& & \text{By DeclSpine} & \\
& \Box & \\
\bullet \text{ Case } & & \text{ inj}_{k_1} e_0 \leftarrow A_1 + A_2 p \gg \Delta & \\
& \Gamma \vdash \vdash \text{ inj}_{k_1} e_0 \leftarrow A_1 + A_2 p \gg \Delta & & \text{Def. of subst.} & 
\end{align*}
\]
Proof of **Theorem 8** *(Soundness of Algorithmic Typing)*  

\[ \Gamma \vdash e_0 \triangleleft A_k \quad \vdash \Delta \]  

**Subderivation**  

\[ [\Omega] \Delta \vdash [\Omega] e_0 \triangleleft [\Omega] A_k \quad \vdash \Delta \]  

By i.h.

\[ [\Omega] \Delta \vdash \text{inj}_k \quad [\Omega] e_0 \triangleright ([\Omega] A_1) + ([\Omega] A_2) \quad \vdash \Delta \]  

By Decl+lk

\[ [\Omega] \Delta \vdash [\Omega] \text{inj}_k e_0 \triangleleft [\Omega] (A_1 + A_2) \quad \vdash \Delta \]  

By def. of substitution

**Case** \[ \Gamma[\alpha_1: *, \alpha_2: \star, \alpha: * \triangleright \alpha_k] \vdash e_0 \triangleleft \alpha_k \quad \vdash \alpha_k \triangleright \Delta \]  

\[ \Gamma[[\alpha]: \star] \vdash \text{inj}_k e_0 \triangleleft \alpha \quad \vdash \alpha \triangleright \Delta \]  

**Subderivation**  

\[ [\Omega] \Delta \vdash [\Omega] [\alpha] e_0 \triangleright \alpha \]  

By i.h.

\[ \Gamma[[\alpha]_1] + ([\alpha]_2) = [\alpha] \]  

By Decl+lk

\[ [\Omega] \Delta \vdash [\Omega] \text{inj}_k e_0 \Rightarrow [\Omega] [\alpha] \quad \vdash \Delta \]  

Similar to the \(+l\_{\alpha}\) case (above)

By above equality / def. of subst.

**Case** \[ \Gamma[\alpha_1: \star, \alpha_2: \star, \alpha: * \triangleright \alpha_k] \vdash e_1 \triangleleft A_1 \quad \vdash \theta \quad \Theta \vdash e_2 \triangleleft [\theta] A_2 \quad \vdash \Theta \quad \vdash \Delta \]  

\[ \vdash (e_1, e_2) \triangleleft (A_1 \times A_2) \quad \vdash \Delta \]  

\[ \vdash (e_1, e_2) \triangleleft (A_1 \times A_2) \quad \vdash \Delta \]  

By def. of subst.

\[ \vdash (e_1, e_2) \triangleleft (A_1 \times A_2) \quad \vdash \Delta \]  

**Subderivation**  

\[ [\Omega] \Delta \vdash [\Omega] e_1 \triangleleft [\Omega] A_1 \quad [\Omega] \Delta \vdash [\Omega] e_2 \triangleleft [\Omega] A_2 \]  

By Lemma 51 (Typing Extension)

\[ [\Omega] \Delta \vdash [\Omega] e_1 \triangleleft [\Omega] A_1 \]  

By Lemma 53 (Extension Transitivity)

\[ [\Omega] \Delta \vdash [\Omega] e_2 \triangleleft [\Omega] A_2 \]  

By Lemma 56 (Confluence of Completeness)

\[ [\Omega] \Delta \vdash [\Omega] (e_1, e_2) \triangleleft ([\Omega] A_1) \times [\Omega] A_2 \]  

By i.h.

\[ [\Omega] \Delta \vdash [\Omega] (e_1, e_2) \triangleleft ([\Omega] A_1) \times [\Omega] A_2 \]  

By i.h.

\[ [\Omega] \Delta \vdash [\Omega] (e_1, e_2) \triangleleft ([\Omega] A_1) \times [\Omega] A_2 \]  

By Dec\times 1

\[ [\Omega] \Delta \vdash [\Omega] (e_1, e_2) \triangleleft ([\Omega] A_1) \times [\Omega] A_2 \]  

By def. of subst.

**Case** \[ \Gamma[\alpha_1: *, \alpha_2: *, \alpha: * \triangleright \alpha_k] \vdash e_1 \triangleleft \alpha_1 \quad \vdash \theta \quad \Theta \vdash e_2 \triangleleft [\theta] \alpha_2 \quad \vdash \Theta \quad \vdash \Delta \]  

\[ \vdash (e_1, e_2) \triangleright \alpha_1 \times \alpha_2 \quad \vdash \Delta \quad \vdash \Delta \]  

\[ \vdash (e_1, e_2) \triangleright \alpha_1 \times \alpha_2 \quad \vdash \Delta \]  

Similar to the \(+l_{\alpha}\) case, but using Lemma 51 (Typing Extension) and Lemma 56 (Confluence of Completeness) to show \([\theta] A_1 = [\theta] A_2\).

\[ \vdash (e_1, e_2) \triangleright \alpha_1 \times \alpha_2 \quad \vdash \Delta \]  

**Subderivation**  

\[ [\Omega] \Theta \vdash [\Omega] e_1 \triangleleft [\Omega] \alpha_1 \]  

By i.h.

\[ [\Omega] \Theta = [\Omega] \Delta \]  

By Lemma 56 (Confluence of Completeness)

\[ [\Omega] \Delta \vdash [\Omega] e_1 \triangleleft [\Omega] \alpha_1 \]  

By above equality

\[ \Theta \vdash e_2 \triangleleft [\theta] \alpha_2 \quad \vdash \Delta \]  

**Subderivation**  

\[ [\Omega] \Delta \vdash [\Omega] e_2 \triangleleft [\Omega] \alpha_2 \]  

By i.h.

\[ [\Omega] \Theta \vdash [\Omega] \alpha_2 \]  

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega] \Delta \vdash [\Omega] e_2 \triangleleft [\Omega] \alpha_2 \]  

By above equality
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \{\Omega\} \Delta \vdash [\Omega]e_1, [\Omega]e_2 \iff ([\Omega]\delta_1) \times [\Omega]\delta_2 \not\vdash \]  
\[ \{\Omega\}\delta_1 \times [\Omega]\delta_2 = [\Omega] \delta \]  
\[ \{\Omega\} \Delta \vdash [\Omega]e_1, e_2 \iff [\Omega] \delta \not\vdash \]

**Case**

\[ \Gamma[\delta_2 : \star, \delta_1 : \alpha, \star : \alpha = \delta_1 \rightarrow \delta_2] \vdash e_0 \cdot s_0 : (\delta_1 \rightarrow \delta_2) \not\vdash C \not\vdash \Delta \]

\[ \text{MatchEmpty} \]

By above equality

\[ \Gamma \vdash \Delta \vdash \Pi \text{ covers } [\Delta]B \]

\[ \Gamma \vdash \text{case}(e_0, \Pi) \iff [\Theta]C \not\vdash \Delta \]

Subderivation

\[ \Theta \rightarrow \Delta \]

By Lemma 51 (Typing Extension)

\[ \Theta \rightarrow \Omega \]

By Lemma 53 (Extension Transitivity)

\[ [\Omega]\Theta \vdash [\Omega]e_0 \Rightarrow [\Theta]B \not\vdash \]

By i.h.

\[ [\Omega]\Delta \vdash [\Omega]e_0 \Rightarrow [\Theta]B \not\vdash \]

By Lemma 56 (Confluence of Completeness)

\[ \Theta \vdash \Pi :: [\Theta]B \iff [\Theta]C \not\vdash \Delta \]

Subderivation

\[ \Gamma \vdash e_0 \Rightarrow B ! \not\vdash \Theta \]

By Lemma 63 (Well-Formed Outputs of Typing) (Synthesis)

\[ \Gamma \vdash C \not\vdash \Theta \]

Given

\[ \Theta \rightarrow \Theta \]

By Lemma 51 (Typing Extension)

\[ \Theta \vdash C \not\vdash \Theta \]

By Lemma 41 (Extension Weakening for Principal Typing)

\[ \Theta \vdash [\Theta]C \not\vdash \Theta \]

By Lemma 40 (Right-Hand Subst. for Principal Typing)

\[ [\Omega]\Delta \vdash [\Omega]\Pi :: [\Theta]B \iff [\Theta]C \not\vdash \Delta \]

By i.h. (v)

\[ [\Omega]\Theta \vdash = [\Theta]C \not\vdash \]

By Lemma 29 (Substitution Monotonicity)

\[ [\Omega]\Delta \vdash [\Theta]\Pi :: [\Theta]B \iff [\Theta]C \not\vdash \Delta \]

By above equalities

\[ \Delta \vdash \Pi \text{ covers } [\Delta]B \]

Subderivation

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

By idempotence of substitution

\[ \Theta \vdash B ! \not\vdash \]

By Lemma 63 (Well-Formed Outputs of Typing)

\[ \Delta \vdash B ! \not\vdash \]

By Lemma 41 (Extension Weakening for Principal Typing)

\[ \Delta \vdash [\Delta]B \not\vdash \]

By Lemma 40 (Right-Hand Subst. for Principal Typing)

\[ [\Omega]\Delta \vdash [\Omega]\Pi \text{ covers } [\Delta]B \]

By Theorem 7 (Soundness of Match Coverage)

\[ [\Delta]B = [\Omega]B \]

By Lemma 39 (Principal Agreement) (i)

\[ [\Omega]\Delta \vdash [\Omega]\Pi \text{ covers } [\Omega]B \]

By above equality

\[ \equiv [\Omega]\Delta \vdash [\Omega]\text{case}(e_0, \Pi) \iff [\Theta]C \not\vdash \]

By DeclCase

Part (v):

**Case** **MatchEmpty**  
Apply rule **DeclMatchEmpty**

**Case**

\[ \Gamma \vdash e \not\vdash C \not\vdash \Delta \]

\[ \Gamma \vdash (\cdot \Rightarrow e) :: \cdot \not\vdash C \not\vdash \Delta \]

MatchBase

Apply the i.h. and **DeclMatchBase**

**Case** **MatchUnit**  
Apply the i.h. and **DeclMatchUnit**

Proof of Theorem 8 (Soundness of Algorithmic Typing)
Proof of Theorem 8 (Soundness of Algorithmic Typing)

- **Case** \( \Gamma \vdash \pi :: \bar{A} \equiv C \vdash \Theta \vdash \Pi' :: \bar{A} \equiv C \vdash \Delta \)
  
  Apply the i.h. to each premise, using lemmas for well-formedness under \( \Theta \); then apply \text{DeclMatchSeq}.

- **Cases** [\text{Match}] [\text{Match} \lor] [\text{MatchWild}]
  
  Apply the i.h. and the corresponding declarative match rule.

- **Cases** [\text{Match} \times] [\text{Match} + k]
  
  We have \( \Gamma \vdash \bar{A} ! \) types, so the first type in \( \bar{A} \) has no free existential variables.
  
  Apply the i.h. and the corresponding declarative match rule.

- **Case** \( A \) not headed by \( \land \) or \( \lor \) \( \exists \Gamma, z : A! \vdash \bar{p} \Rightarrow e' :: \bar{A} \equiv C \vdash \Delta, z : A!, \Delta' \)
  
  \( \Gamma \vdash z, \bar{p} \Rightarrow e :: A, \bar{A} \equiv C \vdash \Delta \)
  
  **MatchNeg**

  Construct \( \Omega' \) and show \( \Delta, z : A!, \Delta' \Rightarrow \Omega' \) as in the \text{Match} case.

  Use the i.h., then apply rule \text{DeclMatchNeg}

**Part (vi):**

- **Case** \( \Gamma / \sigma \equiv \pi : k \vdash _\perp \Gamma' \)
  
  \( \Gamma / \sigma \equiv \pi : k \vdash _\perp \Gamma' \)
  
  Subderivation

  \( \Gamma / \sigma \equiv \pi : k \vdash _\perp \Gamma' \)

  **Match**

  Apply the i.h. and the corresponding declarative match rule.

- **Case** \( \Gamma_\bowtie \vdash \sigma \equiv \pi : k \vdash \Gamma' \)
  
  \( \Gamma_\bowtie \vdash \sigma \equiv \pi : k \vdash \Gamma' \)
  
  Subderivation

  \( \Gamma_\bowtie \vdash \sigma \equiv \pi : k \vdash \Gamma' \)

  **Match**

  Apply the i.h. and the corresponding declarative match rule.

- **Case** \( \Gamma_\bowtie \vdash \sigma \equiv \pi : k \vdash \Gamma' \)
  
  \( \Gamma_\bowtie \vdash \sigma \equiv \pi : k \vdash \Gamma' \)
  
  Subderivation

  \( \Gamma_\bowtie \vdash \sigma \equiv \pi : k \vdash \Gamma' \)

  **Match**

  Apply the i.h. and the corresponding declarative match rule.
L’ Completeness

L’.1 Completeness of Auxiliary Judgments

Lemma 90 (Completeness of Instantiation).
Given $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash \tau : \kappa$ and $\tau = [\Gamma] \kappa$ and $\lambda \in \text{unsolved}(\Gamma)$ and $\lambda \notin \text{FV}(\tau)$:
If $[\Omega] \lambda = [\Omega] \kappa$
then there are $\Delta$, $\Omega'$ such that $\Omega \rightarrow \Omega'$ and $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Gamma \vdash \lambda := \tau : \kappa \rightarrow \Delta$.

Proof. By induction on $\tau$.
We have $[\Omega] \Gamma \vdash [\Omega] \lambda \leq^* [\Omega] \alpha$. We now case-analyze the shape of $\tau$.

- Case $\tau = \beta$:
  - $\lambda \notin \text{FV}(\beta)$ Given
  - $\lambda \neq \beta$ From definition of FV
  - $\beta \in \text{unsolved}(\Gamma)$ From $[\Gamma] \beta = \beta$

  Let $\Omega' = \Omega$.

  ![Extension Reflexivity]

  Now consider whether $\lambda$ is declared to the left of $\beta$, or vice versa.

  - Case $\Gamma = \Gamma_0[\lambda : \kappa][\beta : \kappa]$:
    - Let $\Delta = \Gamma_0[\lambda : \kappa][\beta : \kappa] = \lambda$.
    - $\Gamma \vdash \lambda := \beta : \kappa \rightarrow \Delta$ By InstReach
    - $[\Omega] \lambda = [\Omega] \beta$ Given
    - $\Gamma \rightarrow \Omega$ Given

    ![Parallel Extension Solution]

  - Case $\Gamma = \Gamma_0[\beta : \kappa][\lambda : \kappa]$:
    Similar, but using InstSolve instead of InstReach.

- Case $\tau = \alpha$:
We have $[\Omega] \lambda = \alpha$, so (since $\Omega$ is well-formed), $\alpha$ is declared to the left of $\lambda$ in $\Omega$.
We have $\Gamma \rightarrow \Omega$.

By Lemma 21 (Reverse Declaration Order Preservation), we know that $\alpha$ is declared to the left of $\lambda$ in $\Gamma$, that is, $\Gamma_1[\alpha : \kappa][\lambda : \kappa]$.

Let $\Delta = \Gamma_1[\alpha : \kappa][\lambda : \kappa] \rightarrow [\lambda : \kappa] \rightarrow \Delta$.

By InstReach $\Gamma_1[\alpha : \kappa][\lambda : \kappa] \vdash \lambda := \alpha : \kappa \rightarrow \Delta$.

By Lemma 27 (Parallel Extension Solution), $\Gamma_1[\alpha : \kappa][\lambda : \kappa] \rightarrow \Delta := [\Gamma] \kappa \rightarrow \Omega$.
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 $\tau = \tau_1 \oplus \tau_2$:
Proof of **Lemma 90**

**Completeness of Instantiation**

\[ \Gamma \vdash \varphi \]

Given

\[ \Gamma \vdash \varphi \]

Similarly

\[ \varphi \in \text{unsolved}(\Gamma) \]

By Lemma 23 (Deep Pair Introduction) (iii) twice, inserting unsolved variables \( \hat{\alpha}_2 \) and \( \hat{\alpha}_1 \) on both contexts in the above extension preserves extension:

\[ \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_1, \hat{\alpha}_2] \rightarrow \Omega_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_2] \]

By Lemma 23 (Deep Pair Introduction) (ii)

Straightforward

\[ \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_1, \hat{\alpha}_2] \rightarrow \Omega_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_2] \]

By Lemma 33 (Extension Transitivity)

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

\[ \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_1, \hat{\alpha}_2] \rightarrow \Omega_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_1, \hat{\alpha}_2] \]

By Lemma 28 (Parallel Variable Update)

Since \( \hat{\alpha} \notin \text{FV}(\tau) \), it follows that \( [\Gamma_1]_\tau = [\Gamma]_\tau = \tau \).

Therefore \( \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\tau) \).

Therefore \( \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\tau) \).

By Lemma 55 (Completing Completeness) (i) and (iii), \( \Omega | \Gamma_{1} = | \Omega | \Gamma \) and \( \Omega | \hat{\alpha}_1 = \tau_1 \).

By i.h., there are \( \Delta_2 \) and \( \Omega_2 \) such that \( \Gamma_{1} \vdash \hat{\alpha}_1 = \tau_1 : \Delta_2 \) and \( \Delta_2 \rightarrow \Omega_2 \) and \( \Omega_1 \rightarrow \Omega_2 \).

Next, note that \( \Delta_2 | \Delta_2 = \Delta_2 | \tau_2 \).

By Lemma 64 (Left Unsolvedness Preservation), we know that \( \hat{\alpha}_2 \in \text{unsolved}(\Delta_2) \).

By Lemma 65 (Left Free Variable Preservation), we know that \( \hat{\alpha}_2 \notin \text{FV}(\Delta_2 | \tau_2) \).

By Lemma 33 (Extension Transitivity), \( \Omega \rightarrow \Omega_2 \).

We know \( \Omega_2 | \Delta_2 = | \Omega | \Gamma \) because:

\[ \Omega_2 | \Delta_2 = | \Omega_2 | \Omega_2 \]

By Lemma 54 (Completing Stability)

\[ = | \Omega | \Omega \]

By Lemma 55 (Completing Completeness) (iii)

\[ = | \Omega ) | \Gamma \]

By Lemma 54 (Completing Stability)

By Lemma 55 (Completing Completeness) (i), we know that \( \Omega_2 | \hat{\alpha}_2 = \Omega_1 | \hat{\alpha}_2 = \tau_2 \).

By Lemma 55 (Completing Completeness) (i), we know that \( \Omega_2 | \tau_2 = | \Omega | \tau_2 \).

Hence we know that \( \Omega_2 | \Delta_2 = \Omega_2 | \hat{\alpha}_2 \leq \Omega_2 | \tau_2 \).

By i.h., we have \( \Delta \) and \( \Omega \) such that \( \Delta_2 \vdash \hat{\alpha}_2 : \Delta_2 | \tau_2 \rightarrow \Omega_2 \) and \( \Delta \rightarrow \Omega_2 \).

By rule \text{InstBin}, \( \Gamma \vdash \hat{\alpha} : \Delta \rightarrow \Omega_2 \).

By Lemma 33 (Extension Transitivity), \( \Omega \rightarrow \Omega_2 \).

Case \( \tau = \text{succ}(\tau_0) \):

Similar to the \( \tau = \tau_1 \) case, but simpler. 

Yes
Lemma 91 (Completeness of Checkeq).

Given \( \Gamma \rightarrow \Omega \) and \( \text{dom}(\Gamma) = \text{dom}(\Omega) \)
and \( \Gamma \vdash \sigma : \kappa \) and \( \Gamma \vdash \tau : \kappa \)
and \( [\Omega] \sigma = [\Omega] \tau \)
then \( \Gamma \vdash [\Gamma] \sigma \equiv [\Gamma] \tau : \kappa \rightarrow \Delta \)
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 \([\Gamma] \sigma = [\Gamma] \tau = 1\):
  
  \[ \Gamma \vdash 1 \equiv 1 : \ast \rightarrow \Gamma \Delta \]
  
  By CheckeqUnit
  
  Let \( \Omega' = \Omega \).

  Given

  - Case \( \Delta \rightarrow \Omega' \)
    
    \( \Delta = \Gamma \) and \( \Omega' = \Omega \)

  - Case \( \text{dom}(\Gamma) = \text{dom}(\Omega) \)
    
    Given

  - Case \( \Omega \rightarrow \Omega' \)
    
    By Lemma 32 (Extension Reflexivity)

- Case \([\Gamma] \sigma = [\Gamma] \tau = 0\):
  
  Similar to the case for 1, applying CheckeqZero instead of CheckeqUnit.

- Case \([\Gamma] \sigma = [\Gamma] \tau = \alpha\):
  
  Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit.

- Case \([\Gamma] \sigma = \hat{\alpha} \) and \([\Gamma] \tau = \hat{\beta}\):

  - If \( \hat{\alpha} = \hat{\beta} \):
    
    Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit.

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

    \[ \Gamma \rightarrow \Omega \]

    Given

    \[ \hat{\alpha} \notin \text{FV}(\hat{\beta}) \]

    By definition of FV(\(-\))

    \[ [\Omega] \sigma = [\Omega] \tau \]

    Given

    \[ [\Omega][\Gamma] \sigma = [\Omega][\Gamma] \tau \]

    By Lemma 29 (Substitution Monotonicity) (i) twice

    \[ [\Omega] \hat{\alpha} = [\Omega][\Gamma] \tau \]

    \[ [\Omega] \hat{\alpha} = [\Omega][\Gamma] \tau \]

    Given

    \[ \text{dom}(\Gamma) = \text{dom}(\Omega) \]

    \[ \Gamma \vdash \hat{\alpha} \equiv [\Gamma] \tau : \kappa \rightarrow \Delta \]

    By Lemma 90 (Completeness of Instantiation)

    
    By CheckeqInstL

- Case \([\Gamma] \sigma = \hat{\alpha} \) and \([\Gamma] \tau = 0 \) or zero or \( \alpha\):

  Similar to the previous case, except:

  \[ \hat{\alpha} \notin \text{FV}(\{-1\}) \]

  By definition of FV(\(-\))

  and similarly for 1 and \( \alpha. \)

- Case \([\Gamma] \tau = \hat{\alpha} \) and \([\Gamma] \sigma = 0 \) or zero or \( \alpha\):

  Symmetric to the previous case.

- Case \([\Gamma] \sigma = \hat{\alpha} \) and \([\Gamma] \tau = \text{succ}([\Gamma] t_0)\):

  If \( \hat{\alpha} \notin \text{FV}([\Gamma] t_0) \), then \( \hat{\alpha} \notin \text{FV}([\Gamma] t) \).

  Proceed as in the previous several cases.

  The other case, \( \hat{\alpha} \in \text{FV}([\Gamma] t_0) \), is impossible:
Proof of Lemma 92 \(\text{(Completeness of Checkeq)}\) \(\text{lem:checkeq-completeness}\)

We have \(\hat{\alpha} \preceq [\Gamma]\top\).
Therefore \(\hat{\alpha} < \text{succ}(\Gamma)\top\), that is, \(\hat{\alpha} < [\Gamma]t\).

By a property of substitutions, \([\Omega]\hat{\alpha} \preceq [\Omega][\Gamma]t\).

Since \(\Gamma \rightarrow \Omega\), by Lemma 92 (Substitution Monotonicity) (i), \([\Omega][\Gamma]t = [\Omega]t\), so \([\Omega]\hat{\alpha} < [\Omega]t\).
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** \([\Gamma]\sigma = [\Gamma]t_0 \oplus [\Gamma]t_1\) and \([\Gamma]t = [\Gamma]t_1 \oplus [\Gamma]t_2\):

  \(\Gamma \rightarrow \Omega\)
  \(\Gamma \vdash [\Gamma]\sigma_1 \triangleq [\Gamma]t_1 : \star \lor \Theta\)
  \(\Theta \rightarrow \Omega_0\)
  \(\Theta \rightarrow \Omega_0\)
  \(\text{dom}(\Theta) = \text{dom}(\Omega_0)\)
  \(\exists \Delta \rightarrow \Omega'\)
  \(\text{dom}(\Delta) = \text{dom}(\Omega')\)

  \(\Gamma \vdash [\Gamma]\sigma_2 \triangleq [\Theta][\Gamma]t_2 : \star \lor \Delta\)

  By i.h.

  By Lemma 33 (Extension Transitivity) \(\text{CheqeqBin}\)

  \(\Delta \rightarrow \Omega'\)

  \(\Omega_0 \rightarrow \Omega'\)

  \(\Gamma \vdash [\Gamma]\sigma_1 \lor [\Gamma]\sigma_2 \triangleq [\Gamma]t_1 \lor [\Gamma]t_2 : \star \lor \Delta\)

- **Case** \([\Gamma]\sigma = \hat{\alpha}\) and \([\Gamma]t = t_1 \oplus t_2\):
  Similar to the \(\hat{\alpha}/\text{succ}(\cdot)\) case, showing the impossibility of \(\hat{\alpha} \in \text{FV}([\Gamma]t_k)\) for \(k = 1\) and \(k = 2\).

- **Case** \([\Gamma]t = \hat{\alpha}\) and \([\Gamma]\sigma = \sigma_1 \lor \sigma_2\):
  Symmetric to the previous case.

\(\square\)

**Lemma 92** (Completeness of Elimeq).

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

1. If \(\text{mgu}(\sigma, t) = \emptyset\)
   then \(\Gamma / \sigma \triangleq t : \kappa \vdash (\Gamma, \Delta)\)
   where \(\Delta\) has the form \(\alpha_1 = t_1, \ldots, \alpha_n = t_n\)
   and for all \(u\) such that \(\Gamma \vdash u : \kappa\), it is the case that \([\Gamma, \Delta]u = \emptyset([\Gamma]u)\).

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

**Proof.** By induction on the structure of \([\Gamma]\sigma\) and \([\Gamma]t\).

- **Case** \([\Omega]\sigma = t = \text{zero}\):

  \(\text{mgu}(\text{zero}, \text{zero}) = \emptyset\)

  \(\Gamma / \text{zero} \triangleq \text{zero} : N \vdash \Gamma\)

  By rule \(\text{ElimeqZero}\)

  \(\exists \Delta \vdash \text{zero} : N \vdash \Gamma, \Delta\)

  where \(\Delta = \emptyset\).

  Suppose \(\Gamma \vdash u : \kappa'\).

  \([\Gamma, \Delta]u = [\Gamma]u\)

  where \(\Delta = \emptyset\)

  \(= \emptyset([\Gamma]u)\)

  where \(\emptyset\) is the identity

- **Case** \(\sigma = \text{succ}(\sigma')\) and \(t = \text{succ}(t')\):

  - Case \(\text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \emptyset\):

    \(\text{mgu}(\sigma', t') = \text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \emptyset\)

    By properties of unification

    \(\text{succ}(\sigma') = [\Gamma]\text{succ}(\sigma')\)

    By definition of substitution

    \(\sigma' = [\Gamma]\sigma'\)

    By injectivity of successor

    \(\text{succ}(t') = [\Gamma]\text{succ}(t')\)

    By definition of substitution

    \(t' = [\Gamma]t'\)

    By injectivity of successor

    \(\Gamma / \sigma' \triangleq t' : N \vdash \Gamma, \Delta\)

    By i.h.

    \(\exists [\Gamma, \Delta]u = \emptyset([\Gamma]u)\) for all \(u\) such that \(\ldots\)

    By rule \(\text{ElimeqSucc}\)
Proof of **Lemma 92** (Completeness of Elimeq)

First we establish some properties of the subterms:

\[
\sigma_1 \oplus \sigma_2 = [\Gamma](\sigma_1 \oplus \sigma_2) \quad \text{Given} \\
\sigma_1 = [\Gamma]\sigma_1 \quad \text{By definition of substitution} \\
\sigma_2 = [\Gamma]\sigma_2 \quad \text{By definition of substitution} \\
[\Gamma]t_1 = t_1 \quad \text{By definition of substitution} \\
[\Gamma]t_2 = t_2 \quad \text{By definition of substitution}
\]

- **Case** $\text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \bot$:

  - **Subcase** $\text{mgu}(\sigma', t') = \bot$:

    \[
    \frac{\text{mgu}(\text{succ}(\sigma'), \text{succ}(t')) = \bot}{\text{By properties of unification}}
    \]

    \[
    \frac{\text{succ}(\sigma') = [\Gamma]\text{succ}(\sigma')}{\text{Given}} \\
    \frac{\sigma' = [\Gamma]\sigma'}{\text{By definition of substitution}} \\
    \frac{\text{succ}(t') = [\Gamma]\text{succ}(t')}{\text{Given}} \\
    \frac{t' = [\Gamma]t'}{\text{By definition of successor}} \\
    \frac{\Gamma / \sigma' \triangleq t' : N \vdash \bot}{\text{By i.h.}} \\
    \]

  - **Subcase** $\text{mgu}(\text{succ}(\sigma'), t') = \bot$:

    \[
    \frac{\text{mgu}(\text{succ}(\sigma'), t') = \bot}{\text{By properties of unification}}
    \]

    \[
    \frac{\text{succ}(\sigma') = [\Gamma]\text{succ}(\sigma')}{\text{Given}} \\
    \frac{\sigma' = [\Gamma]\sigma'}{\text{By definition of substitution}} \\
    \frac{\text{succ}(t') = [\Gamma]\text{succ}(t')}{\text{Given}} \\
    \frac{t' = [\Gamma]t'}{\text{By definition of successor}} \\
    \frac{\Gamma / \sigma' \triangleq t' : N \vdash \bot}{\text{By i.h.}} \\
    \]

- **Case** $\sigma = \sigma_1 \oplus \sigma_2$ and $t = t_1 \oplus t_2$:

  First we establish some properties of the subterms:

  \[
  \sigma_1 \oplus \sigma_2 = [\Gamma](\sigma_1 \oplus \sigma_2) \quad \text{Given} \\
  \sigma_1 = [\Gamma]\sigma_1 \quad \text{By definition of substitution} \\
  \sigma_2 = [\Gamma]\sigma_2 \quad \text{By definition of substitution} \\
  t_1 \oplus t_2 = [\Gamma](t_1 \oplus t_2) \quad \text{Given} \\
  t_1 = [\Gamma]t_1 \quad \text{By definition of substitution} \\
  t_2 = [\Gamma]t_2 \quad \text{By definition of substitution}
  \]

  \[
  \text{mgu}(\sigma, t) = \bot \\
  \]

  - **Subcase** $\text{mgu}(\sigma, t) = \bot$:

    \[
    \begin{align*}
    \Gamma / \sigma_1 \triangleq t_1 : \kappa \vdash \bot \quad &\text{By i.h.} \\
    \hline \\
    \Gamma / \sigma_1 \oplus \sigma_2 \triangleq t_1 \oplus t_2 : \kappa \vdash \bot \quad &\text{By rule ElimeqBinBot}
    \end{align*}
    \]

    \[
    \begin{align*}
    \frac{\text{mgu}(\sigma_1, t_1) = \theta_1 \text{ and } \text{mgu}(\sigma_2, t_2) = \theta_2}{\text{Above line with } \sigma_2 \text{ as } u} \\
    \frac{\theta_1(\sigma_2) = [\Gamma]\sigma_2}{\text{Above line with } t_2 \text{ as } u} \\
    \frac{[\Gamma]t_1 \oplus [\Gamma]t_2}{\text{By transitivity of equality}} \\
    \frac{[\Gamma]t_1 \oplus [\Gamma]t_2}{\text{By Lemma 29 (Substitution Monotonicity)}} \\
    \end{align*}
    \]

    \[
    \Gamma / \sigma_1 \oplus \sigma_2 \triangleq t_1 \oplus t_2 : \kappa \vdash \bot \quad \text{By i.h.} \\
    \]

    \[
    \begin{align*}
    \frac{\text{mgu}(\sigma_1, t_1) = \theta_1 \text{ and } \text{mgu}(\sigma_2, t_2) = \theta_2}{\text{Above line with } \sigma_2 \text{ as } u} \\
    \frac{\theta_1(\sigma_2) = [\Gamma]\sigma_2}{\text{Above line with } t_2 \text{ as } u} \\
    \frac{[\Gamma]t_1 \oplus [\Gamma]t_2}{\text{By transitivity of equality}} \\
    \frac{[\Gamma]t_1 \oplus [\Gamma]t_2}{\text{By Lemma 29 (Substitution Monotonicity)}} \\
    \end{align*}
    \]

    \[
    \Gamma / \sigma_1 \oplus \sigma_2 \triangleq t_1 \oplus t_2 : \kappa \vdash \bot \quad \text{By rule ElimeqBin}
    \]

    \[
    \begin{align*}
    \frac{\text{mgu}(\sigma_1, t_1) = \theta_1 \text{ and } \text{mgu}(\sigma_2, t_2) = \theta_2}{\text{Above line with } \sigma_2 \text{ as } u} \\
    \frac{\theta_1(\sigma_2) = [\Gamma]\sigma_2}{\text{Above line with } t_2 \text{ as } u} \\
    \frac{[\Gamma]t_1 \oplus [\Gamma]t_2}{\text{By transitivity of equality}} \\
    \frac{[\Gamma]t_1 \oplus [\Gamma]t_2}{\text{By Lemma 29 (Substitution Monotonicity)}} \\
    \end{align*}
    \]

    \[
    \Gamma / \sigma_1 \oplus \sigma_2 \triangleq t_1 \oplus t_2 : \kappa \vdash \bot \quad \text{By rule ElimeqBin}
    \]
**Definition of context substitution.**

**Proof.**

Part (i): By induction on the given derivation, using the given “for all” at the leaves.

Case \( \sigma = \alpha \):

- Subcase \( \alpha \in \text{FV}(t) \):

  \[
  \text{mgu}(\alpha, t) = t/\alpha \quad \text{By properties of unification}
  \]

- Subcase \( \alpha \notin \text{FV}(t) \):

  \[
  \text{mgu}(\alpha, t) = [t/\alpha] \quad \text{By properties of unification}
  \]

  \[
  \Gamma / \alpha \leq t : \kappa \quad \text{By rule ElimeqUvarL}
  \]

\[
\begin{align*}
\Gamma, \Delta_1 &/ [\Gamma, \Delta_1] \sigma_2 \equiv [\Gamma, \Delta_1] t_2 : \kappa \vdash \Gamma, \Delta_2 & \text{By i.h.} \\
\end{align*}
\]

** Proof of Lemma 94 (Completeness of Elimeq).**

If \( \Delta \) has the form \( \alpha_1 = t_1, \ldots, \alpha_n = t_n \) and, for all \( u \) such that \( \Gamma \vdash u : \kappa \), it is the case that \( [\Gamma, \Delta] u = \theta([\Gamma] u) \),
then:

(i) If \( \Gamma \vdash \text{A} \) type then \( [\Gamma, \Delta] \text{A} = [\Gamma] \text{A} \).

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

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

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

**Proof.** Part (i): By induction on the given derivation, using the given “for all” at the leaves.

Part (ii): By induction on the given derivation, using part (i) in the \( \rightarrow \text{Var} \) case.

Part (iii): By induction on \( \Delta \). In the base case (\( \Delta = . \)), 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 \( e = (e_0 : \Lambda) \) case.

**Lemma 94 (Completeness of Propequiv).**

Given \( \Omega \rightarrow \Omega \) and \( \Gamma \vdash \text{P prop} \) and \( \Gamma \vdash \text{Q prop} \)
and \( [\Omega] \text{P} = [\Omega] \text{Q} \) then \( \Gamma \vdash [\Gamma] \text{P} \equiv [\Gamma] \text{Q} \rightarrow \Delta \)
where \( \Delta \rightarrow \Omega \) and \( \Omega \rightarrow \Omega \).

**Proof.** By induction on the given derivations. There is only one possible case:

- Case \( \Gamma \vdash \sigma_1 : \mathbb{N} \quad \Gamma \vdash \sigma_2 : \mathbb{N} \quad \text{EqProp} \)

\[
\Gamma \vdash \sigma_1 = \sigma_2 \quad \text{prop}
\]

- Case \( \Gamma \vdash \tau_1 : \mathbb{N} \quad \Gamma \vdash \tau_2 : \mathbb{N} \quad \text{EqProp} \)

\[
\Gamma \vdash \tau_1 = \tau_2 \quad \text{prop}
\]
Lemma 95 (Completeness of Checkprop).

If \( \Delta \rightarrow \Omega \) and \( \text{dom}(\Gamma) = \text{dom}(\Omega) \)
and \( \Gamma \vdash P \ prop \)
and \( (\Gamma)P = P \)
and \( (\Omega)\Gamma \vdash (\Omega)P \ true \)
then \( \Gamma \vdash P \ true \ \rightarrow \Delta \)
where \( \Delta \rightarrow \Omega' \) and \( \Omega \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \).

Proof. Only one rule, \text{DeclCheckpropEd} 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 91 (Completeness of Checkeq), \( \Gamma \vdash (\Gamma)t_1 \equiv (\Gamma)t_2 : N \ \rightarrow \Delta \) where \( \Delta \rightarrow \Omega' \) and \( \Omega \rightarrow \Omega' \).
By \text{CheckpropEd}, \( \Gamma \vdash (t_1 = t_2) \ true \ \rightarrow \Delta \).

L'.2 Completeness of Equivalence and Subtyping

Lemma 96 (Completeness of Equiv).

If \( \Delta \rightarrow \Omega \) and \( \Gamma \vdash A \ type \) and \( \Gamma \vdash B \ type \)
and \( (\Omega)\Gamma \vdash (\Omega)A \equiv (\Omega)B \)
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 \( (\Omega)A = (\Omega)B \) 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{impos} (impossible), \text{fully written out}, and 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.
Proof of Lemma 96 (Completeness of Equiv)

\[\Gamma \vdash B\]

<table>
<thead>
<tr>
<th>(\Gamma)</th>
<th>(\beta)</th>
<th>(B')</th>
<th>(\forall\beta. B')</th>
<th>(\exists\beta. B')</th>
<th>(\top)</th>
<th>(\bot)</th>
<th>(A_1 \oplus A_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\vdash)</td>
<td>Implies</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td></td>
</tr>
<tr>
<td>(\wedge)</td>
<td>imposs.</td>
<td>With</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td></td>
</tr>
<tr>
<td>(\forall\alpha. A')</td>
<td>imposs.</td>
<td>imposs.</td>
<td>Forall</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td></td>
</tr>
<tr>
<td>(\exists\alpha. A')</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>Exists</td>
<td>imposs.</td>
<td>imposs.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>2.Units</td>
<td>imposs.</td>
<td>2.BEx.Unit</td>
<td></td>
</tr>
<tr>
<td>[(\Gamma)]A</td>
<td>(\alpha)</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>2.AEx.Unit</td>
</tr>
<tr>
<td>(\hat{\alpha})</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>2.AEx.Unit</td>
<td>2.AEx.Uvar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>2.AEx.Unit</td>
<td>2.AEx.Uvar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A_1 \oplus A_2)</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>imposs.</td>
<td>2.BEx.Bin</td>
<td></td>
</tr>
</tbody>
</table>

- **Case** \(\Gamma \vdash P\ prop\ \Gamma \vdash A_0\ type\)

  \(\Gamma \vdash \top\) by Subderivation

  \(\Gamma \vdash A_0\) by Type Subderivation

  \(\Gamma \vdash Q \supset B_0\) type

  \(\Gamma \vdash Q\ prop\) by Given

  \(\Gamma \vdash B_0\) type

  \(\Gamma \vdash [\Gamma] P \equiv [\Gamma] Q \rightarrow \Theta\)

  \(\Theta \rightarrow Q_0\)

  \(\Omega \rightarrow \Omega_0\)

  \(\Gamma \vdash \Theta\) by Lemma 48 (Prop Equivalence Extension)

  \(\Gamma \vdash A_0\) type

  \(\Gamma \vdash B_0\) type

  [\(\Omega\)]A_0 = [\(\Omega\)]B_0

  [\(\Omega\)]A_0 = [\(\Omega\)]B_0

  [\(\Omega\)]A_0 = [\(\Omega\)]B_0

  \(\Gamma \vdash [\Gamma] A_0 \equiv [\Gamma] B_0 \rightarrow \Delta\) by Lemma 55 (Completing Completeness) (ii) twice

  \(\Delta \rightarrow \Omega'\)

  \(\Omega_0 \rightarrow \Omega'\)

- **Case** [\(\Gamma\)]A

  This case of the rule concluding the first derivation coincides with the Implies entry in the table.

  We have \(\Omega\) | A = \(\Omega\) | B, that is, \([\Omega] P \supset A_0 = [\Omega] B_0\).

  Because \(\Omega\) is predicative, \(B\) must have the form \(Q \supset B_0\), where \(\Omega\) | P = \(\Omega\) | Q and \(\Omega\) | A_0 = \(\Omega\) | B_0.

  \(\Gamma \vdash P\ prop\)

  \(\Gamma \vdash Q\ prop\) by Inversion on rule ImpliesWF

  \(\Gamma \vdash [\Gamma] P \equiv [\Gamma] Q \rightarrow \Theta\)

  \(\Theta \rightarrow Q_0\)

  \(\Omega \rightarrow \Omega_0\)

  \(\Gamma \vdash \Theta\) by Lemma 48 (Prop Equivalence Extension)

  \(\Gamma \vdash A_0\) type

  \(\Gamma \vdash B_0\) type

  \(\Omega\) | A_0 = \(\Omega\) | B_0

  \(\Omega\) | A_0 = \(\Omega\) | B_0

  \(\Omega\) | A_0 = \(\Omega\) | B_0

  \(\Gamma \vdash [\Gamma] A_0 \equiv [\Gamma] B_0 \rightarrow \Delta\) by Lemma 55 (Completing Completeness) (ii) twice

  \(\Delta \rightarrow \Omega'\)

  \(\Omega_0 \rightarrow \Omega'\)

- **Case** [\(\Gamma\)]A

  This case coincides with the Forall entry in the table.

- **Case** \(\Gamma \vdash \forall \alpha: \kappa. A_0\) type

  This case coincides with the Forall entry in the table.
Proof of Lemma 96 (Completeness of Equiv)

\[ \Gamma \rightarrow \Omega \]
\[ \Gamma, \alpha : \kappa \rightarrow \Omega, \alpha : \kappa \]
\[ \Gamma, \alpha : \kappa \vdash \Lambda_0 \text{ type} \]
\[ B = \forall \alpha : \kappa . B_0 \]
\[ [\Omega] \Lambda_0 = [\Omega] B_0 \]
\[ \Gamma, \alpha : \kappa \vdash [\Gamma] \Lambda_0 \equiv [\Gamma] B_0 \vdash \Delta_0 \]
\[ \Omega, \alpha : \kappa \rightarrow \Omega_0 \]
\[ \Gamma, \alpha : \kappa \vdash [\Gamma] B_0 \vdash \Delta \]
\[ \Omega \rightarrow \Omega' \text{ and } \Omega_0 = \{ \Omega', \alpha : \kappa, \ldots \} \]
\[ \Delta_0 = \{ \Delta, \alpha : \kappa, \Delta' \} \]
\[ \Delta \rightarrow \Omega' \]
\[ \Gamma \vdash [\forall \alpha : \kappa . [\Gamma] \Lambda_0 \equiv [\forall \alpha : \kappa . [\Gamma] B_0 \vdash \Delta] \]
\[ \equiv \forall \]
\[ \equiv \forall \]

• Case ExistsWF: Similar to the ForallWF case. (This is the Exists entry in the table.)

• Case BinWF: If BinWF also concluded the second derivation, then the proof is similar to the ImpliesWF case, but on the first premise, using the i.h. instead of Lemma 94 (Completeness of Propequiv). This is the 2.Bins entry in the lower right corner of the table.

If BinWF did not conclude the second derivation, we are in the 2.AEx.Bin or 2.BEx.Bin entries; see below.

In the remainder, we cover the 4 × 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

\[ \equiv \forall \]
\[ \equiv \forall \]

• Case 2.Unvars: [\Gamma] A = [\Gamma] B = \alpha

\[ \equiv \forall \]
\[ \equiv \forall \]

• Case 2.AExUnit: [\Gamma] A = \alpha \text{ and } [\Gamma] B = 1

\[ \equiv \forall \]
\[ \equiv \forall \]
Proof of Lemma 96 (Completeness of Subtyping)

If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash A$ type and $\Gamma \vdash B$ type and $[\Omega][\Gamma]A \triangleleft \triangleright [\Omega][\Gamma]B$ then there exist $\Delta$ and $\Omega'$ such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$ and $\Gamma \vdash [\Gamma]A \leq \leq [\Gamma]B \rightarrow \Delta$.

• Case 2.BExUnit: $[\Gamma]A = 1$ and $[\Gamma]B = \hat{\alpha}$
Symmetric to the 2.AExUnit case.

• Case 2.AEx.Unvar: $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = \alpha$
Similar to the 2.AEx.Unit case, using $\beta = [\Omega][\beta] = [\Gamma][\beta]$ and $\hat{\alpha} \notin \text{FV}(\beta)$.

• Case 2.BExUnvar: $[\Gamma]A = 1$ and $[\Gamma]B = \hat{\alpha}$
Symmetric to the 2.AExUnvar case.

• Case 2.AEx.SameEx: $[\Gamma]A = \hat{\alpha} = \hat{\beta} = [\Gamma]B$

  - $\hat{\alpha} \in \text{FV}([\Gamma][\beta])$:
    We have $\hat{\alpha} \triangleleft [\Gamma][\beta]$. Therefore $\hat{\alpha} = [\Gamma][\beta]$, or $\hat{\alpha} \triangleleft [\Gamma][\beta]$.
    But we are in Case 2.AEx.Unvar, so the former is impossible.
    Therefore, $\hat{\alpha} \triangleleft [\Gamma][\beta]$.
    By a property of substitutions, $[\Omega][\hat{\alpha}] \triangleleft [\Omega][\Gamma][\beta]$.
    Since $\Gamma \rightarrow \Omega$, by Lemma 29 (Substitution Monotonicity) (iii), $[\Omega][\Gamma][\beta] = [\Omega][\hat{\beta}]$, so $[\Omega][\hat{\alpha}] \triangleleft [\Omega][\hat{\beta}]$.
    But it is given that $[\Omega][\hat{\alpha}] = [\Omega][\hat{\beta}]$, a contradiction.
    - $\hat{\alpha} \not\in \text{FV}([\Gamma][\beta])$:
      $\Gamma \vdash \hat{\alpha} := [\Gamma][\beta] : \star \rightarrow \Delta$ By Lemma 90 (Completeness of Instantiation)
      $\Delta \rightarrow \Omega'$ By Lemma 90 (Completeness of Instantiation)
      $\Omega \rightarrow \Omega'$

• Case 2.AEx.Bin: $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = B_1 \oplus B_2$
Since $[\Gamma]B$ is an arrow, it cannot be exactly $\hat{\alpha}$. By the same reasoning as in the previous case (2.AEx.Unvar), $\hat{\alpha} \not\in \text{FV}([\Gamma][\beta])$.

  - $\hat{\alpha} \vdash [\Gamma][\beta] : \star \rightarrow \Delta$ By Lemma 90 (Completeness of Instantiation)
  - $\Delta \rightarrow \Omega'$ By Lemma 90 (Completeness of Instantiation)
  - $\Omega \rightarrow \Omega'$
  - $\Gamma \vdash [\Gamma][\beta] : \star \rightarrow \Delta$ By Lemma 90 (Completeness of Instantiation)

• Case 2.BEx.Bin: $[\Gamma]A = A_1 \oplus A_2$ and $[\Gamma]B = \hat{\beta}$
Symmetric to the 2.AEx.Bin case, applying $\text{InstantiateR}$ instead of $\text{InstantiateL}$.

**Theorem 9** (Completeness of Subtyping).
If $\Gamma \rightarrow \Omega$ and $\text{dom}(\Gamma) = \text{dom}(\Omega)$ and $\Gamma \vdash A$ type and $\Gamma \vdash B$ type and $[\Omega][\Gamma]A \triangleleft \triangleright [\Omega][\Gamma]B$ then there exist $\Delta$ and $\Omega'$ such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$ and $\Gamma \vdash [\Gamma]A \leq \leq [\Gamma]B \rightarrow \Delta$. ✷
Proof of Theorem 9 (Completeness of Subtyping)


It is straightforward to show $\text{dom}(\Delta) = \text{dom}(\Omega')$; for examples of the necessary reasoning, see the proof of Theorem 11 (Completeness of Algorithmic Typing).

We have $[\Omega] \vdash [\Omega]A \leq [\Omega]B$.

- Case $[\Omega][\Gamma] \vdash [\Omega]A$ type nonpos($[\Omega]A$)

$$[\Omega][\Gamma] \vdash [\Omega]A \leq [\Omega]A$$ Refl

First, we observe that, since applying $\Omega$ as a substitution leaves quantifiers alone, the quantifiers that head A must also head B. For convenience, we alpha-vary B to quantify over the same variables as A.

- If $A$ is headed by $\forall$, then $[\Omega]A = (\forall \alpha : \kappa. [\Omega]A_0) = (\forall \alpha : \kappa. [\Omega]B_0) = [\Omega]B$.

Let $\Gamma_0 = (\Gamma, \alpha : \kappa, \beta : \kappa)$. Let $\Omega_0 = (\Omega, \alpha : \kappa, \beta : \kappa, \delta : \kappa = \alpha)$.

* If $\text{pol}(A_0) \in \{\cdot, \cdot\}$, then:

  (We elide the straightforward use of lemmas about context extension.)

$$[\Omega_0][\Gamma_0] \vdash [\Omega]A_0 \leq [\Omega]A_0$$ By $\leq$Refl

$$[\Omega_0][\Gamma_0] \vdash [\Omega][\delta/\alpha]A_0 \leq A_0$$ By def. of subst.

$$\Delta_0 \rightarrow \Omega_0'$$ By i.h. (fewer quantifiers)

$$\Omega_0 \rightarrow \Omega_0'$$

$$\Gamma_0 \vdash [\delta/\alpha][\Gamma_0]A_0 \leq [\Gamma]B_0 \vdash \Delta_0$$

$$\Gamma_0 \vdash [\delta/\alpha][\Gamma]A_0 \leq [\Gamma]B_0 \vdash \Delta_0$$

$\delta$ unsolved in $\Gamma_0$

$$\Gamma_0 \vdash [\delta/\alpha][\Gamma][\Gamma_0]A_0 \leq [\Gamma][\Gamma]B_0 \vdash \Delta_0$$

$\Gamma_0$ substitutes as $\Gamma$.

$$\Gamma, \alpha : \kappa \vdash \forall \alpha : \kappa. [\Gamma][\Gamma_0]A_0 \leq [\Gamma][\Gamma]B_0 \vdash \Delta, \alpha : \kappa, \Theta$$ By $\ll\forall L$

$$\Gamma \vdash \forall \alpha : \kappa. [\Gamma]A_0 \leq \forall \alpha : \kappa. [\Gamma]B_0 \vdash \Delta$$ By $\ll\forall R$

* $\Delta \rightarrow \Omega$ By Lemma 106 (Completeness of Equiv)

* $\Omega \rightarrow \Omega'$ By Lemma 106 (Completeness of Equiv)

* $\Gamma \vdash [\Gamma][\forall \alpha : \kappa. A_0] \leq [\Gamma][\forall \alpha : \kappa. B_0] \vdash \Delta$ By def. of subst.

- If $A$ is not headed by $\forall$:

We have $\text{nonneg}([\Omega]A)$. Therefore $\text{nonneg}(A)$, and thus $A$ is not headed by $\exists$. Since the same quantifiers must also head $B$, the conditions in rule $\ll\exists L$ are satisfied.

$$\Gamma \rightarrow \Omega$$ Given

$$\Gamma \vdash [\Gamma]A \equiv [\Gamma]B \vdash \Delta$$ By Lemma 96 (Completeness of Equiv)

$$\Delta \rightarrow \Omega'$$

$$\Omega \rightarrow \Omega'$$

$$\Gamma \vdash [\Gamma][\forall \alpha : \kappa. A_0] \leq [\Gamma][\forall \alpha : \kappa. B_0] \vdash \Delta$$ By $\ll\exists L$

- Case $\ll\forall L < \ll\forall R$

Symmetric to the $\ll\forall L$ case, using $\ll\exists L$ (or $\ll\exists R$), and $\ll\exists R < \ll\exists L$ instead of $\ll\exists R < \ll\exists L$.

- Case $\ll\forall L$

$$\ll\forall L < \ll\forall R$

Symmetric to the $\ll\forall L$ case, using $\ll\exists L$ (or $\ll\exists R$), and $\ll\exists R < \ll\exists L$ instead of $\ll\forall L$.

$$\ll\forall L$$

We begin by considering whether or not $[\Omega]B$ is headed by a universal quantifier.

- $[\Omega]B = (\forall \beta : \kappa'. B')$:

$$[\Omega][\Gamma, \beta : \kappa' \vdash [\Omega]A \leq [\Omega]B$$ By Lemma 5 (Subtyping Inversion)

The remaining steps are similar to the $\ll\forall R$ case.
**Case**

$$\exists \beta : \kappa . [\Omega]B \not\vdash [\Omega]B$$

Subderivation

$$\Gamma \vdash \tau : \kappa$$

Given

$$\Gamma' \Rightarrow \Omega$$

By $$\Rightarrow$$-Marker

$$\Gamma', \Delta, \hat{\alpha} : \kappa \vdash \Omega, \Delta, \hat{\alpha} : \kappa = \tau$$

By $$\Rightarrow$$-Solve

$$[\Omega] \Gamma = [\Omega_0][\Gamma, \Delta, \hat{\alpha} : \kappa]$$

By definition of context application (lines 16, 13)

$$[\Omega] \Gamma \vdash [\tau / \alpha][\Omega]A_0 \leq [\Omega]B$$

Subderivation

$$[\Omega_0][\Gamma', \Delta, \hat{\alpha} : \kappa] \vdash [\tau / \alpha][\Omega]A_0 \leq [\Omega]B$$

By above equality

$$[\Omega_0][\Gamma', \Delta, \hat{\alpha} : \kappa] \vdash [\Omega_0][\Delta / \alpha][\Omega]A_0 \leq [\Omega]B$$

By definition of substitution

$$[\Omega_0][\Gamma', \Delta, \hat{\alpha} : \kappa] \vdash [\Omega_0][\Delta / \alpha][\Omega]A_0 \leq [\Omega_0]B$$

By definition of substitution

$$[\Omega_0][\Gamma', \Delta, \hat{\alpha} : \kappa] \vdash [\Omega_0][\hat{\alpha} / \alpha][\Omega]A_0 \leq [\Omega_0]B$$

By distributivity of substitution

$$\Gamma', \Delta, \hat{\alpha} : \kappa \vdash [\Gamma'][\hat{\alpha} / \alpha][\Omega]A_0 < [\Gamma'_B] \Delta_0$$

By i.h. (A lost a quantifier)

$$\Delta_0 \Rightarrow \Omega''$$

$$\Omega_0 \Rightarrow \Omega''$$

$$\Gamma', \Delta, \hat{\alpha} : \kappa \vdash [\Gamma'][\hat{\alpha} / \alpha][\Omega]A_0 \leq [\Gamma']B \vdash \Delta_0$$

By definition of substitution

$$\Gamma', \Delta, \hat{\alpha} : \kappa \vdash \Delta_0$$

By Lemma 50 (Subtyping Extension)

$$\Gamma \Rightarrow \Delta$$

By Lemma 22 (Extension Inversion) (ii)

$$\Omega'' = (\Omega', \Delta, \hat{\alpha}, \Theta)$$

By Lemma 22 (Extension Inversion) (ii)

$$\Delta \Rightarrow \Omega'$$

$$\Omega' \Rightarrow \Omega''$$

Above

$$\Omega, \Delta, \hat{\alpha} : \kappa \vdash \tau : \Omega', \Delta, \hat{\alpha} : \Theta$$

By above equalities

$$\Omega \Rightarrow \Omega'$$

By Lemma 22 (Extension Inversion) (ii)

$$\Gamma', \Delta, \hat{\alpha} : \kappa \vdash [\Gamma][\hat{\alpha} / \alpha][\Gamma_0]A_0 \leq [\Gamma']B \vdash \Delta, \Delta, \hat{\alpha}, \Theta$$

By above equality $$\Delta_0 = (\Delta, \Delta, \hat{\alpha}, \Theta)$$

$$[\Gamma]B$$ not headed by $$\forall$$

From the case assumption

$$\Gamma \vdash \forall \alpha : \kappa . [\Gamma][\alpha]A_0 \leq [\Gamma']B \vdash \Delta$$

By $$\forall$$-Substitution

$$\Gamma \vdash [\Gamma][\forall \alpha : \kappa . A]_0 \leq [\Gamma']B \vdash \Delta$$

By definition of substitution

$$\exists \beta : \kappa . [\Omega]B \leq [\forall \beta : \kappa . [\Omega]B]_0$$

$$\Omega$$ predicated

$$[\Omega] \Gamma \vdash [\Omega]A \leq [\forall \beta : \kappa . [\Omega]B]_0$$

Given

$$[\Omega] \Gamma \vdash [\forall \beta : \kappa . [\Omega]B]_0$$

By above equality

$$[\Omega] \Gamma \vdash [\forall \beta : \kappa . [\Omega]B]_0$$

Subderivation

$$[\Omega, \beta : \kappa] \vdash [\Omega, \beta : \kappa]A \leq [\forall \beta : \kappa . [\Omega]B]_0$$

By definitions of substitution

$$\Gamma, \beta : \kappa \vdash [\Gamma, \beta : \kappa]A \leq [\Gamma, \beta : \kappa]B \vdash \Delta'$$

By i.h. (B lost a quantifier)

$$\Delta' \Rightarrow \Omega_0'$$

$$\Omega, \beta : \kappa \vdash \Omega_0'$$

$$\Gamma, \beta : \kappa \vdash [\Gamma]A \leq [\Gamma]B \vdash \Delta'$$

By definition of substitution

$$\Gamma, \beta : \kappa \vdash \Delta'$$

By Lemma 43 (Instantiation Extension)

$$\Delta' = (\Delta, \beta : \kappa, \Theta)$$

By Lemma 22 (Extension Inversion) (i)

$$\Gamma \Rightarrow \Delta$$

$$\Delta, \beta : \kappa, \Theta \Rightarrow \Omega_0'$$

By $$\Delta' \Rightarrow \Omega_0'$$ and above equality

$$\Omega_0' = (\Omega', \beta : \kappa, \Omega_R)$$

By Lemma 22 (Extension Inversion) (i)

$$\Delta \Rightarrow \Omega'$$

$$\Rightarrow$$-Solve
Proof of Theorem 9 (Completeness of Subtyping)  

\[ \Gamma, \beta : \kappa \vdash \Gamma \alpha <: \Gamma \beta B_0 \vdash \Delta, \beta : \kappa, \Theta \]

By above equality

\[ \Omega, \beta : \kappa \rightarrow \Omega', \beta : \kappa, \Omega_R \]

By above equality

\[ \Omega \rightarrow \Omega' \]

By Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash [\Gamma]A <: \forall \beta : \kappa. [\Gamma]B_0 \vdash \Delta \]

By \( \vdash \forall \)

\[ \Gamma \vdash [\Gamma]A < :: [\Gamma] (\forall \beta : \kappa. B_0) \vdash \Delta \]

By definition of substitution

\[ \frac{\text{Case}}{[\Omega] \Gamma, \alpha : k \vdash [\Omega] A_0 \leq + \exists \alpha : k. [\Omega] B} {[\Omega] \Gamma \vdash \exists \alpha : k. [\Omega] A_0 \leq + \exists \alpha : k. [\Omega] B} \leq \exists \alpha \]

\( \Omega \) predicative

\[ [\Omega] \Gamma \vdash [\Omega] A \leq + [\Omega] B \]

Given

\[ [\Omega] \Gamma \vdash [\Omega] \exists \alpha : k. A_0 \leq + [\Omega] B \]

By above equality

\[ [\Omega] \Gamma, \alpha : k \vdash [\Omega] A_0 \leq + [\Omega] B \]

Subderivation

\[ [\Omega, \alpha : k] (\Gamma, \alpha : k) \vdash [\Omega, \alpha : k] A_0 \leq + [\Omega, \alpha : k] B \]

By definitions of substitution

\[ \Gamma, \alpha : k \vdash \Gamma, \beta : k A_0 \leq + \Gamma, \beta : k B \vdash \Delta' \]

By i.h. (\( A \) lost a quantifier)

\[ \Delta' \rightarrow \Omega'_0 \]

By definition of substitution

\[ \Omega, \alpha : k \rightarrow \Omega'_0 \]

By definition of substitution

\[ \Gamma, \alpha : k \vdash [\Gamma] A < : [\Gamma] B \vdash \Delta' \]

By above equality

\[ \Delta' = (\Delta, \alpha : k, \Theta) \]

By Lemma 43 (Instantiation Extension) (i)

\[ \Gamma \rightarrow \Delta \]

By definition of substitution

\[ \Delta, \alpha : k, \Theta \rightarrow \Omega'_0 \]

By \( \Delta' \rightarrow \Omega'_0 \) and above equality

\[ \Omega'_0 = (\Omega', \alpha : k, \Omega_R) \]

By Lemma 22 (Extension Inversion) (i)

\[ \Delta \rightarrow \Omega' \]

By definition of substitution

\[ \Gamma, \alpha : k \vdash [\Gamma] A_0 \leq + [\Gamma] B \vdash \Delta, \alpha : k, \Theta \]

By above equality

\[ \Omega, \alpha : k \rightarrow \Omega', \alpha : k, \Omega_R \]

By above equality

\[ \Omega \rightarrow \Omega' \]

By Lemma 33 (Extension Transitivity)

\[ \Gamma \vdash \exists \alpha : k. [\Gamma] A_0 \leq + [\Gamma] B \vdash \Delta \]

By \( \vdash \exists \forall \)

\[ \Gamma \vdash [\Gamma] (\exists \alpha : k. A_0) \leq + [\Gamma] B \vdash \Delta \]

By definition of substitution

We consider whether \( [\Omega] A \) is headed by an existential.

If \( [\Omega] A = \exists \alpha : k'. A' \)

\[ [\Omega] \Gamma, \alpha : k' \vdash A' \leq + [\Omega] B \]

By Lemma 5 (Subtyping Inversion)

The remaining steps are similar to the \( \leq \exists \) case.

If \( [\Omega] A \) not headed by \( \exists \):

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

Subderivation

\[ \Gamma \rightarrow \Omega \]

Given

\[ \Gamma, \Gamma \vdash \Omega, \Gamma \]

By \( \vdash \)

\[ \Gamma, \Gamma, \Gamma : \alpha : \kappa \rightarrow \Gamma \vdash \Omega, \Gamma, \Gamma : \alpha : \kappa \rightarrow \tau \]

By \( \vdash \)

\[ [\Omega] \Gamma = [\Omega, \Gamma, \Gamma : \alpha : \kappa] \]

By definition of context application (lines 16, 13)

\[ [\Omega] \Gamma \vdash [\Omega] A \leq + [\tau / \beta] B_0 \]

Subderivation

\[ [\Omega_0] (\Gamma, \Gamma, \Gamma : \alpha : \kappa) \vdash [\Omega] A \leq + [\tau / \beta][\Omega] B_0 \]

By above equality

\[ [\Omega_0] (\Gamma, \Gamma, \Gamma : \alpha : \kappa) \vdash [\Omega] A \leq + [\tau / \beta][\Omega_0] B_0 \]

By definition of substitution

\[ [\Omega_0] (\Gamma, \Gamma, \Gamma : \alpha : \kappa) \vdash [\Omega] A \leq + [\tau / \beta][\Omega_0] B_0 \]

By definition of substitution

\[ [\Omega_0] (\Gamma, \Gamma, \Gamma : \alpha : \kappa) \vdash [\Omega] A \leq + [\tau / \beta][\Omega_0] B_0 \]

By distributivity of substitution
L’.3 Completeness of Typing

**Theorem 10** (Completeness of Match Coverage).
If $\mathcal{O} \vdash \mathcal{O} \Pi \text{ covers } [\mathcal{O}] \bar{A}$ and $\Gamma \rightarrow \Omega$ and $\Gamma \vdash \bar{A}$ types and $[\Gamma] \bar{A} = \bar{A}$
then $\Gamma \vdash \Pi \text{ covers } \bar{A}$.

**Proof.** By induction on the derivation of the given coverage rule.

- **Case**
  
  $[\mathcal{O}] \Gamma \vdash \cdot \Rightarrow e_1 \ldots \text{ covers } \cdot$  
  
  Apply **DeclCoversEmpty**

- **Cases**
  
  DeclCoversVar DeclCovers1 DeclCovers× DeclCovers− DeclCovers−

  Use the i.h. and apply the corresponding algorithmic coverage rule.

- **Case**
  
  $\theta = \text{mgu}(t_1, t_2)$  
  
  $[\mathcal{O}] \Gamma \vdash [\mathcal{O}] \Pi \text{ covers } \theta \bar{A}$  
  
  $\text{Subderivation}$

  $\text{mgu}(t_1, t_2) = \theta$  
  
  $\Gamma / t_1 : t_1 : \kappa \vdash \Gamma \Theta$  
  
  By Lemma 92 (Completeness of Elimeq) (1)

  $\Gamma / [\Gamma] t_1 : \kappa \vdash [\Gamma] \Theta$  
  
  Follows from given assumption

  $\text{Subderivation}$

  $[\theta][\mathcal{O}] \Gamma \vdash [\theta] \mathcal{O} \Pi \text{ covers } [\theta] \bar{A}$  
  
  By Lemma 93 (Substitution Upgrade) (iii)

  $[\theta][\mathcal{O}] \Pi = [\mathcal{O}, \bar{O}] \Pi$  
  
  By Lemma 93 (Substitution Upgrade) (iv)

  $([\theta] \bar{A}, [\theta] \bar{A}) = ([\Gamma] \bar{A}, [\Gamma] \bar{A})$  
  
  By Lemma 95 (Substitution Upgrade) (i)

  $[\mathcal{O} \Theta] [\mathcal{O} \Theta] \Pi \text{ covers } [\mathcal{O} \Theta] \bar{A}$  
  
  By above equalities

  $\Gamma, \Theta \vdash [\Gamma, \Theta] \Pi \text{ covers } [\Gamma, \Theta] A_0, [\Gamma, \Theta] \bar{A}$  
  
  By i.h.

  $\Gamma \vdash \Pi \text{ covers } A_0 \wedge (t_1 = t_2), \bar{A}$  
  
  By **CoversEq**

- **Case**
  
  $\text{mgu}(t_1, t_2) = \bot$  
  
  $[\mathcal{O}] \Gamma \vdash [\mathcal{O}] \Pi \text{ covers } A_0 \wedge (t_1 = t_2), \bar{A}$

  By **DeclCoversEqBot**
Proof of Theorem 10 (Completeness of Match Coverage). Given \( \Gamma \rightarrow \Omega \) such that \( \text{dom}(\Gamma) = \text{dom}(\Omega) \): 

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

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

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

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

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

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

Proof. By induction, using the measure in Definition 7.

- Case \( \{ x : A : p \} \in (\Omega) \Gamma \)
  \[ (x : A : p) \in (\Omega) \Gamma \]
  Premise
  Given
  \( \Gamma \rightarrow \Omega \)
  From definition of context application

  \( \{ x : A' : p \} \in \Gamma \) where \( (\Omega)A' = A \)

  Let \( \Delta = \Gamma \).
  Let \( \Omega' = \Omega \).

  \( \Gamma \rightarrow \Omega \)
  \( \Omega \rightarrow \Omega \)
  \( \Gamma \vdash x \Rightarrow [\Gamma]A' \) \( p \mapsto \Gamma \)
  \( \Gamma \vdash \bar{A} : [\Gamma]A' \)

  By Lemma 32 (Extension Reflexivity)

  \( \Gamma \vdash \bar{A} : [\Gamma]A' \)

  By idempotence of substitution

  \( \text{dom}(\Gamma) = \text{dom}(\Omega) \)

  \( \Gamma \rightarrow \Omega \)

  \( [\Omega][\Gamma]A' = [\Omega]A' \)

  By Lemma 29 (Substitution Monotonicity) (iii)

  \( = A \)

  By above equality
Proof of Theorem 11 (Completeness of Algorithmic Typing)

• Case $\Gamma \vdash [\Omega]\Gamma \vdash [\Omega]e \Rightarrow B \quad q \quad [\Omega]\Gamma \vdash B \leq [\Omega]A \quad p$

$\frac{[\Omega]\Gamma \vdash [\Omega]e \Rightarrow B \quad q}{[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A}$

Subderivation

$\Gamma \vdash e \Rightarrow B' \quad q \quad \Theta$

By i.h.

$B = [\Omega]B'$

""

$\Theta \rightarrow \Omega_0$

""

$\Omega \rightarrow \Omega_0$

""

$\text{dom}(\Theta) = \text{dom}(\Omega_0)$

""

$\Gamma \rightarrow \Omega$

Given

$\Gamma \rightarrow \Omega_0$

By Lemma 33 (Extension Transitivity)

$[\Omega]\Gamma \vdash B \leq [\Omega]A$

Subderivation

$[\Omega]\Gamma = [\Omega]\Theta$

By Lemma 53 (Confluence of Completeness)

$[\Omega]\Theta \vdash B \leq [\Omega]A$

By above equalities

$\Theta \rightarrow \Omega_0$

Above

$\Theta \vdash B' \leq [\Omega]A \rightarrow \Delta$

By Theorem 9 (Completeness of Subtyping)

$\Omega_0 \rightarrow \Omega'$

""

$\Delta \rightarrow \Omega'$

By Lemma 33 (Extension Transitivity)

$\Omega \rightarrow \Omega'$

By Lemma 33 (Extension Transitivity)

$\Gamma \vdash e \Leftarrow A \quad p \quad \Delta$

By Sub

$[\Omega]\Gamma \vdash [\Omega]A \quad \text{type}$

$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega]A$

Subderivation

$[\Omega]A = [\Omega][\Gamma]A$

By Lemma 29 (Substitution Monotonicity)

$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega][\Gamma]A$

By above equality

$\Gamma \vdash e_0 \Leftarrow [\Gamma]A \rightarrow \Delta$

By i.h.

$\Delta \rightarrow \Omega$

""

$\Omega \rightarrow \Omega'$

""

$\text{dom}(\Delta) = \text{dom}(\Omega')$

""

$\Delta \rightarrow \Omega'$

By Lemma 33 (Extension Transitivity)

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

Given

$\Gamma \vdash (e_0 : A) \Rightarrow [\Delta]A \rightarrow \Delta$

By Anno

$[\Delta]A = [\Delta][\Delta]A$

By idempotence of substitution

$A = [\Omega]A$

Above

$= [\Omega']A$

By Lemma 55 (Completing Completeness) (ii)

$= [\Omega'][\Delta]A$

By Lemma 29 (Substitution Monotonicity)

• Case $[\Omega]\Gamma \vdash () \Leftarrow 1$

We have $[\Omega]A = 1$. Either $[\Gamma]A = 1$, or $[\Gamma]A = \hat{\alpha}$ where $\hat{\alpha} \in \text{unsolved}(\Gamma)$.

In the former case:

Let $\Delta = \Gamma$.

Let $\Omega' = \Omega$.

$\Delta \rightarrow \Omega$

Given

$\Omega \rightarrow \Omega'$

By Lemma 32 (Extension Reflexivity)

$\text{dom}(\Gamma) = \text{dom}(\Omega)$

Given

$\Gamma \vdash () \Leftarrow 1 \quad p \quad \Gamma$

By \[\text{1}\]

$\Gamma \vdash () \Leftarrow [\Gamma]1 \quad p \quad \Gamma$

$1 = [\Gamma]1$
Proof of Theorem 11 (Completeness of Algorithmic Typing)

In the latter case, since $A = \hat{a}$ and $\Gamma \vdash \hat{a} p$ type is given, it must be the case that $p = \hat{f}$.

\[
\Gamma_0[\hat{a} : \ast] \vdash (\cdot) \leftarrow \hat{f} \vdash \Gamma_0[\hat{a} : \ast] = 1 \quad \text{By i.h.}
\]

\[
\Gamma_0[\hat{a} : \ast] \vdash (\cdot) \leftarrow \hat{f} \vdash \Gamma_0[\hat{a} : \ast] = 1 \quad \text{By def. of subst.}
\]

\[
\Gamma_0[\hat{a} : \ast] \rightarrow \Omega \quad \text{Given}
\]

\[
\Gamma_0[\hat{a} : \ast] \rightarrow \Omega \quad \text{By Lemma 27 (Parallel Extension Solution)}
\]

\[
\Omega \rightarrow \Omega \quad \text{By Lemma 32 (Extension Reflexivity)}
\]

**Case** \( v \text{ chk-I} \) \[
\subalign{\Gamma \vdash \Omega \vdash \forall \alpha \colon \kappa . \ A_\alpha p} & \quad \text{DeclvI}
\]

\[
\Gamma \vdash \Omega \vdash \forall \alpha \colon \kappa . A_\alpha p \quad \text{By i.h.}
\]

\[
\Delta' \rightarrow \Omega_0'
\]

\[
\Omega, \alpha : \kappa \rightarrow \Omega_0'
\]

\[
\text{dom}(\Delta') = \text{dom}(\Omega_0')
\]

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

\[
\Delta' = (\Delta, \alpha : \kappa, \Theta)
\]

\[
\Delta, \alpha : \kappa, \Theta \rightarrow \Omega_0'
\]

\[
\Omega_0' = (\Omega', \alpha : \kappa, \Omega Z)
\]

\[
\Delta \rightarrow \Omega'
\]

\[
\text{dom}(\Delta) = \text{dom}(\Omega')
\]

\[
\Omega \rightarrow \Omega'
\]

\[
\text{By Lemma 22 (Extension Inversion) on } \Omega, \alpha : \kappa \rightarrow \Omega_0'
\]

\[
\Gamma, \alpha : \kappa \vdash \forall \alpha : \kappa . [\Gamma'] A_\alpha p \vdash \Delta, \alpha : \kappa, \Theta \quad \text{By definition of substitution}
\]

\[
\Gamma, \alpha : \kappa \vdash \forall \alpha : \kappa . [\Gamma'] A_\alpha p \vdash \Delta, \alpha : \kappa, \Theta \quad \text{By definition of def. substitution}
\]

**Case** \( v \text{ DeclvSpine} \) \[
\subalign{\Omega \vdash \tau : \kappa} & \quad \text{DeclvSpine}
\]

\[
\Gamma \vdash \tau : \kappa
\]

\[
\Gamma \rightarrow \Omega
\]

\[
\Gamma, \hat{a} : \kappa \rightarrow \Omega, \hat{a} : \kappa = \tau
\]

\[
\text{Subderivation}
\]

\[
\Gamma \rightarrow \Omega
\]

\[
\Gamma, \hat{a} : \kappa \rightarrow \Omega, \hat{a} : \kappa = \tau
\]

\[
\text{By Solve}
\]

\[
\text{Subderivation}
\]

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

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

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

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
\text{By distributivity of substitution}
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
\text{By definition of subst. context application}
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
[\tau/\alpha] A_\alpha = [\tau/\alpha] \Omega, \hat{a} : \kappa = \tau A_\alpha
\]

\[
\text{By definition of subst. context application}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ [\Omega, \vec{\alpha} : \tau] = [\eta] [\vec{\alpha} / \hat{\alpha} : \tau] \]
We have \( \text{mgu}(\sigma, \tau) = \theta \), and will need to apply Lemma 92 (Completeness of Elimeq) (1). That lemma has five side conditions, which can be shown exactly as in the case above.

\[ \text{mgu}(\sigma, \tau) = \theta \] Premise

\( \text{Let } \Omega_0 = (\Omega, \Gamma, \Theta) \).

\( \Gamma \rightarrow \Omega \) Given

\( \Gamma, \Gamma, \Theta \rightarrow \Omega_0 \) By Marker

\( \text{dom}(\Gamma) = \text{dom}(\Omega) \) Given

\( \text{dom}(\Gamma, \Theta) = \text{dom}(\Omega_0) \) By def. of dom(–)

\( \Gamma, \Gamma, \Theta \rightarrow \Omega_0 \) By Lemma 92 (Completeness of Elimeq) (1)

\( \text{EQ0 for all } \Gamma, \Gamma, \Theta \rightarrow \Gamma, \Gamma, \Theta \)

\( \text{By Lemma 92 (Completeness of Elimeq) (1)} \)

\( \text{EQ0 for all } \Gamma, \Gamma, \Theta \rightarrow \Gamma, \Gamma, \Theta \)

\( \text{By inversion} \)

\( \text{Let } \Omega_1 = (\Omega, \Gamma, \Theta). \)

\( \theta(\Omega) \rightarrow \theta(\Omega) \) ! Subderivation

\( \Gamma, \Gamma, \Theta \rightarrow \Omega_1 \) By induction on \( \Theta \)

\( \theta(\Omega) \rightarrow \theta(\Omega) \) ! By above equality EQa

\( \theta(\Omega) \rightarrow \theta(\Omega) \) ! By i.h.

\( \text{By Lemma 92 (Substitution Upgrade) (i) (with EQ0)} \)

\( \text{By Lemma 92 (Substitution Upgrade) (ii)} \)

\( \text{By Lemma 92 (Substitution Upgrade) (iii)} \)

\( \text{By Lemma 92 (Substitution Upgrade) (iv)} \)

\[ \text{By above equalities} \]

\[ \text{dom}(\Gamma, \Theta) = \text{dom}(\Omega_1) \] dom(\( \Gamma \) = dom(\( \Omega \))

\[ \text{dom}(\Gamma, \Theta) = \text{dom}(\Omega_0) \) By above equality EQa

\[ \text{dom}(\Gamma, \Theta) = \text{dom}(\Omega_0) \) By above equality EQa

\[ \text{dom}(\Delta) = \text{dom}(\Omega) \) By above equalities

\[ \text{By above equalities} \]

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

\( \text{Case } [\Omega] \Gamma \vdash [\Omega]P \text{ true } [\Omega] \Gamma \vdash [\Omega](e \cdot s_0) : [\Omega]A_0 \vdash B q \) DeclCheck
Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

\[ \text{[\Omega]} \vdash \text{[\Omega]} P \quad \text{true} \quad \text{Subderivation} \]

\[ \text{[\Omega]} \vdash \text{[\Omega]} [\Gamma] P \quad \text{true} \quad \text{By Lemma 29 (Substitution Monotonicity) (ii)} \]

\[ \Gamma \vdash [\Gamma] P \quad \text{true} - \Theta \quad \text{By Lemma 99 (Completeness of Checkprop)} \]

\[ \Theta \rightarrow \Omega_1 \quad \text{"} \]

\[ \Omega \rightarrow \Omega_1 \quad \text{"} \]

\[ \text{dom}(\Theta) = \text{dom}(\Omega_1) \quad \text{"} \]

\[ \Gamma \rightarrow \Omega \quad \text{Given} \]

\[ \text{[\Omega]} \vdash [\Omega] \{\Theta\} \quad \text{By Lemma 57 (Multiple Confluence)} \]

\[ [\Omega] A_0 = [\Omega_1] A_0 \quad \text{By Lemma 55 (Completing Completeness) (ii)} \]

\[ [\Omega] [\Gamma] \vdash [\Omega] (e \cdot s_0) : [\Omega] A_0 \quad \text{p} \gg B \quad q \quad \text{Subderivation} \]

\[ [\Omega_1] \Theta \vdash [\Omega] (e \cdot s_0) : \text{[\Omega]} A_0 \quad \text{p} \gg B \quad q \quad \text{By above equalities} \]

\[ \Theta \vdash e \cdot s_0 : \text{[\Theta]} A_0 \quad \text{p} \gg B' \quad q \quad \text{By above equality} \]

\[ \text{\Theta} \vdash e \cdot s_0 : [\Theta] (P \gg \Gamma) A_0 \gg B' \quad q \quad \text{By \thesis spine} \]

\[ \text{\Theta} \vdash e \cdot s_0 : \text{[\Theta]} \Gamma A_0 \gg B' \quad q \quad \text{By \thesis def. of subst.} \]

\[ \text{Case}: \quad [\Omega] [\Gamma] \vdash [\Omega] e_0 \Leftrightarrow A_k' p \quad \text{Decl+Ink} \]

\[ [\Omega] [\Gamma] \vdash \text{inj}_k [\Omega] e_0 \Leftrightarrow A_1' + A_2' p \quad \text{Dec+Ink} \]

Either \( [\Gamma] A = A_1 + A_2 \) (where \( [\Omega] A_k = A_k' \)) or \( [\Gamma] A = \alpha \in \text{unsolved}(\Gamma) \).

In the former case:

\[ [\Omega] [\Gamma] \vdash [\Omega] e_0 \Leftrightarrow A_k' p \quad \text{Subderivation} \]

\[ [\Omega] [\Gamma] \vdash [\Omega] e_0 \Leftrightarrow [\Omega] A_k' p \quad \text{[\Omega]} A_k = A_k' \]

\[ \Gamma \vdash e_0 \Leftrightarrow [\Gamma] A_k p \gg \Delta \quad \text{By \thesis i.h.} \]

\[ \Gamma \vdash e \cdot s_0 : [\Gamma] (\Delta \gg \Omega) \quad \text{p} \gg B' \quad q \quad \text{By \thesis def. of subst.} \]

In the latter case, \( A = \alpha \) and \( [\Omega] A = [\Omega] \alpha = A_1' + A_2' \) = \( \tau_1 + \tau_2 \).

By inversion on \( \Gamma \vdash \alpha \quad \text{type} \), it must be the case that \( p = \not\emptyset \).

\[ \Gamma \rightarrow \Omega \quad \text{Given} \]

\[ \Gamma = \Gamma_0[\alpha : \not\emptyset] \quad \alpha \in \text{unsolved}(\Gamma) \]

\[ \Omega = \Omega_0[\alpha : \not\emptyset = \tau_0] \quad \text{By Lemma 22 (Extension Inversion) (vi)} \]

Let \( \Omega_2 = \Omega_0[\alpha_1 : \tau_1, \alpha_2 : \tau_2, \not\emptyset : \not\emptyset = \alpha_1 + \alpha_2] \).

Let \( \Gamma_2 = \Gamma_0[\alpha_1 : \not\emptyset, \alpha_2 : \not\emptyset, \alpha : \not\emptyset = \alpha_1 + \alpha_2] \).

\[ \Gamma \rightarrow \Gamma_2 \quad \text{By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (ii)} \]

\[ \Omega \rightarrow \Omega_2 \quad \text{By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (iii)} \]

\[ \Gamma_2 \rightarrow \Omega_2 \quad \text{By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

Since \( \tau' = \tau'' \) \( \Leftrightarrow \) \( \vdash \tau' \rightarrow \tau'' \) the context \( \Gamma \) must have the form \( \Gamma = \Gamma_0[\bar{\alpha} : \kappa] \).

Let \( \Gamma_2 = \Gamma_0[\bar{\alpha}_1 : \kappa, \bar{\alpha}_2 : \kappa, \bar{\alpha} : \kappa] \).

In the latter case \( (\Gamma' \bar{\alpha} = \bar{\alpha}' \rightarrow \bar{\alpha}_2) \) and \( (\Omega \bar{\alpha} = \bar{\alpha}_1 \rightarrow \bar{\alpha}_2) \):

By inversion on \( \vdash \bar{\alpha} \lor \bar{\alpha}_2 \) type, it must be the case that \( \bar{\alpha} = \phi \).

Since \( \bar{\alpha} \in \text{unsolved}(\Gamma) \), the context \( \Gamma \) must have the form \( \Gamma_0[\bar{\alpha} : \kappa] \).

Let \( \Gamma_2 = \Gamma_0[\bar{\alpha}_1 : \kappa, \bar{\alpha}_2 : \kappa, \bar{\alpha} : \kappa] \).

\[ \begin{align*}
[\Omega] & \vdash \del{\Omega_2 \bar{\alpha}_k} \not\Rightarrow \text{Subd. and } \Lambda'_k = \tau'_k = \del{\Omega_2 \bar{\alpha}_k} \\
[\Omega] & \vdash \del{\Omega_2 \bar{\alpha}_k} \not\Rightarrow \text{By Lemma 57 (Multiple Confluence)} \\
[\Omega] & \vdash \del{\Omega_2 \bar{\alpha}_k} \not\Rightarrow \text{By above equality} \\
\Gamma & \vdash \del{\Gamma_2 \bar{\alpha}_k} \not\Rightarrow \text{By i.h.} \\
\Delta & \leadsto \Omega' \\
\Delta & \leadsto \Omega' \\
\Omega & \vdash \inj \bar{\alpha}_k \not\Rightarrow \text{By Lemma 33 (Extension Transitivity)} \\
\Omega & \vdash \inj \bar{\alpha}_k \not\Rightarrow \text{By } \vdash \bar{\alpha} \lor \bar{\alpha}_2 \\
\bar{\alpha} & \in \text{unsolved}(\Gamma) \\
\end{align*} \]

\[ \begin{align*}
\text{Case } \Gamma & : \Delta \\
\Omega & : \Delta \\
\Omega & : \Delta \\
\end{align*} \]

We have \( \Omega \mid A = A'_1 \rightarrow A'_2 \). Either \( \Gamma \mid A = A_1 \rightarrow A_2 \) where \( A'_1 = \Omega \mid A_1 \) and \( A'_2 = \Omega \mid A_2 \)---or \( \Gamma \mid A = \bar{\alpha} \) and \( \Omega \mid \bar{\alpha} = \bar{\alpha}_1 \rightarrow \bar{\alpha}_2 \).

In the former case:

\[ \begin{align*}
\Gamma & : \Delta \\
\Omega & : \Delta \\
\Omega & : \Delta \\
\Omega & : \Delta \\
\end{align*} \]

By definition of context application

By above equality

Given

By \( \vdash \bar{\alpha} \lor \bar{\alpha}_2 \) type, it must be the case that \( \bar{\alpha} = \phi \).

Since \( \bar{\alpha} \in \text{unsolved}(\Gamma) \), the context \( \Gamma \) must have the form \( \Gamma_0[\bar{\alpha} : \kappa] \).

Let \( \Gamma_2 = \Gamma_0[\bar{\alpha}_1 : \kappa, \bar{\alpha}_2 : \kappa, \bar{\alpha} : \kappa] \).
Proof of Theorem 11 (Completeness of Algorithmic Typing)  

\[ \Gamma \rightarrow \Gamma_2 \quad \text{By Lemma 23 (Deep Evar Introduction) (iii) twice} \]

\[ [\Omega] \hat{\alpha} = \tau'_1 \rightarrow \tau'_2 \quad \text{Known in this subcase} \]

\[ \Omega \rightarrow \Omega \quad \text{Given} \]

\[ \Omega = \Omega_0[\hat{\alpha} : \alpha = \tau_0] \quad \text{By Lemma 22 (Extension Inversion) (vi)} \]

Let \( \Omega_2 = \Omega_0[\hat{\alpha} : \alpha = \tau'_1, \hat{\alpha}_1 : \alpha = \tau'_2, \hat{\alpha} : \alpha = \hat{\alpha}_1 \rightarrow \hat{\alpha}_2]. \)

\[ \Gamma \rightarrow \Gamma_2 \quad \text{By Lemma 23 (Deep Evar Introduction) (iii) twice} \]

\[ \Omega \rightarrow \Omega_2 \quad \text{By Lemma 23 (Deep Evar Introduction) (iii) twice} \]

\[ \Gamma_2 \rightarrow \Omega_2 \quad \text{By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)} \]

\[ [\Omega]_1, x : \tau'_1 \not\vdash [\Omega]_0 e_0 \leftrightarrow \tau'_2 \not\vdash \text{Subderivation} \]

\[ [\Omega]_1 = [\Omega]_2 \|_2 \quad \text{By Lemma 57 (Multiple Confluence)} \]

\[ \tau'_2 = [\Omega] \hat{\alpha}_2 \quad \text{From above equality} \]

\[ = [\Omega_2] \hat{\alpha}_2 \quad \text{By Lemma 55 (Completing Completeness) (i)} \]

\[ \tau'_1 = [\Omega_2] \hat{\alpha}_1 \quad \text{Similar} \]

\[ [\Omega_2] \Gamma_2, x : \tau'_1 \not\vdash [\Omega_2, x : \tau'_1 \not\vdash] \Gamma_2, x : \hat{\alpha}_1 \not\vdash \text{By def. of context application} \]

\[ [\Omega_2, x : \tau'_1 \not\vdash] \Gamma_2, x : \hat{\alpha}_1 \not\vdash \vdash [\Omega_2, \hat{\alpha}_2] \not\vdash \text{By above equalities} \]

\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \quad \text{Given} \]

\[ \text{dom}(\Gamma_2, x : \hat{\alpha}_1) = \text{dom}(\Omega_2, x : \tau'_1) \quad \text{By def. of } \Gamma_2 \text{ and } \Omega_2 \]

\[ \Gamma_2, x : \hat{\alpha}_1 \not\vdash e_0 \vdash [\Gamma_2, x : \hat{\alpha}_1 \not\vdash] \hat{\alpha}_2 \not\vdash \Delta^+ \quad \text{By i.h.} \]

\[ \Delta^+ \rightarrow \Omega^+ \]

\[ \text{dom}(\Delta^+) = \text{dom}(\Omega^+) \quad \text{"} \]

\[ \Omega_2 \rightarrow \Omega^+ \quad \text{"} \]

\[ \Gamma_2, x : \hat{\alpha}_1 \not\vdash \Delta^+ \quad \text{By Lemma 51 (Typing Extension)} \]

\[ \Delta^+ = (\Delta, x : \hat{\alpha}_1 \not\vdash, \Delta_Z) \quad \text{By Lemma 22 (Extension Inversion) (v)} \]

\[ \Omega^+ = (\Omega', x : \ldots \not\vdash, \Omega_Z) \quad \text{By Lemma 22 (Extension Inversion) (v)} \]

\[ \Gamma \vdash \lambda x. e_0 \leftrightarrow \hat{\alpha} \not\vdash \Delta \quad \text{By i.h.} \]

\[ \hat{\alpha} = [\Gamma] \hat{\alpha} \quad \hat{\alpha} \in \text{unsolved}(\Gamma) \]

\[ \Gamma \vdash \lambda x. e_0 \leftrightarrow [\Gamma] \hat{\alpha} \not\vdash \Delta \quad \text{By above equality} \]

\[ \text{Case} \quad [\Omega] \Gamma \vdash [\Omega]_0 e_0 \Rightarrow A \quad q \quad [\Omega] \Gamma \vdash [\Omega]_0 s_0 : A \quad q \Rightarrow C \quad [p] \quad \text{Decl--E} \]

\[ [\Omega] \Gamma \vdash [\Omega]_0 e_0 \Rightarrow A \quad q \quad \text{Subderivation} \]

\[ \Gamma \vdash e_0 \Rightarrow A' \quad q \not\vdash \Theta \quad \text{By i.h.} \]

\[ \text{dom}(\Theta) = \text{dom}(\Omega_0) \quad " \]

\[ \Omega \rightarrow \Omega_0 \quad " \]

\[ A = [\Omega_0] A' \quad " \]

\[ A' = [\Theta] A' \quad " \]
Given 

\[ \Gamma \rightarrow \Omega \]

\[ [\Omega] \Gamma = [\Omega_\Theta] \Theta \]

\[ [\Omega] \Gamma \vdash [\Omega \Theta]_0 : \text{A} \ q \gg C \ [p] \]

\[ [\Omega_\Theta] \Theta \vdash [\Omega \Theta]_0 : [\Omega_\Theta] \text{A'} \ q \gg C \ [p] \]

\[ \Theta \vdash s_0 : [\Theta] \text{A'} \ q \gg C' \ [p] \vdash \Delta \]

\[ C' = [\Delta] C' \]

\[ \Delta \rightarrow \Omega' \]

\[ \text{By above equalities} \]

\[ \text{By i.h.} \]

\[ \text{By Lemma 57 (Multiple Confluence)} \]

\[ \text{Subderivation} \]

\[ \text{By Lemma 33 (Extension Transitivity)} \]
Proof of Theorem 11 \textit{(Completeness of Algorithmic Typing)}

\textbf{Case}

for all \(C_2\),
\[
\begin{align*}
&\text{if } [\Delta][\Gamma]s : [\Delta]A ! \gg C_2 \not\in C \not\in
&\text{then } C_2 = C.
\end{align*}
\]

\[
\Delta \rightarrow \Omega
\]

\[
[\Delta][\Gamma]s : [\Delta]A ! \gg C_2 \not\in C \not\in
\]

\[
\Gamma \rightarrow \Omega
\]

\[
[\Delta][\Gamma]s : [\Delta]A ! \gg C_2 \not\in C \not\in
\]

\[
\begin{align*}
&\text{Subderivation} \\
&\text{By i.h.}
\end{align*}
\]

\[
\begin{align*}
&\text{By Lemma 60 (Split Solutions)} \\
&\text{By Lemma 72 (Separation—Main) (Spines)}
\end{align*}
\]

Suppose, for a contradiction, that \(\text{FEV}(\Delta|C') \neq \emptyset\).

That is, there exists some \(\hat{\alpha} \in \text{FEV}(\Delta|C')\).

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 \(\Delta\).

\[
\begin{align*}
&\text{From (NEQ) and (EQ)} \\
&\text{By Theorem 8 (Soundness of Algorithmic Typing)}
\end{align*}
\]

\[
\begin{align*}
&\text{By Lemma 13 (Right-Hand Substitution for Typing)} \\
&\text{By Lemma 18 (Equal Domains)}
\end{align*}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

\[ \Gamma \vdash \forall \Gamma A \text{ type} \]

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

\[ \Omega \vdash \forall \Gamma A = [\Omega] [\Gamma] A = [\Omega] [\Gamma] A \]

By Lemma 38 (Extension Weakening (Types))


By Lemma 55 (Completing Completeness) (ii) twice

\[ \Omega \Gamma = [\Omega'] \Gamma \]

By Lemma 57 (Multiple Confluence)

\[ = [\Omega_1] \Gamma \]

By Lemma 57 (Multiple Confluence)

\[ = [\Omega_2] \Gamma \]

Follows from \( \forall \alpha \not\in \text{dom}(\Gamma) \)

\[ [\Omega_2] s = [\Omega] s \]

\( \Omega \) and \( \Omega \) differ only in \( \alpha \)

\[ [\Omega'] \Gamma \vdash [\Omega] s : [\Omega] A ! \gg [\Omega_2] C' \]

By above equalities

\[ C = [\Omega'] C' \]

Above

\[ [\Omega'] C' \neq [\Omega_2] C' \]

By def. of subst.

\[ C \neq [\Omega_2] C' \]

By above equality

\[ C = [\Omega_2] C' \]

Instantiating “for all \( C_2 \) with \( C_2 = [\Omega_2] C' \)

\[ \Rightarrow \]

\[ \text{FEV}([\Delta] C') = \emptyset \]

By contradiction

\[ \Gamma \vdash s : [\Gamma] A ! \gg C' \]

By \( \Gamma \vdash \Gamma \)

\[ \Gamma \vdash s : [\Gamma] A ! \gg C' \]

By \( \text{SpineRecover} \)

\[ \text{DeclSpinePass} \]

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A \]

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \text{DeclSpinePass} \]

\[ [\Omega] \Gamma \vdash [\Omega] s : [\Omega] A \gg C \]

\[ \text{DeclSpinePass} \]

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \Rightarrow \]

\[ \Delta \rightarrow [\Omega'] \]

""

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]

""

\[ \Omega \rightarrow [\Omega'] \]

""

\[ [\Delta] C' \]

""

\[ C \rightarrow [\Omega'] C' \]

""

\[ C = [\Omega'] C' \]

""

We distinguish cases as follows:

- If \( p = f \) or \( q = ! \), then we can just apply \( \text{SpinePass} \),

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \text{By i.h.} \]

- Otherwise, \( p = f \) and \( q = ! \). If \( \text{FEV}(C) \neq \emptyset \), we can apply \( \text{SpinePass} \), as above. If \( \text{FEV}(C) = \emptyset \), then we instead apply \( \text{SpinePass} \), as above.

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \Gamma \vdash s : [\Gamma] A \gg C \]

\[ \text{By i.h.} \]

Here, \( q' = ! \) and \( q = f \), so \( q' \subseteq q \).

\[ \text{DeclEmptySpine} \]

\[ [\Omega] \Gamma \vdash \cdot : [\Omega] A \gg [\Omega] A \]

\[ \Gamma \vdash \cdot : [\Gamma] A \gg [\Gamma] A \]

\[ \text{By EmptySpine} \]

\[ [\Gamma] A = [\Gamma] [\Gamma] A \]

\[ \text{By idempotence of substitution} \]

\[ \Gamma \rightarrow [\Omega] \]

\[ \text{Given} \]

\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \]

\[ \text{Given} \]


\[ \text{By Lemma 29 (Substitution Monotonicity)) (iii) } \]

\[ \Omega \rightarrow [\Omega] \]

\[ \text{By Lemma 32 (Extension Reflexivity)} \]

\[ \text{DeclSpine} \]

\[ \text{Case} \]

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \leftarrow [\Omega] A_1 \]

\[ [\Omega] \Gamma \vdash [\Omega] s_0 : [\Omega] A_2 \gg B \]

\[ [\Omega] \Gamma \vdash [\Omega] (e_0 \cdot s_0) : ([\Omega] A_1) \rightarrow ([\Omega] A_2) \gg B \]

\[ \text{Decl} \rightarrow \]

\[ \text{By} \]

\[ \text{DeclSpine} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
\Theta \rightarrow \Omega_\Theta & \\
\Omega \rightarrow \Omega_\Theta & \\
A = [\Omega_\Theta]A' & \\
A' = [\Theta]A' & \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
\Gamma \vdash e_0 : A_1 \rightarrow A_2 & \\
\Gamma \vdash e_0 \equiv A' \rightarrow \Theta & \\
\end{align*}
\]

\[
\begin{align*}
\Theta \rightarrow & \quad \Omega_\Theta \\
\Omega \rightarrow & \quad \Omega_\Theta \\
A = [\Omega_\Theta]A' & \\
A' = [\Theta]A' & \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
\Delta \rightarrow & \quad \Omega' \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
dom(\Delta) = \text{dom}(\Omega') & \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
\Omega \rightarrow & \quad \Omega' \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
B' = \text{dom}(\Omega') & \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
B = [\Omega']B' & \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
\Gamma \vdash e_0 \cdot s_0 : A_1 \rightarrow A_2 & \\
\Gamma \vdash \Delta & \\
\end{align*}
\]

\[
\begin{align*}
\text{Subderivation} & \quad \text{By i.h.} \\
\Gamma \vdash e_0 \cdot s_0 & \\
\Gamma \vdash \Delta & \\
\end{align*}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing)  

Case \[ \Omega; \Gamma \vdash [\Omega]P \text{ true} \quad [\Omega; \Gamma \vdash [\Omega]e \Leftarrow [\Omega]A_0 \ p \quad \frac{\text{Decl} \land I}{[\Omega; \Gamma \vdash [\Omega]e \Leftarrow ([\Omega]A_0) \land [\Omega]P \ p} \]

If \( e \) not a case, then:

\[ [\Omega; \Gamma \vdash [\Omega]P \text{ true} \quad \frac{\text{Subderivation}}{\Gamma \vdash P \text{ true } \rightarrow \Theta} \quad \text{By Lemma 95 (Completeness of Checkprop)} \]

\[ \Theta \rightarrow \Omega_0' \quad \text{"} \]

\[ \Omega \rightarrow \Omega_0' \quad \text{"} \]

\[ \Gamma \rightarrow \Omega \quad \text{Given} \]

\[ \Gamma \rightarrow \Omega_0' \quad \text{By Lemma 33 (Extension Transitivity)} \]

\[ [\Omega; \Gamma = [\Omega]\Omega \quad \frac{\text{By Lemma 54 (Completing Stability)}}{[\Omega; \Gamma = [\Omega]_0' \Omega_0'} \quad \text{"} \]

\[ = [\Omega_0' \theta]_0' \quad \text{By Lemma 55 (Completing Completeness) (iii)} \]

\[ \Gamma \vdash A_0 \land P \ p \text{ type} \quad \text{By inversion} \]

\[ [\Omega]A_0 = [\Omega_0' \theta]A_0 \quad \text{By Lemma 55 (Completing Completeness) (ii)} \]

\[ [\Omega; \Gamma \vdash [\Omega]e \Leftarrow [\Omega]A_0 \ p \text{ true} \quad [\Omega; \Gamma \vdash [\Omega]e \Leftarrow [\Omega_0' \theta]A_0 \ p \text{ true} \quad \frac{\text{Subderivation}}{\Theta \vdash e \Leftarrow [\Theta]A_0 \ p \rightarrow \Delta} \quad \text{By i.h.} \]

\[ \Delta \rightarrow \Omega' \quad \text{"} \]

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \quad \text{"} \]

\[ \Omega_0' \rightarrow \Omega' \quad \text{"} \]

\[ \Omega \rightarrow \Omega' \quad \text{By Lemma 33 (Extension Transitivity)} \]

\[ \Gamma \vdash e \Leftarrow A_0 \land P \ p \rightarrow \Delta \quad \text{By } \text{Case} \]

Otherwise, we have \( e = \text{case}(e_0, \Pi) \). Let \( n \) be the height of the given derivation.

\[
\begin{align*}
\text{n } - 1 & [\Omega; \Gamma \vdash [\Omega]([\text{case}(e_0, \Pi)]) \Leftarrow [\Omega]A_0 \ p \quad \text{Subderivation} \\
\text{n } - 2 & [\Omega; \Gamma \vdash [\Omega]e_0 \Rightarrow B \ ! \\
\text{n } - 2 & [\Omega; \Gamma \vdash [\Omega]\Pi : \delta \Leftarrow [\Omega]A_0 \ p \\
\text{n } - 2 & [\Omega; \Gamma \vdash [\Omega]\Pi \text{ covers } B \\
\text{n } - 1 & [\Omega; \Gamma \vdash [\Omega]P \text{ true} \quad [\Omega; \Gamma \vdash [\Omega]\Pi : \delta \Leftarrow [\Omega]A_0 \land P \ p \quad \text{By def. of subst.} \\
\text{n } - 1 & [\Omega; \Gamma \vdash [\Omega]\Pi : \delta \Leftarrow [\Omega]A_0 \land P \ p \quad \text{By i.h.} \\
\text{n } - 1 & \Theta \rightarrow \Omega_0' \quad \text{"} \]
\end{align*}
\]

\[ \frac{\text{Subderivation}}{\Omega \rightarrow \Omega_0' \quad \text{"} \]

\[ B = \left[ \Omega_0' \delta \right] B' \quad \text{"} \]

\[ \frac{\text{By Lemma 30 (Substitution Invariance)}}{\text{By Lemma 62 (Case Invertibility)}} \]

\[ [\Omega; \Gamma = [\Omega]_0' \theta] \quad \text{By Lemma 57 (Multiple Confluence) (ii)} \]

\[ [\Omega](A_0 \land P) = [\Omega_0'](A_0 \land P) \quad \text{By Lemma 55 (Completing Completeness) (ii)} \]

\[ n - 1 \quad [\Omega_0'](\theta) \vdash [\Omega]\Pi : [\Omega_0'](\theta)B' \Leftarrow [\Omega]_0'\Pi([\Omega]_0'\land P) \ p \quad \text{By equalities} \\
\text{Latin text} \quad \text{By i.h.} \\
\text{Latin text} \quad \text{By Theorem 10 (Completeness of Match Coverage)} \]

\[ \frac{[\Omega; \Gamma = [\Omega]_0' \theta] \quad \text{By Lemma 33 (Extension Transitivity)}}{\text{By Case} \quad \frac{\text{By Lemma 56 (Confluence of Completeness)}}{[\Omega; \Gamma \vdash \text{case}(e_0, \Pi) \Leftarrow A_0 \land P \ p \rightarrow \Delta} \quad \text{"} \]

\[ \text{Latin text} \quad \text{By Case} \]

\[ \text{Latin text} \quad \text{By Case} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

Case

\[ \Omega \vdash [\Omega]e_1 \triangleleft A'_1 p \quad [\Omega] \vdash [\Omega]e_2 \triangleleft A'_2 p \]

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 \(A'_2 = [\Omega]A_2\).

\[ \Omega \vdash [\Omega]e_1 \triangleleft [\Omega]A_1 p \]

Subderivation

\[ \Gamma \vdash e_1 \triangleleft [\Gamma]A_1 p \quad \Theta \rightarrow \Omega \]

By i.h.

\[ \text{dom}(\Theta) = \text{dom}(\Omega) \]

By above equalities

\[ \Delta \rightarrow \Omega' \]

By Lemma 33 (Extension Transitivity)

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]

By def. of subst.

\[ \Omega' \rightarrow \Omega' \]

By Lemma 32 (Extension Transitivity)

\[ \Gamma \vdash \langle e_1, e_2 \rangle \triangleleft ([\Gamma]A_1) \times ([\Gamma]A_2) p \triangleright \Delta \]

By def. of subst.

- In the second case, where \([\Gamma]A = \alpha\), combine the corresponding subcase for [\(\text{Decl+I}_3\)] with some straightforward additional reasoning about contexts (because here we have two subderivations, rather than one).

Case

\[ \Omega \vdash [\Omega]e_0 \rightarrow C ! \]

Subderivation

\[ \Gamma \vdash e_0 \rightarrow C' ! \quad \Theta \rightarrow \Omega \]

By i.h.

\[ \text{dom}(\Theta) = \text{dom}(\Omega) \]

By def. of subst.

\[ \text{C} = [\Omega_\Theta]C' \]

By Lemma 63 (Well-Formed Outputs of Typing)

\[ \text{FEV}(C') = \emptyset \]

By inversion

\[ [\Omega_\Theta]C' = C' \]

By a property of substitution
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ \Gamma \rightarrow \Omega \]
\[ \Delta \rightarrow \Omega \]
\[ \Theta \rightarrow \Omega \]

\[ [\Omega] \Gamma = [\Omega] \Theta = [\Omega] \Delta \]

\[ \Gamma \rightarrow \Theta \]
\[ \Gamma \rightarrow [\Omega] \Theta \]

\[ [\Omega] \Gamma = [\Omega] \Theta \]

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

\[ [\Omega] \Gamma = [\Omega] \Theta \]
\[ \Gamma \vdash A \text{ type} \]

\[ [\Omega] A \Gamma = [\Omega] \Theta \]
\[ [\Omega] \Gamma \vdash [\Omega] \Pi :: C \preceq [\Omega] A \]

\[ [\Omega] \Theta \vdash [\Omega] \Pi :: [\Omega] \Theta C' \preceq [\Omega] A \]
\[ \Theta \vdash [\Psi] \Pi :: [\Psi] C' \preceq [\Psi] A \]

\[ \Theta \vdash [\varpi] A \]

\[ \Theta \vdash \text{inversion} \]

\[ \text{Subderivation} \]

\[ \text{Above} \]

\[ \text{By Lemma 33 (Extension Transitivity)} \]

\[ \text{Given + inversion} \]

\[ \text{By Lemma 33 (Extension Transitivity)} \]

\[ \text{Given} \]

\[ \text{Subderivation} \]

\[ \text{By Lemma 33 (Extension Transitivity)} \]

\[ \text{By above equalities} \]

\[ \text{By Lemma 51 (Typing Extension)} \]
\[ \text{& 33} \]

\[ \Theta \vdash [\varpi] A \]

\[ \text{By Lemma 41 (Extension Weakening for Principal Typing)} \]

\[ \text{By above equalities} \]

\[ \text{By Lemma 33 (Extension Transitivity)} \]

\[ \text{By Lemma 57 (Multiple Confluence)} \]

\[ \text{By i.h. (v)} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Case} \]

\[ \text{Decl} \times 1 \]

\[ [\Omega] \Gamma \vdash [\Omega] \Theta e_1 \iff A_1 p \]
\[ [\Omega] \Gamma \vdash [\Omega] \Theta e_2 \iff A_2 p \]

\[ [\Omega] \Gamma \vdash (\Theta e_1, \Theta e_2) \iff A_1 \times A_2 p \]

\[ [\Omega] \Gamma \vdash [\Omega] \Theta \]

Either \( A = \emptyset \) where \( [\Omega] \emptyset = A_1 \times A_2 \), or \( A = A_1' \times A_2' \) where \( A_1 = [\Omega] A_1' \) and \( A_2 = [\Omega] A_2' \).

In the former case \( (A = \emptyset) \):

We have \( [\Omega] \emptyset = A_1 \times A_2 \). Therefore \( A_1 = [\Omega] A_1' \) and \( A_2 = [\Omega] A_2' \). Moreover, \( \Gamma = \Gamma_0 [\emptyset : \kappa] \).

\[ [\Omega] \Gamma \vdash [\Omega] e_1 \iff [\Omega] A_1' p \]

Let \( \Gamma' = \Gamma_0 [\emptyset_1 : \kappa, \emptyset_2 : \kappa, \emptyset : \kappa = \emptyset_1 + \emptyset_2] \).

\[ [\Omega] \Gamma' \vdash [\Omega] e_1 \iff [\Omega] A_1' p \]

\[ \Gamma' \vdash e_1 \iff [\Gamma'] A_1' p \]

\[ \text{By above equality} \]

\[ \Theta \vdash e_1 \iff [\Theta] A_1' p \]

\[ \text{By i.h.} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]

\[ \text{Subderivation} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

• Case \(\text{DeclMatchSeq}\) Apply rule \(\text{MatchEmpty}\)

• Case \(\text{DeclMatchBase}\) Apply the i.h. (i) and rule \(\text{MatchBase}\)

• Case \(\text{DeclMatchUnit}\) Apply the i.h. and rule \(\text{MatchUnit}\)

• Case \(\text{DeclMatch×}\) By i.h. and rule \(\text{Match×}\)

• Case \(\text{DeclMatch+1}\) By i.h. and rule \(\text{Match+1}\)

• Case \(\text{DeclMatch}\) Apply rule \(\text{Match}\)

Now we turn to parts (v) and (vi), completeness of matching.

Case

\[\begin{align*}
[\Omega]\Gamma &= [\Omega_1]\Theta \\
[\Omega]A_2 &= [\Omega_1]A_2' \\
[\Omega_1]\Theta \vdash [\Omega]e_2 \leftarrow [\Omega_1]A_2' p \\
\Theta \vdash e_2 &\leftarrow (\Theta)A_2' p' \vdash \Delta \\
\end{align*}\]

By Lemma 57 (Multiple Confluence)

By above equalities

By i.h.

\[\begin{align*}
\text{dom}(\Delta) &= \text{dom}(\Omega') \\
\Delta &\rightarrow \Omega' \\
\Omega_1 &\rightarrow \Omega' \\
\end{align*}\]

By Lemma 33 (Extension Transitivity)

By \(\times\)

In the latter case \((A = A_1' \times A_2')\):

\[\begin{align*}
[\Omega]\Gamma \vdash [\Omega]e_1 \leftarrow A_1 p \\
[\Omega]\Gamma \vdash [\Omega]e_1 \leftarrow [\Omega]A_1' p \\
\Gamma \vdash e_1 \leftarrow [\Gamma]A_1' p \vdash \Theta \\
\Theta \rightarrow \Omega_0 \\
\text{dom}(\Theta) &= \text{dom}(\Omega_0) \\
\Omega &\rightarrow \Omega_0 \\
\end{align*}\]

Subderivation

\[\begin{align*}
[\Omega]\Gamma \vdash [\Omega]e_2 \leftarrow A_2 p \\
[\Omega]\Gamma \vdash [\Omega]e_2 \leftarrow [\Omega]A_2' p \\
\Gamma \vdash A_1' \times A_2' p \text{ type} \\
\Gamma \vdash A_2' \text{ type} \\
\Gamma &\rightarrow \Omega \\
\end{align*}\]

Given \((A = A_1' \times A_2')\)

By inversion

By Lemma 33 (Extension Transitivity)

By \(\times\)

\[\begin{align*}
\Omega_0 &\rightarrow A_1' \text{ type} \\
[\Omega]\Gamma \vdash [\Omega]e_2 \leftarrow [\Omega_0]A_2' p \\
[\Omega]\Gamma \vdash [\Omega]e_2 \leftarrow [\Omega_0]\Theta A_2' p \\
\Theta &\vdash e_2 \leftarrow (\Theta)A_2' p \vdash \Delta \\
\end{align*}\]

By i.h.

By Lemma 55 (Completing Completeness) (ii)

By Lemma 29 (Substitution Monotonicity) (iii)

By Lemma 57 (Multiple Confluence)

By \(\times\)

\[\begin{align*}
\Delta &\rightarrow \Omega' \\
\text{dom}(\Delta) &= \text{dom}(\Omega') \\
\Omega_0 &\rightarrow \Omega' \\
\end{align*}\]

By Lemma 33 (Extension Transitivity)

By \(\times\)

\[\begin{align*}
\Gamma \vdash \langle e_1, e_2 \rangle &\leftarrow [\Omega](A_1 \times A_2) p \vdash \Delta \\
\end{align*}\]

By def. of substitution

To apply the i.h. (vi), we will show (1) \(\Gamma \vdash (A, \vec{A}) \not\models \) types, (2) \(\Gamma \vdash P \text{ prop} \), (3) \(\text{FEV}(P) = \emptyset \), (4) \(\Gamma \vdash C \text{ p type} \), (5) \([\Omega]\Gamma / [\Omega]P \vdash \vec{\rho} \Rightarrow [\Omega]e : [\Omega]\vec{A} \leftarrow [\Omega]C p \), and (6) \(p' \subseteq p\).
\[ \Gamma \vdash (A \land P, \tilde{A}) ! \text{ types} \quad \text{Given} \]
\[ \Gamma \vdash (A \land P) ! \text{ type} \quad \text{By inversion on } \text{PrincipalTypevecWF} \]
\[ \Gamma \vdash A ! \text{ type} \quad \text{By Lemma 42 (Inversion of Principal Typing)} \]

(2) \[ \Gamma \vdash P \text{ prop} \]

(3) \[ \text{FEV}(\rho) = \emptyset \quad \text{By inversion} \]

(1) \[ \Gamma \vdash (A, \tilde{A}) ! \text{ types} \quad \text{By inversion and } \text{PrincipalTypevecWF} \]

(4) \[ \Gamma \vdash C \text{ p type} \quad \text{Given} \]

(5) \[ [\Omega] \Gamma / \rho \Rightarrow [\Omega] e :: [\Omega] A, [\Omega] \tilde{A} \leftarrow [\Omega] C p \quad \text{Subderivation} \]

(6) \[ p' \subseteq p \quad \text{Given} \]

\[ \Gamma / [\Gamma] \rho \Rightarrow e :: [\Gamma] (A, \tilde{A}) \leftarrow [\Gamma] C p' \leftarrow \Delta \quad \text{By i.h. (vi)} \]

\[ \Delta \mapsto \Omega' \]

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \]

\[ \Gamma / [\Gamma] \rho \Rightarrow e :: [\Gamma] A, [\Gamma] \tilde{A} \leftarrow [\Gamma] C p' \leftarrow \Delta \quad \text{By def. of subst.} \]

\[ \Gamma / [\Gamma] \rho \Rightarrow e :: [\Gamma] ((A \land P), \tilde{A}) \leftarrow [\Gamma] C p' \leftarrow \Delta \quad \text{By def. of subst.} \]

- **Case** \text{DeclMatchNeg} \quad \text{By i.h. and rule } \text{MatchNeg}

- **Case** \text{DeclMatchWild} \quad \text{By i.h. and rule } \text{MatchWild}

- **Case** \[ \text{mgu}([\Omega] \sigma, [\Omega] \tau) = \perp \quad \text{Given} \]

\[ [\Omega] \Gamma / [\Omega] \sigma = [\Omega] \tau / [\Omega] (\rho \Rightarrow e) :: [\Omega] \tilde{A} \leftarrow [\Omega] C p \]

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

\[ \text{FEV}(\sigma = \tau) = \emptyset \quad \text{Given} \]

\[ [\Omega] \sigma = [\Gamma] \sigma \quad \text{By Lemma 39 (Principal Agreement) (i)} \]

\[ [\Omega] \tau = [\Gamma] \tau \quad \text{By above equalities} \]

\[ \text{mgu}([\Omega] \sigma, [\Omega] \tau) = \perp \quad \text{Given} \]

\[ \text{mgu}([\Gamma] \sigma, [\Gamma] \tau) = \perp \quad \text{By above equalities} \]

\[ \Gamma / [\sigma \vdash \tau : \kappa \rightarrow \perp] \quad \text{By Lemma 92 (Completeness of Elimeq) (2)} \]

\[ \Omega \mapsto \Omega \quad \text{By Lemma 92 (Completeness of Elimeq) (2)} \]

\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \quad \text{By Lemma 32 (Extension Reflexivity)} \]

- **Case** \[ \text{mgu}([\Omega] \sigma, [\Omega] \tau) = \emptyset \quad \text{Given} \]

\[ \text{mgu}([\Gamma] \sigma, [\Gamma] \tau) = \emptyset \quad \text{By above equalities} \]

\[ \Gamma / [\sigma \vdash \tau : \kappa \rightarrow \perp] \quad \text{By Lemma 92 (Completeness of Elimeq) (1)} \]

\[ \text{By Lemma 92 (Completeness of Elimeq) (1)} \]

\[ \text{By above equalities} \]

\[ \Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n) \quad \text{"} \]

\[ [\Gamma, \Theta] u = \theta([\Gamma] u) \quad \text{" for all } \Gamma : u : \kappa \]

\[ \Theta([\Omega] \Gamma) / \theta(\rho \Rightarrow [\Omega] e) :: \theta([\Omega] \tilde{A}) \leftarrow \theta([\Omega] C) \quad \text{Subderivation} \]

\[ \theta([\Omega] \Gamma) = [\Omega, \triangleright_p, \Theta]([\Gamma, \triangleright_p, \Theta) \quad \text{By Lemma 93 (Substitution Upgrade) (iii)} \]

\[ \theta([\Omega] \tilde{A}) = [\Omega, \triangleright_p, \Theta] \tilde{A} \quad \text{By Lemma 93 (Substitution Upgrade) (i) (over } \tilde{A}) \]

\[ \theta([\Omega] C) = [\Omega, \triangleright_p, \Theta] C \quad \text{By Lemma 93 (Substitution Upgrade) (i)} \]

\[ \theta(\rho \Rightarrow [\Omega] e) = [\Omega, \triangleright_p, \Theta] (\rho \Rightarrow e) \quad \text{By Lemma 93 (Substitution Upgrade) (iv)} \]

\[ [\Omega, \triangleright_p, \Theta]([\Gamma, \triangleright_p, \Theta) / [\Omega, \triangleright_p, \Theta] (\rho \Rightarrow e) :: \theta([\Omega, \triangleright_p, \Theta] \tilde{A}) \leftarrow [\Omega, \triangleright_p, \Theta] C \quad \text{By above equalities} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)\[\text{thm:typing-completeness}\]

\[\Gamma, \triangleright P, \Theta \vdash (\bar{\rho} \Rightarrow e) :: [\Gamma, \triangleright P, \Theta] \overrightarrow{\bar{A}} \leftarrow [\Gamma, \triangleright P, \Theta] C \ p \vdash \Delta, \triangleright P, \Delta' \quad \text{By i.h.}\]
\[\Delta, \triangleright P, \Delta' \rightarrow \Omega', \triangleright P, \Omega'' \quad ''\]
\[\Omega, \triangleright P, \Theta \rightarrow \Omega', \triangleright P, \Omega'' \quad ''\]
\[\text{dom}(\Delta, \triangleright P, \Delta') = \text{dom}(\Omega', \triangleright P, \Omega'') \quad ''\]
\[\exists \Delta \rightarrow \Omega' \quad \text{By Lemma 22 (Extension Inversion) (ii)}\]
\[\exists \text{dom}(\Delta) = \text{dom}(\Omega') \quad ''\]
\[\exists \Omega \rightarrow \Omega' \quad \text{By Lemma 22 (Extension Inversion) (ii)}\]
\[\exists \Gamma / [\Gamma] \sigma = [\Gamma] \tau \vdash \bar{\rho} \Rightarrow e :: [\Gamma] \bar{A} \leftarrow [\Gamma] C \ p \vdash \Delta \quad \text{By MatchUnify} \]