Sound and Complete Bidirectional Typechecking for Higher-Rank Polymorphism and Indexed Types: Lemmas and Proofs

Joshua Dunfield Neelakantan R. Krishnaswami

March 2, 2015

Contents

<p>| A Definitions | 7 |
| B Properties of the Declarative System | 23 |
| 1 Lemma (Declarative Weakening) | 23 |
| 2 Lemma (Declarative Term Substitution) | 23 |
| 3 Lemma (Reflexivity of Declarative Subtyping) | 23 |
| 4 Lemma (Subtyping Inversion) | 23 |
| 5 Lemma (Subtyping Polarity Flip) | 23 |
| 6 Lemma (Transitivity of Declarative Subtyping) | 23 |
| C Substitution and Well-formedness Properties | 23 |
| 7 Lemma (Substitution—Well-formedness) | 23 |
| 8 Lemma (Uvar Preservation) | 24 |
| 9 Lemma (Sorting Implies Typing) | 24 |
| 10 Lemma (Right-Hand Substitution for Sorting) | 24 |
| 11 Lemma (Right-Hand Substitution for Propositions) | 24 |
| 12 Lemma (Right-Hand Substitution for Typing) | 24 |
| 13 Lemma (Substitution for Sorting) | 24 |
| 14 Lemma (Substitution for Prop Well-Formedness) | 24 |
| 15 Lemma (Substitution for Type Well-Formedness) | 24 |
| 16 Lemma (Substitution Stability) | 24 |
| 17 Lemma (Equal Domains) | 24 |
| D Properties of Extension | 24 |
| 18 Lemma (Declaration Preservation) | 24 |
| 19 Lemma (Declaration Order Preservation) | 24 |
| 20 Lemma (Reverse Declaration Order Preservation) | 24 |
| 21 Lemma (Extension Inversion) | 24 |
| 22 Lemma (Deep Evar Introduction) | 25 |
| 23 Lemma (Soft Extension) | 25 |
| 25 Lemma (Parallel Admissibility) | 25 |
| 26 Lemma (Parallel Extension Solution) | 25 |
| 27 Lemma (Parallel Variable Update) | 25 |
| 28 Lemma (Substitution Monotonicity) | 25 |
| 29 Lemma (Substitution Invariance) | 26 |
| 30 Lemma (Split Extension) | 26 |
| D.1 Reflexivity and Transitivity | 26 |
| 31 Lemma (Extension Reflexivity) | 26 |
| 32 Lemma (Extension Transitivity) | 26 |
| D.2 Weakening | 26 |
| 33 Lemma (Suffix Weakening) | 26 |
| 34 Lemma (Suffix Weakening) | 26 |</p>
<table>
<thead>
<tr>
<th>Page</th>
<th>Lemma (Extension Weakening (Sorts))</th>
<th>Lemma (Extension Weakening (Props))</th>
<th>Lemma (Extension Weakening (Types))</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.3</td>
<td>Principal Typing Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Lemma (Principal Agreement)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Lemma (Right-Hand Subst. for Principal Typing)</td>
<td>Lemma (Extension Weakening for Principal Typing)</td>
<td>Lemma (Inversion of Principal Typing)</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.4</td>
<td>Instantiation Extends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Instantiation Extension)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.5</td>
<td>Equivalence Extends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Elimeq Extension)</td>
<td>Lemma (Elimprop Extension)</td>
<td>Lemma (Checkeq Extension)</td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Checkprop Extension)</td>
<td></td>
<td>Lemma (Prop Equivalence Extension)</td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Equivalence Extension)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.6</td>
<td>Subtyping Extends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Subtyping Extension)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.7</td>
<td>Typing Extends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Typing Extension)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D.8</td>
<td>Unfiled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Context Partitioning)</td>
<td>Lemma (Completing Stability)</td>
<td>Lemma (Completing Completeness)</td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Completing Completeness)</td>
<td>Lemma (Confluence of Completeness)</td>
<td>Lemma (Multiple Confluence)</td>
</tr>
<tr>
<td>27</td>
<td>Lemma (Canonical Completion)</td>
<td></td>
<td>Lemma (Split Solutions)</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Internal Properties of the Declarative System</td>
<td>Lemma (Interpolating With and Exists)</td>
<td>Lemma (Case Invertibility)</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>Miscellaneous Properties of the Algorithmic System</td>
<td>Lemma (Well-Formed Outputs of Typing)</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>Decidability of Instantiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Lemma (Left Unsolvedness Preservation)</td>
<td>Lemma (Left Free Variable Preservation)</td>
<td>Lemma (Instantiation Size Preservation)</td>
</tr>
<tr>
<td>29</td>
<td>Lemma (Decidability of Instantiation)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>Separation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Lemma (Transitivity of Separation)</td>
<td>Lemma (Separation Truncation)</td>
<td>Lemma (Separation for Auxiliary Judgments)</td>
</tr>
<tr>
<td>30</td>
<td>Lemma (Separation for Subtyping)</td>
<td>Lemma (Separation—Main)</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Decidability of Algorithmic Subtyping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Lemmas for Decidability of Subtyping</td>
<td>Lemma (Substitution Isn’t Large)</td>
<td>Lemma (Instantiation Solves)</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Lemma (Checkeq Solving)</td>
<td>Lemma (Prop Equiv Solving)</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Lemma (Equiv Solving)</td>
<td>Lemma (Decidability of Propositional Judgments)</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Lemma (Decidability of Equivalence)</td>
<td></td>
</tr>
</tbody>
</table>
I.2 Decidability of Subtyping 31
1 Theorem (Decidability of Subtyping) 31

I.3 Decidability of Matching and Coverage 31
2 Lemma (Decidability of Expansion Judgments) 31
3 Theorem (Decidability of Coverage) 31

I.4 Decidability of Typing 31
4 Theorem (Decidability of Typing) 31

J Determinancy 32
5 Lemma (Determinacy of Auxiliary Judgments) 32
6 Lemma (Determinacy of Equivalence) 32
4 Theorem (Determinacy of Subtyping) 32
5 Theorem (Determinacy of Typing) 32

K Properties of Algorithmic Subtyping 33

L Soundness 33

L.1 Soundness of Instantiation 33
1 Lemma (Soundness of Instantiation) 33

L.2 Soundness of Checkeq 33
2 Lemma (Soundness of Checkeq) 33

L.3 Soundness of Equivalence (Propositions and Types) 33
3 Lemma (Soundness of Propositional Equivalence) 33
4 Lemma (Soundness of Algorithmic Equivalence) 33

L.4 Soundness of Checkprop 33
5 Lemma (Soundness of Checkprop) 33

L.5 Soundness of Eliminations (Equality and Proposition) 33
6 Lemma (Soundness of Equality Elimination) 33
7 Theorem (Soundness of Algorithmic Subtyping) 33

L.6 Soundness of Typing 33
8 Theorem (Soundness of Match Coverage) 33
9 Lemma (Well-formedness of Algorithmic Typing) 33
10 Theorem (Soundness of Algorithmic Typing) 33

M Completeness 35

M.1 Completeness of Auxiliary Judgments 35
1 Lemma (Completeness of Instantiation) 35
2 Lemma (Completeness of Checkeq) 35
3 Lemma (Completeness of Elimeq) 35
4 Lemma (Substitution Upgrade) 35
5 Lemma (Completeness of Propequiv) 35
6 Lemma (Completeness of Checkprop) 35

M.2 Completeness of Equivalence and Subtyping 36
7 Lemma (Completeness of Equiv) 36
8 Theorem (Completeness of Subtyping) 36

M.3 Completeness of Typing 36
9 Theorem (Completeness of Match Coverage) 36
10 Theorem (Completeness of Algorithmic Typing) 36

Proofs 38

B Properties of the Declarative System 38

1 Proof of Lemma (Declarative Weakening) 38
2 Proof of Lemma (Declarative Term Substitution) 38
3 Proof of Lemma (Reflexivity of Declarative Subtyping) 38
4 Proof of Lemma (Subtyping Inversion) 39
5 Proof of Lemma (Subtyping Polarity Flip) 39
6 Proof of Lemma (Transitivity of Declarative Subtyping) 39
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Substitution and Well-formedness Properties</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution—Well-formedness)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Uvar Preservation)</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Sorting Implies Typing)</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Substitution for Sorting)</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Substitution for Propositions)</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Substitution for Typing)</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution for Sorting)</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution for Prop Well-Formed)</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution for Type Well-Formed)</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution Stability)</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Equal Domains)</td>
<td>45</td>
</tr>
<tr>
<td>D</td>
<td>Properties of Extension</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Declaration Preservation)</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Declaration Order Preservation)</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Reverse Declaration Order Preservation)</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Inversion)</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Deep Evar Introduction)</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Parallel Admissibility)</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Parallel Extension Solution)</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Parallel Variable Update)</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution Monotonicity)</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Substitution Invariance)</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Soft Extension)</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Split Extension)</td>
<td>62</td>
</tr>
<tr>
<td>D'1</td>
<td>Reflexivity and Transitivity</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Reflexivity)</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Transitivity)</td>
<td>63</td>
</tr>
<tr>
<td>D'2</td>
<td>Weakening</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Suffix Weakening)</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Suffix Weakening)</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Weakening (Sorts))</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Weakening (Props))</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Weakening (Types))</td>
<td>66</td>
</tr>
<tr>
<td>D'3</td>
<td>Principal Typing Properties</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Principal Agreement)</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Right-Hand Subst. for Principal Typing)</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Extension Weakening for Principal Typing)</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Inversion of Principal Typing)</td>
<td>67</td>
</tr>
<tr>
<td>D'4</td>
<td>Instantiation Extends</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Instantiation Extension)</td>
<td>68</td>
</tr>
<tr>
<td>D'5</td>
<td>Equivalence Extends</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Elimeq Extension)</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Elimprop Extension)</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Checkeq Extension)</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Checkprop Extension)</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Prop Equivalence Extension)</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Equivalence Extension)</td>
<td>71</td>
</tr>
<tr>
<td>D'6</td>
<td>Subtyping Extends</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Subtyping Extension)</td>
<td>71</td>
</tr>
<tr>
<td>D'7</td>
<td>Typing Extends</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Typing Extension)</td>
<td>72</td>
</tr>
<tr>
<td>D'8</td>
<td>Unfiled</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Context Partitioning)</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Completing Stability)</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Completing Completeness)</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>Proof of Lemma (Confluence of Completeness)</td>
<td>75</td>
</tr>
</tbody>
</table>

March 2, 2015
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>Proof of Lemma (Multiple Confluence)</td>
<td>75</td>
</tr>
<tr>
<td>58</td>
<td>Proof of Lemma (Canonical Completion)</td>
<td>75</td>
</tr>
<tr>
<td>59</td>
<td>Proof of Lemma (Split Solutions)</td>
<td>75</td>
</tr>
<tr>
<td>60</td>
<td>Internal Properties of the Declarative System</td>
<td>75</td>
</tr>
<tr>
<td>61</td>
<td>Proof of Lemma (Interpolating With and Exists)</td>
<td>76</td>
</tr>
<tr>
<td>62</td>
<td>Proof of Lemma (Case Invertibility)</td>
<td>76</td>
</tr>
<tr>
<td>63</td>
<td>Miscellaneous Properties of the Algorithmic System</td>
<td>76</td>
</tr>
<tr>
<td>64</td>
<td>Proof of Lemma (Well-Formed Outputs of Typing)</td>
<td>76</td>
</tr>
<tr>
<td>65</td>
<td>Decidability of Instantiation</td>
<td>77</td>
</tr>
<tr>
<td>66</td>
<td>Proof of Lemma (Left Unsolvedness Preservation)</td>
<td>77</td>
</tr>
<tr>
<td>67</td>
<td>Proof of Lemma (Left Free Variable Preservation)</td>
<td>78</td>
</tr>
<tr>
<td>68</td>
<td>Proof of Lemma (Instantiation Size Preservation)</td>
<td>79</td>
</tr>
<tr>
<td>69</td>
<td>Proof of Lemma (Decidability of Instantiation)</td>
<td>80</td>
</tr>
<tr>
<td>70</td>
<td>Separation</td>
<td>81</td>
</tr>
<tr>
<td>71</td>
<td>Proof of Lemma (Transitivity of Separation)</td>
<td>81</td>
</tr>
<tr>
<td>72</td>
<td>Proof of Lemma (Separation Truncation)</td>
<td>81</td>
</tr>
<tr>
<td>73</td>
<td>Proof of Lemma (Separation for Auxiliary Judgments)</td>
<td>83</td>
</tr>
<tr>
<td>74</td>
<td>Proof of Lemma (Separation for Subtyping)</td>
<td>83</td>
</tr>
<tr>
<td>75</td>
<td>Proof of Lemma (Separation—Main)</td>
<td>84</td>
</tr>
<tr>
<td>76</td>
<td>Decidability of Algorithmic Subtyping</td>
<td>89</td>
</tr>
<tr>
<td>77</td>
<td>Lemmas for Decidability of Subtyping</td>
<td>89</td>
</tr>
<tr>
<td>78</td>
<td>Proof of Lemma (Substitution Isn’t Large)</td>
<td>89</td>
</tr>
<tr>
<td>79</td>
<td>Proof of Lemma (Instantiation Solves)</td>
<td>89</td>
</tr>
<tr>
<td>80</td>
<td>Proof of Lemma (Checkeq Solving)</td>
<td>90</td>
</tr>
<tr>
<td>81</td>
<td>Proof of Lemma (Prop Equiv Solving)</td>
<td>90</td>
</tr>
<tr>
<td>82</td>
<td>Proof of Lemma (Equiv Solving)</td>
<td>91</td>
</tr>
<tr>
<td>83</td>
<td>Proof of Lemma (Decidability of Propositional Judgments)</td>
<td>91</td>
</tr>
<tr>
<td>84</td>
<td>Proof of Lemma (Decidability of Equivalence)</td>
<td>92</td>
</tr>
<tr>
<td>85</td>
<td>Decidability of Subtyping</td>
<td>93</td>
</tr>
<tr>
<td>86</td>
<td>Proof of Theorem (Decidability of Subtyping)</td>
<td>93</td>
</tr>
<tr>
<td>87</td>
<td>Decidability of Matching and Coverage</td>
<td>95</td>
</tr>
<tr>
<td>88</td>
<td>Proof of Lemma (Decidability of Expansion Judgments)</td>
<td>95</td>
</tr>
<tr>
<td>89</td>
<td>Proof of Theorem (Decidability of Coverage)</td>
<td>95</td>
</tr>
<tr>
<td>90</td>
<td>Decidability of Typing</td>
<td>95</td>
</tr>
<tr>
<td>91</td>
<td>Proof of Theorem (Decidability of Typing)</td>
<td>95</td>
</tr>
<tr>
<td>92</td>
<td>Determinacy</td>
<td>97</td>
</tr>
<tr>
<td>93</td>
<td>Proof of Lemma (Determinacy of Auxiliary Judgments)</td>
<td>97</td>
</tr>
<tr>
<td>94</td>
<td>Proof of Lemma (Determinacy of Equivalence)</td>
<td>99</td>
</tr>
<tr>
<td>95</td>
<td>Proof of Theorem (Determinacy of Subtyping)</td>
<td>99</td>
</tr>
<tr>
<td>96</td>
<td>Proof of Theorem (Determinacy of Typing)</td>
<td>99</td>
</tr>
<tr>
<td>97</td>
<td>Properties of Algorithmic Subtyping</td>
<td>101</td>
</tr>
<tr>
<td>98</td>
<td>Soundness</td>
<td>101</td>
</tr>
<tr>
<td>99</td>
<td>Instantiation</td>
<td>101</td>
</tr>
<tr>
<td>100</td>
<td>Proof of Lemma (Soundness of Instantiation)</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>Proof of Lemma (Soundness of Checkeq)</td>
<td>102</td>
</tr>
<tr>
<td>102</td>
<td>Proof of Lemma (Soundness of Propositional Equivalence)</td>
<td>103</td>
</tr>
<tr>
<td>103</td>
<td>Proof of Lemma (Soundness of Algorithmic Equivalence)</td>
<td>103</td>
</tr>
<tr>
<td>104</td>
<td>Soundness of Checkprop</td>
<td>105</td>
</tr>
<tr>
<td>105</td>
<td>Proof of Lemma (Soundness of Checkprop)</td>
<td>105</td>
</tr>
<tr>
<td>106</td>
<td>Soundness of Eliminations (Equality and Proposition)</td>
<td>105</td>
</tr>
<tr>
<td>107</td>
<td>Proof of Lemma (Soundness of Equality Elimination)</td>
<td>105</td>
</tr>
<tr>
<td>108</td>
<td>Proof of Theorem (Soundness of Algorithmic Subtyping)</td>
<td>107</td>
</tr>
</tbody>
</table>
CONTENTS

L'.4 Soundness of Typing .................................................. 109
7 Proof of Theorem (Soundness of Match Coverage) .................. 109
88 Proof of Lemma (Well-formedness of Algorithmic Typing) ....... 110
81 Proof of Theorem (Soundness of Algorithmic Typing) .......... 112

M' Completeness ............................................................... 122
M'.1 Completeness of Auxiliary Judgments .............................. 122
89 Proof of Lemma (Completeness of Instantiation) ................. 122
90 Proof of Lemma (Completeness of Checkeq) ...................... 124
91 Proof of Lemma (Completeness of Elimeq) ....................... 125
92 Proof of Lemma (Substitution Upgrade) ............................ 127
93 Proof of Lemma (Completeness of Propequiv) .................... 127
94 Proof of Lemma (Completeness of Checkprop) ................... 128

M'.2 Completeness of Equivalence and Subtyping ...................... 128
95 Proof of Lemma (Completeness of Equiv) .......................... 128
9 Proof of Theorem (Completeness of Subtyping) .................. 131

M'.3 Completeness of Typing ................................................. 135
10 Proof of Theorem (Completeness of Match Coverage) .......... 135
11 Proof of Theorem (Completeness of Algorithmic Typing) ....... 136
## A Definitions

<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 4</td>
</tr>
<tr>
<td>$\Psi \vdash P \ prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 4</td>
</tr>
<tr>
<td>$\Psi \vdash A \ type$</td>
<td>Type is well-formed</td>
<td>Figure 4</td>
</tr>
<tr>
<td>$\Psi \vdash \vec{A} \ types$</td>
<td>Type vector is well-formed</td>
<td>Figure 4</td>
</tr>
<tr>
<td>$\Psi \vdash \text{ctx}$</td>
<td>Declarative context is well-formed</td>
<td>Figure 4</td>
</tr>
<tr>
<td>$\Psi \vdash A \leq \pm B$</td>
<td>Declarative subtyping</td>
<td>Figure 5</td>
</tr>
<tr>
<td>$\Psi \vdash P \ true$</td>
<td>Declarative truth</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash e \Leftarrow A \ p$</td>
<td>Declarative checking</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash e \Rightarrow A \ p$</td>
<td>Declarative synthesis</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash s : A \ p \Rightarrow C \ q$</td>
<td>Declarative spine typing</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi : \vec{A} \Leftarrow C \ p$</td>
<td>Declarative spine typing, recovering principality</td>
<td>Figure 7</td>
</tr>
<tr>
<td>$\Psi \vdash \Pi \ covers \vec{A}$</td>
<td>Declarative pattern matching</td>
<td>Figure 8</td>
</tr>
<tr>
<td>$\Gamma \vdash \tau : \kappa$</td>
<td>Index term/monotype is well-formed</td>
<td>Figure 11</td>
</tr>
<tr>
<td>$\Gamma \vdash P \ prop$</td>
<td>Proposition is well-formed</td>
<td>Figure 11</td>
</tr>
<tr>
<td>$\Gamma \vdash A \ type$</td>
<td>Polytype is well-formed</td>
<td>Figure 11</td>
</tr>
<tr>
<td>$\Gamma \vdash \text{ctx}$</td>
<td>Algorithmic context is well-formed</td>
<td>Figure 11</td>
</tr>
<tr>
<td>$\Gamma \vdash P \ true \ \vdash \Delta$</td>
<td>Check proposition</td>
<td>Figure 13</td>
</tr>
<tr>
<td>$\Gamma \vdash P \vdash \Delta^\perp$</td>
<td>Assume proposition</td>
<td>Figure 13</td>
</tr>
<tr>
<td>$\Gamma \vdash s \equiv t : \kappa \vdash \Delta$</td>
<td>Check equation</td>
<td>Figure 14</td>
</tr>
<tr>
<td>$\Gamma \vdash s \equiv t : \kappa \vdash \Delta^\perp$</td>
<td>Assume/eliminate equation</td>
<td>Figure 15</td>
</tr>
<tr>
<td>$\Gamma \vdash A \Leftarrow \pm B \vdash \Delta$</td>
<td>Algorithmic subtyping</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash A \Leftarrow \pm B \vdash \Delta$</td>
<td>Assume/eliminate proposition</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash A \equiv Q \vdash \Delta$</td>
<td>Equivalence of propositions</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash A \equiv B \vdash \Delta$</td>
<td>Equivalence of types</td>
<td>Figure 17</td>
</tr>
<tr>
<td>$\Gamma \vdash \vec{A} \Leftarrow t : \kappa \vdash \Delta$</td>
<td>Instantiate</td>
<td>Figure 18</td>
</tr>
<tr>
<td>$\text{e chk-I}$</td>
<td>Checking intro form</td>
<td>Figure 19</td>
</tr>
<tr>
<td>$\Gamma \vdash e \Leftarrow A \ p \vdash \Delta$</td>
<td>Algorithmic checking</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash e \Rightarrow A \ p \vdash \Delta$</td>
<td>Algorithmic synthesis</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash s : A \ p \Rightarrow C \ q \vdash \Delta$</td>
<td>Algorithmic spine typing</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash s : A \ p \Rightarrow C \ q \vdash \Delta$</td>
<td>Algorithmic spine typing, recovering principality</td>
<td>Figure 20</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi : \vec{A} \Leftarrow C \ p \vdash \Delta$</td>
<td>Algorithmic pattern matching</td>
<td>Figure 21</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi : \vec{A} \Leftarrow C \ p \vdash \Delta$</td>
<td>Algorithmic pattern matching (assumption)</td>
<td>Figure 21</td>
</tr>
<tr>
<td>$\Gamma \vdash \Pi \ covers \vec{A}$</td>
<td>Algorithmic match coverage</td>
<td>Figure 22</td>
</tr>
<tr>
<td>$\Gamma \rightarrow \Delta$</td>
<td>Context extension</td>
<td>Figure 23</td>
</tr>
<tr>
<td>$\Omega \Gamma$</td>
<td>Apply complete context</td>
<td>Figure 24</td>
</tr>
</tbody>
</table>

Figure 1: List of judgments
A Definitions

### Expressions
\[ e ::= x \mid () \mid \lambda x. e \mid e_1 \ (e_2 \ - s) \mid (e : A) \]
\[ \mid \langle e_1, e_2 \rangle \mid \text{inj}_1 e \mid \text{inj}_2 e \mid \text{case}(e, \Pi) \]

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

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

### Patterns
\[ \rho ::= x \mid (\rho_1, \rho_2) \mid \text{inj}_1 \rho \mid \text{inj}_2 \rho \]

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

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

Figure 2: Source syntax

---

### Universal variables
\[ \alpha, \beta, \gamma \]

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

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

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

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

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

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

### Binary connectives
\[ \oplus ::= \to \mid + \mid \times \]

### Principalities
\[ p, q ::= ! \mid \mathcal{F} \]

sometimes omitted

Figure 3: Syntax of declarative types and contexts
\[ \Psi \vdash t : \kappa \] Under context \( \Psi \), term \( t \) has sort \( \kappa \)

\[ \frac{\alpha : \kappa}{\Psi \vdash \alpha : \kappa} \] UvarSort
\[ \Psi \vdash 1 : \ast \] UnitSort
\[ \frac{\Psi \vdash t_1 : \ast \quad \Psi \vdash t_2 : \ast}{\Psi \vdash t_1 \oplus t_2 : \ast} \] BinSort
\[ \Psi \vdash \text{zero} : \text{N\text{ZeroSort}} \]
\[ \Psi \vdash \text{succ}(t) : \text{N\text{SuccSort}} \]

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

\[ \frac{\Psi \vdash t : \text{N} \quad \Psi \vdash t' : \text{N}}{\Psi \vdash t = t' : \ast} \] EqDeclProp

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

\[ \frac{\alpha : \ast \in \Psi}{\Psi \vdash \alpha : \kappa} \] DeclUvarWF
\[ \Psi \vdash 1 \text{ type} \] DeclUnitWF
\[ \frac{\Psi \vdash A \text{ type} \quad \Psi \vdash B \text{ type}}{\Psi \vdash A \oplus B \text{ type}} \oplus \in \{\rightarrow, \times, +\} \] DeclBinWF
\[ \frac{\Psi, \alpha : \kappa \vdash A \text{ type}}{\Psi \vdash \forall \alpha : \kappa. A \text{ type}} \] DeclAllWF
\[ \frac{\Psi, \alpha : \kappa \vdash A \text{ type}}{\Psi \vdash \exists \alpha : \kappa. A \text{ type}} \] DeclExistsWF
\[ \frac{\Psi \vdash P \text{ prop} \quad \Psi \vdash A \text{ type}}{\Psi \vdash P \supset A \text{ type}} \] DeclImpliesWF
\[ \frac{\Psi \vdash P \text{ prop} \quad \Psi \vdash A \text{ type}}{\Psi \vdash A \land P \text{ type}} \] DeclWithWF

\[ \Psi \vdash \vec{\Lambda} \text{ types} \] Under context \( \Psi \), types in \( \vec{\Lambda} \) are well-formed

for all \( A \in \vec{\Lambda} \).
\[ \frac{\Psi \vdash A \text{ type}}{\Psi \vdash \vec{\Lambda} \text{ types}} \] DeclTypevecWF

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

\[ \frac{\Psi \text{ ctx}}{\cdot \text{ctx}} \] EmptyDeclCtx
\[ \frac{\Psi \text{ ctx} \quad x \notin \text{dom}(\Psi)}{\Psi, x : A \text{ ctx}} \] HypDeclCtx
\[ \frac{\Psi \text{ ctx} \quad \alpha \notin \text{dom}(\Psi)}{\Psi, \alpha : \kappa \text{ ctx}} \] VarDeclCtx

Figure 4: Sorting; well-formedness of propositions, types, and contexts in the declarative system
\[ \psi \vdash A \leq^\pm B \]

Under context \( \psi \), type \( A \) is a subtype of \( B \), decomposing head connectives of polarity \( \pm \)

\[
\frac{\psi \vdash A \text{ type} \quad \text{nonpos}(A) \quad \text{nonneg}(A)}{\psi \vdash A \leq^\pm A} \leq_{\text{Refl}^\pm} \\
\frac{\psi \vdash A \leq^\pm B \quad \text{nonpos}(A) \quad \text{nonpos}(B)}{\psi \vdash A \leq^+ B} \leq^+ \\
\frac{\psi \vdash A \leq^\pm B \quad \text{nonneg}(A) \quad \text{nonneg}(B)}{\psi \vdash A \leq^- B} \leq^- \\
\frac{\psi \vdash \tau : \kappa \quad \psi \vdash [\tau/\alpha] A \leq^\pm B}{\psi \vdash \forall \alpha : \kappa. A \leq^\pm B} \leq_{\forall L} \\
\frac{\psi, \beta : \kappa \vdash A \leq^- B}{\psi \vdash A \leq^- \forall \beta : \kappa. B} \leq_{\forall R} \\
\frac{\psi, \alpha : \kappa \vdash A \leq^+ B}{\psi \vdash \exists \alpha : \kappa. A \leq^+ B} \leq_{\exists L} \\
\frac{\psi \vdash \tau : \kappa \quad \psi \vdash A \leq^+ [\tau/\beta] B}{\psi \vdash A \leq^+ \exists \beta : \kappa. B} \leq_{\exists R}
\]

Figure 5: Subtyping in the declarative system

\[ \psi \vdash P \text{ true} \]

Under context \( \psi \), check \( P \)

\[
\frac{\psi \vdash (t = t) \text{ true}}{\text{DeclCheckpropEq}}
\]

Figure 6: Declarative truth
A Definitions

\[
\begin{align*}
\Psi \vdash e \triangleq A \ p & \quad \text{Under context } \Psi, \text{ expression } e \text{ checks against input type } A \\
\Psi \vdash e \Rightarrow A \ p & \quad \text{Under context } \Psi, \text{ expression } e \text{ synthesizes output type } A \\
\Psi \vdash s : A \ p \gg C q & \quad \text{Under context } \Psi, \text{ passing spine } s \text{ to a function of type } A \text{ synthesizes type } C; \\
& \quad \text{in the } [q] \text{ form, recover principality in } q \text{ if possible} \\
\Psi \vdash P \ true & \quad \text{Under context } \Psi, \text{ check } P
\end{align*}
\]

\[
\begin{align*}
\Psi \vdash (t = t) \ true & \quad \text{DeclCheckpropEq} \\
\Psi \vdash x : A \ p \in \Psi \quad & \text{DeclVar} \\
\Psi \vdash x \Rightarrow A \ p & \quad \text{DeclSub} \\
\Psi \vdash A \ type \quad & \text{DeclAnno} \\
\Psi \vdash (e : A) \Rightarrow A ! & \quad \text{DeclII} \\
\end{align*}
\]

\[
\begin{align*}
v \ \text{chk-I} & \quad \Psi, \alpha : \ k \vdash v \ \in \ A \ p \\
\Psi \vdash v \leftarrow \forall \alpha : \ k. A \ p & \quad \text{DeclI} \\
\Psi, x : A \ p \vdash e \ \in \ B \ p & \quad \text{DeclI} \\
\Psi \vdash \lambda x. e \ \in \ A \Rightarrow B \ p & \quad \text{Decl→I} \\
\Psi \vdash s : A \ ! \gg C \ ! & \quad \text{for all } C'. \\
& \quad \text{if } \Psi \vdash s : A \ ! \gg C' \ ! \text{ then } C' = C \\
\Psi \vdash s : A \ ! \gg C \ ! & \quad \text{DeclSpineRecover} \\
\Psi \vdash s : A \ p \gg C \ q & \quad \text{DeclSpinePass} \\
\Psi \vdash : A \ p \gg A \ p & \quad \text{DeclEmptySpine} \\
\Psi \vdash e \ \in \ A_k \ p & \quad \text{DeclI_k} \\
\Psi \vdash \text{inj}_k e \ \in \ A_1 \ + \ A_2 \ p & \quad \text{DeclI} \\
\Psi \vdash e \ \in \ A ! & \quad \text{DeclCase} \\
\Psi \vdash \Pi : \ A \Rightarrow C \ p & \quad \text{DeclCase} \\
\end{align*}
\]

\[
\begin{align*}
\Psi / P \vdash e \ \in \ C \ p & \quad \text{Under context } \Psi, \text{ incorporate proposition } P \\
& \quad \text{and check } e \text{ against } C'
\end{align*}
\]

\[
\begin{align*}
\text{mgu}(\sigma, \tau) = \bot & \quad \text{DeclCheck\bot} \\
\Psi / (\sigma = \tau) \vdash e \ \in \ C \ p & \quad \text{DeclCheckUnify} \\
\end{align*}
\]

Figure 7: Declarative typing
\[ \Psi \vdash \Pi :: \vec{A} \leftarrow Cp \] Under context \( \Psi \), check branches \( \Pi \) with patterns of type \( \vec{A} \) and bodies of type \( C \)

- **DeclMatchEmpty**: \( \Psi \vdash \cdot :: \vec{A} \leftarrow Cp \)
- **DeclMatchBase**: \( \Psi \vdash e :: Cp \)
- **DeclMatchUnit**: \( \Psi \vdash \emptyset :: \vec{A} \leftarrow Cp \)
- **DeclMatchSeq**: \( \Psi \vdash \Pi :: \vec{A} \leftarrow Cp \)
- **DeclMatchPlus**
- **DeclMatchTimes**
- **DeclMatchOr**
- **DeclMatchAnd**: \( \Psi, \alpha : \kappa \vdash \Pi :: A, \vec{A} \leftarrow Cp \)
- **DeclMatchOr**: \( \Psi \vdash e :: \exists \alpha : \kappa. A, \vec{A} \leftarrow Cp \)
- **DeclMatchPlus**: \( \Psi \vdash p, \rho \Rightarrow e :: A_k, \vec{A} \leftarrow Cp \)
- **DeclMatchTimes**: \( \Psi \vdash (\rho_1, \rho_2), \vec{\rho} \Rightarrow e :: A_1 \times A_2, \vec{A} \leftarrow Cp \)
- **DeclMatchOr**: \( \Psi \vdash \rho \Rightarrow e :: A_1 + A_2, \vec{A} \leftarrow Cp \)
- **DeclMatchAnd**: \( \Psi \vdash \rho \Rightarrow e :: A, \vec{A} \leftarrow Cp \)

- **DeclMatchNeg**
- **DeclMatchWild**

**DeclMatch**: Under context \( \Psi \), incorporate proposition \( P \) while checking branches \( \Pi \) with patterns of type \( \vec{A} \) and bodies of type \( C \)

- **mgu(\sigma, \tau) = \bot**
- **mgu(\sigma, \tau) = \theta**

Figure 8: Declarative pattern matching
$\Psi \vdash \Pi \text{ covers } \vec{A}$ \hfill Patterns $\Pi$ cover the types $\vec{A}$ in context $\Psi$

$\Psi \vdash (\cdot \Rightarrow e_1) \Pi' \text{ covers } \cdot$ \hfill DeclCoversEmpty

$\Pi \vdash \Pi' \text{ covers } \vec{A}$ \hfill $\Psi \vdash \Pi' \text{ covers } \vec{A}$ \hfill DeclCoversVar

$\Psi \vdash \Pi' \text{ covers } \vec{1}, \vec{A}$ \hfill $\Psi \vdash \Pi \text{ covers } \vec{1}, \vec{A}$ \hfill DeclCovers1

$\Pi \vdash \Pi' \text{ covers } A_1, A_2, \vec{A}$ \hfill $\Psi \vdash \Pi' \text{ covers } A_1, A_2, \vec{A}$ \hfill DeclCovers×

$\Psi \vdash \Pi \text{ covers } A_1 \times A_2, \vec{A}$ \hfill DeclCovers×

$\Pi \vdash \Pi \text{ covers } \vec{A}$ \hfill $\Psi \vdash \Pi \text{ covers } A_1, A_2, \vec{A}$ \hfill DeclCovers+

$\Pi \vdash \Pi \text{ covers } \vec{A}$ \hfill $\Psi \vdash \Pi \text{ covers } A_0 \wedge (t_1 = t_2), \vec{A}$ \hfill DeclCoversEq

$\Psi \vdash \Pi \text{ covers } \exists \alpha : \kappa. A, \vec{A}$ \hfill DeclCovers∃

$\theta = \text{mgu}(t_1, t_2)$ \hfill $\Psi \vdash \Pi \text{ covers } A_0 \wedge (t_1 = t_2), \vec{A}$ \hfill DeclCoversEq

$\Psi \vdash \Pi \text{ covers } A_0 \wedge (t_1 = t_2), \vec{A}$ \hfill DeclCoversEqBot

$\Pi \rightsquigarrow \Pi'$ \hfill Expand head pair patterns in $\Pi$

$\Pi \rightsquigarrow \Pi'$ \hfill $(\rho_1, \rho_2, \vec{\varphi} \Rightarrow e) \Pi \rightsquigarrow (\rho_1, \rho_2, \vec{\varphi} \Rightarrow e) \Pi'$ \hfill $\rho \in \{z_1\}$

$\Pi \rightsquigarrow \Pi'$ \hfill $(\rho, \vec{\varphi} \Rightarrow e) \Pi \rightsquigarrow (\rho, \vec{\varphi} \Rightarrow e) \Pi'$ \hfill $\Pi \rightsquigarrow \Pi'$

$\Pi \rightsquigarrow \Pi_L \parallel \Pi_R$ \hfill Expand head sum patterns in $\Pi$ into left $\Pi_L$ and right $\Pi_R$ sets

$\Pi \rightsquigarrow \Pi_L \parallel \Pi_R$ \hfill $\rho \in \{u_n, \_\}$

$\Pi \rightsquigarrow \Pi_L \parallel \Pi_R$ \hfill $(\rho, \vec{\varphi} \Rightarrow e) \Pi \rightsquigarrow (\rho, \vec{\varphi} \Rightarrow e) \Pi_L \parallel \Pi_R$

$\Pi \rightsquigarrow \Pi_L \parallel \Pi_R$ \hfill $(\text{inj}_1 \rho, \vec{\varphi} \Rightarrow e) \Pi \rightsquigarrow (\rho, \vec{\varphi} \Rightarrow e) \Pi_L \parallel \Pi_R$

$\Pi \rightsquigarrow \Pi_L \parallel \Pi_R$ \hfill $(\text{inj}_2 \rho, \vec{\varphi} \Rightarrow e) \Pi \rightsquigarrow (\rho, \vec{\varphi} \Rightarrow e) \Pi_L \parallel \Pi_R$

$\Pi \rightsquigarrow \Pi'$ \hfill Remove head variable and wildcard patterns from $\Pi$

$\Pi \parallel \Pi' \rightsquigarrow \Pi'$ \hfill $\Pi \rightsquigarrow \Pi'$ \hfill Remove head variable, wildcard, and unit patterns from $\Pi$

$\Pi \rightsquigarrow \Pi'$ \hfill $\rho \in \{u_n, \_\}$

$\Pi \rightsquigarrow \Pi'$ \hfill $\rho \in \{u_n, \_\}$

$\Pi \rightsquigarrow \Pi'$ \hfill $\rho \in \{u_n, \_\}$

Figure 9: Match coverage
Universal variables \( \alpha, \beta, \gamma \)
Existential variables \( \hat{\alpha}, \hat{\beta}, \hat{\gamma} \)

Variables
\[ u ::= \alpha | \hat{\alpha} \]

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

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

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

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

Contexts
\[ \Gamma, \Delta, \Theta ::= \cdot | \Gamma, u : \kappa | \Gamma, x : A p \]
\[ | \Gamma, \hat{\alpha} : \kappa = \tau | \Gamma, \alpha = t | \Gamma, \triangleright u \]

Complete contexts
\[ \Omega ::= \cdot | \Omega, \alpha : \kappa | \Omega, x : A p \]
\[ | \Omega, \hat{\alpha} : \kappa = \tau | \Omega, \alpha = t | \Omega, \triangleright u \]

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

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

Under context $\Gamma$, term $\tau$ has sort $\kappa$

$$\Gamma \vdash \tau : \kappa$$

$$(\alpha : \kappa) \in \Gamma$$  VarSort  $$\Gamma \vdash \alpha : \kappa$$  SolvedVarSort  $$\Gamma \vdash \top : \top$$

$$\Gamma \vdash \tau_1 : \top$$  $$\Gamma \vdash \tau_2 : \top$$  BinSort  $$\Gamma \vdash \top \circ \tau_1 \top \circ \tau_2 : \top$$

$$(\alpha : \kappa = \tau) \in \Gamma$$

$$\Gamma \vdash u : \kappa \in \Gamma$$  VarSort  $$\Gamma \vdash u : \kappa$$  SolvedVarSort  $$\Gamma \vdash \bot : \bot$$

$$\Gamma \vdash t : \bot$$

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

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

$$\Gamma \vdash t : \bot$$

$$\Gamma \vdash t' : \bot$$

$$\Gamma \vdash t = t' \text{ prop}$$

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

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

$$\Gamma, \alpha : \kappa \vdash A \text{ type}$$  ExistsWF  $$\Gamma \vdash P \text{ prop}$$

$$\Gamma \vdash A \text{ type}$$  SolvedVarWF  $$\Gamma \vdash A \bigcirc P$$  ImpliesWF  $$\Gamma \vdash A \text{ type}$$  WithWF

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

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

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

$$\Gamma \vdash \text{FEV}(\Gamma \mid A) = \emptyset$$  PrincipalWF  $$\Gamma \vdash A \text{ p type}$$

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

NonPrincipalWF

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

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

for all $A \in \vec{A}$.

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

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

for all $A \in \vec{A}$.

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

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

Algorithmic context $\Gamma$ is well-formed

$$\Gamma \text{ ctx}$$

$$\Gamma \text{ ctx}$$

$$\Gamma \vdash A \text{ type}$$  EmptyCtx  $$\Gamma \vdash x \notin \text{dom}(\Gamma)$$

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

$$\Gamma \vdash \text{HypCtx}$$  HypCtx  $$\Gamma \vdash x \notin \text{dom}(\Gamma)$$

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

$$\Gamma \vdash \text{FEV}(\Gamma \mid A) = \emptyset$$  Hyp!Ctx

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

$$\Gamma \vdash u \notin \text{dom}(\Gamma)$$  VarCtx  $$\Gamma \vdash \alpha \notin \text{dom}(\Gamma)$$

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

$$\Gamma \vdash \text{SolvedCtx}$$  SolvedCtx  $$\Gamma \vdash t : \kappa$$

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

$$\Gamma \vdash \alpha : \kappa \in \Gamma$$  EqnVarCtx  $$\Gamma \vdash \alpha : \kappa \in \Gamma$$

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

$$\Gamma \vdash \vec{u} \notin \Gamma$$  MarkerCtx

Figure 11: Well-formedness of types and contexts in the algorithmic system
\[\Gamma \vdash P \text{ true} \vdash \Delta\]

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

\[
\begin{align*}
\Gamma \vdash t_1 \equiv t_2 : \kappa \vdash \Delta & \quad \text{CheckeqVar} \\
\Gamma \vdash 1 \equiv 1 : \ast \vdash \Gamma & \quad \text{CheckeqUnit} \\
\Gamma \vdash \tau_1 \equiv \tau_1' : \ast \vdash \Theta & \quad \Theta \vdash [\Theta] \tau_2 \equiv [\Theta] \tau_2' : \ast \vdash \Delta & \quad \text{CheckeqBin} \\
\Gamma \vdash \text{zero} \equiv \text{zero} : \mathbb{N} \vdash \Gamma & \quad \text{CheckeqZero} \\
\Gamma \vdash \text{succ}(t_1) \equiv \text{succ}(t_2) : \mathbb{N} \vdash \Delta & \quad \text{CheckeqSucc} \\
\Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := t : \kappa \vdash \Delta & \quad \hat{\alpha} \not\in \text{FV}(t) & \quad \text{CheckeqInstL} \\
\Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := t : \kappa \vdash \Delta & \quad \hat{\alpha} \not\in \text{FV}(t) & \quad \text{CheckeqInstR}
\end{align*}
\]

Figure 14: Checking equations

\[t_1 \neq t_2\]

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

Figure 15: Head constructor clash
Unify $\sigma$ and $\tau$, taking $\Gamma$ to $\Delta$, or to inconsistency $\bot$.

\[ \Gamma / \sigma \trianglelefteq \tau : \kappa \rightarrow \Delta \]

\underline{ElimeqUnit}

\[ \Gamma / 1 \trianglelefteq 1 : \ast \rightarrow \Gamma \]

\underline{ElimeqZero}

\[ \Gamma / \text{zero} \trianglelefteq \text{zero} : N \rightarrow \Gamma \]

\underline{ElimeqSucc}

\[ \Gamma / \text{succ}(\sigma) \trianglelefteq \text{succ}(\tau) : N \rightarrow \Delta \]

\underline{ElimeqUvarRefI}

\[ \Gamma / \alpha \trianglelefteq \alpha : \kappa \rightarrow \Gamma \]

\underline{ElimeqUvarL}

\[ \alpha \not\in \text{FV}(\tau) \quad (\alpha = -) \not\in \Gamma \]

\[ \Gamma / \alpha \trianglelefteq \tau : \kappa \rightarrow \Gamma, \alpha = \tau \]

\underline{ElimeqUvarL_⊥}

\[ t \neq \alpha \quad \alpha \in \text{FV}(\tau) \]

\[ \Gamma / \alpha \trianglelefteq \tau : \kappa \rightarrow \bot \]

\underline{ElimeqUvarR}

\[ \alpha \not\in \text{FV}(\tau) \quad (\alpha = -) \not\in \Gamma \]

\[ \Gamma / \tau \trianglelefteq \alpha : \kappa \rightarrow \Gamma, \alpha = \tau \]

\underline{ElimeqUvarR_⊥}

\[ t \neq \alpha \quad \alpha \in \text{FV}(\tau) \]

\[ \Gamma / \tau \trianglelefteq \alpha : \kappa \rightarrow \bot \]

\underline{ElimeqBin}

\[ \Gamma / \tau_1 \trianglelefteq \tau_1' : \ast \rightarrow \Theta \quad \Theta / [\Theta]\tau_2 \trianglelefteq [\Theta]\tau_2' : \ast \rightarrow \Delta \]

\[ \Gamma / \tau_1 \oplus \tau_2 \trianglelefteq \tau_1' \oplus \tau_2' : \ast \rightarrow \Delta \]

\underline{ElimeqBinBot}

\[ \Gamma / \tau_1 \trianglelefteq \tau_1' : \ast \rightarrow \bot \]

\[ \Gamma / \tau_1 \oplus \tau_2 \trianglelefteq \tau_1' \oplus \tau_2' : \ast \rightarrow \bot \]

\underline{ElimeqClash}

\[ \sigma \neq \tau \]

\[ \Gamma / \sigma \trianglelefteq \tau : \kappa \rightarrow \bot \]

Figure 16: Eliminating equations
\[ \Gamma \vdash A \lessdot B \dashv \Delta \]

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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \vdash A \lessdot B \dashv \Delta )</td>
<td>B not headed by ( \forall )</td>
</tr>
<tr>
<td>( \vdash \forall \Theta \vdash \Gamma \vdash A \equiv B \dashv \Delta )</td>
<td>( A ) not headed by ( \forall )</td>
</tr>
<tr>
<td>( \vdash \forall \Theta \vdash \Gamma \vdash A \lessdot B \dashv \Delta )</td>
<td>( B ) not headed by ( \forall )</td>
</tr>
<tr>
<td>( \Gamma, \alpha \vdash A \lessdot B \dashv \Delta )</td>
<td>( \alpha ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma, \alpha \vdash \exists \alpha : \kappa \vdash A \dashv B \dashv \Delta )</td>
<td>( \alpha ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash \neg (\alpha) \vdash A \lessdot B \dashv \Delta )</td>
<td>( \neg (\alpha) ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma, \theta_1 \vdash \exists \theta_1 \vdash A \lessdot B \dashv \Delta )</td>
<td>( \exists \theta_1 ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash \neg (\beta) \vdash A \lessdot B \dashv \Delta )</td>
<td>( \neg (\beta) ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma, \beta \vdash \exists \beta : \kappa \vdash A \lessdot B \dashv \Delta )</td>
<td>( \exists \beta : \kappa ) not headed by ( \Gamma )</td>
</tr>
</tbody>
</table>

\[ \Gamma \vdash P \equiv Q \dashv \Delta \]

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

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

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

\[ \Gamma \vdash \alpha \equiv \alpha \dashv \Gamma \equiv \text{Var} \]
<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Gamma \vdash \alpha \equiv \alpha \dashv \Gamma \equiv \text{Var} )</td>
<td>( \alpha ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash \alpha \equiv \alpha \dashv \Gamma \equiv \text{Exvar} )</td>
<td>( \alpha ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash 1 \equiv 1 \dashv \Gamma \equiv \text{Unit} )</td>
<td>( 1 ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash A_1 \equiv B_1 \dashv \Theta \equiv \text{Equiv} )</td>
<td>( A_1 ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta \equiv \text{Equiv} )</td>
<td>( A_2 ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \dashv \Delta \equiv \text{Equiv} )</td>
<td>( (A_1 \oplus A_2) ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash (\forall \alpha : \kappa \vdash A) \equiv (\forall \alpha : \kappa \vdash B) \dashv \Delta \equiv \text{Equiv} )</td>
<td>( (\forall \alpha : \kappa \vdash A) ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash (\exists \alpha : \kappa \vdash A) \equiv (\exists \alpha : \kappa \vdash B) \dashv \Delta \equiv \text{Equiv} )</td>
<td>( (\exists \alpha : \kappa \vdash A) ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash P \equiv Q \dashv \Theta \equiv \text{Equiv} )</td>
<td>( P ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Theta \vdash [\Theta]A \equiv [\Theta]B \dashv \Delta \equiv \text{Equiv} )</td>
<td>( [\Theta]A ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \Gamma \vdash (P \lor Q) \equiv (B \lor Q) \dashv \Delta \equiv \text{Equiv} )</td>
<td>( (P \lor Q) ) not headed by ( \Gamma )</td>
</tr>
</tbody>
</table>

\[ \Gamma \vdash [\Delta] \vdash \alpha \equiv \tau : \dashv \Delta \equiv \text{Instantiatel} \]
<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Gamma \vdash [\Delta] \vdash \alpha \equiv \tau : \dashv \Delta \equiv \text{Instantiatel} )</td>
<td>( \alpha ) not headed by ( \Gamma )</td>
</tr>
<tr>
<td>( \alpha \not\in \text{FV}(\tau) )</td>
<td>( \alpha ) not headed by ( \Gamma )</td>
</tr>
</tbody>
</table>

\[ \Gamma \vdash [\Delta] \vdash \tau \equiv \dashv \Delta \equiv \text{Instantiatel} \]

Figure 17: Algorithmic equivalence and subtyping
$\Gamma \vdash \check{\alpha} := t : \kappa \rightarrow \Delta$

Under input context $\Gamma$, instantiate $\check{\alpha}$ such that $\check{\alpha} = t$ with output context $\Delta$

- $\Gamma_0 \vdash \tau : \kappa$
  - $\Gamma_0, \check{\alpha} : \kappa, \Gamma_1 \vdash \check{\alpha} := \tau : \kappa \rightarrow \Gamma_2, \check{\alpha} : \kappa = \tau, \Gamma_1$ \hspace{1cm} \text{InstSolve}

- $\check{\beta} \in \text{unsolved}(\Gamma[\check{\alpha} : \kappa][\check{\beta} : \kappa])$
  - $\Gamma[\check{\alpha} : \kappa][[\check{\beta} : \kappa]] \vdash \check{\alpha} := \check{\beta} : \kappa \rightarrow \Gamma[\check{\alpha} : \kappa][[\check{\beta} : \kappa] = \check{\alpha}]$ \hspace{1cm} \text{InstReach}

- $\Gamma[\check{\alpha}_2 : \star, \check{\alpha}_1 : \star, \check{\alpha} : \star = \check{\alpha}_1 \oplus \check{\alpha}_2] \vdash \check{\alpha}_1 := \tau_1 : \star \rightarrow \Theta \vdash \check{\alpha}_2 := [\Theta \tau_2 : \star \rightarrow \Delta]$ \hspace{1cm} \text{InstBin}

- $\Gamma[\check{\alpha} : \star] \vdash \check{\alpha} := \tau_1 \oplus \tau_2 : \star \rightarrow \Delta$

- $\Gamma[\check{\alpha} : \star] \vdash \check{\alpha} := \text{zero} : \star \rightarrow \Gamma[\check{\alpha} : \star = \text{zero}]$ \hspace{1cm} \text{InstZero}

- $\Gamma[\check{\alpha}_1 : \star, \check{\alpha}_2 : \star = \text{succ}(\check{\alpha}_1)] \vdash \check{\alpha}_1 := t_1 : \star \rightarrow \Delta \vdash \check{\alpha}_2 := \text{succ}(t_1) : \star \rightarrow \Delta$ \hspace{1cm} \text{InstSucc}

**Figure 18: Instantiation**

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

- $\lambda x. e \text{ \textit{chk-I}}$
- $() \text{ \textit{chk-I}}$
- $\langle e_1, e_2 \rangle \text{ \textit{chk-I}}$
- $\text{inj}_k e \text{ \textit{chk-I}}$

**Figure 19: “Checking intro form”**
A Definitions

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

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

Γ ⊢ s : A p ⇒ C q ⊢ Δ
Γ ⊢ s : A p ⇒ C [q] ⊢ Δ
Under input context Γ, passing spine s to a function of type A synthesizes type C; in the [q] form, recover principality in q if possible

\[ \{ x : A p \} \in \Gamma \]
\[ \Gamma \vdash x \Rightarrow [\Gamma]A p \vdash \Gamma \]

Sub
\[ \Gamma \vdash (e : A) \Rightarrow [\Gamma]A ! \vdash \Gamma \]

Anno

\[ \Gamma \vdash \emptyset \Leftarrow 1 p \vdash \Gamma \]

11\r

\[ \Gamma, \alpha : k \vdash v \Leftarrow A p \vdash \Delta, \alpha, \kappa, \Theta \]
\[ \Gamma \vdash v \Leftarrow \forall \alpha : k. A p \vdash \Delta \]

\forall I

v \text{ chk-I} \[ \Gamma \vdash v \Rightarrow P \vdash \Delta \]
\[ \Theta \vdash e \Leftarrow [\Theta]A p \vdash \Delta \]

∧ I

v \text{ chk-I} \[ \Gamma, \alpha \vdash p / P \vdash \Delta \]
\[ \Theta \vdash v \Leftarrow [\Theta]A p \vdash \Delta, \alpha, \kappa, \Theta \]
\[ \Gamma \vdash v \Leftarrow P \vdash A ! \vdash \Gamma \]

⇒ I

\[ \Gamma \vdash P \vdash \Delta \]

⇒ Spine

\[ \Gamma, x : A p \vdash e \Leftarrow B p \vdash \Delta, x : A p, \Theta \]
\[ \Gamma \vdash \lambda x. e \Leftarrow A \rightarrow B p \vdash \Delta \]

⇒ I

\[ \Gamma \vdash e \Rightarrow A p \vdash \Theta \]
\[ \Theta \vdash s : A p \Rightarrow C [q] \vdash \Delta \]

⇒ E

\[ \Gamma \vdash e \Rightarrow A p \vdash \Theta \]
\[ \Theta \vdash s : A p \Rightarrow C q \vdash \Delta \]

⇒ Spine

\[ \Gamma \vdash e \Leftarrow A p \vdash \Theta \]
\[ \Theta \vdash s : A p \Rightarrow C q \vdash \Delta \]

⇒ Spine

\[ \Gamma \vdash e \Leftarrow A p \vdash \Theta \]
\[ \Theta \vdash s : A p \Rightarrow C q \vdash \Delta \]

⇒ Spine

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

⇒ I\k

\[ \Gamma \vdash e_1 \Leftarrow A_1 p \vdash \Theta \]
\[ \Theta \vdash e_2 \Leftarrow [\Theta]A p \vdash \Delta \]

↓ I

\[ \Gamma \vdash \langle e_1, e_2 \rangle \Leftarrow A_1 \times A_2 p \vdash \Delta \]

⇒ Spine

\[ \Gamma \vdash [\alpha : \star] \Leftarrow \langle e_1, e_2 \rangle \Leftarrow \alpha \vdash \Delta \]

⇒ Spine

\[ \Gamma \vdash e \Rightarrow A ! \vdash \Theta \]
\[ \Theta \vdash \Pi : [\Theta]A \Leftarrow [\Theta]C p \vdash \Delta \]
\[ \Delta \vdash \Pi \text{ covers } [\Delta]A \]
\[ \Gamma \vdash \text{ case}(e, \Pi) \Leftarrow C p \vdash \Delta \]

Figure 20: Algorithmic typing
Figure 21: Algorithmic pattern matching
A Definitions

\[ \Gamma \vdash \Pi \text{ covers } \vec{A} \]

Under context \( \Gamma \), patterns \( \Pi \) cover the types \( \vec{A} \)

\[
\begin{align*}
\Gamma \vdash (\cdot \Rightarrow e) \Pi \text{ covers } \cdot & \quad \text{(CoversEmpty)} \\
\Pi \vdash \Pi' \text{ covers } \vec{A} & \Rightarrow \Pi \vdash \Pi' \text{ covers } \vec{A}, \vec{A} & \quad \text{(CoversVar)} \\
\Pi \vdash \Pi' \text{ covers } \vec{A} & \Rightarrow \Pi \vdash \Pi' \text{ covers } \vec{A}_1, \vec{A} & \quad \text{(Covers1)} \\
\Pi \vdash \Pi \sqcup \Pi \text{ covers } A_1, \vec{A} & \Rightarrow \Pi \vdash \Pi \text{ covers } A_1 \times A_2, \vec{A} & \quad \text{(Covers×)} \\
\Pi \vdash \Pi \text{ covers } \vec{A} & \Rightarrow \Pi \vdash \Pi \text{ covers } A_1 + A_2, \vec{A} & \quad \text{(Covers+)} \\
\Gamma, \alpha : \kappa \vdash \Pi \text{ covers } \vec{A} & \Rightarrow \Gamma \vdash \Pi \text{ covers } \exists \alpha : \kappa, A, \vec{A} & \quad \text{(Covers∃)} \\
\Gamma \vdash [\Gamma]t_1 \not\equiv [\Gamma]t_2 : \kappa \Rightarrow \Delta \vdash [\Delta] \Pi \text{ covers } [\Delta]\vec{A}, [\Delta]\vec{A} & \quad \text{(CoversEq)} \\
\Gamma \vdash \Pi \text{ covers } \vec{A} & \Rightarrow \Gamma \vdash \Pi \text{ covers } A_0 \land (t_1 = t_2), \vec{A} & \quad \text{(CoversEqBot)}
\end{align*}
\]

Figure 22: Algorithmic match coverage

\[ \Gamma \rightarrow \Delta \]

\( \Gamma \) is extended by \( \Delta \)

\[
\begin{align*}
\Gamma \rightarrow \Delta & \Rightarrow \text{Id} & \quad \Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa & \quad \text{(Var)} \\
\Gamma \rightarrow \Delta & \Rightarrow \text{Add} & \quad \Gamma \rightarrow \Delta & \Rightarrow \text{Eqn} \\
\Gamma, x : A \rightarrow \Delta, x : A' \rightarrow & \quad \Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa & \quad \text{(UVar)} \\
\Gamma \rightarrow \Delta, \vec{A} : \kappa & \Rightarrow \text{Solved} \\
\Gamma \rightarrow \Delta, \vec{A} : \kappa & \Rightarrow \text{Add} \\
\Gamma \rightarrow \Delta, \vec{A} : \kappa & \Rightarrow \text{AddSolved}
\end{align*}
\]

Figure 23: Context extension

\[
\begin{align*}
[\cdot] & = [\cdot] \\
[\Omega, x : A \ p]([\Gamma, x : A \Gamma] p) & = [\Omega] [\Gamma, x : A] \ p \text{ if } [\Omega] A = [\Omega] A_\Gamma \\
[\Omega, \alpha : \kappa]([\Gamma, \alpha : \kappa]) & = [\Omega] [\Gamma, \alpha : \kappa] \\
[\Omega, \vartriangleright u ([\Gamma, \vartriangleright u)] & = [\Omega] [\Gamma] \\
[\Omega, \alpha : t ([\Gamma, \alpha : t')] & = [\Omega] [\Gamma] [\Omega] [\Gamma'] \text{ if } [\Omega] t = [\Omega] t' \\
[\Omega, \vartriangleright \alpha : \kappa : t] & = \begin{cases} [\Omega] [\Gamma'] \text{ when } \Gamma = (\Gamma', \alpha : \kappa = t') \\
[\Omega] [\Gamma'] \text{ when } \Gamma = (\Gamma', \alpha : \kappa) \\
[\Omega] [\Gamma] \text{ otherwise}
\end{cases}
\end{align*}
\]

Figure 24: Applying a complete context \( \Omega \) to a context
B Properties of the Declarative System

Lemma 1 (Declarative Weakening). \[\text{Go to proof}\]

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

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

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

(iv) If \(\Psi_0, \Psi_1 \vdash A\) type then \(\Psi_0, \Psi, \Psi_1 \vdash A\) type.

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

Suppose \(\Psi \vdash t : \kappa\). Then:

1. If \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash t' : \kappa\) then \(\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]t' : \kappa\).
2. If \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash \text{prop}\) then \(\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]\text{prop}\).
3. If \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash A\) type then \(\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A\) type.
4. If \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash \ell \leq_\pm B\) then \(\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A \leq_\pm [t/\alpha]B\).
5. If \(\Psi_0, \alpha : \kappa, \Psi_1 \vdash \text{true}\) then \(\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]\text{true}\).

Lemma 3 (Reflexivity of Declarative Subtyping). \[\text{Go to proof}\]

Given \(\Psi \vdash A\) type, we have that \(\Psi \vdash A \leq^+ A\).

Lemma 4 (Subtyping Inversion). \[\text{Go to proof}\]

- If \(\Psi \vdash \exists \alpha : \kappa. A \leq^+ B\) then \(\Psi, \alpha : \kappa \vdash A \leq^+ B\).
- If \(\Psi \vdash A \leq^- \forall \beta : \kappa. B\) then \(\Psi, \beta : \kappa \vdash A \leq^- B\).

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

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

Lemma 6 (Transitivity of Declarative Subtyping). \[\text{Go to proof}\]

Given \(\Psi \vdash A\) type and \(\Psi \vdash B\) type and \(\Psi \vdash C\) type:

(i) If \(D_1 \vdash \Psi \vdash A \leq^\pm B\) and \(D_2 \vdash \Psi \vdash B \leq^\pm C\) 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, \(\text{FEV}(e) = \emptyset\).

C Substitution and Well-formedness Properties

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

Lemma 7 (Substitution—Well-formedness). \[\text{Go to proof}\]

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

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

Moreover, if \(\lambda = !\) and \(\text{FEV}([\Gamma]\lambda) = \emptyset\) then \(\text{FEV}([\Gamma][\tau/\alpha]\lambda) = \emptyset\).
Lemma 8 (Uvar Preservation). [Go to proof]
If $\Delta \rightarrow \Omega$ then:

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

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

Lemma 9 (Sorting Implies Typing). [Go to proof]
If $\Gamma \vdash t : \kappa$ then $\Gamma \vdash [\Gamma]t : \kappa$.

Lemma 10 (Right-Hand Substitution for Sorting). [Go to proof]
If $\Gamma \vdash t : \kappa$ then $\Gamma \vdash [\Gamma]t : \kappa$.

Lemma 11 (Right-Hand Substitution for Propositions). [Go to proof]
If $\Gamma \vdash P$ prop then $\Gamma \vdash [\Gamma]P$ prop.

Lemma 12 (Right-Hand Substitution for Typing). [Go to proof]
If $\Gamma \vdash \Delta$ type then $\Gamma \vdash [\Gamma]\Delta$ type.

Lemma 13 (Substitution for Typing). [Go to proof]
If $\Omega \vdash t : \kappa$ then $[\Omega]\Omega \vdash [\Omega]t : \kappa$.

Lemma 14 (Substitution for Prop Well-Formedness). [Go to proof]
If $\Omega \vdash A$ type then $[\Omega]\Omega \vdash [\Omega]A$ type.

Lemma 15 (Substitution for Type Well-Formedness). [Go to proof]
If $\Omega \vdash A$ type then $[\Omega]\Omega \vdash [\Omega]A$ type.

Lemma 16 (Substitution Stability). [Go to proof]
If $(\Omega, \Omega_2)$ is well-formed and $\Omega_2$ is soft and $\Omega \vdash A$ type then $[\Omega]A = [\Omega, \Omega_2]A$.

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

D Properties of Extension

Lemma 18 (Declaration Preservation). [Go to proof]
If $\Gamma \rightarrow \Delta$ and $u$ is declared in $\Gamma$, then $u$ is declared in $\Delta$.

Lemma 19 (Declaration Order Preservation). [Go to proof]
If $\Gamma \rightarrow \Delta$ and $u$ is declared to the left of $\nu$ in $\Gamma$, then $u$ is declared to the left of $\nu$ in $\Delta$.

Lemma 20 (Reverse Declaration Order Preservation). [Go to proof]
If $\Gamma \rightarrow \Delta$ and $u$ and $\nu$ are both declared in $\Gamma$ and $u$ is declared to the left of $\nu$ in $\Delta$, then $u$ is declared to the left of $\nu$ in $\Gamma$.

An older paper had a lemma

"Substitution Extension Invariance"
If $\Theta \vdash A$ type and $\Theta \rightarrow \Gamma$ then $[\Gamma]A = [\Gamma][\Theta]A$ and $[\Gamma]A = [\Theta][\Gamma]A$.

For the second part, $[\Gamma]A = [\Gamma][\Theta]A$, use Lemma 28 (Substitution Monotonicity) (i) or (iii) instead. The first part $[\Gamma]A = [\Gamma][\Theta]A$ hasn’t been proved in this system.

Lemma 21 (Extension Inversion). [Go to proof]

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

(ii) If $\Delta \vdash \Gamma_0, \bullet u, \Gamma_1 \rightarrow \Delta$
then there exist unique $\Delta_0$ and $\Delta_1$
such that $\Delta = (\Delta_0, \bullet u, \Delta_1)$ and $\Delta' \vdash \Gamma_0 \rightarrow \Delta_0$ where $\Delta' < \Delta$.
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 $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$.

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

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

Moreover, if $\Gamma_1$ is soft, then $\Delta_1$ is soft. Moreover, if $\text{dom}(\Gamma_0, x : A, \Gamma_1) = \text{dom}(\Delta)$ then $\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)$.

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

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

**Lemma 22** (Deep Evar Introduction). [Go to proof]
(i) If $\Gamma_0, \Gamma_1$ is well-formed and $\bar{\alpha}$ is not declared in $\Gamma_0, \Gamma_1$ then $\Gamma_0, \Gamma_1 \rightarrow \Gamma_0, \bar{\alpha} : \kappa, \Gamma_1$.

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

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

**Lemma 23** (Soft Extension). [Go to proof]
If $\Gamma \rightarrow \Delta$ and $\Gamma, \Theta \text{ctx}$ and $\Theta$ is soft, then there exists $\Omega$ such that $\text{dom}(\Theta) = \text{dom}(\Omega)$ and $\Gamma, \Theta \rightarrow \Delta, \Omega$.

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

$$
\begin{align*}
[\cdot] & = . \\
[\Gamma, x : A] & = [\Gamma], x : A \\
[\Gamma, \alpha : \kappa] & = [\Gamma], \alpha : \kappa \\
[\Gamma, \alpha = t] & = [\Gamma], \alpha = t \\
[\Gamma, \bar{\alpha} : \kappa = t] & = [\Gamma], \bar{\alpha} : \kappa = t \\
[\Gamma, \bar{\alpha} : \star] & = [\Gamma], \bar{\alpha} : \star = 1 \\
[\Gamma, \bar{\alpha} : \text{N}] & = [\Gamma], \bar{\alpha} : \text{N} = \text{zero}
\end{align*}
$$

**Lemma 24** (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 $[\cdot]$ and applying the rules for $\rightarrow$. ∎

**Lemma 25** (Parallel Admissibility). [Go to proof]
If $\Gamma_L \rightarrow \Delta_L$ and $\Gamma_L, \Gamma_R \rightarrow \Delta_L, \Delta_R$ then:

(i) $\Gamma_L, \bar{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa, \Delta_R$

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

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

**Lemma 26** (Parallel Extension Solution). [Go to proof]
If $\Gamma_L, \bar{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau', \Delta_R$ and $\Gamma_L \vdash \tau : \kappa$ and $[\Delta_L]\tau = [\Delta_L]\tau'$ then $\Gamma_L, \bar{\alpha} : \kappa = \tau, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau', \Delta_R$.

**Lemma 27** (Parallel Variable Update). [Go to proof]
If $\Gamma_L, \bar{\alpha} : \kappa, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau_0, \Delta_R$ and $\Gamma_L \vdash \tau_1 : \kappa$ and $\Delta_L \vdash \tau_2 : \kappa$ and $[\Delta_L]\tau_0 = [\Delta_L]\tau_1 = [\Delta_L]\tau_2$ then $\Gamma_L, \bar{\alpha} : \kappa = \tau_1, \Gamma_R \rightarrow \Delta_L, \bar{\alpha} : \kappa = \tau_2, \Delta_R$.

**Lemma 28** (Substitution Monotonicity). [Go to proof]
(i) If $\Gamma \rightarrow \Delta$ and $\Gamma \vdash t : \kappa$ then $[\Delta][\Gamma]t = [\Delta]t$.

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

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

March 2, 2015
Lemma 29 (Substitution Invariance). \footnote{Go to proof}

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

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

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

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

Lemma 30 (Split Extension). \footnote{Go to proof}

If \(\Delta \rightarrow \Omega\) and \(\hat{\kappa} \in \text{unsolved}(\Delta)\) and \(\Omega = \Omega_1[\hat{\kappa} : \kappa = t_1]\) and \(\Omega\) is canonical (Definition 3) and \(\Gamma \vdash t_2 : \kappa\) then \(\Delta \rightarrow \Omega_1[\hat{\kappa} : \kappa = t_2]\).

D.1 Reflexivity and Transitivity

Lemma 31 (Extension Reflexivity). \footnote{Go to proof}

If \(\Gamma\ context\) then \(\Gamma \rightarrow \Gamma\).

Lemma 32 (Extension Transitivity). \footnote{Go to proof}

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

D.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 33 (Suffix Weakening). \footnote{Go to proof}

If \(\Gamma \vdash t : \kappa\) then \(\Gamma, \Theta \vdash t : \kappa\).

Lemma 34 (Suffix Weakening). \footnote{Go to proof}

If \(\Gamma \vdash A\ type\) then \(\Gamma, \Theta \vdash A\ type\).

The following proposed lemma is false.

“Extension Weakening (Truth)”
If \(\Gamma \vdash P\ true \rightarrow \Delta\) and \(\Gamma \rightarrow \Gamma'\) then there exists \(\Delta'\) such that \(\Delta \rightarrow \Delta'\) and \(\Gamma' \vdash P\ true \rightarrow \Delta'\).

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

Lemma 35 (Extension Weakening (Sorts)). \footnote{Go to proof}

If \(\Gamma \vdash t : \kappa\) and \(\Gamma \rightarrow \Delta\) then \(\Delta \vdash t : \kappa\).

Lemma 36 (Extension Weakening (Props)). \footnote{Go to proof}

If \(\Gamma \vdash P\ prop\) and \(\Gamma \rightarrow \Delta\) then \(\Delta \vdash P\ prop\).

Lemma 37 (Extension Weakening (Types)). \footnote{Go to proof}

If \(\Gamma \vdash A\ type\) and \(\Gamma \rightarrow \Delta\) then \(\Delta \vdash A\ type\).

D.3 Principal Typing Properties

Lemma 38 (Principal Agreement). \footnote{Go to proof}

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

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

Lemma 39 (Right-Hand Subst. for Principal Typing). \footnote{Go to proof}

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

Lemma 40 (Extension Weakening for Principal Typing). \footnote{Go to proof}

If \(\Gamma \vdash A\ p\ type\) and \(\Gamma \rightarrow \Delta\) then \(\Delta \vdash A\ p\ type\).

Lemma 41 (Inversion of Principal Typing). \footnote{Go to proof}

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

(2) If \(\Gamma \vdash (P \Rightarrow A)\ p\ type\) 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 \(\Gamma \vdash A\ p\ type\).
D.4 Instantiation Extends

Lemma 42 (Instantiation Extension). Go to proof
If $\Gamma \vdash \Delta ::= \tau : \kappa \rightarrow \Delta$ then $\Gamma \rightarrow \Delta$.

D.5 Equivalence Extends

Lemma 43 (Elimeq Extension). Go to proof
If $\Gamma / s \triangleleft t : \kappa \rightarrow \Delta$ then there exists $\Theta$ such that $\Gamma, \Theta \rightarrow \Delta$.

Lemma 44 (Elimprop Extension). Go to proof
If $\Gamma / P \triangleleft \Delta$ then there exists $\Theta$ such that $\Gamma, \Theta \rightarrow \Delta$.

Lemma 45 (Checkeq Extension). Go to proof
If $\Gamma \vdash A \equiv B \triangleleft \Delta$ then $\Gamma \rightarrow \Delta$.

Lemma 46 (Checkprop Extension). Go to proof
If $\Gamma \vdash P \text{ true} \triangleleft \Delta$ then $\Gamma \rightarrow \Delta$.

Lemma 47 (Prop Equivalence Extension). Go to proof
If $\Gamma \vdash A \equiv B \triangleleft \Delta$ then $\Gamma \rightarrow \Delta$.

D.6 Subtyping Extends

Lemma 49 (Subtyping Extension). Go to proof
If $\Gamma \vdash A <: \tau \triangleleft \Delta$ then $\Gamma \rightarrow \Delta$.

D.7 Typing Extends

Lemma 50 (Typing Extension). 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 $\Gamma \vdash \Pi :: \vec{A} \leftarrow C \ p \rightarrow \Delta$
or $\Gamma / P \vdash \Pi :: \vec{A} \leftarrow C \ p \rightarrow \Delta$
then $\Gamma \rightarrow \Delta$.

D.8 Unfiled

Lemma 51 (Context Partitioning). Go to proof
If $\Delta, \Theta \rightarrow \Omega, \vec{A}, \Omega_2$ where $\Delta \rightarrow \Omega$ and $\Theta$ is soft, then $[\Omega, \Omega_2]\{(\Delta, \Theta) = [\Omega]\Delta\}$.

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

Lemma 53 (Completing Stability). Go to proof
If $\Gamma \rightarrow \Omega$ then $[\Omega]\Gamma = [\Omega]\Omega$.

Lemma 54 (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 55 (Confluence of Completeness). Go to proof
If $\Delta_1 \rightarrow \Omega$ and $\Delta_2 \rightarrow \Omega$ then $[\Omega]\Delta_1 = [\Omega]\Delta_2$.

Lemma 56 (Multiple Confluence). Go to proof
If $\Delta \rightarrow \Omega$ and $\Omega \rightarrow \Omega'$ and $\Delta' \rightarrow \Omega'$ then $[\Omega]\Delta = [\Omega']\Delta'$.
Lemma 57 (Bundled Substitution for Sorting). If \( \Gamma \vdash t : \kappa \) and \( \Gamma \rightarrow \Omega \) then \( [\Omega] \Gamma \vdash [\Omega] t : \kappa \).

Proof.

\[
\begin{align*}
\Gamma \vdash t : \kappa & \quad \text{Given} \\
\Omega \vdash t : \kappa & \quad \text{By Lemma 35 (Extension Weakening (Sorts))} \\
[\Omega] \Omega \vdash [\Omega] t : \kappa & \quad \text{By Lemma 13 (Substitution for Sorting)} \\
\Omega \rightarrow \Omega & \quad \text{By Lemma 31 (Extension Reflexivity)} \\
[\Omega] \Omega = [\Omega] \Gamma & \quad \text{By Lemma 55 (Confluence of Completeness)} \\
\iff [\Omega] \Gamma \vdash [\Omega] t : \kappa & \quad \text{By above equality}
\end{align*}
\]

Lemma 58 (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 \( \alpha : \kappa \rightarrow \tau \) 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 56 (Multiple Confluence) that \( [\Omega_{\text{canon}}] \Gamma = [\Omega] \Gamma \).

Lemma 59 (Split Solutions). Go to proof
If \( \Delta \rightarrow \Omega \) and \( \alpha \in \text{unsolved}(\Delta) \)
then there exists \( \Omega_1 = \Omega[\{\alpha : \kappa = t_1\}] \) such that \( \Omega_1 \rightarrow \Omega \) and \( \Omega_2 = \Omega[\{\alpha : \kappa = t_2\}] \) where \( \Delta \rightarrow \Omega_2 \) and \( t_2 \neq t_1 \) and \( \Omega_2 \) is canonical.

E Internal Properties of the Declarative System

Lemma 60 (Interpolating With and Exists). Go to proof
(1) If \( D : \Psi \vdash \Pi : \vec{A} \leftarrow C \ p \) and \( \Psi \vdash P_0 \) true
then \( D' : \Psi : \Pi : \vec{A} \leftarrow C \land P_0 \ p. \)

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

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

Lemma 61 (Case Invertibility). Go to proof
If \( \Psi \vdash \text{case}(e_0, \Pi) \leftarrow C \ p \)
then \( \Psi \vdash \pi_0 \Rightarrow A \ 1 \) 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.

F Miscellaneous Properties of the Algorithmic System

Lemma 62 (Well-Formed Outputs of Typing). Go to proof

(Spines) If \( \Gamma \vdash s : A \ q \gg C \ p \) \(-\Delta \) or \( \Gamma \vdash s : A \ q \gg [p] \) \(-\Delta \)
and \( \Gamma \vdash A \ q \) type
then \( \Delta \vdash C \ p \) type.

(Synthesis) If \( \Gamma \vdash e \Rightarrow A \ p \) \(-\Delta \)
then \( A \vdash p \) type.

G Decidability of Instantiation

Lemma 63 (Left Unresolvedness Preservation). Go to proof
If \( \Gamma_0, \delta, \Gamma_1 \vdash \delta : A : \kappa \rightarrow \Delta \) and \( \delta \in \text{unsolved}(\Gamma_0) \) then \( \delta \in \text{unsolved}(\Delta) \).
Lemma 65 (Instantiation Size Preservation). If $\Gamma_{0}, \delta : \kappa, \Gamma \vdash \alpha := t : \kappa \dashv \Delta$ and $\Gamma \vdash s : \kappa'$ and $\delta \notin \text{FV}(|\Gamma|)$ and $\beta \notin \text{FV}(|\Gamma|)$, then $\beta \notin \text{FV}(|\Delta|)$.

Lemma 66 (Decidability of Instantiation). If $\Gamma = \Gamma_{0}[\delta : \kappa']$ and $\Gamma \vdash t : \kappa$ such that $|\Gamma|t = t$ and $\delta \notin \text{FV}(t)$, then:

1. Either there exists $\Delta$ such that $\Gamma_{0}[\delta : \kappa'] \vdash \alpha := t : \kappa \dashv \Delta$, or not.

H Separation

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

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

Definition 5 (Separation-Preserving Extension).
The separated context $\Gamma_{1} \ast \Gamma_{R}$ extends to $\Delta_{L} \ast \Gamma_{R}$, written

$$(\Gamma_{L} \ast \Gamma_{R}) \dashv \ast (\Delta_{L} \ast \Delta_{R})$$

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 $\Gamma_{1}$, and $\Delta_{R}$ may add existential variables to $\Gamma_{R}$, but no variable from $\Gamma_{1}$ ends up in $\Delta_{R}$ and no variable from $\Gamma_{R}$ ends up in $\Delta_{L}$.

It is necessary to write $(\Gamma_{L} \ast \Gamma_{R}) \dashv \ast (\Delta_{L} \ast \Delta_{R})$ rather than $(\Gamma_{L} \ast \Gamma_{R}) \rightarrow (\Delta_{L} \ast \Delta_{R})$, because only $\dashv$ includes the domain conditions. For example, $(\delta \ast \beta) \rightarrow (\delta, \beta = \delta) \ast$, but the variable $\beta$ has “crossed over” to the left of $\ast$ in the context $(\delta, \beta = \delta) \ast$.

Lemma 67 (Transitivity of Separation). If $(\Gamma_{1} \ast \Gamma_{R}) \dashv \ast (\Theta_{1} \ast \Theta_{R})$ and $(\Theta_{L} \ast \Theta_{R}) \dashv \ast (\Delta_{L} \ast \Delta_{R})$ then $(\Gamma_{1} \ast \Gamma_{R}) \dashv \ast (\Delta_{L} \ast \Delta_{R})$.

Lemma 68 (Separation Truncation).
If $H$ has the form $\alpha : \kappa$ or $\ast \gamma$ or $\ast \rho$ and $(\Gamma_{L} \ast (\Gamma_{R}, H)) \dashv \ast (\Delta_{L} \ast \Delta_{R})$ then $(\Gamma_{L} \ast \Gamma_{R}) \dashv (\Delta_{L} \ast \Delta_{0})$ where $\Delta_{0} = (\Delta_{0}, H, \Theta)$.

Lemma 69 (Separation for Auxiliary Judgments).

(i) If $\Gamma_{L} \ast \Gamma_{R} \vdash \sigma : \kappa \dashv \Delta$
and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) \subseteq \text{dom}(\Gamma_{R})$
then $\Delta = (\Delta_{L} \ast \Delta_{R})$ and $(\Gamma_{L} \ast \Gamma_{R}) \dashv (\Delta_{L} \ast \Delta_{R})$.

(ii) If $\Gamma_{L} \ast \Gamma_{R} \vdash \text{P true} \dashv \Delta$
and $\text{FEV}(\text{P}) \subseteq \text{dom}(\Gamma_{R})$
then $\Delta = (\Delta_{L} \ast \Delta_{R})$ and $(\Gamma_{L} \ast \Gamma_{R}) \dashv (\Delta_{L} \ast \Delta_{R})$.

(iii) If $\Gamma_{L} \ast \Gamma_{R} \vdash \sigma \dashv \tau : \kappa \dashv \Delta$
and $\text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset$
then $\Delta = (\Delta_{L} \ast \Delta_{R})$ and $(\Gamma_{L} \ast \Gamma_{R}) \dashv (\Delta_{L} \ast \Delta_{R})$.

(iv) If $\Gamma_{L} \ast \Gamma_{R} \vdash \text{P} \dashv \Delta$
and $\text{FEV}(\text{P}) = \emptyset$
then $\Delta = (\Delta_{L} \ast \Delta_{R})$ and $(\Gamma_{L} \ast \Gamma_{R}) \dashv (\Delta_{L} \ast \Delta_{R})$.

(v) If $\Gamma_{L} \ast \Gamma_{R} \vdash \delta : \kappa \dashv \Delta$
and $\text{FEV}(\delta) \cup \{\delta\} \subseteq \text{dom}(\Gamma_{R})$
then $\Delta = (\Delta_{L} \ast \Delta_{R})$ and $(\Gamma_{L} \ast \Gamma_{R}) \dashv (\Delta_{L} \ast \Delta_{R})$. 

March 2, 2015
(vi) If $\Gamma_L \ast \Gamma_R \vdash P \equiv Q \to \Delta$
and $\text{FEV}(P) \cup \text{FEV}(Q) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$.

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

**Lemma 70 (Separation for Subtyping).** Go to proof
If $\Gamma_L \ast \Gamma_R \vdash A \equiv B \to \Delta$
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 $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$.

**Lemma 71 (Separation—Main).** Go to proof

- **(Spines)** If $\Gamma_L \ast \Gamma_R \vdash s : A \gg C q \to \Delta$
or $\Gamma_L \ast \Gamma_R \vdash s : A \gg C q \equiv \Delta$
and $\Gamma_L \ast \Gamma_R \vdash A$ p type
and $\text{FEV}(A) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$ and $\text{FEV}(C) \subseteq \text{dom}(\Delta_R)$.

- **(Checking)** If $\Gamma_L \ast \Gamma_R \vdash e \equiv C p \to \Delta$
and $\Gamma_L \ast \Gamma_R \vdash C$ p type
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$.

- **(Synthesis)** If $\Gamma_L \ast \Gamma_R \vdash \Pi :: \vec{A} \equiv C p \to \Delta$
and $\text{FEV}(\vec{A}) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$.

- **(Match)** If $\Gamma_L \ast \Gamma_R \vdash \Pi :: \vec{A} \equiv C p \to \Delta$
and $\text{FEV}(\vec{A}) = \emptyset$
and $\text{FEV}(C) \subseteq \text{dom}(\Gamma_R)$
then $\Delta = (\Delta_L \ast \Delta_R)$ and $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$.

- **(Match Elim.)** If $\Gamma_L \ast \Gamma_R \vdash P \equiv \Pi :: \vec{A} \equiv C p \to \Delta$
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 $(\Gamma_L \ast \Gamma_R) \to (\Delta_L \ast \Delta_R)$.

\section{Decidability of Algorithmic Subtyping}

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

\[
\forall \quad \exists \quad \land
\]

A type is large if 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 $\#\text{large}(A)$.

\subsection{I.1 Lemmas for Decidability of Subtyping}

**Lemma 72 (Substitution Isn’t Large).** Go to proof
For all contexts $\Theta$, we have $\#\text{large}(\Theta \vec{A}) = \#\text{large}(A)$.

**Lemma 73 (Instantiation Solves).** Go to proof
If $\Gamma \vdash \vec{A} := \tau : \kappa \to \Delta$ and $\|\tau\| \equiv \tau$ and $\vec{A} \notin \text{FV}(\|\Gamma\|\vec{\tau})$ then $\text{unsolved}(\Gamma) = |\text{unsolved}(\Gamma)| + 1$.

**Lemma 74 (Checkeq Solving).** Go to proof
If $\Gamma \vdash s \equiv t : \kappa \to \Delta$ then either $\Delta = \Gamma$ or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$. 

March 2, 2015
The following judgments are decidable, with $\Delta$ as output in (1)–(3), and $\bot$ as output in (4) and (5).

We assume $\sigma = [\Gamma] \sigma$ and $t = [\Gamma] t$ in (1) and (4). Similarly, in the other parts we assume $P = [\Gamma] P$ and (in part (3)) $Q = [\Gamma] Q$.

(1) $\Gamma \vdash \sigma \downarrow t : \kappa \dashv \Delta$
(2) $\Gamma \vdash P \text{ true} \dashv \Delta$
(3) $\Gamma \vdash P \equiv Q \dashv \Delta$
(4) $\Gamma / \sigma \downarrow t : \kappa \dashv \Delta \bot$
(5) $\Gamma / P \dashv \Delta \bot$

Given a context $\Gamma$ and types $A, B$ such that $\Gamma \vdash A \text{ type}$ and $\Gamma \vdash B$ type and $[\Gamma]A = A$ and $[\Gamma]B = B$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash A \equiv B \dashv \Delta$.

Theorem 1 (Decidability of Subtyping). Given a context $\Gamma$ and types $A, B$ such that $\Gamma \vdash A \text{ type}$ and $\Gamma \vdash B$ type and $[\Gamma]A = A$ and $[\Gamma]B = B$, it is decidable whether there exists $\Delta$ such that $\Gamma \vdash A <: \pm B \dashv \Delta$.

I.3 Decidability of Matching and Coverage

Lemma 79 (Decidability of Expansion Judgments). Given branches $\Pi$, it is decidable whether:

(1) there exists $\Pi'$ such that $\Pi \overset{\sim}{\rightarrow} \Pi'$;
(2) there exist $\Pi_L$ and $\Pi_R$ such that $\Pi \overset{\sim}{\rightarrow} \Pi_L \parallel \Pi_R$;
(3) there exists $\Pi'$ such that $\Pi \overset{\text{var}}{\rightarrow} \Pi'$;
(4) there exists $\Pi'$ such that $\Pi \overset{\sim}{\rightarrow} \Pi'$.

Theorem 2 (Decidability of Coverage). Given a context $\Gamma$, branches $\Pi$ and types $A$, it is decidable whether $\Gamma \vdash \Pi$ covers $A$ is derivable.

I.4 Decidability of Typing

Theorem 3 (Decidability of Typing). (i) Synthesis: Given a context $\Gamma$, a principality $p$, and a term $e$, it is decidable whether there exist a type $A$ and a context $\Delta$ such that $\Gamma \vdash e \Rightarrow A p \dashv \Delta$.

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

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

March 2, 2015
(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 \( \Gamma \vdash \Pi :: \vec{A} \equiv C \vdash p \rightarrow \Delta \).

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

### J Determinacy

**Lemma 80** (Determinacy of Auxiliary Judgments). Go to proof

1. **Elimineq:** Given \( \Gamma, \sigma, t, \kappa \) such that \( \text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset \) and \( D_1 : \Gamma / \sigma \equiv t : \kappa \rightarrow \Delta_1 \) and \( D_2 : \Gamma / \sigma \equiv t : \kappa \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
2. **Instantiation:** Given \( \Gamma, \vec{B}, t, \kappa \) such that \( \vec{B} \in \text{unsolved}(\Gamma) \) and \( \Gamma \vdash t : \kappa \) and \( \vec{B} \notin \text{FV}(t) \) and \( D_1 : \Gamma \vdash \vec{B} : \kappa \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash \vec{B} : \kappa \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
3. **Symmetric instantiation:** Given \( \Gamma, \vec{B}, \vec{\beta}, \kappa \) such that \( \vec{B}, \vec{\beta} \in \text{unsolved}(\Gamma) \) and \( \vec{B} \neq \vec{\beta} \) and \( D_1 : \Gamma \vdash \vec{B} : \kappa \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash \vec{\beta} : \kappa \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
4. **Checkeq:** Given \( \Gamma, \sigma, t, \kappa \) such that \( D_1 : \Gamma \vdash \sigma \equiv t : \kappa \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash \sigma \equiv t : \kappa \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
5. **Elimprop:** Given \( \Gamma, P \) such that \( D_1 : \Gamma \vdash P \equiv Q \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash P \equiv Q \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
6. **Checkprop:** Given \( \Gamma, P \) such that \( D_1 : \Gamma \vdash P \) true \( \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash P \) true \( \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

**Lemma 81** (Determinacy of Equivalence). Go to proof

1. **Propositional equivalence:** Given \( \Gamma, P, Q \) such that \( D_1 : \Gamma \vdash P \equiv Q \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash P \equiv Q \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
2. **Type equivalence:** Given \( \Gamma, A, B \) such that \( D_1 : \Gamma \vdash A \equiv B \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash A \equiv B \rightarrow \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

**Theorem 4** (Determinacy of Subtyping). Go to proof

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

**Theorem 5** (Determinacy of Typing). Go to proof

1. **Checking:** Given \( \Gamma, e, A, p \) such that \( D_1 : \Gamma \vdash e \leftarrow A \vdash \Delta_1 \) and \( D_2 : \Gamma \vdash e \leftarrow A \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
2. **Synthesis:** Given \( \Gamma, e \) such that \( D_1 : \Gamma \vdash e \Rightarrow B_1 p_1 \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash e \Rightarrow B_2 p_2 \rightarrow \Delta_2 \), it is the case that \( B_1 = B_2 \) and \( p_1 = p_2 \) and \( \Delta_1 = \Delta_2 \).
3. **Spine judgments:**
   - Given \( \Gamma, e, A, p \) such that \( D_1 : \Gamma \vdash e : A \vdash C_1 q_1 \rightarrow \Delta_1 \) and \( D_2 : \Gamma \vdash e : A \vdash C_2 q_2 \rightarrow \Delta_2 \), it is the case that \( C_1 = C_2 \) and \( q_1 = q_2 \) and \( \Delta_1 = \Delta_2 \).
   - The same applies for derivations of the principality-recovering judgments \( \Gamma \vdash e : A \vdash C_k [q_k] \rightarrow \Delta_k \).
4. **Match judgments:**
   - Given \( \Gamma, \Pi, \vec{A}, p, C \) such that \( D_1 : \Gamma \vdash \Pi : \vec{A} \leftarrow C \vdash \Delta_1 \) and \( D_2 : \Gamma \vdash \Pi : \vec{A} \leftarrow C \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
   - Given \( \Gamma, P, \Pi, \vec{A}, p, C \) such that \( D_1 : \Gamma / P \vdash \Pi : \vec{A} \leftarrow C \vdash \Delta_1 \) and \( D_2 : \Gamma / P \vdash \Pi : \vec{A} \leftarrow C \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
K Properties of Algorithmic Subtyping

L Soundness

L.1 Soundness of Instantiation

Lemma 82 (Soundness of Instantiation). If \( \Gamma \vdash \alpha : \tau \triangleleft \kappa \triangleleft \Delta \) and \( \alpha \not\in \text{FV}(\Gamma[\tau]) \) and \( \Gamma[\tau] = \tau \) and \( \Delta \rightarrow \Omega \) then \( \Omega[\alpha] = \Omega[\tau] \).

L.2 Soundness of Checkeq

Lemma 83 (Soundness of Checkeq). If \( \Gamma \vdash \sigma \triangleright t : \kappa \triangleleft \Delta \) where \( \Delta \rightarrow \Omega \) then \( \Omega[\sigma] = \Omega[t] \).

L.3 Soundness of Equivalence (Propositions and Types)

Lemma 84 (Soundness of Propositional Equivalence). If \( \Gamma \vdash P \equiv Q \triangleleft \kappa \triangleleft \Delta \) where \( \Delta \rightarrow \Omega \) then \( \Omega[P] = \Omega[Q] \).

Lemma 85 (Soundness of Algorithmic Equivalence). If \( \Gamma \vdash A \equiv B \triangleleft \Delta \) where \( \Delta \rightarrow \Omega \) then \( \Omega[A] = \Omega[B] \).

L.4 Soundness of Checkprop

Lemma 86 (Soundness of Checkprop). If \( \Gamma \vdash P \text{ true} \triangleleft \Delta \) and \( \Delta \rightarrow \Omega \) then \( \Psi \vdash \Omega[P \text{ true}] \).

L.5 Soundness of Eliminations (Equality and Proposition)

Lemma 87 (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 \( \Gamma / \sigma \triangleright t : \kappa \triangleleft \Delta \) then \( \Delta = (\Gamma,\Theta) \) where \( \Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n) \) and for all \( \Omega \) such that \( \Gamma \rightarrow \Omega \) and all \( t' \) such that \( \Omega[t'] : \kappa' \), it is the case that \( [\Omega,\Theta][t'] = [\emptyset][\Omega][t'] \), where \( \emptyset = \text{mgu}(\sigma,t) \).

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

L.6 Soundness of Subtyping

Theorem 6 (Soundness of Algorithmic Subtyping). If \( \Gamma[\tau] = A \) and \( \Gamma[t] = t \) and \( \Gamma \vdash A : \tau \) and \( \Gamma \vdash t : \kappa \) and \( \Delta \rightarrow \Omega \) and \( \Gamma \vdash A <;\Delta \rightarrow \tau \) and \( \Delta \rightarrow \Omega \) then \( \Omega[A] \triangleleft \Omega[\tau] \).

L.7 Soundness of Typing

Theorem 7 (Soundness of Match Coverage). If \( \Pi \) covers \( \vec{A} \) and \( \Gamma \rightarrow \Omega \) and \( \Gamma \vdash \vec{A} : \text{types} \) and \( \Gamma[\vec{A}] = \vec{A} \) then \( \Omega[\vec{A}] \triangleright \Pi \rightarrow \vec{A} \).

Lemma 88 (Well-formedness of Algorithmic Typing). Given \( \Gamma \text{ ctx} \):

(i) If \( \Gamma \vdash e \Rightarrow A : p \triangleleft \Delta \) then \( \Delta \vdash A : p \text{ type} \).

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

Definition 7 (Measure). Let measure \( \mathcal{M} \) on typing judgments be a lexicographic ordering:
1. first, the subject expression e, spine s, or matches \( \Pi \)—regarding all types in annotations as equal in size;

2. second, the partial order on judgment forms where an ordinary spine judgment is smaller than a principality-recovering spine judgment—and with all other judgment forms considered equal in size; and,

3. third, the derivation height.

\[
\left\{ \begin{array}{c}
\text{ordinary spine judgment} \\
\text{recovering spine judgment}
\end{array} \right\} \quad \text{ height}(\mathcal{D})
\]

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

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

Algorithmic rules (soundness cases):

- \( \text{Var} [1] [I] \alpha \) and \( \text{EmptySpine} \) have no premises.
- \( \text{Sub} \) \( (=, =, <) \)
- \( \text{Anno} \) \( (<, -, -) \)
- \( \forall [\neg \text{Spine}] \) \( (=, =, <) \)
- \( \exists [\text{Spine}] \) \( (=, =, <) \)
- \( \exists [\neg \text{Spine}] \) has only an auxiliary judgment, to which we need not apply the i.h., putting it in the same class as the rules with no premises.
- \( \rightarrow [\neg \text{Spine}] \) \( (=, =, <) \)
- \( \rightarrow [\text{Spine}] \) \( (=, =, <) \)
- \( \rightarrow [\text{SpineRec}] \) \( (=, =, <) \)
- \( \rightarrow [\text{SpinePass}] \) \( (=, =, <) \)
- \( ^\alpha \text{Spine} \) \( (=, =, <) \)
- \( \text{Case} \) \( (<, -, -) \)

Declarative rules (completeness cases):

- \( \text{DeclVar} \) \( \text{DeclI} [1] \) and \( \text{DeclEmptySpine} \) have no premises.
- \( \text{DeclSub} \) \( (=, =, <) \)
- \( \text{DeclAnno} \) \( (<, -, -) \)
- \( \text{DeclI} \) \( \text{DeclSpine}, \text{Decl} \neg \text{I} [1], \text{Decl} \neg \text{Spine} \) \( (=, =, <) \)
- \( \text{Decl} \rightarrow [\neg \text{I}] \) \( \text{Decl} \rightarrow [\text{E}] \) \( (<, -, -) \)
- \( \text{DeclSpineRec} \) \( (=, =, <) \)
- \( \text{DeclSpinePass} \) \( (=, =, <) \)
- \( \text{Decl} \rightarrow [\text{Spine}] \) \( \text{Decl} + [1], \text{Decl} \times [1], \text{DeclCase} \) \( (<, -, -) \)
Theorem 8 (Soundness of Algorithmic Typing). \(\text{Go to proof}\)
Given \(\Delta \rightarrow \Omega:\)

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

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

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

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

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

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

M Completeness

M.1 Completeness of Auxiliary Judgments

Lemma 89 (Completeness of Instantiation). \(\text{Go to proof}\)
Given \(\Gamma \rightarrow \Omega\) and \(\text{dom}(\Gamma) = \text{dom}(\Omega)\) and \(\Gamma \vdash \kappa \text{ and } \tau = [\Gamma]\tau\) and \(\kappa \in \text{unsolved}(\Gamma)\) and \(\kappa \notin \text{FV}(\tau)\):
If \([\Omega] \Delta = [\Omega]\tau\)
then there are \(\Delta, \Omega'\) such that \(\Omega \rightarrow \Omega'\) and \(\Delta \rightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Gamma \vdash \kappa := \tau : \kappa \rightarrow \Delta\).

Lemma 90 (Completeness of Checkeq). \(\text{Go to proof}\)
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 \(\Gamma \vdash [\Gamma]\sigma \triangleleft [\Gamma]\tau : \kappa \rightarrow \Delta\)
where \(\Delta \rightarrow \Omega'\) and \(\text{dom}(\Delta) = \text{dom}(\Omega')\) and \(\Omega \rightarrow \Omega'\).

Lemma 91 (Completeness of Elimeq). \(\text{Go to proof}\)
If \([\Gamma]\sigma = \tau\) 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 \(\Gamma / \sigma \triangleleft t : \kappa \rightarrow \Gamma\)
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 \(\Gamma / \sigma \triangleleft t : \kappa \rightarrow \perp\).

Lemma 92 (Substitution Upgrade). \(\text{Go to proof}\)
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:

(i) If \(\Gamma \vdash A \text{ type}\) then \([\Gamma, \Delta]A = \emptyset([\Gamma]A)\).

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

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

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

Lemma 93 (Completeness of Propequiv). \(\text{Go to proof}\)
Given \(\Gamma \rightarrow \Omega\)
and \(\Gamma \vdash P \text{ prop and } \Gamma \vdash Q \text{ prop}\)
and \([\Omega]P = [\Omega]Q\)
then \(\Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \vdash \Delta\)
where \(\Delta \rightarrow \Omega'\) and \(\Omega \rightarrow \Omega'\).
Lemma 94 (Completeness of Checkprop). \(\text{Go to proof}\)

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

M.2 Completeness of Equivalence and Subtyping

Lemma 95 (Completeness of Equiv). \(\text{Go to proof}\)

If \(\Gamma \rightarrow \Omega\) and \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type
and \(\Omega\Gamma \vdash [\Omega]A \rightarrow [\Omega]B\)
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). \(\text{Go to proof}\)

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

M.3 Completeness of Typing

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

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

Theorem 11 (Completeness of Algorithmic Typing). \(\text{Go to proof}\)

Given \(\Gamma \rightarrow \Omega\) such that \(\text{dom}(\Gamma) = \text{dom}(\Omega)\):

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

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

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

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

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

(vi) If \(\Gamma \vdash A !\) ! types and \(\Gamma \vdash P\) prop and \(\text{FEV}(P) = \emptyset\) and \(\Gamma \vdash C p\) type
and \([\Omega]\Gamma / \Omega]P \vdash [\Omega]A \leftarrow [\Omega]C p\)
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 $\Gamma / [\Gamma] P \vdash \Pi : [\Gamma] \vec{A} \leftrightarrow [\Gamma] C p' \rightarrow \Delta.$
Proofs

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

B’  Properties of the Declarative System

Lemma 1 (Declarative Weakening).

(i) If $\Psi, \alpha : \kappa, \Pi_1 \vdash t : \kappa$ then $\Psi, \Pi_1 \vdash [t/\alpha] t : \kappa$.

(ii) If $\Psi, \Pi_1 \vdash P \prop$ then $\Psi, \Pi_1 \vdash [t/\alpha] P \prop$.

(iii) If $\Psi, \Pi_1 \vdash P \true$ then $\Psi, \Pi_1 \vdash [t/\alpha] P \true$.

Proof. By induction on the derivation.

Lemma 2 (Declarative Term Substitution). Suppose $\Psi \vdash t : \kappa$. Then:

1. If $\Psi, \alpha : \kappa, \Pi_1 \vdash t' : \kappa$ then $\Psi, [t/\alpha] \Pi_1 \vdash [t/\alpha] t' : \kappa$.

2. If $\Psi, \alpha : \kappa, \Pi_1 \vdash P \prop$ then $\Psi, [t/\alpha] \Pi_1 \vdash [t/\alpha] P \prop$.

3. If $\Psi, \alpha : \kappa, \Pi_1 \vdash A \type$ then $\Psi, [t/\alpha] \Pi_1 \vdash [t/\alpha] A \type$.

4. If $\Psi, \alpha : \kappa, \Pi_1 \vdash A \leq^\pm B$ then $\Psi, [t/\alpha] \Pi_1 \vdash [t/\alpha] A \leq^\pm [t/\alpha] B$.

5. If $\Psi, \alpha : \kappa, \Pi_1 \vdash P \true$ then $\Psi, [t/\alpha] \Pi_1 \vdash [t/\alpha] P \true$.

Proof. By induction on the derivation of the substitutee.

Lemma 3 (Reflexivity of Declarative Subtyping).

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

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

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

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

- Case $A = \exists b : k. B, p = +$:
  $\Psi, b : k \vdash B \leq^+ B$ By i.h. (one less quantifier)
  $\Psi, b : k \vdash b : k$ By rule $U\text{varSort}$
  $\Psi, b : k \vdash \exists b : k. B$ By rule $\leq\exists R$
  $\Psi \vdash \exists b : k. B \leq^+ \exists b : k. B$ By rule $\leq\exists L$

- Case $A = \exists b : k. B, p = -$:
  $\Psi \vdash \exists b : k. B \leq^- \exists b : k. B$ By i.h. (polarities match)
  $\Psi \vdash \exists b : k. B \leq^- \exists b : k. B$ By $\leq^-\exists R$

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

- Case $A = \forall b : k. B, p = -$:
  $\Psi, b : k \vdash B \leq^- B$ By i.h. (one less quantifier)
  $\Psi, b : k \vdash b : k$ By rule $U\text{varSort}$
  $\Psi, b : k \vdash \forall b : k. B \leq^- B$ By rule $\leq^-\forall L$
  $\Psi \vdash \forall b : k. B \leq^- \forall b : k. B$ By rule $\leq^-\forall R$
Lemma 4 (Subtyping Inversion).

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

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

Lemma 5 (Subtyping Polarity Flip).

- If nonpos(A) and nonpos(B) and $\forall \alpha : \kappa. A \leq^+ B$ then $\forall \alpha \vdash A \leq^+ B$ by a derivation of the same or smaller size.
- If nonneg(A) and nonneg(B) and $\forall \alpha : \kappa. A \leq^- B$ then $\forall \alpha \vdash A \leq^- B$ by a derivation of the same or smaller size.
- If nonpos(A) and nonneg(B) and nonneg(B) and $\forall \alpha : \kappa. A \leq^+ B$ then $\forall \alpha \vdash A = B$.

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

Lemma 6 (Transitivity of Declarative Subtyping).

Given $\forall \beta : \kappa. A \leq^+ B$ and $\forall \beta : \kappa. C \leq^- B$.

(i) If $\forall \alpha : \kappa. A \leq^+_\kappa, \beta \leq^+ B$ and $\forall \beta : \kappa. B \leq^- C$ then $\forall \alpha : \kappa. A \leq^+_\kappa, \beta \leq^- C$.

(ii) If $\forall \alpha : \kappa. A \leq^-_{\kappa, \beta} B$ and $\forall \beta : \kappa. B \leq^- C$ then $\forall \alpha : \kappa. A \leq^-_{\kappa, \beta} C$.

**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** $\forall \tau : \kappa_1. \forall \alpha : \kappa_1. A' \leq^- B$  
    
    $\Psi \vdash \forall \alpha : \kappa_1. A' \leq^- B$  
    
    $\Psi \vdash \forall \tau : \kappa_1. A' \leq^- B$  
    
    $\Psi \vdash \tau : \kappa_1$  
    
    Subderivation  
    
    $\Psi \vdash \tau : \kappa_1$  
    
    Given  
    
    $\Psi \vdash [\tau/\alpha]A' \leq^- B$  
    
    $\Psi \vdash B \leq^- C$  
    
    Given  
    
    $\Psi \vdash [\tau/\alpha]A' \leq^- C$  
    
    By i.h. (A lost a quantifier)  
    
    $\Psi \vdash A \leq^- C$  
    
    By rule  

  - **Case** $\forall \beta : \kappa_2. A \leq^- B'$  
    
    $\Psi \vdash A \leq^- \forall \beta : \kappa_2. B'$  
    
    $\Psi \vdash A \leq^- \forall \beta : \kappa_2. B' \leq^- R$

  We case-analyze $D_2$:

  * **Case** $\forall \tau : \kappa_2. \forall \beta : \kappa_2, B' \leq^- C$  
    
    $\Psi \vdash \forall \beta : \kappa_2, B' \leq^- C$  
    
    $\Psi \vdash \tau : \kappa_2$  
    
    Subderivation  
    
    $\Psi \vdash [\tau/\beta]B' \leq^- C$  
    
    $\Psi \vdash A \leq^- [\tau/\beta]B'$  
    
    By Lemma 2 (Declarative Term Substitution) on $D_1$  
    
    $\Psi \vdash A \leq^- [\tau/\beta]B'$  
    
    By i.h. (B lost a quantifier)  

  * **Case** $\forall \kappa_3. \forall \kappa. C' \leq^- B$  
    
    $\Psi \vdash B \leq^- \forall \kappa : \kappa_3. C'$  
    
    $\Psi \vdash A \leq^- B$  
    
    Given  
    
    $\Psi \vdash A \leq^- B$  
    
    By Lemma 1 (Declarative Weakening)  
    
    $\Psi \vdash C \leq^- C'$  
    
    $\Psi \vdash C \leq^- C'$  
    
    Subderivation  
    
    $\Psi \vdash C \leq^- C'$  
    
    By i.h. (C lost a quantifier)  
    
    $\Psi \vdash B \leq^- \forall \kappa : \kappa_3. C'$  
    
    By $\leq^- R$
• Case \textit{nonpos}(B), polarity = +:
  Now we case-analyze \(D_1\):

  - \textbf{Case} \(\Psi, \alpha : \tau \vdash A' \leq^+ B\)
    \[\frac{}{\Psi \vdash \exists \alpha : \kappa_1, A' \leq^+ B}\]
    Subderivation
    \(\Psi, \alpha : \tau \vdash A' \leq^+ B\) By Lemma 1 (Declarative Weakening) \((D_2)\)
    \(\Psi, \alpha : \tau \vdash A' \leq^+ C\) By i.h. (A lost a quantifier)
    \[\frac{}{\Psi \vdash \exists \alpha : \kappa_1, A' \leq^+ C}\]
    \[\frac{}{\Psi \vdash \exists \alpha : \kappa_1, A' \leq^+ C}\]
  - \textbf{Case} \(\Psi \vdash A \leq^- B\)
    \[\frac{}{\Psi \vdash A \leq^- B}\]
    \(\Psi \vdash A \leq^- B\) Subderivation of \(D_1\)
    \(\Psi \vdash B \leq^- C\) Subderivation of \(D_2\)
    \(\Psi \vdash A \leq^- \exists c : \kappa_3, C'\) By i.h. (C lost a quantifier)
    \[\frac{}{\Psi \vdash A \leq^- \exists c : \kappa_3, C'}\]

• Case \(B = \exists \beta : \kappa_2, B'\), polarity = +:
  Now we case-analyze \(D_2\):

  - \textbf{Case} \(\Psi \vdash \tau : \kappa_3\)
    \[\frac{}{\Psi \vdash B \leq^+ \exists \alpha : \kappa_3, C'}\]
    \(\Psi \vdash B \leq^+ C\) Subderivation of \(D_2\)
    \(\Psi \vdash A \leq^+ B\) Given
    \(\Psi \vdash A \leq^+ \exists c : \kappa_3, C'\) By i.h. (C lost a quantifier)
    \[\frac{}{\Psi \vdash A \leq^+ \exists c : \kappa_3, C'}\]
    \[\frac{}{\Psi \vdash A \leq^+ \exists c : \kappa_3, C'}\]
  - \textbf{Case} \(\Psi, \beta : \kappa_2 \vdash B' \leq^+ C\)
    \[\frac{}{\Psi \vdash \exists \beta : \kappa_2, B' \leq^+ C}\]
    \[\frac{}{\Psi \vdash \exists \beta : \kappa_2, B' \leq^+ C}\]

Now we case-analyze \(D_1\):

  - \textbf{Case} \(\Psi \vdash \tau : \kappa_2\)
    \[\frac{}{\Psi \vdash A \leq^+ \exists \beta : \kappa_2, B'}\]
    \[\frac{}{\Psi \vdash A \leq^+ \exists \beta : \kappa_2, B'}\]
Proof of Lemma 6 (Transitivity of Declarative Subtyping)

\[ Ψ, β \vdash \kappa_2 \vdash B' \leq^+ C \]
\[ Ψ \vdash \tau : \kappa_1 \]
\[ Ψ \vdash A \leq^+ [τ/β]B' \]
\[ Ψ \vdash [τ/β]B' \leq^+ C \]
\[ Ψ \vdash A \leq^+ C \]

Subderivation of \(D_2\)

Subderivation of \(D_1\)

By Lemma 2 (Declarative Term Substitution)

By i.h. (B lost a quantifier)

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

\[ Ψ, \alpha : \kappa_1 \vdash B \leq^+ C \]
\[ Ψ, \alpha : \kappa_1 \vdash A' \leq^+ B \]
\[ Ψ, \alpha : \kappa_1 \vdash A' \leq^+ C \]
\[ Ψ \vdash \exists \alpha : \kappa_1 . A' \leq^+ C \]

By Lemma 1 (Declarative Weakening)

By i.h. (A lost a quantifier)

By \(\leq^+\)

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

\[ Ψ, \alpha : \kappa_1 \vdash B \leq^+ C \]
\[ Ψ, \alpha : \kappa_1 \vdash A \leq^+ B \]
\[ Ψ, \alpha : \kappa_1 \vdash A \leq^+ C \]
\[ Ψ \vdash \exists \alpha : \kappa_1 . A' \leq^+ C \]

By i.h. (C lost a quantifier)

By \(\leq^+\)

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

We case-analyze \(D_2\):

\[ \text{– Case } Ψ, \alpha : \kappa_1 \vdash A \leq^+ B \]
\[ Ψ \vdash \exists \alpha : \kappa_1 . A' \leq^+ B \]

\[ Ψ, \alpha : \kappa_1 \vdash A \leq^+ C \]
\[ Ψ, \alpha : \kappa_1 \vdash A' \leq^+ B \]
\[ Ψ, \alpha : \kappa_1 \vdash A' \leq^+ C \]
\[ Ψ \vdash \exists \alpha : \kappa_1 . A' \leq^+ C \]

By Lemma 1 (Declarative Weakening)

By i.h. (C lost a quantifier)

By \(\leq^+\)

\[ \text{– Case } Ψ \vdash B \leq^+ C \]
\[ \text{nonneg}(B) \quad \text{nonneg}(C) \]

We case-analyze \(D_1\):

\[ \text{– Case } Ψ \vdash B \leq^+ C \]
\[ Ψ \vdash B \leq^+ C \]

\[ Ψ, \alpha : \kappa_1 \vdash A \leq^+ B \]
\[ Ψ \vdash [\tau/\alpha]A' \leq^+ B \]

\[ Ψ \vdash \forall \alpha : \kappa_1 . A' \leq^+ B \]

\[ Ψ \vdash B \leq^+ C \]
\[ Ψ \vdash \exists \alpha : \kappa_1 . A' \leq^+ C \]

Given

Subderivation of \(D_1\)

By i.h. (A lost a quantifier)

By \(\leq^+\)

\[ \text{– Case } Ψ \vdash B \leq^+ C \]
\[ \text{nonneg}(A) \quad \text{nonneg}(B) \]

\[ Ψ \vdash A \leq^+ B \]
\[ Ψ \vdash [\tau/\alpha]A' \leq^+ B \]

\[ Ψ \vdash \forall \alpha : \kappa_1 . A' \leq^+ B \]

\[ Ψ \vdash A \leq^+ C \]
\[ Ψ \vdash [\tau/\alpha]A' \leq^+ C \]

\[ Ψ \vdash \forall \alpha : \kappa_1 . A' \leq^+ C \]

By i.h. (\(D_1\) and \(D_2\) smaller)

\[ \text{nonneg}(A) \quad \text{nonneg}(B) \]

\[ \text{Subderivation of } D_2 \]

\[ \text{Subderivation of } D_2 \]

\[ \text{Subderivation of } D_2 \]

\[ \text{Subderivation of } D_2 \]

\[ \text{Subderivation of } D_2 \]

\[ \text{Subderivation of } D_2 \]

\[ \text{Substitution and Well-formedness Properties} \]

Lemma 7 (Substitution—Well-formedness).
(i) If \( \Gamma \vdash A \ p \ \text{type} \) and \( \Gamma \vdash \tau \ p \ \text{type} \) then \( \Gamma \vdash [\tau/\alpha]A \ p \ \text{type} \).

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

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

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

Lemma 8 (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 24).

Lemma 9 (Sorting Implies Typing). If \( \Gamma \vdash t : \star \) then \( \Gamma \vdash t \ \text{type} \).

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

Lemma 10 (Right-Hand Substitution for Sorting). If \( \Gamma \vdash t : \kappa \) then \( \Gamma \vdash [\Gamma]t : \kappa \).

Proof. By induction on \( [\Gamma]t \) (the size of \( t \) under \( \Gamma \)).

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

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

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

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

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

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

  By Lemma 33 (Suffix Weakening), \( \Gamma \vdash \tau : \kappa \). Since \( [\Gamma]\tau < [\Gamma]\alpha \), we can apply the i.h., giving

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

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

- **Case** SolvedVarSort: In this case \( t = \hat{\alpha} \) and \( \Gamma = (\Gamma_L, \hat{\alpha} = \tau, \Gamma_R) \). Thus \( [\Gamma]t = [\Gamma]\hat{\alpha} = [\Gamma_1]\tau \).

  We assume contexts are well-formed, so all free variables in \( \tau \) are declared in \( \Gamma_L \). Consequently, \( [\Gamma_L]\tau = [\Gamma]\tau \), which is less than \( [\Gamma]\hat{\alpha} \). We can therefore apply the i.h. to \( \tau \), yielding \( \Gamma \vdash [\Gamma]\tau : \kappa \).

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

- **Case** BinSort: 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 BinSort, \( \Gamma \vdash ([\Gamma]t_1) \oplus ([\Gamma]t_2) : \kappa \), which by the definition of substitution is \( \Gamma \vdash [\Gamma](t_1 \oplus t_2) : \kappa \).

Lemma 11 (Right-Hand Substitution for Propositions). If \( \Gamma \vdash P \ \text{prop} \) then \( \Gamma \vdash [\Gamma]P \ \text{prop} \).

Proof. Use inversion (EqProp), apply Lemma 10 (Right-Hand Substitution for Sorting) to each premise, and apply EqProp again.

Lemma 12 (Right-Hand Substitution for Typing). If \( \Gamma \vdash A \ \text{type} \) then \( \Gamma \vdash [\Gamma]A \ \text{type} \).

Proof. By induction on \( [\Gamma]A \) (the size of \( A \) under \( \Gamma \)).

Several cases correspond to cases in the proof of Lemma 10 (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 9 (Sorting Implies Typing) to move from a sorting judgment to a typing judgment.
Proof of Lemma 12 \((\text{Right-Hand Substitution for Typing})\) \(\text{lem:substitution-tp}\)

- the case for \([\text{BinWF}]\) is like the case for \([\text{BinSort}]\).

Now, the new cases:

- **Case** \([\text{ForallWF}]\) : In this case \(A = \forall \alpha : \kappa. A_\alpha\). By i.h., \(\Gamma, \alpha : \kappa \vdash \Gamma \{ \alpha \} A_\alpha\) type. By the definition of substitution, \(\Gamma, \alpha : \kappa \vdash \Gamma \{ \alpha \} A_\alpha\) type, so by \([\text{ForallWF}]\) \(\Gamma \vdash \forall \alpha. \Gamma \{ \alpha \} A_\alpha\) type, which by the definition of substitution is \(\Gamma \vdash \Gamma \{ \Gamma \} (\forall \alpha. A_\alpha)\) type.

- **Case** \([\text{ExistsWF}]\) : Similar to the \([\text{ForallWF}]\) case.

- **Case** \([\text{ImpliesWF}]\), \([\text{WithWF}]\) : Use the i.h. and Lemma 11 \((\text{Right-Hand Substitution for Propositions})\), then apply \([\text{ImpliesWF}]\) or \([\text{WithWF}]\).

Lemma 13 \((\text{Substitution for Sorting})\). If \(\Omega \vdash t : \kappa\) then \([\Omega] \Omega \vdash [\Omega] t : \kappa\).

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

- **Case** \(u : \kappa \in \Omega \vdash u : \kappa\) \([\text{VarSort}]\)
  - **Subderivation** \(\Omega = (\Omega_1, a : \kappa = \tau, \Omega_2)\) Decomposing \(\Omega\)
  - \(\Omega_1 \vdash \tau : \kappa\) By implicit assumption that \(\Omega\) is well-formed
  - \(\Omega_1, a : \kappa = \tau, \Omega_2 \vdash \tau : \kappa\) By Lemma 33 \((\text{Suffix Weakening})\)
  - \([\Omega]_1 \vdash [\Omega] a : \kappa\) By Lemma 10 \((\text{Right-Hand Substitution for Sorting})\)
  - \([\Omega]_1 \vdash [\Omega] a : \kappa\) By implicit assumption that \(\Omega\) is well-formed

- **Case** \(\Omega \vdash 1 : \ast\) \([\text{UnitSort}]\)
  - Since \(1 = [\Omega] 1\), applying \([\text{UnitSort}]\) gives the result.

- **Case** \(\Omega \vdash \tau_1 : \ast\) \(\vdash \tau_2 : \ast\) \(\vdash \tau_1 \oplus \tau_2 : \ast\) \([\text{BinSort}]\)
  - By i.h. on each premise, rule \([\text{BinSort}]\) and the definition of substitution.

- **Case** \(\Omega \vdash \text{zero} : \text{N}\) \([\text{ZeroSort}]\)
  - Since \(\text{zero} = [\Omega] \text{zero}\), applying \([\text{ZeroSort}]\) gives the result.

- **Case** \(\Omega \vdash t : \text{N}\)
  - \(\Omega \vdash \text{succ}(t) : \text{N}\) \([\text{SuccSort}]\)
  - By i.h., rule \([\text{SuccSort}]\) and the definition of substitution.

Lemma 14 \((\text{Substitution for Prop Well-Formedness})\).
If \(\Omega \vdash P\) prop then \([\Omega]_1 \Omega \vdash [\Omega] P\) prop.
Proof of Lemma 14 (Substitution for Prop Well-Formedness)

Proof. Only one rule derives this judgment form:

- **Case** \(\Omega \vdash t : N\) \(\Omega \vdash t' : N\) \(\Omega \vdash t = t'\) prop
  \[\Omega \vdash t : N\] Subderivation
  \[\langle \Omega \rangle t : N\] By Lemma 13 (Substitution for Prop Well-Formedness)
  \[\Omega \vdash t' : N\] Subderivation
  \[\langle \Omega \rangle t' : N\] By Lemma 13 (Substitution for Prop Well-Formedness)
  \[\langle \Omega \rangle t = \langle \Omega \rangle t'\] prop By EqProp
  \[\langle \Omega \rangle \Omega \vdash (\langle \Omega \rangle t) = (\langle \Omega \rangle t')\] prop By EqProp
  \[\langle \Omega \rangle \Omega \vdash (\langle \Omega \rangle t = \langle \Omega \rangle t')\] prop By def. of subst.

Lemma 15 (Substitution for Type Well-Formedness). If \(\Omega \vdash A\) type then \(\langle \Omega \rangle \Omega \vdash [\langle \Omega \rangle] A\) type.

Proof. By induction on \(\mid \Omega \vdash A\mid\):

1. Several cases correspond to those in the proof of Lemma 13 (Substitution for Sorting):
   - the UnitWF case is like the UnitSort case (using DeclUnitWF instead of UnitSort);
   - the VarWF case is like the VarSort case (using DeclUvarWF instead of UvarSort);
   - the SolvedVarWF case is like the SolvedVarSort case.

2. However, uses of Lemma 10 (Right-Hand Substitution for Sorting) are replaced by uses of Lemma 12 (Right-Hand Substitution for Typing).

   Now, the new cases:

   - **Case** \(\Omega, \alpha : \kappa \vdash A_0\) type, \(\Omega \vdash \forall \alpha : \kappa. A_0\) type
     \(\Omega, \alpha : \kappa \vdash A_0 : \kappa'\) Subderivation
     \[\langle \Omega \rangle \langle \Omega \rangle, \alpha : \kappa \vdash [\langle \Omega \rangle] A_0 : \kappa'\] By i.h.
     \[\langle \Omega \rangle \langle \Omega \rangle, \alpha : \kappa \vdash \forall \alpha : \kappa. [\langle \Omega \rangle] A_0 : \kappa'\] By def. of completion
     \[\langle \Omega \rangle \langle \Omega \rangle \vdash \forall \alpha : \kappa. [\langle \Omega \rangle] A_0 : \kappa'\] By DeclAllWF
     \[\langle \Omega \rangle \langle \Omega \rangle \vdash [\langle \Omega \rangle \langle \Omega \rangle] \forall \alpha : \kappa. A_0 : \kappa'\] By def. of subst.

   - **Case** ExistsWF Similar to the ForallWF case, using DeclExistsWF instead of DeclAllWF

   - **Case** \(\Omega \vdash A_1\) type \(\Omega \vdash A_2\) type
     \(\Omega \vdash A_1 \oplus A_2\) type BinWF
     By i.h. on each premise, rule DeclBinWF, and the definition of substitution.

   - **Case** \(\Omega \vdash P\) prop \(\Omega \vdash A_0\) type
     \(\Omega \vdash P \supset A_0\) type ImpliesWF
     \[\langle \Omega \rangle \langle \Omega \rangle \vdash [\langle \Omega \rangle] P : [\langle \Omega \rangle] A_0\] prop By Lemma 14 (Substitution for Prop Well-Formedness)
     \[\langle \Omega \rangle \langle \Omega \rangle \vdash [\langle \Omega \rangle] P \supset [\langle \Omega \rangle] A_0\] type By i.h.
     \[\langle \Omega \rangle \langle \Omega \rangle \vdash [\langle \Omega \rangle] (P \supset A_0)\] type By DeclImpliesWF
     \[\langle \Omega \rangle \langle \Omega \rangle \vdash [\langle \Omega \rangle] (P \supset A_0)\] type By def. of subst.

   - **Case** \(\Omega \vdash P\) prop \(\Omega \vdash A_0\) type
     \(\Omega \vdash A_0 \land P\) type WithWF
     Similar to the ImpliesWF case.
Lemma 16 (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) $\Omega Z = (\Omega', \& : \kappa)$ or (3) $\Omega Z = (\Omega', \& : \kappa = t)$. 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 $\text{FV}(A) \cap \text{dom}(\Omega Z) = \emptyset$.

Lemma 17 (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.

D’ Properties of Extension

Lemma 18 (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 $\frac{}{\Gamma \rightarrow \Delta}$. This case is impossible, since by hypothesis $u$ is declared in $\Gamma$.

• Case $\frac{\Delta \vdash A}{\Gamma, x : A \rightarrow \Delta, x : A'}$. Immediate.

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

• Case $\frac{\Gamma \rightarrow \Delta}{\Gamma, \delta : \kappa \rightarrow \Delta, \delta : \kappa}$. Similar to the $\rightarrow \text{Var}$ case.

• Case $\frac{\Gamma \rightarrow \Delta}{\Gamma, \| \alpha = t \rightarrow \Delta, \| \alpha = t'}$. It is given that $u$ is declared in $\langle \Gamma, \alpha = t \rangle$. Since $\alpha = t$ is not a declaration, $u$ is declared in $\Gamma$. By i.h., $u$ is declared in $\Delta$, and therefore declared in $\langle \Delta, \alpha = t' \rangle$.

• Case $\frac{\Gamma \rightarrow \Delta}{\Gamma, \| \delta \rightarrow \Delta, \| \delta}$. Similar to the $\rightarrow \text{Eqn}$ case.
• Case \( \Gamma \rightarrow \Delta \)
\[
\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa = t \xrightarrow{\text{Solve}}
\]
Similar to the \texttt{-Var} case.

• Case \( \Gamma \rightarrow \Delta \)
\[
\Gamma \rightarrow \Delta, \alpha : \kappa \xrightarrow{\text{Add}}
\]
It is given that \( u \) is declared in \( \Gamma \). By i.h., \( u \) is declared in \( \Delta \), and therefore declared in \((\Delta, \alpha : \kappa)\).

• Case \( \Gamma \rightarrow \Delta \)
\[
\Gamma \rightarrow \Delta, \alpha : \kappa \xrightarrow{\text{AddSolved}}
\]
Similar to the \texttt{-Add} case. \( \square \)

\textbf{Lemma 19} (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 \( \Delta \).

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

• Case \( \rightarrow \xrightarrow{\text{-Id}} \)

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

• Case \( \Gamma \rightarrow \Delta \)
\[
[\Delta]A = [\Delta]A' \xrightarrow{\text{-Var}}
\]
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 18 (Declaration Preservation), \( u \) is declared in \( \Delta \).

Therefore \( u \) is declared to the left of \( v \) in \( (\Delta, x : A') \).

– Case \( v \neq x \):

Here, \( v \) is declared in \( \Gamma \). By i.h., \( u \) is declared to the left of \( v \) in \( \Delta \).

Therefore \( u \) is declared to the left of \( v \) in \( (\Delta, x : A') \).

• Case \( \Gamma \rightarrow \Delta \)
\[
\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \xrightarrow{\text{Uvar}}
\]
Similar to the \texttt{-Var} case.

• Case \( \Gamma \rightarrow \Delta \)
\[
\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \xrightarrow{\text{Unsolved}}
\]
Similar to the \texttt{-Var} case.

• Case \( \Gamma \rightarrow \Delta \)
\[
[\Delta]t = [\Delta]t' \xrightarrow{\text{Solved}}
\]
Similar to the \texttt{-Var} case.

• Case \( \Gamma \rightarrow \Delta \)
\[
\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' = t \xrightarrow{\text{Solve}}
\]
Similar to the \texttt{-Var} case.
Proof. \(\Gamma \rightarrow \Delta\) if \(\Delta = \{\alpha : \kappa, \Delta_1\}\). By i.h., \(\alpha\) must be declared in \(\Gamma\).

Therefore \(\alpha\) is declared to the left of \(\kappa\) in \(\Delta\).

\[\alpha = \tau'\]  

The equation \(\alpha = \tau'\) does not declare any variables, so \(\alpha\) and \(\kappa\) must be declared in \(\Gamma\).

Therefore \(\kappa\) is declared to the left of \(\alpha\) in \(\Delta\).

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

Similar to the \[\text{Eqn}\] case.

Lemma 20 (Reverse Declaration Order Preservation). If \(\Gamma \rightarrow \Delta\) and \(\alpha\) and \(\kappa\) are both declared in \(\Gamma\) and \(\alpha\) is declared to the left of \(\kappa\) in \(\Delta\), then \(\alpha\) is declared to the left of \(\kappa\) in \(\Gamma\).

Proof. It is given that \(\alpha\) and \(\kappa\) are declared in \(\Gamma\). Either \(\alpha\) is declared to the left of \(\kappa\) in \(\Gamma\), or \(\kappa\) is declared to the left of \(\alpha\). Suppose the latter (for a contradiction). By Lemma 19 (Declaration Order Preservation), \(\alpha\) is declared to the left of \(\kappa\) in \(\Delta\). But we know that \(\alpha\) is declared to the left of \(\kappa\) in \(\Delta\): contradiction. Therefore \(\alpha\) is declared to the left of \(\kappa\) in \(\Gamma\).

Lemma 21 (Extension Inversion).

(i) If \(\Delta = \{\alpha : \kappa, \Delta_1\}\) then there exist unique \(\Delta_0\) and \(\Delta_1\) such that \(\Delta = \{\Delta_0, \alpha : \kappa, \Delta_1\}\) and \(\Delta' :: \Gamma_0 \rightarrow \Delta_0\) where \(\Delta' < \Delta\).

Moreover, if \(\Gamma_1\) is soft, then \(\Delta_1\) is soft.

(ii) If \(\Delta = \{\alpha : \kappa, \Delta_1\}\) then there exist unique \(\Delta_0\) and \(\Delta_1\) such that \(\Delta = \{\Delta_0, \alpha : \kappa, \Delta_1\}\) and \(\Delta' :: \Gamma_0 \rightarrow \Delta_0\) where \(\Delta' < \Delta\).

Moreover, if \(\Gamma_1\) is soft, then \(\Delta_1\) is soft.

Moreover, if \(\text{dom}(\Gamma_0, \Gamma_1)\) then \(\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)\).

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

Moreover, if \(\Gamma_1\) is soft, then \(\Delta_1\) is soft.

Moreover, if \(\text{dom}(\Gamma_0, \Gamma_1)\) then \(\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)\).

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

Moreover, if \(\Gamma_1\) is soft, then \(\Delta_1\) is soft.

Moreover, if \(\text{dom}(\Gamma_0, \Gamma_1)\) then \(\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)\).

(v) If \(\Delta = \{\alpha : \kappa, \Delta_1\}\) then there exist unique \(\Delta_0\), \(\Lambda'\), and \(\Delta_1\) such that \(\Delta = \{\Delta_0, \alpha : \Lambda', \Delta_1\}\) and \(\Delta' :: \Gamma_0 \rightarrow \Delta_0\) and \(\Lambda' = \Lambda\) where \(\Delta' < \Delta\).

Moreover, if \(\Gamma_1\) is soft, then \(\Delta_1\) is soft.

Moreover, if \(\text{dom}(\Gamma_0, \Gamma_1)\) then \(\text{dom}(\Gamma_0) = \text{dom}(\Delta_0)\).

(vi) If \(\Delta = \{\alpha : \kappa, \Delta_1\}\) then either
• there exist unique $\Delta_0$, $\tau'$, and $\Delta_1$
  such that $\Delta = (\Delta_0, \hat{\kappa} : \kappa = \tau', \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ where $D' < D$, or
• there exist unique $\Delta_0$ and $\Delta_1$
  such that $\Delta = (\Delta_0, \hat{\kappa} : \kappa, \Delta_1)$ and $D' :: \Gamma_0 \rightarrow \Delta_0$ 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 $\text{Var}$ case is impossible.

Throughout this proof, we shadow $\Delta$ so that it refers to the largest proper prefix of the $\Delta$ in the statement of the lemma. For example, in the $\text{Var}$ case of part (i), we really have $\Delta = (\Delta_0, x : A')$, but we call $\Delta_{00}$ "$\Delta$".

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

• Case $\Gamma \rightarrow \Delta$

  $\Gamma \rightarrow \Delta$

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

  $\Gamma[\alpha : x, \Gamma_1]

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

  Given

  Since the last element must be equal

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

  By transitivity

  $\Gamma \rightarrow \Delta$

  Subderivation

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

  By equality

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

  By i.h.

  $\Gamma_0 \rightarrow \Delta_0$

  "$

  $\Gamma_1'$ soft then $\Delta_1$ soft

  "$

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

  By congruence

  $\Gamma_1, x : A$ soft then $\Delta_1, x : A'$ soft

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

• Case $\Gamma \rightarrow \Delta$

  $\Gamma \rightarrow \Delta$

  $\beta : \kappa'$

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

  There are two cases:

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

    $\Gamma \rightarrow \Delta$

    $\beta \kappa' \rightarrow \Delta$

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

    Given

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

    $\Gamma, \alpha : \kappa, \Gamma_1'$

    Since the last element must be equal

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

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

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

    By injectivity of syntax

    $\Gamma_1'$ soft then $\Delta_1$ soft

    "$

    (\Delta, \alpha : \kappa)$

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

    By i.h.

    $\Gamma_0 \rightarrow \Delta_0$

    "$

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

    By equality

    $\Gamma_0 \rightarrow \Delta_0$

    "$

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

    By congruence

    $\Gamma_1', \beta : \kappa'$ soft then $\Delta_1, \beta : \kappa'$ soft

    Since $\Gamma_1', \beta : \kappa'$ is not soft

  - Case $\alpha \neq \beta$:

    $\Gamma \rightarrow \Delta$

    $\Gamma \rightarrow \Delta$

    $\beta : \kappa' \rightarrow \Delta$

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

    Given

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

    $\Gamma, \alpha : \kappa, \Gamma_1'$

    Since the last element must be equal

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

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

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

    By injectivity of syntax

    $\Gamma_1'$ soft then $\Delta_1$ soft

    "$

    (\Delta, \beta : \kappa')$

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

    By equality

    $\Gamma_0 \rightarrow \Delta_0$

    "$

    (\Delta, \beta : \kappa')$

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

    By congruence

    $\Gamma_1', \beta : \kappa'$ soft then $\Delta_1, \beta : \kappa'$ soft

    Since $\Gamma_1', \beta : \kappa'$ is not soft
Proof of Lemma 21 (Extension Inversion) lem:extension-inversion

\[
\begin{align*}
\{\Gamma, \& : \kappa'\} &= (\Gamma_0, \alpha : \kappa, \Gamma_1) \\
&= (\Gamma_0, \alpha : \kappa, \Gamma'_1, \& : \kappa') \\
\Gamma &= (\Gamma_0, \alpha : \kappa, \Gamma'_1)
\end{align*}
\]
Given
Since the last element must be equal
By injectivity of syntax

\[
\begin{align*}
\Gamma &\longrightarrow \Delta \\
\Gamma_0, \alpha : \kappa, \Gamma'_1 &\longrightarrow \Delta \\
\Delta &= (\Delta_0, \alpha : \kappa, \Delta_1)
\end{align*}
\]
Subderivation
By equality
By i.h.

\[
\begin{align*}
\Gamma_0 &\longrightarrow \Delta_0 \\
\text{if } \Gamma'_1 \text{ soft then } \Delta_1 \text{ soft}
\end{align*}
\]

\[
\begin{align*}
\{\Delta, \& : \kappa'\} &= (\Delta_0, \alpha : \kappa, \Delta_1, \& : \kappa') \\
\text{if } \Gamma'_1, \& : \kappa' \text{ soft then } \Delta_1, \& : \kappa' \text{ soft}
\end{align*}
\]
By congruence

Case \( \Gamma \longrightarrow \Delta \)

\[
\begin{align*}
[\Delta] t &= [\Delta] t' \\
\Gamma, \alpha : \kappa, \Gamma'_1 &\longrightarrow \Delta, \alpha : \kappa = t' \\
\text{Solved}
\end{align*}
\]

\[
\begin{align*}
(\Gamma, \beta = t) &= (\Gamma_0, \alpha : \kappa, \Gamma_1) \\
&= (\Gamma_0, \alpha : \kappa, \Gamma'_1, \beta = t) \\
\Gamma &= (\Gamma_0, \alpha : \kappa, \Gamma'_1)
\end{align*}
\]
Given
Since the last element must be equal
By injectivity of syntax

\[
\begin{align*}
\Gamma &\longrightarrow \Delta \\
\Gamma_0, \alpha : \kappa, \Gamma'_1 &\longrightarrow \Delta \\
\Delta &= (\Delta_0, \alpha : \kappa, \Delta_1)
\end{align*}
\]
Subderivation
By equality
By i.h.

\[
\begin{align*}
\Gamma_0 &\longrightarrow \Delta_0 \\
\text{if } \Gamma'_1 \text{ soft then } \Delta_1 \text{ soft}
\end{align*}
\]

\[
\begin{align*}
(\Delta, \beta = t') &= (\Delta_0, \alpha : \kappa, \Delta_1, \beta = t') \\
\text{if } \Gamma'_1, \beta = t \text{ soft then } \Delta_1, \beta = t' \text{ soft}
\end{align*}
\]
By congruence
Since \( \Gamma'_1, \beta = t \) is not soft

Case \( \Gamma \longrightarrow \Delta \)

\[
\begin{align*}
\Gamma, \& : \kappa, \Gamma'_1 &\longrightarrow \Delta, \& \text{ Marker}
\end{align*}
\]

\[
\begin{align*}
(\Gamma, \& : \kappa) &= (\Gamma_0, \alpha : \kappa, \Gamma_1) \\
&= (\Gamma_0, \alpha : \kappa, \Gamma'_1, \& : \kappa') \\
\Gamma &= (\Gamma_0, \alpha : \kappa, \Gamma'_1)
\end{align*}
\]
Given
Since the last element must be equal
By injectivity of syntax

\[
\begin{align*}
\Gamma &\longrightarrow \Delta \\
\Gamma_0, \alpha : \kappa, \Gamma'_1 &\longrightarrow \Delta \\
\Delta &= (\Delta_0, \alpha : \kappa, \Delta_1)
\end{align*}
\]
Subderivation
By equality
By i.h.

\[
\begin{align*}
\Gamma_0 &\longrightarrow \Delta_0 \\
\text{if } \Gamma'_1 \text{ soft then } \Delta_1 \text{ soft}
\end{align*}
\]

\[
\begin{align*}
\Delta, \& : \kappa &= (\Delta_0, \alpha : \kappa, \Delta_1, \& : \kappa) \\
\text{if } \Gamma'_1, \& : \kappa \text{ soft then } \Delta_1, \& : \kappa \text{ soft}
\end{align*}
\]
By congruence
Since \( \Gamma'_1, \& : \kappa \) is not soft
Proof of Lemma 21 (Extension Inversion) lem:extension-inversion

• Case \[ \Gamma \rightarrow \Delta \]

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

\[ \Rightarrow \Delta, \hat{\alpha} : \kappa \quad \text{Add} \]

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

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

if \( \Gamma_1 \) soft then \( \Delta_1 \) soft

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

Suppose \( \Gamma_1 \) soft.

\( \Delta_1 \) soft

\( \Delta_1, \hat{\alpha} : \kappa' \) soft

By definition of softness

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

• Case \[ \Gamma \rightarrow \Delta \]

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

\[ \Rightarrow \Delta, \hat{\alpha} : \kappa' = t \quad \text{AddSolved} \]

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

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

if \( \Gamma_1 \) soft then \( \Delta_1 \) soft

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

Suppose \( \Gamma_1 \) soft.

\( \Delta_1 \) soft

\( (\Delta_1, \hat{\alpha} : \kappa' = t) \) soft

By definition of softness

\[ \Rightarrow \text{ if } \Gamma_1 \text{ soft then } \Delta_1, \hat{\alpha} : \kappa' = t \text{ soft} \quad \text{Implication intro} \]

• Case \[ \Gamma \rightarrow \Delta \]

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

\[ \Rightarrow (\Gamma, \hat{\beta} : \kappa' = t) \quad \text{Solve} \]

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

\[ = (\Gamma_0, \alpha : \kappa, \Gamma_1, \hat{\beta} : \kappa') \quad \text{Since the final elements are equal} \]

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

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

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

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

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

if \( \Gamma_1' \) soft then \( \Delta_1 \) soft

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

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

\( \Gamma_1' \) soft

\( \Delta_1 \) soft

\( \Gamma_1', \hat{\beta} : \kappa' = t \) soft

By definition of softness

\[ \Rightarrow \text{ if } \Gamma_1', \hat{\beta} : \kappa' \text{ soft then } \Delta_1, \hat{\beta} : \kappa' = t \text{ soft} \quad \text{Implication intro} \]

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

(iii) We have \( \Gamma_0, \alpha = \tau, \Gamma_1 \rightarrow \Delta \).
Proof of Lemma 21 (Extension Inversion)

• Case \( \Gamma \rightarrow \Delta \)
  \[ \Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa' \longrightarrow \text{Uvar} \]
  \[ \Gamma'_{\alpha = \tau, \Gamma_1} \]
  \[ (\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \beta : \kappa') \]
  \[ = (\Gamma_0, \alpha = \tau, \Gamma'_1, \beta : \kappa') \]
  \[ = (\Gamma_0, \alpha = \tau, \Gamma'_{1}, \beta : \kappa') \]
  \[ \Gamma = (\Gamma_0, \alpha = \tau, \Gamma'_{1}) \]

- Case \( \Delta = (\Delta_0, \alpha = \tau', \Delta_1) \)
  By i.h.

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

- Case \( \Gamma_0 \rightarrow \Delta_0 \)
  By i.h.

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

• Case \( \Gamma \rightarrow \Delta \)
  \[ |\Delta| \alpha = |\Delta| \alpha' \longrightarrow \text{Var} \]
  \[ |\Gamma| x : A \rightarrow \Delta, x : A' \]
  \[ |\Gamma_0, \alpha = \tau, \Gamma_1| \]
  Similar to the \( \rightarrow \text{Uvar} \) case.

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

• Case \( \Gamma \rightarrow \Delta \)
  \[ |\Delta| t = |\Delta| t' \longrightarrow \text{Solved} \]
  \[ |\Gamma_0, \alpha = \tau, \Gamma_1| t = t \rightarrow \Delta, \alpha : \kappa' = \tau \]
  Similar to the \( \rightarrow \text{Uvar} \) case.

• Case \( \Gamma \rightarrow \Delta \)
  \[ |\Delta| t = |\Delta| t' \longrightarrow \text{Solve} \]
  \[ |\Gamma_0, \alpha = \tau, \Gamma_1| t = t \rightarrow \Delta, \beta = t' \]
  Similar to the \( \rightarrow \text{Uvar} \) case.

• Case \( \Gamma \rightarrow \Delta \)
  \[ |\Delta| t = |\Delta| t' \longrightarrow \text{Eqn} \]
  \[ |\Gamma_0, \alpha = \tau, \Gamma_1| t = t \rightarrow \Delta, \beta = t' \]
  There are two cases:

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

  - Case \( \alpha \neq \beta \):
    \[ (\Gamma_0, \alpha = \tau, \Gamma_1) = (\Gamma, \beta = t) \]
    \[ = (\Gamma_0, \alpha = \tau, \Gamma'_1, \beta = t) \]
    \[ \Gamma = (\Gamma_0, \alpha = \tau, \Gamma'_1) \]
    Given
    \[ \Delta = (\Delta_0, \alpha = \tau', \Delta_1) \]
    Since the final elements must be equal
    By injectivity of context syntax
    \[ \Rightarrow [\Delta_0] \tau = [\Delta_0] \tau' \]
    By i.h.
    \[ \Rightarrow \Gamma_0 \rightarrow \Delta_0 \]
    By congruence of equality
    \[ \Rightarrow (\Delta, \beta = t') = (\Delta_0, \alpha = \tau', \Delta_1, \beta = t') \]
Proof of Lemma 21 (Extension Inversion)

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

Case $\Gamma \rightarrow \Delta$

\[ \Gamma \rightarrow \Delta, \hat{\alpha} : \kappa' \rightarrow \text{Add} \]

\[ \Gamma_2, \alpha = \tau, \Gamma_1 \]

$\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ By i.h.

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

\[ \Rightarrow \Gamma_0 \rightarrow \Delta_0 \]

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

By congruence of equality

Case $\Gamma \rightarrow \Delta$

\[ \Gamma \rightarrow \Delta, \hat{\alpha} : \kappa' = t \rightarrow \text{AddSolved} \]

\[ \Gamma_2, \alpha = \tau, \Gamma_1 \]

$\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ By i.h.

\[ [\Delta_0] \tau = [\Delta_0] \tau' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

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

By congruence of equality

Case $\Gamma \rightarrow \Delta$

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

\[ \Gamma_2, \alpha = \tau, \Gamma_1 \]

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

Given

Since the final elements must be equal

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

By injectivity of context syntax

\[ [\Delta_0] \tau = [\Delta_0] \tau' \]

\[ \Gamma_0 \rightarrow \Delta_0 \]

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

By congruence of equality

Case $\Gamma \rightarrow \Delta$

\[ \Gamma \rightarrow \Delta, \lambda : \kappa \rightarrow \text{Var} \]

Similar to the Uvar case.

Case $\Gamma \rightarrow \Delta$

\[ \Gamma \rightarrow \Delta, \beta \rightarrow \text{Marker} \]

Similar to the Uvar case.

Case $\Gamma \rightarrow \Delta$

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

Similar to the Uvar case.

Case $\Gamma \rightarrow \Delta$

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

Similar to the Uvar case.

There are two cases.

Case $\hat{\alpha} = \hat{\beta}$:

\[ \kappa' = \kappa \]

\[ t = \tau \]

\[ \Gamma_1 = \cdot \]

\[ \Gamma = \Gamma_0 \]

By injectivity of syntax

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

where $\tau' = t'$ and $\Delta_1 = \cdot$ and $\Delta = \Delta_0$

\[ \Gamma_0 \rightarrow \Delta \]

From subderivation $\Gamma \rightarrow \Delta$

\[ [\Delta_0] \tau = [\Delta_0] \tau' \]

From premise $[\Delta] t = [\Delta] t'$ and $x$
Proof of Lemma 21 (Extension Inversion) lem:extension-inversion

– Case \( \hat{\alpha} \not= \hat{\beta} \):

\[
\begin{align*}
(\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) &= (\Gamma_0, \hat{\beta} : \kappa' = t) \\
&= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \hat{\beta} : \kappa' = t) \\
&= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') \\
\Delta &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \\
\end{align*}
\]

Given

Since the final elements must be equal

By injectivity of context syntax

By i.h.

\[
[\Delta_0]_\tau = [\Delta_0]_\tau' \quad " \quad [\Delta_0]_\tau' = [\Delta_0]_\tau' \quad " \\
\Gamma_0 \rightarrow \Delta_0 \quad " \\
\Gamma_0 \rightarrow \Delta_0 \quad " \\
(\Delta, \hat{\beta} : \kappa' = t') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') \quad By \ congruence \ of \ equality
\]

• Case

\[
\begin{align*}
\Gamma &\rightarrow \Delta \\
[\Delta t = [\Delta]t'] &\rightarrow \Delta, \beta = t' \quad \text{Eqn}
\end{align*}
\]

\[
\begin{align*}
(\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) &= (\Gamma, \beta = t) \\
&= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \beta = t) \\
&= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') \\
\Delta &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \\
\end{align*}
\]

Given

Since the final elements must be equal

By injectivity of context syntax

By i.h.

\[
[\Delta_0]_\tau = [\Delta_0]_\tau' \quad " \\
\Gamma_0 \rightarrow \Delta_0 \quad " \\
(\Delta, \hat{\beta} = t') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} = t') \quad By \ congruence \ of \ equality
\]

• Case

\[
\begin{align*}
\Gamma &\rightarrow \Delta \\
\Delta, \beta : \kappa' &\rightarrow \Delta_1, \beta : \kappa' \quad \text{Add}
\end{align*}
\]

\[
\begin{align*}
\Delta &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \\
[\Delta_0]_\tau = [\Delta_0]_\tau' \quad " \\
\Gamma_0 \rightarrow \Delta_0 \quad " \\
(\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') \quad By \ congruence \ of \ equality
\end{align*}
\]

• Case

\[
\begin{align*}
\Gamma &\rightarrow \Delta \\
\Delta_1, \beta : \kappa' &\rightarrow t \quad \text{AddSolved}
\end{align*}
\]

\[
\begin{align*}
\Delta &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \\
[\Delta_0]_\tau = [\Delta_0]_\tau' \quad " \\
\Gamma_0 \rightarrow \Delta_0 \quad " \\
(\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t) \quad By \ congruence \ of \ equality
\end{align*}
\]

• Case

\[
\begin{align*}
\Gamma &\rightarrow \Delta \\
\Delta_1, \hat{\beta} : \kappa' = t &\rightarrow \Delta_1, \hat{\beta} : \kappa' = t \quad \text{Solve}
\end{align*}
\]

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

Given

Since the last elements must be equal

By injectivity of syntax

Subderivation

By equality

By i.h.

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

(v) We have \( \Gamma_0, \alpha : A, \Gamma_1 \rightarrow \Delta \). This proof is similar to the proof of part (i), except for the domain condition, which we handle similarly to part (ii).
Proof of **Lemma 21** *(Extension Inversion)*

**(vi)** We have $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \rightarrow \Delta$.

- **Case**

$$
\Gamma \rightarrow \Delta
\frac{\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'}{\Gamma, \hat{\alpha}, \kappa, \Gamma_1}
$$

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

*Given*

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

*Since the final elements must be equal*

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

*By injectivity of context syntax*

(By induction, there are two possibilities:)

- $\hat{\alpha}$ is not solved:

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

*By i.h.*

- $\hat{\alpha}$ is solved:

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

*By i.h.*

- **Case**

$$
\Gamma \rightarrow \Delta
\frac{[\Delta]A = [\Delta]A'}{\Gamma, \hat{\alpha}, \kappa, \Gamma_1}
$$

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

- **Case**

$$
\Gamma \rightarrow \Delta
\frac{\Gamma, \beta \rightarrow \Delta, \beta}{\Gamma, \beta \rightarrow \Delta, \beta}
$$

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

- **Case**

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

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

- **Case**

$$
\Gamma \rightarrow \Delta
\frac{\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'}{\Gamma, \alpha, \hat{\alpha}, \kappa, \Gamma_1}
$$

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

- **Case**

$$
\Gamma \rightarrow \Delta
\frac{\Gamma, \beta : \kappa' \rightarrow \Delta, \beta : \kappa'}{\Gamma, \hat{\alpha}, \kappa, \Gamma_1}
$$

*Case* $\hat{\alpha} \neq \hat{\beta}$:

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

*Given*

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

*Since the final elements must be equal*

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

*By injectivity of context syntax*

(By induction, there are two possibilities:)

- $\hat{\alpha}$ is not solved:

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

*By i.h.*

- $\hat{\alpha}$ is solved:

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

*By i.h.*
Proof of Lemma 21

Case $\alpha = \beta$:
\[
\kappa' = \kappa \text{ and } \Gamma_0 = \Gamma \text{ and } \Gamma_1 = 
\]
By injectivity of syntax
\[
\{ \Delta, \beta : \kappa' \} = \{ \Delta_0, \alpha : \kappa, \Delta_1 \}
\]
where $\Delta_0 = \Delta$ and $\Delta_1 = 
\]
\[
\Gamma_0 \rightarrow \Delta_0
\]
From premise $\Gamma \rightarrow \Delta$

Case $\Gamma \rightarrow \Delta$
\[
\Gamma_0, \alpha : \kappa, \Gamma_1
\]
\[
\rightarrow \Delta, \beta : \kappa' \rightarrow \text{Add}
\]
By induction, there are two possibilities:
- $\alpha$ is not solved:
\[
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)
\]
By i.h.
\[
\Gamma_0 \rightarrow \Delta_0
\]
[“]
\[
\{ \Delta, \beta : \kappa' \} = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa')
\]
By congruence of equality

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

Case $\Gamma \rightarrow \Delta$
\[
\Gamma_2, \alpha : \kappa, \Gamma_1
\]
\[
\rightarrow \Delta, \beta : \kappa' = t \rightarrow \text{AddSolved}
\]
By induction, there are two possibilities:
- $\alpha$ is not solved:
\[
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)
\]
By i.h.
\[
\Gamma_0 \rightarrow \Delta_0
\]
[“]
\[
\{ \Delta, \beta : \kappa' = t \} = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') = t
\]
By congruence of equality

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

Case $\alpha \neq \beta$:
\[
(\Gamma_0, \alpha : \kappa, \Gamma_1) = (\Gamma, \beta : \kappa')
\]
Given
\[
(\Gamma_0, \alpha : \kappa, \Gamma_1', \beta : \kappa')
\]
Since the final elements must be equal
\[
\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1')
\]
By injectivity of context syntax
By induction, there are two possibilities:
* $\alpha$ is not solved:
\[
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)
\]
By i.h.
\[
\Gamma_0 \rightarrow \Delta_0
\]
[“]
\[
\{ \Delta, \beta : \kappa' = t \} = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa' = t)
\]
By congruence of equality

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

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

March 2, 2015
Lemma 22 (Deep Evar Introduction). (i) If $\Gamma_0, \Gamma_1$ is well-formed and $\delta$ is not declared in $\Gamma_0, \Gamma_1$ then $\Gamma_0, \Gamma_1 \rightarrow \Gamma_0, \delta : \kappa, \Gamma_1$.

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

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

Proof.

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

- Case $\Gamma_1 = \cdot$:
  - $\Gamma_0 \text{ ctx}$
    - Given
  - $\delta \not\in \text{dom}(\Gamma_0)$
    - Given
  - $\Gamma_0, \delta : \kappa \text{ ctx}$
    - By rule VarCtx
  - $\Gamma_0 \rightarrow \Gamma_0$
    - By Lemma 31 (Extension Reflexivity)
  - $\Rightarrow \Gamma_0 \rightarrow \Gamma_0, \delta : \kappa$
    - By rule Add

- Case $\Gamma_1 = \Gamma_1', \alpha : \kappa'$:
  - $\Gamma_0, \Gamma_1', x : A \text{ ctx}$
    - Given
  - $\Gamma_0, \Gamma_1', x : A \text{ type}$
    - By inversion
  - $\delta \not\in \text{dom}(\Gamma_0, \Gamma_1', x : A)$
    - By inversion (1)
  - $\delta \neq x$
    - By inversion (2)
  - $\Gamma_0, \delta : \kappa, \Gamma_1', x : A$
    - By i.h.

- Case $\Gamma_1 = \Gamma_1', \beta : \kappa'$:
  - $\Gamma_0, \Gamma_1', \beta : \kappa' \text{ ctx}$
    - Given
  - $\Gamma_0, \Gamma_1', \beta : \kappa' \text{ type}$
    - By inversion
  - $\delta \not\in \text{dom}(\Gamma_0, \Gamma_1', \beta : \kappa')$
    - By inversion (1)
  - $\delta \neq \beta$
    - By inversion (2)
  - $\Gamma_0, \delta : \kappa, \Gamma_1', x : A$
    - By i.h.

- Case $\Gamma_1 = (\Gamma_1', \beta : \kappa' = t)$:
(ii) Assume $\Gamma_0, \alpha : \kappa, \Gamma_1$ ctx. We proceed by induction on $\Gamma_1$:

- Case $\Gamma_1 = \void$: 
  $\Gamma_0, t : \kappa$ Given
  $\Gamma_0, t : \kappa$ Given
  $\Gamma_0$ ctx
  Since $\Gamma_1 = \void$
  $\Rightarrow \Gamma_0, \alpha : \kappa \rightarrow \Gamma_0, \alpha : \kappa$ By rule —solve

- Case $\Gamma_1 = (\Gamma_0', \kappa : A)$:
  $\Gamma_0, t : \kappa$ Given
  $\Gamma_0, \alpha : \kappa, \Gamma_1, x : A$ ctx
  $\Gamma_0, \alpha : \kappa, \Gamma_1, x : A$ ctx
  $\Gamma_0, \alpha : \kappa, \Gamma_1$ ctx
  $\Gamma_0, \alpha : \kappa, \Gamma_1$ ctx
  $\Rightarrow \Gamma_0, \alpha : \kappa, \Gamma_1$ By rule —Var

Proof of Lemma 22 (Deep Evar Introduction) lem:deep-existential
Proof of Lemma 22 (Deep Evar Introduction)

(iii) Apply parts (i) and (ii) as lemmas, then Lemma 32 (Extension Transitivity).

Lemma 25 (Parallel Admissibility).
If $\Gamma_L \rightarrow \Delta_L$ and $\Gamma_R \rightarrow \Delta_R, \Delta_L$ then:

(i) $\Gamma_L, \alpha : \kappa, \Gamma_R \rightarrow \Delta_L, \alpha : \kappa, \Delta_R$
Proof. By induction on \( \Delta_R \). As always, we assume that all contexts mentioned in the statement of the lemma are well-formed. Hence, \( \delta \notin \text{dom}(\Gamma_1) \cup \text{dom}(\Gamma_R) \cup \text{dom}(\Delta_L) \cup \text{dom}(\Delta_R) \).

(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 \( \Gamma_1, \Gamma_R \to \Delta_L, \Delta_R \), the context \( \Delta_R \) becomes smaller.

The only tricky part of the proof is that to apply the i.h., we need \( \Gamma_1, \Gamma_R \to \Delta_L \). To avoid this, we need to reason by contradiction, using Lemma 18 (Declaration Preservation).

- Case \( \Delta_R = \cdot \): We have \( \Gamma_1 \to \Delta_L \). Applying \(--\text{Unsolved}--\) to that derivation gives the result.

- Case \( \Delta_R = (\Delta_R', \hat{\beta}: \kappa = \tau) \): We have \( \hat{\beta} \neq \hat{\alpha} \) by the well-formedness assumption.

  The concluding rule of \( \Gamma_1, \Gamma_R \to \Delta_L, \Delta_R' \) must have been \(--\text{Unsolved}--\) or \(--\text{Add}--\). In both cases, the result follows by i.h. and applying \(--\text{Unsolved}--\) or \(--\text{Add}--\).

  Note: In \(--\text{Add}--\), the left-hand context doesn’t change, so we clearly maintain \( \Gamma_1 \to \Delta_L \). In \(--\text{Unsolved}--\) we can correctly apply the i.h. because \( \Gamma_R \neq \cdot \). Suppose, for a contradiction, that \( \Gamma_R = \cdot \). Then \( \Gamma_1 = (\Gamma_1', \hat{\beta}) \). It was given that \( \Gamma_1 \to \Delta_L \); that is, \( \Gamma_1', \hat{\beta} \to \Delta_L \). By Lemma 18 (Declaration Preservation), \( \Delta_L \) has a declaration of \( \hat{\beta} \). But then \( \Delta = (\Delta_L, \Delta_R', \hat{\beta}) \) is not well-formed: contradiction. Therefore \( \Gamma_R \neq \cdot \).

- Case \( \Delta_R = (\Delta_R', \hat{\alpha}: \kappa = \tau) \): We have \( \hat{\beta} \neq \hat{\alpha} \) by the well-formedness assumption.

  The concluding rule must have been \(--\text{Solved}--\) or \(--\text{AddSolved}--\). In each case, apply the i.h. and then the corresponding rule. (In \(--\text{Solved}--\) and \(--\text{AddSolved}--\) use Lemma 18 (Declaration Preservation) to show \( \Gamma_R \neq \cdot \).)

- Case \( \Delta_R = (\Delta_R', \alpha) \): The concluding rule must have been \(--\text{Uvar}--\). The result follows by i.h. and applying \(--\text{Uvar}--\).

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

- Case \( \Delta_R = (\Delta_R', x: A) \): Similar to the previous case, with rule \(--\text{Marker}--\).

- Case \( \Delta_R = (\Delta_R', \cdot) \): Similar to the previous case, with rule \(--\text{Var}--\).

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

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

Lemma 26 (Parallel Extension Solution).

If \( \Gamma_1, \hat{\alpha}: \kappa, \Gamma_R \to \Delta_L, \hat{\alpha}: \kappa = \tau', \Delta_R \) and \( \Gamma_1 \vdash \tau: \kappa \) and \( [\Delta_L] \tau = [\Delta_L] \tau' \) then \( \Gamma_1, \hat{\alpha}: \kappa = \tau, \Gamma_R \to \Delta_L, \hat{\alpha}: \kappa = \tau', \Delta_R \).

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

In the case where \( \Delta_R = \cdot \), we know that rule \(--\text{Solve}--\) must have concluded the derivation (we can use Lemma 18 (Declaration Preservation) to get a contradiction that rules out \(--\text{AddSolved}--\)); then we have a subderivation \( \Gamma_1 \to \Delta_L \), to which we can apply \(--\text{Solved}--\). \(\blacksquare\)

Lemma 27 (Parallel Variable Update).

If \( \Gamma_1, \hat{\alpha}: \kappa, \Gamma_R \to \Delta_L, \hat{\alpha}: \kappa = \tau_0, \Delta_R \) and \( \Gamma_1 \vdash \tau_1: \kappa \) and \( [\Delta_L] \tau_0 = [\Delta_L] \tau_1 = [\Delta_L] \tau_2 \) then \( \Gamma_1, \hat{\alpha}: \kappa = \tau_1, \Gamma_R \to \Delta_L, \hat{\alpha}: \kappa = \tau_2, \Delta_R \).

Proof. By induction on \( \Delta_R \). Similar to the proof of Lemma 26 (Parallel Extension Solution), but applying \(--\text{Solved}--\) at the end. \(\blacksquare\)

Lemma 28 (Substitution Monotonicity).
Lemma 28 (Substitution Monotonicity)
Proof of Lemma 28 (Substitution Monotonicity) lem:substitution-monotonicity

- Case
  \[ \Gamma \vdash \text{zero} : \text{ZeroSort} \]
  \[ [\Delta] \text{zero} = \text{zero} = [\Delta][\Gamma] \text{zero} \]
  Since \( \text{FV}(\text{zero}) = \emptyset \)

- Case
  \[ \Gamma \vdash \text{succ} (t) : \text{SuccSort} \]
  \[ [\Delta][\Gamma] t = [\Delta] t \]
  By i.h.
  \[ \text{succ} ([\Delta][\Gamma] t) = \text{succ} ([\Delta] t) \]
  By congruence of equality
  \[ [\Delta][\Gamma] \text{succ} (t) = [\Delta] \text{succ} (t) \]
  By definition of substitution

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

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

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

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

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

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

- Case
  \[ \Gamma \vdash A_1 \text{ type} \quad \Gamma \vdash A_2 \text{ type} \]
  \[ \Gamma \vdash A_1 \oplus A_2 \text{ type} \text{ BinWF} \]
  \[ [\Delta][\Gamma] A_1 = [\Delta] A_1 \]
  By i.h.
  \[ [\Delta][\Gamma] A_2 = [\Delta] A_2 \]
  By i.h.
  \[ [\Delta][\Gamma] A_1 \oplus [\Delta][\Gamma] A_2 = [\Delta] A_1 \oplus [\Delta] A_2 \]
  By congruence of equality
  \[ [\Delta][\Gamma] (A_1 \oplus A_2) = [\Delta] (A_1 \oplus A_2) \]
  Definition of substitution

- Case
  \[ \Gamma, \alpha : \kappa \vdash A_0 \text{ type} \]
  \[ \Gamma \vdash \forall \alpha : \kappa. A_0 \text{ ForallWF} \]
  \[ \Gamma \rightarrow \Delta \]
  Given
  \[ \Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa \]
  By rule \( \rightarrow Uvar \)
  \[ [\Delta, \alpha : \kappa] [\Gamma, \alpha : \kappa] A_0 = [\Delta, \alpha : \kappa] A_0 \]
  By i.h.
  \[ [\Delta] [\Gamma] A_0 = [\Delta] A_0 \]
  By definition of substitution
  \[ \forall \alpha : \kappa. [\Delta][\Gamma] A_0 = \forall \alpha : \kappa. [\Delta] A_0 \]
  By congruence of equality
  \[ [\Delta][\Gamma] (\forall \alpha : \kappa. A_0) = [\Delta] (\forall \alpha : \kappa. A_0) \]
  By definition of substitution
Proof of Lemma 28 (Substitution Monotonicity).

(i) If $\Gamma \mapsto \Delta$ and $\Gamma \vdash t : \kappa$ and FEV($\Gamma[t]$) = $\emptyset$ then $[\Delta]\Gamma t = [\Gamma]t$.

(ii) If $\Gamma \mapsto \Delta$ and $\Gamma \vdash P$ prop and FEV($\Gamma[P]$) = $\emptyset$ then $[\Delta]\Gamma P = [\Gamma]P$.

(iii) If $\Gamma \mapsto \Delta$ and $\Gamma \vdash A$ type and FEV($\Gamma[A]$) = $\emptyset$ then $[\Delta]\Gamma A = [\Gamma]A$.

Proof. Each part is a separate induction, relying on the proofs of the earlier parts. In each part, the result follows by an induction on the derivation of $\Gamma \mapsto \Delta$.

The main observation is that $\Delta$ adds no equations for any variable of $t$, $P$, and $A$ that $\Gamma$ does not already contain, and as a result applying $\Delta$ as a substitution to $\Gamma[t]$ does nothing.

Lemma 29 (Substitution Invariance).

(i) If $\Gamma \mapsto \Delta$ and $\Gamma \vdash t : \kappa$ and FEV($\Gamma[t]$) = $\emptyset$ then $[\Delta]\Gamma t = [\Gamma]t$.

(ii) If $\Gamma \mapsto \Delta$ and $\Gamma \vdash P$ prop and FEV($\Gamma[P]$) = $\emptyset$ then $[\Delta]\Gamma P = [\Gamma]P$.

(iii) If $\Gamma \mapsto \Delta$ and $\Gamma \vdash A$ type and FEV($\Gamma[A]$) = $\emptyset$ then $[\Delta]\Gamma A = [\Gamma]A$.

Proof. Each part is a separate induction, relying on the proofs of the earlier parts. In each part, the result follows by an induction on the derivation of $\Gamma \mapsto \Delta$.

The main observation is that $\Delta$ adds no equations for any variable of $t$, $P$, and $A$ that $\Gamma$ does not already contain, and as a result applying $\Delta$ as a substitution to $\Gamma[t]$ does nothing.

Lemma 23 (Soft Extension).

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

Proof. By induction on $\Theta$.

- Case $\Theta = :$ We have $\Gamma \mapsto \Delta$. Let $\Omega = :$. Then $\Gamma, \Theta \mapsto \Delta, \Omega$.

- Case $\Theta = (\Theta', \& : \kappa = t)$:
  
  $\Gamma, \Theta' \mapsto \Gamma, \Omega'$

  If $\kappa = s$, let $t = 1$; if $\kappa = N$, let $t = \text{zero}$.

  $\Gamma, \Theta' \mapsto \Gamma, \Omega'$

  By i.h. $\text{Solved}$

- Case $\Theta = (\Theta', \& : \kappa)$:
  
  If $\kappa = s$, let $t = 1$; if $\kappa = N$, let $t = \text{zero}$.

  $\Gamma, \Theta' \mapsto \Gamma, \Omega'$

  By i.h. $\text{Solved}$

Lemma 30 (Split Extension).

If $\Delta \mapsto \Omega$

and $\& \in \text{unsolved}(\Delta)$

and $\Omega = \Omega_1[\& : \kappa = t_1]$ and $\Omega$ is canonical (Definition 3)

and $\Omega \vdash t_2 : \kappa$

then $\Delta \mapsto \Omega_1[\& : \kappa = t_2]$.

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

Proof of Lemma 31 (Extension Reflexivity)
D’.1 Reflexivity and Transitivity

**Lemma 31** (Extension Reflexivity).
If $\Gamma \ ctx$ then $\Gamma \rightarrow \Gamma$.

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

- **Case**
  \[ \begin{array}{l}
  \hline
  \text{EmptyCtx} \\
  \hline
  \Gamma \ ctx \\
  \end{array} \]
  \[ \begin{array}{l}
  \rightarrow \cdot \\
  \text{By rule } \rightarrow \text{Id} \\
  \end{array} \]

- **Case**
  \[ \begin{array}{l}
  \hline
  \text{HypCtx} \\
  \hline
  \Gamma \ ctx \quad x \notin \text{dom}(\Gamma) \quad \Gamma \vdash \text{A type} \\
  \hline
  \Gamma, x : \text{A ctx} \\
  \end{array} \]
  \[ \begin{array}{l}
  \Gamma \rightarrow \Gamma \\
  \text{By i.h.} \\
  [\Gamma]A = [\Gamma]A \\
  \Gamma, x : \text{A} \rightarrow \Gamma, x : \text{A} \\
  \text{By rule } \rightarrow \text{Var} \\
  \end{array} \]

- **Case**
  \[ \begin{array}{l}
  \hline
  \text{VarCtx} \\
  \hline
  \Gamma \ ctx \quad u : k \notin \text{dom}(\Gamma) \\
  \hline
  \Gamma, u : k \ ctx \\
  \end{array} \]
  \[ \begin{array}{l}
  \Gamma \rightarrow \Gamma \\
  \text{By i.h.} \\
  \Gamma, u : k \rightarrow \Gamma, u : k \\
  \text{By rule } \rightarrow \text{Uvar} \text{ or } \rightarrow \text{Unsolved} \\
  \end{array} \]

- **Case**
  \[ \begin{array}{l}
  \hline
  \text{SolvedCtx} \\
  \hline
  \Gamma \ ctx \quad \& \notin \text{dom}(\Gamma) \quad \Gamma \vdash t : k \\
  \hline
  \Gamma, \& : k = t \ ctx \\
  \end{array} \]
  \[ \begin{array}{l}
  \Gamma \rightarrow \Gamma \\
  \text{By i.h.} \\
  [\Gamma]t = [\Gamma]t \\
  \Gamma, \& : k = t \rightarrow \Gamma, \& : k = t \\
  \text{By rule } \rightarrow \text{Solved} \\
  \end{array} \]

- **Case**
  \[ \begin{array}{l}
  \hline
  \text{EqnVarCtx} \\
  \hline
  \Gamma \ ctx \quad \alpha : k \in \Gamma \quad (\alpha = -) \notin \Gamma \quad \Gamma \vdash \tau : k \\
  \hline
  \Gamma, \alpha = \tau \ ctx \\
  \end{array} \]
  \[ \begin{array}{l}
  \Gamma \rightarrow \Gamma \\
  \text{By i.h.} \\
  [\Gamma]t = [\Gamma]t \\
  \Gamma, \alpha = t \rightarrow \Gamma, \alpha = t \\
  \text{By rule } \rightarrow \text{Eqn} \\
  \end{array} \]

- **Case**
  \[ \begin{array}{l}
  \hline
  \text{MarkerCtx} \\
  \hline
  \Gamma \ ctx \quad \triangleright u \notin \Gamma \\
  \hline
  \Gamma, \triangleright u \ ctx \\
  \end{array} \]
  \[ \begin{array}{l}
  \Gamma \rightarrow \Gamma \\
  \text{By i.h.} \\
  \Gamma, \triangleright u \rightarrow \Gamma, \triangleright u \\
  \text{By rule } \rightarrow \text{Marker} \\
  \end{array} \]

**Lemma 32** (Extension Transitivity).
If $D : \Gamma \rightarrow \Theta$ and $D' : \Theta \rightarrow \Delta$ then $\Gamma \rightarrow \Delta$.

*Proof.* By induction on $D'$.

- **Case**
  \[ \begin{array}{l}
  \hline
  \hline
  \rightarrow \text{Id} \\
  \end{array} \]
  \[ \begin{array}{l}
  \rightarrow \cdot \\
  \text{By inversion on } \mathcal{D} \\
  \end{array} \]
  \[ \begin{array}{l}
  \rightarrow \text{Id} \\
  \text{By rule } \rightarrow \text{Id} \\
  \end{array} \]
  \[ \begin{array}{l}
  \Gamma \rightarrow \Delta \\
  \text{Since } \Gamma = \Delta = \cdot \\
  \end{array} \]
Proof of Lemma 32 (Extension Transitivity)

Case \( \Theta' \rightarrow \Delta' \)

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

\[
\Gamma = (\Gamma', x : A'') \quad \text{By inversion on } D
\]

\[
[\Theta]A'' = [\Theta]A \quad \text{By inversion on } D
\]

\[
\Gamma' \rightarrow \Theta' \quad \text{By inversion on } D
\]

\[
\Gamma' \rightarrow \Delta' \quad \text{By i.h.}
\]

\[
[\Delta']\Theta'A'' = [\Delta']\Theta'A \quad \text{By congruence of equality}
\]

\[
[\Delta']A'' = [\Delta']A \quad \text{By Lemma 28 (Substitution Monotonicity)}
\]

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

Case \( \Theta' \rightarrow \Delta' \)

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

\[
\Gamma = (\Gamma', \alpha : \kappa) \quad \text{By inversion on } D
\]

\[
\Gamma' \rightarrow \Theta' \quad \text{By inversion on } D
\]

\[
\Gamma' \rightarrow \Delta' \quad \text{By i.h.}
\]

\[
\Gamma', \alpha : \kappa \rightarrow \Delta', \alpha : \kappa \quad \text{By } \text{Uvar}
\]

Case \( \Theta' \rightarrow \Delta' \)

\[
\Theta', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa \quad \text{Unsolved}
\]

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

- Case \( \Gamma' \rightarrow \Theta' \)

\[
\Gamma', \hat{\alpha} : \kappa \rightarrow \Theta', \hat{\alpha} : \kappa \quad \text{Unsolved}
\]

\[
\Gamma' \rightarrow \Delta' \quad \text{By i.h.}
\]

\[
\Gamma', \hat{\alpha} : \kappa \rightarrow \Delta', \hat{\alpha} : \kappa \quad \text{By rule } \text{Add}
\]

- Case \( \Gamma \rightarrow \Theta' \)

\[
\Gamma \rightarrow \Theta', \hat{\alpha} : \kappa \quad \text{Add}
\]

\[
\Gamma \rightarrow \Delta' \quad \text{By i.h.}
\]

\[
\Gamma \rightarrow \Delta', \hat{\alpha} : \kappa \quad \text{By rule } \text{Add}
\]

Case \( \Theta' \rightarrow \Delta' \)

\[
[\Delta']t = [\Delta']t' \quad \text{Solved}
\]

\[
\Theta', \hat{\alpha} : \kappa = t \rightarrow \Delta', \hat{\alpha} : \kappa = t' \quad \text{Solved}
\]

Two rules could have concluded \( D : \Gamma \rightarrow (\Theta', \hat{\alpha} : \kappa = t) \):

- Case \( \Gamma' \rightarrow \Theta' \)

\[
[\Theta']t'' = [\Theta']t \quad \text{Solved}
\]

\[
[\Delta']\Theta' = [\Delta']\Theta \quad \text{Premise}
\]

\[
[\Delta']\Theta' = [\Delta']\Theta \quad \text{Applying } \Delta' \text{ to both sides}
\]

\[
[\Delta']t'' = [\Delta']t \quad \text{By Lemma 28 (Substitution Monotonicity)}
\]

\[
[\Delta']t' = [\Delta']t \quad \text{By premise } \Delta' = [\Delta']t'
\]

\[
\Gamma', \hat{\alpha} : \kappa \rightarrow t'' \rightarrow \Delta', \hat{\alpha} : \kappa = t' \quad \text{By rule } \text{Solved}
\]
Proof of Lemma 32 (Extension Transitivity) lem:extension-transitivity

\[ \text{– Case } \Gamma \rightarrow \Theta' \]
\[ \Gamma \rightarrow \Theta', \alpha : \kappa = t' \]
\[ \rightarrow \text{AddSolved} \]
\[ \Gamma \rightarrow \Delta', \alpha : \kappa = t' \]
\[ \text{By rule } \rightarrow \text{AddSolved} \]

- Case \[ \Theta' \rightarrow \Delta' \]
\[ [\Delta']t = [\Delta']t' \]
\[ \rightarrow \text{Eqn} \]
\[ \Gamma = (\Gamma', \alpha = t'') \]
\[ \text{By inversion on } D \]
\[ \Gamma' \rightarrow \Theta' \]
\[ \text{By inversion on } D \]
\[ [\Theta']t'' = [\Theta']t \]
\[ \text{By inversion on } D \]
\[ [\Delta']t'' = [\Delta']t \]
\[ \text{Applying } \Delta' \text{ to both sides} \]
\[ \text{By Lemma 28 (Substitution Monotonicity)} \]
\[ \Gamma' \rightarrow \Delta' \]
\[ \text{By i.h.} \]
\[ [\Delta']t'' = [\Delta']t' \]
\[ \text{By premise } [\Delta']t = [\Delta']t' \]
\[ \Gamma', \alpha = t'' \rightarrow \Delta', \alpha = t' \]
\[ \text{By rule } \rightarrow \text{Eqn} \]

- Case \[ \Theta \rightarrow \Delta' \]
\[ \Theta \rightarrow \Delta', \alpha : \kappa \]
\[ \rightarrow \text{Add} \]
\[ \Gamma \rightarrow \Delta' \]
\[ \text{By i.h.} \]
\[ \Gamma \rightarrow \Delta', \alpha : \kappa = t \]
\[ \text{By rule } \rightarrow \text{Add} \]

- Case \[ \Theta \rightarrow \Delta' \]
\[ \Theta \rightarrow \Delta', \alpha : \kappa = t \]
\[ \rightarrow \text{AddSolved} \]
\[ \Gamma \rightarrow \Delta' \]
\[ \text{By i.h.} \]
\[ \Gamma \rightarrow \Delta', \alpha : \kappa = t \]
\[ \text{By rule } \rightarrow \text{AddSolved} \]

- Case \[ \Theta' \rightarrow \Delta' \]
\[ \Theta' \rightarrow \Delta', \alpha : \kappa \]
\[ \rightarrow \text{Marker} \]
\[ \Gamma = \Gamma', \alpha = t'' \]
\[ \text{By inversion on } D \]
\[ \Gamma' \rightarrow \Theta' \]
\[ \text{By inversion on } D \]
\[ \Gamma' \rightarrow \Delta' \]
\[ \text{By i.h.} \]
\[ \Gamma', \alpha = t'' \rightarrow \Delta', \alpha = t' \]
\[ \text{By } \rightarrow \text{Uvar} \]

D’.2 Weakening

Lemma 33 (Suffix Weakening). If \( \Gamma \vdash t : \kappa \) then \( \Gamma, \Theta \vdash t : \kappa \).

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

Lemma 34 (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. \[ \square \]

Lemma 35 (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 21 (Extension Inversion) (i) or (v). In the SolvedVarSort case, use Lemma 21 (Extension Inversion) (iv). In the other cases, apply the i.h. to all subderivations, then apply the rule. \[ \square \]
Lemma 36 (Extension Weakening (Props)). If $\Gamma \vdash P$ prop and $\Gamma \rightarrow \Delta$ then $\Delta \vdash P$ prop.

Proof. By inversion on rule $\text{Eq}^\text{prop}$ and Lemma 35 (Extension Weakening (Sorts)) twice. \hfill $\square$

Lemma 37 (Extension Weakening (Types)). If $\Gamma \vdash A$ type and $\Gamma \rightarrow \Delta$ then $\Delta \vdash A$ type.

Proof. By a straightforward induction on $\Gamma \vdash A$ type.

In the $\text{VarWF}$ case, use Lemma 21 (Extension Inversion) (i) or (v). In the $\text{SolvedVarWF}$ case, use Lemma 21 (Extension Inversion) (iv).

In the other cases, apply the i.h. and/or (for $\text{ImpliesWF}$ and $\text{WithWF}$) Lemma 36 (Extension Weakening (Props)) to all subderivations, then apply the rule. \hfill $\square$

D’.3 Principal Typing Properties

Lemma 38 (Principal Agreement).

(i) If $\Gamma \vdash A \mid A$ type and $\Gamma \rightarrow \Delta$ then $[\Delta]A = [\Gamma]A$.

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

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

Part (i):

- Case $\Gamma_0 \rightarrow \Delta_0 \mid [\Delta_0]t = [\Delta_0]t'$

If $\alpha \not\in \text{FV}(A)$, then:

$\begin{align*}
[\Gamma_0, \alpha = t]A &= [\Gamma_0]A & \text{By def. of subst.} \\
&= [\Delta_0]A & \text{By i.h.} \\
&= [\Delta_0, \alpha = t']A & \text{By def. of subst.}
\end{align*}$

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

- $\Gamma_0 \vdash t \mid t$ type
- $\Gamma_0 \vdash [\Gamma_0]t \mid t$ type By Lemma 12 (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.

$\begin{align*}
\text{FEV}([\Gamma_0]t) &\neq \emptyset & \text{Assumption (for contradiction)} \\
[\Gamma_0]t &= [\Gamma]t & \text{By def. of subst.} \\
\text{FEV}([\Gamma]t) &\neq \emptyset & \text{By above equality} \\
\alpha &\in \text{FV}(A) & \text{Above} \\
\text{FEV}([\Gamma]A) &\neq \emptyset & \text{By a property of subst.} \\
\Gamma &\vdash A \mid A \text{ type} & \text{Given} \\
\text{FEV}([\Gamma]A) = \emptyset & & \text{By inversion} \\
\implies & & \\
\text{FEV}([\Gamma_0]t) &= \emptyset & \text{By contradiction} \\
\Gamma_0 &\vdash t \mid t \text{ type} & \text{By PrincipalWF} \\
[\Gamma_0]t &= [\Delta_0]t & \text{By i.h.} \\
\Gamma_0 &\vdash [\Delta_0]t \mid t & \text{By above equality} \\
\text{FEV}([\Delta_0]t) &= \emptyset & \text{By above equality} \\
\Gamma_0 &\vdash ([\Delta_0]t/\alpha)A \mid A \text{ type} & \text{By Lemma 7 (Substitution—Well-formedness) (i)} \\
[\Gamma_0]([\Delta_0]t/\alpha)A &= [\Delta_0][\Delta_0]t/\alpha)A & \text{By i.h. (at $[\Delta_0]t/\alpha)A$) } \\
\Gamma_0, \alpha = t]A &= [\Gamma_0][\Gamma_0]t/\alpha)A & \text{By def. of subst.} \\
&= [\Gamma_0][\Delta_0]t/\alpha)A & \text{By above equality} \\
&= [\Delta_0][\Delta_0]t/\alpha)A & \text{By above equality} \\
&= [\Delta_0][\Delta_0]t'/\alpha)A & \text{By $[\Delta_0]t = [\Delta_0]t'$} \\
&= [\Delta]A & \text{By def. of subst.}
\end{align*}$
Lemma 39 (Right-Hand Subst. for Principal Typing). If $\Gamma \vdash A \ p$ type then $\Gamma \vdash [\Gamma] A \ p$ type.

Proof. By cases of $p$:

1. Case $p = 1$:
   $\Gamma \vdash A$ type  
   $\text{By inversion}$
   $\text{FEV}([\Gamma] A) = \emptyset$  
   $\text{By inversion}$
   $\Gamma \vdash [\Gamma] A$ type  
   $\text{By Lemma 12 (Right-Hand Substitution for Typing)}$
   $\Gamma \vdash \Gamma \rightarrow \Gamma$  
   $\text{By Lemma 31 (Extension Reflexivity)}$
   $[\Gamma][\Gamma] A = [\Gamma] A$  
   $\text{By Lemma 28 (Substitution Monotonicity)}$
   $\text{FEV}([\Gamma][\Gamma] A) = \emptyset$  
   $\text{By inversion}$
   $\Gamma \vdash [\Gamma] A \ !$ type  
   $\text{By rule PrincipalWF}$

2. Case $p = f$:
   $\Gamma \vdash A$ type  
   $\text{By inversion}$
   $\Gamma \vdash [\Gamma] A$ type  
   $\text{By Lemma 12 (Right-Hand Substitution for Typing)}$
   $\Gamma \vdash A \ f$ type  
   $\text{By rule NonPrincipalWF}$

3. Case $p = 1$:
   $\Gamma \vdash A$ type  
   $\text{By inversion}$
   $\text{FEV}([\Gamma] A) = \emptyset$  
   $\text{By inversion}$
   $\Delta \vdash A$ type  
   $\text{By Lemma 37 (Extension Weakening (Types))}$
   $\Delta \vdash A \ f$ type  
   $\text{By rule NonPrincipalWF}$

Lemma 40 (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 $p$:

1. Case $p = f$:
   $\Delta \vdash A$ type  
   $\text{By Lemma 37 (Extension Weakening (Types))}$
   $\Delta \vdash A \ f$ type  
   $\text{By rule NonPrincipalWF}$

2. Case $p = 1$:
   $\Delta \vdash A$ type  
   $\text{By inversion}$
   $\text{FEV}([\Delta] A) = \emptyset$  
   $\text{By congruence of equality}$
   $\Delta \vdash [\Delta] A \ !$ type  
   $\text{By rule PrincipalWF}$

Lemma 41 (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.

1. Case $p = f$:
   $\Gamma \vdash A \rightarrow B$ type  
   $\text{By inversion}$
   $\Gamma \vdash A$ type  
   $\text{By inversion on 1}$
   $\Gamma \vdash B$ type  
   $\text{By inversion on 1}$
   $\Gamma \vdash A \ f$ type  
   $\text{By rule NonPrincipalWF}$
   $\Gamma \vdash B \ f$ type  
   $\text{By rule NonPrincipalWF}$
Proof of Lemma 41 (Inversion of Principal Typing)

\[ \text{lem:principal-inversion} \]

Case \( p = 1 \):

1. \( \Gamma \vdash A \rightarrow B \text{ type} \) By inversion on \( \Gamma \vdash A \rightarrow B \text{ ! type} \)
2. \( \emptyset = \text{FEV}(\Gamma | (A \rightarrow B)) \) By definition of \( \Gamma \)
3. \( \text{FEV}(\Gamma | A) \rightarrow \text{FEV}(\Gamma | B) \) By definition of \( \text{FEV}(-) \)
4. \( \text{FEV}(\Gamma | A) = \text{FEV}(\Gamma | B) = \emptyset \) By properties of empty sets and unions
5. \( \Gamma \vdash A \text{ type} \) By inversion on \( 1 \)
6. \( \Gamma \vdash B \text{ type} \) By inversion on \( 1 \)
7. \( \Gamma \vdash \Lambda \text{ type} \) By rule \[ \text{PrincipalWF} \]
8. \( \Gamma \vdash B \text{ ! type} \) By rule \[ \text{PrincipalWF} \]

Part 2: We have \( \Gamma \vdash P \sqsupset A \ p \text{ type} \). Similar to Part 1.
Part 3: We have \( \Gamma \vdash A \land P \ p \text{ type} \). Similar to Part 2.

D'.4 Instantiation Extends

Lemma 42 (Instantiation Extension).

If \( \Gamma \vdash \alpha := \tau : \kappa \vdash \Delta \) then \( \Gamma \vdash \alpha \vdash \Delta \).

Proof. By induction on the given derivation.

- Case \[ \Gamma_1 \vdash \tau : \kappa \]
  \[ \Gamma_1, \alpha : \kappa, \Gamma_R \vdash \alpha := \tau : \kappa \vdash \Gamma_1, \alpha : \kappa = \tau, \Gamma_R \]
  Follows by Lemma 22 (Deep Evar Introduction) (ii).

- Case \( \beta \in \text{unsolved}(\Gamma_0[\alpha : \kappa][\beta : \kappa]) \)
  \[ \Gamma_0[\alpha : \kappa][\beta : \kappa] \vdash \alpha := \beta : \kappa \vdash \Gamma_0[\alpha : \kappa][\beta : \kappa] \]
  Follows by Lemma 22 (Deep Evar Introduction) (ii).

- Case \( \Gamma_0[\beta_2 : *, \beta_1 : *, \beta : * = \beta_1 \oplus \beta_2] \vdash \beta_1 := \tau_1 : * \vdash \Theta \vdash \beta_2 := [\Theta] \tau_2 : * \vdash \Delta \)
  \[ \Gamma_0[\alpha : \kappa] \vdash \alpha := \tau_1 \oplus \tau_2 : * \vdash \Delta \]
  Follows by Lemma 22 (Deep Evar Introduction) (ii).

- Case \( \Gamma_0[\beta_2 : *, \beta_1 : *, \beta : * = \beta_1 \oplus \beta_2] \vdash \beta_1 := \tau_1 : * \vdash \Theta \)
  \[ \Theta \vdash \beta_2 := [\Theta] \tau_2 : * \vdash \Delta \]
  By i.h.

- Case \( \Gamma_0[\beta_2 : *, \beta_1 : *, \beta : * = \beta_1 \oplus \beta_2] \rightarrow \Theta \)
  \[ \Theta \vdash \beta_2 := [\Theta] \tau_2 : * \vdash \Delta \]
  By i.h.

- Case \( \Gamma_0[\beta_2 : *, \beta_1 : *, \beta : * = \beta_1 \oplus \beta_2] \rightarrow \Delta \)
  By Lemma 32 (Extension Transitivity)

- Case \( \Gamma_0[\beta : *] \rightarrow \Gamma_0[\beta_2 : *, \beta_1 : *, \beta : * = \beta_1 \oplus \beta_2] \)
  By Lemma 22 (Deep Evar Introduction) (parts (i), (i), and (ii), using Lemma 32 (Extension Transitivity))

- Case \( \Gamma_0[\beta : *] \rightarrow \Delta \)
  By Lemma 32 (Extension Transitivity)

- Case \( \Gamma_0[\alpha : N] \vdash \alpha := \text{zero} : N \vdash \Gamma_0[\alpha : N = \text{zero}] \)
  Follows by Lemma 22 (Deep Evar Introduction) (ii).

- Case \( \Gamma[\alpha_1 : N, \beta : N = \text{succ} \alpha_1] \vdash \alpha_1 := t_1 : N \vdash \Delta \)
  \[ \Gamma[\alpha : N] \vdash \alpha := \text{succ}(t_1) : N \vdash \Delta \]
  By reasoning similar to the \[ \text{InstBin} \] case.
D’.5 Equivalence Extends

Lemma 43 (Elimeq Extension).
If $\Gamma / s \doteq t : \kappa \vdash \Delta$ then there exists $\Theta$ such that $\Gamma, \Theta \longrightarrow \Delta$.

Proof. By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context $\Delta$.

- Case $\Gamma / \alpha \doteq \alpha : \kappa \vdash \Gamma$ (ElimeqUvarRefl)
  
  Since $\Delta = \Gamma$, applying Lemma 31 (Extension Reflexivity) suffices (let $\Theta = \cdot$).

- Case $\Gamma / \text{zero} \doteq \text{zero} : \mathbb{N} \vdash \Gamma$ (ElimeqZero)
  
  Similar to the ElimeqUvarRefl case.

- Case $\Gamma / \sigma \doteq t : \mathbb{N} \vdash \Delta$
  
  $\Gamma / \text{succ}(\sigma) \doteq \text{succ}(t) : \mathbb{N} \vdash \Delta$ (ElimeqSucc)
  
  Follows by i.h.

- Case $\Gamma_0[\hat{\alpha} : \kappa] \vdash \hat{\alpha} \equiv t : \kappa \vdash \Delta$ (ElimeqInstL)
  
  $\Gamma_0[\hat{\alpha} : \kappa] / \hat{\alpha} \doteq t : \kappa \vdash \Delta$ (ElimeqInstL)
  
  $\Gamma \vdash \hat{\alpha} \equiv t : \kappa \vdash \Delta$ Subderivation
  
  $\Gamma \longrightarrow \Delta$ By Lemma 42 (Instantiation Extension)
  
  Let $\Theta = \cdot$.

- Case $\alpha \notin \text{FV}(\Gamma)[t]$ (i.e. $\alpha = t$) (ElimeqUvarL)
  
  Let $\Theta$ be $(\alpha = t)$.
  
  $\Gamma, \alpha \equiv t \vdash \Gamma, \alpha = t$ By Lemma 31 (Extension Reflexivity)

- Cases ElimeqInstR, ElimeqUvarR
  
  Similar to the respective L cases.

- Case $\sigma \neq t$
  
  $\Gamma / \sigma \doteq t : \kappa \vdash \bot$ (ElimeqClash)
  
  The statement says that the output is a (consistent) context $\Delta$, so this case is impossible. \hfill $\square$

Lemma 44 (Elimprop Extension).
If $\Gamma / P \vdash \Delta$ then there exists $\Theta$ such that $\Gamma, \Theta \longrightarrow \Delta$.

Proof. By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context $\Delta$.

- Case $\Gamma / \sigma \doteq t : \mathbb{N} \vdash \Delta$ (ElimpropEq)
  
  $\Gamma / \sigma \doteq t : \mathbb{N} \vdash \Delta$ Subderivation
  
  $\Gamma, \Theta \longrightarrow \Delta$ By Lemma 43 (Elimeq Extension) \hfill $\square$
Lemma 45 (Checkeq Extension).
If \( \Gamma \vdash A \equiv B \not\vdash \Delta \) then \( \Gamma \not\rightarrow \Delta \).

Proof. By induction on the given derivation.

- Case \( \Gamma \vdash u \equiv u : \kappa \vdash \Gamma \) \( \text{CheckeqVar} \)
  
  Since \( \Delta = \Gamma \), applying Lemma \( \text{31} \) (Extension Reflexivity) suffices.

- Cases \( \text{CheckeqUnit} \), \( \text{CheckeqZero} \) Similar to the \( \text{CheckeqVar} \) case.

- Case \( \Gamma \vdash \tau_1 \equiv \tau_1' : \star \vdash \Theta \quad \Theta \vdash [\Theta] \tau_2 \equiv [\Theta] \tau_2' : \star \vdash \Delta \)
  
  \( \Gamma \vdash \tau_1 \oplus \tau_2 \equiv \tau_1' \oplus \tau_2' : \star \vdash \Delta \) \( \text{CheckeqBin} \)
  
  \( \Gamma \not\rightarrow \Theta \) By i.h.
  \( \Theta \not\rightarrow \Delta \) By i.h.
  \( \therefore \Gamma \not\rightarrow \Delta \) By Lemma \( \text{32} \) (Extension Transitivity)

- Case \( \Gamma \vdash \sigma \equiv t : N \vdash \Delta \)
  
  \( \Gamma \vdash \sigma = t \text{ true} \vdash \Delta \) \( \text{CheckpropEq} \)
  
  \( \Gamma \vdash \sigma \equiv t : N \vdash \Delta \) Subderivation
  
  \( \therefore \Gamma \not\rightarrow \Delta \) By Lemma \( \text{45} \) (Checkeq Extension)

Lemma 46 (Checkprop Extension).
If \( \Gamma \vdash P \not\vdash \Delta \) then \( \Gamma \not\rightarrow \Delta \).

Proof. By induction on the given derivation.

- Case \( \Gamma \vdash \sigma \equiv t : N \vdash \Delta \)
  
  \( \Gamma \vdash \sigma = t \text{ true} \vdash \Delta \) \( \text{CheckpropEq} \)
  
  \( \Gamma \vdash \sigma \equiv t : N \vdash \Delta \) Subderivation
  
  \( \therefore \Gamma \not\rightarrow \Delta \) By Lemma \( \text{45} \) (Checkeq Extension)

Lemma 47 (Prop Equivalence Extension).
If \( \Gamma \vdash P \equiv Q \not\vdash \Delta \) then \( \Gamma \not\rightarrow \Delta \).

Proof. By induction on the given derivation.

- Case \( \Gamma \vdash \sigma_1 \equiv \tau_1 : N \vdash \Theta \quad \Theta \vdash \sigma_2 \equiv \tau_2 : N \vdash \Delta \)
  
  \( \Gamma \vdash (\sigma_1 = \sigma_2) \equiv (\tau_1 = \tau_2) \vdash \Delta \) \( \equiv \text{PropEq} \)
  
  \( \Gamma \vdash \sigma_1 \equiv \tau_1 : N \vdash \Theta \) Subderivation
  \( \Gamma \not\rightarrow \Theta \) By Lemma \( \text{45} \) (Checkeq Extension)
  \( \Theta \not\rightarrow \Delta \) By Lemma \( \text{45} \) (Checkeq Extension)
  \( \therefore \Gamma \not\rightarrow \Delta \) By Lemma \( \text{32} \) (Extension Transitivity)

Proof of Lemma 48 (Equivalence Extension)
Lemma 48 (Equivalence Extension).

If $\Gamma \vdash A \equiv B \vdash \Delta$ then $\Gamma \rightarrow \Delta$.

Proof. By induction on the given derivation.

- Case
  $\Gamma \vdash \exists x. \equiv \exists x \rightarrow \Gamma$

  Here $\Delta = \Gamma$, so Lemma 31 (Extension Reflexivity) suffices.

- Case
  $\Gamma \vdash \forall \alpha : \exists \equiv \forall \alpha \rightarrow \Gamma$

  Similar to the $\equiv \exists \equiv$ case.

- Case
  $\Gamma \vdash \exists x. \equiv \exists x \rightarrow \Gamma$

  Similar to the $\equiv \exists \equiv$ case.

- Case
  $\Gamma \vdash A_1 \equiv B_1 \vdash \Theta \mid A_2 \equiv \Theta B_2 \vdash \Delta$

  $\Gamma \vdash A_1 \equiv B_1 \vdash \Theta$
  Subderivation
  $\Gamma \rightarrow \Theta$
  By i.h.
  $\Theta \vdash \Theta A_2 \equiv \Theta B_2 \vdash \Delta$
  Subderivation
  $\Theta \rightarrow \Delta$
  By i.h.
  $\Gamma \rightarrow \Delta$
  By Lemma 32 (Extension Transitivity)

- Cases $\equiv \exists \equiv \forall$
  Similar to the $\equiv \exists \equiv$ case, but with Lemma 47 (Prop Equivalence Extension) on the first premise.

- Case
  $\Gamma, \alpha : \kappa \vdash A_0 \equiv B \vdash \Delta, \alpha : \kappa, \Delta'$

  $\Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B \vdash \Delta$
  Subderivation
  $\Gamma, \alpha : \kappa \rightarrow \Delta, \alpha : \kappa, \Delta'$
  By i.h.
  $\Gamma \rightarrow \Delta$
  By Lemma 21 (Extension Inversion) (i)

- Case
  $\Gamma_0[\bar{x}] \vdash \bar{\alpha} := \tau : \star \rightarrow \Delta$

  $\Gamma_0[\bar{x}] \vdash \bar{\alpha} \equiv \tau \rightarrow \Delta$
  InstantiationL
  $\Gamma_0[\bar{x}] \rightarrow \Delta$
  By Lemma 42 (Instantiation Extension)

- Case $\equiv \text{InstantiateR}$
  Similar to the $\equiv \text{InstantiateL}$ case.

D’.6 Subtyping Extends

Lemma 49 (Subtyping Extension). If $\Gamma \vdash A \ll B \vdash \Delta$ then $\Gamma \rightarrow \Delta$.

Proof. By induction on the given derivation.

- Case
  $\Gamma \vdash \forall \alpha : \kappa. A \ll \forall \alpha : \kappa. B \rightarrow \Delta$

  $\Gamma \vdash \forall \alpha : \kappa. A \ll \forall \alpha : \kappa. B \rightarrow \Delta$
  Subderivation
  $\Gamma \rightarrow \Delta$
  By i.h. (i)

- Case
  $\Gamma \vdash \exists x. \neq \exists x \rightarrow \Gamma$

  $\Gamma \vdash \exists x. \neq \exists x \rightarrow \Gamma$
  By Lemma 21 (Extension Inversion) (ii)
Proof of Lemma 49 (Subtyping Extension)

• Case \( \mathbb{C} \vdash \mathbb{R} \): Similar to the \( \mathbb{C} \vdash \mathbb{V} \) case.

• Case \( \Gamma, \alpha : \kappa \vdash A < : \beta \Delta, \alpha : \kappa, \Theta \)
  \[
  \Gamma \vdash A < : \forall \alpha : \kappa. B < : \Delta
  \]
  Similar to the \( \mathbb{C} : \mathbb{V} \) case, but using part (i) of Lemma 21 (Extension Inversion).

• Case \( \mathbb{C} : \mathbb{E} \) Similar to the \( \mathbb{C} : \mathbb{V} \) case.

• Case \( \Gamma \vdash A \equiv B < : \Delta \)
  \[
  \Gamma \vdash A < : \forall \alpha : \kappa. B < : \Delta
  \]
  Subderivation
  \[
  \Gamma \longrightarrow \Delta
  \]
  By Lemma 48 (Equivalence Extension) \( \Box \)

D'7 Typing Extends

Lemma 50 (Typing Extension).
If \( \Gamma \vdash e \leftarrow A \rightarrow \Delta \)
or \( \Gamma \vdash e \rightarrow A \rightarrow \Delta \)
or \( \Gamma \vdash s : A \rightarrow B \rightarrow \Delta \)
or \( \Gamma \vdash \Pi :: \vec{A} \leftarrow C \rightarrow \Delta \)
or \( \Gamma / P \vdash \Pi :: \vec{A} \leftarrow C \rightarrow \Delta \)
then \( \Gamma \longrightarrow \Delta \).

Proof. By induction on the given derivation.

• Match judgments:
  In rule MatchEmpty, \( \Delta = \Gamma \), so the result follows by Lemma 31 (Extension Reflexivity).
  Rules MatchBase, Match×, Match+, 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 32 (Extension Transitivity) is also needed.
  In rule Match∃ apply the i.h., then use Lemma 21 (Extension Inversion) (i).
  Match∧ Use the i.h.
  MatchNeg Use the i.h. and Lemma 21 (Extension Inversion) (v).
  Match⊥ Immediate by Lemma 31 (Extension Reflexivity).
  MatchUnify
  \[
  \Gamma, \vec{p}, \Theta' \longrightarrow \Theta
  \]
  By Lemma 43 (Elimeq Extension)
  \[
  \Theta \longrightarrow \Delta, \vec{p}, \Delta'
  \]
  By i.h.
  By Lemma 32 (Extension Transitivity)

• Synthesis, checking, and spine judgments:
  In rules Var, Id, EmptySpine and -> Spine the output context \( \Delta \) is exactly \( \Gamma \), so the result follows by Lemma 31 (Extension Reflexivity).
  – Case \( \Box \) Use the i.h. and Lemma 32 (Extension Transitivity).
  – Case \( \forall \vec{p} \) By \( \longrightarrow Add \), \( \Gamma \longrightarrow \Gamma, \vec{\alpha} : \kappa. \)
The result follows by i.h. and Lemma 32 (Extension Transitivity).
  – Cases \( \forall \vec{p} \rightarrow \vec{p} \)
  Use Lemma 46 (Checkprop Extension), the i.h., and Lemma 32 (Extension Transitivity).
  – Case \( \exists \) Use the i.h.
  – Case \( \exists \vec{p} \)
  \[
  \Gamma, \vec{p}, \Theta' \longrightarrow \Theta
  \]
  By Lemma 44 (Elimprop Extension)
  \[
  \Theta \longrightarrow \Delta, \vec{p}, \Delta
  \]
  By i.h.
  By Lemma 32 (Extension Transitivity)

Proof of Lemma 50 (Typing Extension)
Proof of Lemma 50 \((\text{Typing Extension})\) \lem:typing-extension

- **Case \([\cdot]\):** Use the i.h. andLemma 21 \((\text{Extension Inversion})\).
- **Cases** Sub, Anno \(\rightarrow E\) \(\rightarrow \text{E-1}\) \(\rightarrow \text{Spine}\) \(\rightarrow \text{I}\) \(\rightarrow \text{I}\)
  Use the i.h., andLemma 32 \((\text{Extension Transitivity})\) as needed.
- **Case** \(\alpha\) \(\text{Spine}\) \(\rightarrow \text{I}\)
  Use Lemma 22 \((\text{Deep Evar Introduction})\) (i) twice, Lemma 22 \((\text{Deep Evar Introduction})\) (ii), the i.h. and Lemma 21 \((\text{Extension Inversion})\) (v).
- **Case** \(\rightarrow \text{I}\)
  Use Lemma 22 \((\text{Deep Evar Introduction})\) (i) twice, Lemma 22 \((\text{Deep Evar Introduction})\) (ii), the i.h. and Lemma 21 \((\text{Extension Inversion})\) (v).
- **Case** Use the i.h. on the synthesis premise and the match premise, and then Lemma 32 \((\text{Extension Transitivity})\).

**D’8 Unfiled**

**Lemma 51** \((\text{Context Partitioning})\).

If \(\Delta, \triangleright_\alpha, \Theta \rightarrow \Omega, \triangleright_\alpha, \Omega_Z\) then there is a \(\Psi\) such that \([\Omega, \triangleright_\alpha, \Omega_Z]; \Delta, \triangleright_\alpha, \Theta = \Omega \Delta, \Psi\).

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

- **Case** \(\rightarrow \text{Id}\) Impossible: \(\Delta, \triangleright_\alpha, \Theta\) cannot have the form \(\cdot\).
- **Case** \(\rightarrow \text{Var}\) We have \(\Omega_Z = (\Omega_Z', x : A)\) and \(\Theta = (\Theta', x : A')\). By i.h., there is \(\Psi'\) such that \([\Omega, \triangleright_\alpha, \Omega_Z]; \Delta, \triangleright_\alpha, \Theta = \Omega \Delta, \Psi'\). Then by the definition of context application, \([\Omega, \triangleright_\alpha, \Omega_Z, x : A]; \Delta, \triangleright_\alpha, \Theta', x : A' = \Omega \Delta, \Psi', x : [\Omega']A\). Let \(\Psi = (\Psi', x : [\Omega']A)\).
- **Case** \(\rightarrow \text{Uvar}\) Similar to the \(\rightarrow \text{Var}\) case, with \(\Psi = (\Psi', \alpha : k)\).
- **Cases** \(\rightarrow \text{Eqn}\) \(\rightarrow \text{Unsolved}\) \(\rightarrow \text{Solved}\) \(\rightarrow \text{Solve}\) \(\rightarrow \text{Add}\) \(\rightarrow \text{AddSolved}\) \(\rightarrow \text{Marker}\)
  Broadly similar to the \(\rightarrow \text{Uvar}\) case, but the rightmost context element disappears in context application, so we let \(\Psi = \Psi'\).

**Lemma 53** \((\text{Completing Stability})\).

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** \(\rightarrow \text{Var}\)
  \[
  \begin{array}{l}
  \Gamma_0 \rightarrow \Omega_0 \\
  \overline{[\Omega_0] A = [\Omega_0] A'} \\
  \overline{\Gamma_0, x : A \rightarrow \Omega_0, x : A'} \\
  \overline{\rightarrow \text{Var}} \\
  \end{array}
  \]
  Subderivation
  \[
  \Gamma_0 \rightarrow \Omega_0 \\
  \overline{[\Omega_0] A = [\Omega_0] A'} \\
  \overline{\rightarrow \text{Var}} \\
  \]
  By i.h.
- **Case** \(\rightarrow \text{Var}\)
  \[
  \begin{array}{l}
  \Gamma_0, \alpha : k \rightarrow \Omega_0, \alpha : k \\
  \overline{\rightarrow \text{Uvar}} \\
  \end{array}
  \]
  Similar to \(\rightarrow \text{Var}\)
- **Case** \(\rightarrow \text{Var}\)
  \[
  \begin{array}{l}
  \Gamma_0 \rightarrow \Omega_0 \\
  \overline{\rightarrow \text{Unsolved}} \\
  \end{array}
  \]
  Similar to \(\rightarrow \text{Var}\)
Proof of Lemma 53 (Completing Completeness).

(i) If $\Omega \vdash t : \kappa$ then $[\Omega]t = [\Omega']t$.

(ii) If $\Omega \vdash \Delta$ type then $[\Omega]\Delta = [\Omega']\Delta$.

(iii) If $\Omega \vdash \Delta'$ then $[\Omega]\Delta = [\Omega']\Delta'$.

Proof.

• Part (i):
  By Lemma 28 (Substitution Monotonicity), (i), $[\Omega']t = [\Omega'][\Omega]t$.
  Now we need to show $[\Omega']t = [\Omega]t$. Considered as a substitution, $\Omega'$ is the identity everywhere except existential variables $\exists$ and universal variables $\forall$. 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 23 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 28 (Substitution Monotonicity) (iii) instead of (i).
• **Part (iii):** By induction on the given derivation of \( \Omega \rightarrow \Omega' \).

  Only cases \( \rightarrow \text{Id} \), \( \rightarrow \text{Var} \), \( \rightarrow \text{Uvar} \), \( \rightarrow \text{Eqn} \), \( \rightarrow \text{AddSolved} \), \( \rightarrow \text{Solved} \), \( \rightarrow \text{DomSolved} \), \( \rightarrow \text{Solved} \), \( \rightarrow \text{Marker} \), \( \rightarrow \text{Eqn} \), \( \rightarrow \text{Marker} \), \( \rightarrow \text{Eqn} \), \( \rightarrow \text{Marker} \) are possible. In all of these cases, we use the i.h. and the definition of context application; in cases \( \rightarrow \text{Var} \), \( \rightarrow \text{Eqn} \), \( \rightarrow \text{Marker} \), we also use the equality in the premise of the respective rule.

**Lemma 55** (Confluence of Completeness).
If \( \Delta_1 \rightarrow \Omega \) and \( \Delta_2 \rightarrow \Omega \) then \( \Omega \Delta_1 = \Omega \Delta_2 \).

**Proof.**
\[\Delta_1 \rightarrow \Omega \quad \text{Given} \]
\[\{\Omega\} \Delta_1 = [\Omega] \Omega \quad \text{By Lemma 53 (Completing Stability)} \]
\[\Delta_2 \rightarrow \Omega \quad \text{Given} \]
\[\{\Omega\} \Delta_2 = [\Omega] \Omega \quad \text{By Lemma 53 (Completing Stability)} \]
\[\{\Omega\} \Delta_1 = [\Omega] \Delta_2 \quad \text{By transitivity of equality} \]

**Lemma 56** (Multiple Confluence).
If \( \Delta \rightarrow \Omega \) and \( \Omega \rightarrow \Omega' \) and \( \Delta' \rightarrow \Omega' \) then \( [\Omega] \Delta = [\Omega'] \Delta' \).

**Proof.**
\[\Delta \rightarrow \Omega \quad \text{Given} \]
\[\{\Omega\} \Delta = [\Omega] \Omega \quad \text{By Lemma 53 (Completing Stability)} \]
\[\Omega \rightarrow \Omega' \quad \text{Given} \]
\[\{\Omega\} \Omega = [\Omega'] \Omega' \quad \text{By Lemma 54 (Completing Completeness) (iii)} \]
\[\{\Omega\} \Delta = [\Omega'] \Omega' \quad \text{By Lemma 53 (Completing Stability) (\( \Delta' \rightarrow \Omega' \) given)} \]

**Lemma 58** (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 \( \forall \alpha : \kappa = \tau \) and \( \forall \alpha = \tau \) 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 \( \Omega(\Gamma) \).

**Lemma 59** (Split Solutions).
If \( \Delta \rightarrow \Omega \) and \( \forall \alpha \in \text{unsolved}(\Delta) \)
then there exists \( \Omega_1 = \Omega^{\prime}_1[\forall \alpha : \kappa = t_1] \) such that \( \Delta \rightarrow \Omega_1 \) and \( \Omega_2 = \Omega^{\prime}_1[\forall \alpha : \kappa = t_2] \) where \( \Delta \rightarrow \Omega_2 \) and \( t_2 \neq t_1 \) and \( \Omega_2 \) is canonical.

**Proof.** Use Lemma 58 (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 \( \Omega_{\text{canon}} = \Omega^{\prime}_1[\forall \alpha : \kappa = t_1] \), where \( \text{FEV}(t_1) = \emptyset \). Therefore \( \Rightarrow \Omega^{\prime}_1[\forall \alpha : \kappa = t_1] \rightarrow \Omega \).
Now choose \( t_2 \) as follows:
- If \( t_2 \neq t_1 \), let \( t_2 = t_1 \rightarrow t_1 \).
- If \( t_2 = \star \), let \( t_2 = \text{succ}(t_1) \).

Thus, \( \Rightarrow t_2 \neq t_1 \). Let \( \Omega_2 = \Omega^{\prime}_1[\forall \alpha : \kappa = t_2] \).
\( \Rightarrow \Delta \rightarrow \Omega_2 \) By Lemma 30 (Split Extension)

**E’ Internal Properties of the Declarative System**

**Lemma 60** (Interpolating With and Exists).

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

2. If \( \Gamma \vdash \Pi : \tilde{A} \leftrightarrow [\sigma/\alpha]C_0 \ p \) and \( \Psi \vdash \tau : \kappa \)
   then \( \Gamma' \vdash \Psi \vdash \Pi : \tilde{A} \leftrightarrow (\exists \alpha : \kappa . C_0) \ p \).
Proof of Lemma 60 (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 DeclMatchBase case, for part (1), apply rule $\sqcap I$. For part (2), apply rule $\exists I$.

In the DeclMatchNeg case, part (1), use Lemma 1 (Declarative Weakening) (iii). In part (2), use Lemma 1 (Declarative Weakening) (i).

Lemma 61 (Case Invertibility).

If $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C p$ 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 $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A q \quad \text{pol}(B) \vdash \Psi \leq^* AB$ $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow B p$

DecSub

Impossible, because $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A q$ is not derivable.

• Cases $\text{Decl}\forall$, $\text{Decl}\rightarrow$: Impossible: these rules have a value restriction, but a case expression is not a value.

• Case $\Psi \vdash P \text{true}$ $\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C_0 p$

Dec\lor

$\Rightarrow < n - 1 \quad \Psi \vdash e_0 \Rightarrow A !$ By i.h.
$\Rightarrow < n - 1 \quad \Psi \vdash \Pi :: A \leftarrow C_0 p$ "
$\Rightarrow < n - 1 \quad \Psi \vdash \Pi$ covers $A$
$\Rightarrow < n \quad \Psi \vdash \Pi$ covers $A$

Subderivation

By lemma 60 (Interpolating With and Exists) (1)

• Cases $\text{Decl}\Pi$, $\text{Decl} \rightarrow$, $\text{Decl} \rightarrow \Pi$, $\text{Decl} \rightarrow k$, $\text{Decl} \times$ Impossible, because in these rules $e$ cannot have the form $\text{case}(e_0, \Pi)$.

• Case $\Psi \vdash \text{case}(e_0, \Pi) \Rightarrow A !$ $\Psi \vdash \Pi :: A \leftarrow C p$ $\Psi \vdash \Pi$ covers $A$

$\Psi \vdash \text{case}(e_0, \Pi) \leftarrow C p$

DecCase

Immediate.

$\square$

F'  Miscellaneous Properties of the Algorithmic System

Lemma 62 (Well-Formed Outputs of Typing).

(Spines) If $\Gamma \vdash \text{s : A q} \gg \text{C p} \vdash \Delta$ or $\Gamma \vdash \text{s : A q} \gg \text{C [p] \vdash} \Delta$

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

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

(Synthesis) If $\Gamma \vdash \text{e \Rightarrow A p} \vdash \Delta$

then $\text{A \vdash p type}$.

Proof. By induction on the given derivation.

• Case $\text{Anno}$: Use Lemma 50 (Typing Extension) and Lemma 40 (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, $\Gamma, \vec{\alpha} : \kappa \vdash [\vec{\alpha}/\alpha]A_0 q$ type.

Now apply the i.h.
Proof of Lemma 62 (Well-Formed Outputs of Typing)

- Case Spine: Use Lemma 41 (Inversion of Principal Typing), (2), Lemma 46 (Checkprop Extension), and Lemma 40 (Extension Weakening for Principal Typing).

- Case SpineRecover:
  By i.h., $\Delta \vdash C \not\in \text{type}$.
  We have as premise $FEV(i,C) = \emptyset$.
  Therefore $\Delta \vdash C \not\in \text{type}$.

- Case SpinePass: By i.h.

- Case EmptySpine: Immediate.

- Case $\to$Spine: Use Lemma 41 (Inversion of Principal Typing) (1), Lemma 50 (Typing Extension), and Lemma 40 (Extension Weakening for Principal Typing).

- Case $\times$Spine: Show that $\alpha_1 \to \alpha_2$ is well-formed, then use the i.h. \hfill \Box

G' Decidability of Instantiation

Lemma 63 (Left Unsolvedness Preservation).

If $\Gamma_0, \alpha, \Gamma_1 \vdash \alpha := \lambda : \kappa \vdash \Delta$ and $\hat{\beta} \in \text{unsolved}(\Gamma_0)$ then $\hat{\beta} \in \text{unsolved}(\Delta)$.

Proof. By induction on the given derivation.

- Case $\Gamma_0 \vdash \tau : \kappa$
  $\begin{array}{c}
  \frac{\Gamma_0, \alpha : \kappa, \Gamma_1 \vdash \alpha := \tau : \kappa \vdash \Gamma_0, \alpha : \kappa = \tau, \Gamma_1}{\Delta} \cdot \text{InstSolve}
  
  \end{array}$
  Immediate, since to the left of $\alpha$, the contexts $\Delta$ and $\Gamma$ are the same.

- Case $\hat{\beta} \in \text{unsolved}(\Gamma'[\alpha : \kappa][\hat{\beta} : \kappa])$
  $\begin{array}{c}
  \frac{\Gamma'[\alpha : \kappa][\hat{\beta} : \kappa] \vdash \alpha := \hat{\beta} : \kappa \vdash \Gamma'[\alpha : \kappa][\hat{\beta} : \kappa] = \alpha}{\Delta} \cdot \text{InstReach}
  
  \end{array}$
  Immediate, since to the left of $\alpha$, the contexts $\Delta$ and $\Gamma$ are the same.

- Case $\Gamma_0, \alpha_2 : \ast, \alpha_1 : \ast, \alpha : \ast = \alpha_1 \oplus \alpha_2, \Gamma_1 \vdash \alpha_1 := \tau_1 : \ast \vdash \Theta \vdash \alpha_2 := [\Theta]\tau_2 : \ast \vdash \Delta$
  $\begin{array}{c}
  \frac{\Gamma_0, \alpha : \ast, \Gamma_1 \vdash \alpha := \tau_1 \oplus \tau_2 : \ast \vdash \Delta}{\Delta} \cdot \text{InstBin}
  
  \end{array}$
  We have $\hat{\beta} \in \text{unsolved}(\Gamma_0)$. Therefore $\hat{\beta} \in \text{unsolved}(\Gamma_0, \alpha_2 : \ast)$.
  Clearly, $\alpha_2 \in \text{unsolved}(\Gamma_0, \alpha_2 : \ast)$.
  We have two subderivations:
  \[
  \begin{align*}
  \Gamma_0, \alpha_2 : \ast, \alpha_1 : \ast, \alpha : \ast = \alpha_1 \oplus \alpha_2, \Gamma_1 \vdash \alpha_1 := A_1 : \ast \vdash \Theta \vdash \alpha_2 := [\Theta]A_2 : \ast \vdash \Delta & \quad (1) \\
  \Theta \vdash \alpha_2 := [\Theta]A_2 : \ast \vdash \Delta & \quad (2)
  \end{align*}
  \]
  By induction on (1), $\hat{\beta} \in \text{unsolved}(\Theta)$.
  Also by induction on (1), with $\hat{\alpha}_2$ playing the role of $\hat{\beta}$, we get $\hat{\alpha}_2 \in \text{unsolved}(\Theta)$.
  Since $\hat{\beta} \in \Gamma_0$, it is declared to the left of $\hat{\alpha}_2$ in $\Gamma_0, \hat{\alpha}_2 : \ast, \alpha_1 : \ast, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1$.
  Hence by Lemma 19 (Declaration Order Preservation), $\hat{\beta}$ is declared to the left of $\hat{\alpha}_2$ in $\Theta$. That is, $\Theta = (\Theta_0, \hat{\alpha}_2 : \ast, \Theta_1)$, where $\hat{\beta} \in \text{unsolved}(\Theta_0)$.
  By induction on (2), $\hat{\beta} \in \text{unsolved}(\Delta)$.

- Case $\Gamma'[\alpha : \kappa] \vdash \alpha := \text{zero} : \kappa \vdash \Gamma'[\alpha : \kappa = \text{zero}]$
  $\begin{array}{c}
  \frac{\Gamma'[\alpha : \kappa] \vdash \alpha := \text{zero} : \kappa \vdash \Gamma'[\alpha : \kappa = \text{zero}]}{\Delta} \cdot \text{InstZero}
  
  \end{array}$
  Immediate, since to the left of $\alpha$, the contexts $\Delta$ and $\Gamma$ are the same.
Lemma 64

Proof of Lemma 63

Proof.

Case

\[
\frac{\Gamma[\delta_1 : N, \lambda : N = \text{succ}(\delta_1)] \vdash \delta_1 := t_1 : N \vdash \Lambda}{\Gamma[\lambda : N] \vdash \lambda := \text{succ}(t_1) : N \vdash \Delta \quad \text{InstSucc}}
\]

We have \(\hat{\beta} \in \text{unsolved}(\Gamma_0)\). Therefore \(\hat{\beta} \in \text{unsolved}(\Gamma_0, \delta_1 : N)\). By i.h., \(\hat{\beta} \in \text{unsolved}(\Delta)\).

\[\square\]

Lemma 64 (Left Free Variable Preservation). If \(\Gamma_0, \delta : \kappa, \Gamma_1 \vdash \delta := t : \kappa \vdash \Delta \) and \(\Gamma \vdash s : \kappa'\) and \(\alpha \notin \text{FV}(\Gamma')\) and \(\beta \notin \text{FV}(\Gamma')\), then \(\hat{\beta} \notin \text{FV}(\Delta')\).

Proof. By induction on the given instantiation derivation.

Case

\[
\frac{\Gamma_0 \vdash \tau : \kappa \quad \Gamma_0, \delta : \kappa, \Gamma_1 \vdash \delta := t : \kappa \vdash \Gamma \quad \Delta}{\Gamma \vdash \alpha := \kappa \vdash \Delta \quad \text{InstReach}}
\]

We have \(\hat{\alpha} \notin \text{FV}(\Gamma')\). Since \(\Delta\) differs from \(\Gamma\) only in \(\hat{\alpha}\), it must be the case that \(\Gamma_0 \sigma = \Delta_0 \sigma\). It is given that \(\hat{\beta} \notin \text{FV}(\Gamma_0 \sigma)\), so \(\hat{\beta} \notin \text{FV}(\Delta_0 \sigma)\).

Case

\[
\frac{\Gamma' \vdash \tau' : \kappa'}{\Gamma{\delta_2} : *, \delta_1 : *, \delta : \kappa = \delta_1 \otimes \delta_2} \vdash \delta_1 := \tau_1 : * \vdash \Theta \quad \Theta \vdash \delta_2 := [\Theta] \tau_2 : * \vdash \Delta \quad \text{InstBin}
\]

We have \(\Gamma' \vdash \sigma\) type and \(\delta \notin \text{FV}(\Gamma')\) and \(\hat{\beta} \notin \text{FV}(\Gamma')\).

By weakening, we get \(\Gamma' \vdash \sigma : \kappa'\); since \(\delta \notin \text{FV}(\Gamma')\) and \(\Gamma'\) only adds a solution for \(\hat{\delta}\), it follows that \(\Gamma' \sigma = \Gamma \sigma\).

Therefore \(\hat{\alpha}_1 \notin \text{FV}(\Gamma'_0 \sigma)\) and \(\hat{\alpha}_2 \notin \text{FV}(\Gamma'_0 \sigma)\) and \(\hat{\beta} \notin \text{FV}(\Gamma'_0 \sigma)\).

Since we have \(\hat{\beta} \notin \Gamma_0\), we also have \(\hat{\beta} \notin (\Gamma_0, \hat{\delta}_2 : *)\).

By induction on the first premise, \(\hat{\beta} \notin \text{FV}(\Theta \sigma)\).

Also by induction on the first premise, with \(\hat{\alpha}_2\) playing the role of \(\hat{\beta}\), we have \(\hat{\alpha}_2 \notin \text{FV}(\Theta \sigma)\).

Note that \(\hat{\alpha}_2 \notin \text{unsolved}(\Gamma_0, \delta_2 : *)\).

By Lemma 63 (Left Unsolvedness Preservation), \(\hat{\alpha}_2 \notin \text{unsolved}(\Theta)\).

Therefore \(\Theta\) has the form \((\Theta_0, \hat{\delta}_2 : * \Theta_1)\).

Since \(\hat{\beta} \neq \hat{\alpha}_2\), we know that \(\hat{\beta}\) is declared to the left of \(\hat{\alpha}_2\) in \((\Gamma_0, \hat{\delta}_2 : *)\), so by Lemma 19 (Declaration Order Preservation), \(\hat{\beta}\) is declared to the left of \(\hat{\alpha}_2\) in \(\Theta\). Hence \(\hat{\beta} \in \Theta_0\).

Furthermore, by Lemma 42 (Instantiation Extension), we have \(\Gamma' \quad \rightarrow \Theta\).

Then by Lemma 35 (Extension Weakening (Sorts)), we have \(\Delta \vdash \sigma : \kappa'\).

Using induction on the second premise, \(\hat{\beta} \notin \text{FV}(\Delta \sigma)\).

Case

\[
\frac{\Gamma'[\alpha : N] \vdash \alpha := \text{zero} : N \vdash \Gamma'[\alpha : N = \text{zero}] \quad \text{InstZero}}{\Gamma \vdash \quad \Delta}
\]

We have \(\hat{\alpha} \notin \text{FV}(\Gamma')\). Since \(\Delta\) differs from \(\Gamma\) only in \(\hat{\alpha}\), it must be the case that \(\Gamma' \sigma = \Delta_0 \sigma\). It is given that \(\hat{\beta} \notin \text{FV}(\Gamma' \sigma)\), so \(\hat{\beta} \notin \text{FV}(\Delta_0 \sigma)\).

Case

\[
\frac{\Gamma'[\delta_1 : N, \lambda : N = \text{succ}(\delta_1)] \vdash \delta_1 := t_1 : N \vdash \Delta \quad \text{InstSucc}}{\Gamma'[\lambda : N] \vdash \lambda := \text{succ}(t_1) : N \vdash \Delta \quad \text{InstSucc}}
\]
Lemma 65 (Instantiation Size Preservation). If $\Gamma_0, \alpha, \Gamma_1 \vdash \alpha := \tau : \kappa \vdash \Delta$ and $\Gamma \vdash s : \kappa'$ and $\alpha \notin \text{FV}(\Gamma)[s]$, then $|\Gamma|s = |\Delta|s$, where $|C|$ is the plain size of the term $C$.

Proof. By induction on the given derivation.

- Case

  $\Gamma \vdash \tau : \kappa$

  $\Gamma_0, \alpha : \kappa, \Gamma_1 \vdash \alpha := \tau : \kappa \vdash \Gamma_0, \alpha : \kappa = \tau, \Gamma_1$

  Since $\Delta$ differs from $\Gamma$ only in solving $\alpha$, and we know $\alpha \notin \text{FV}(\Gamma)[\sigma]$, we have $|\Delta|\sigma = |\Gamma|\sigma$; therefore $|\Delta|\sigma = |\Gamma|\sigma$.

- Case

  $\Gamma'[\alpha : \kappa] \vdash \alpha := \text{zero} : \kappa \vdash \Gamma'[\alpha : \kappa = \alpha]$ \hspace{1cm} \text{InstZero}

  Similar to the InstSolve case.

- Case

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

  $\Gamma'[\alpha : \kappa] \vdash \hat{\alpha} := \hat{\beta} : \kappa \vdash \Gamma'[\hat{\alpha} : \kappa = \hat{\alpha}]$ \hspace{1cm} \text{InstReach}

  Here, $\Delta$ differs from $\Gamma$ only in solving $\hat{\beta}$ to $\hat{\alpha}$. However, $\hat{\alpha}$ has the same size as $\hat{\beta}$, so even if $\hat{\beta} \in \text{FV}(\Gamma)[\sigma]$, we have $|\Delta|\sigma = |\Gamma|\sigma$.

- Case

  $\Gamma'$

  $\Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha}_1 : \kappa = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 : \tau_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := \{\Theta\}\tau_2 : \star \vdash \Delta$

  We have $\Gamma \vdash \tau_1 : \star \vdash \Theta \vdash \hat{\alpha}_2 := \{\Theta\}\tau_2 : \star \vdash \Delta$

  Since $\hat{\alpha}_1, \hat{\alpha}_2 \notin \text{dom}(\Gamma')$, we have $\hat{\alpha}_1, \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\Gamma)[\sigma]$.

  By Lemma 22 (Deep Evar Introduction), $\Gamma'[\hat{\alpha} : \kappa] \rightarrow \Gamma'$.

  By Lemma 35 (Extension Weakening (Sorts)), $\Gamma' \vdash \tau_1 : \star$.

  Since $\hat{\alpha} \notin \text{FV}(\Gamma)[\sigma]$, 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 19 (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 63 (Left Unsolvedness Preservation), since $\hat{\alpha}_2 \in \text{unsolved}(\Gamma')$, it is unsolved in $\Theta$: that is, $\Theta = (\Theta_0, \hat{\alpha}_2 : \star, \Theta_1)$.

  By Lemma 42 (Instantiation Extension), we have $\Gamma' \rightarrow \Theta$.

  By Lemma 35 (Extension Weakening (Sorts)), $\Theta \vdash \tau_2 : \star \vdash \Delta$.

  Since $\hat{\alpha}_2 \notin \text{FV}(\Gamma)[\sigma]$, Lemma 64 (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 64 (Left Free Variable Preservation)
Proof of Lemma 65 (Instantiation Size Preservation) lem:instantiation-size-preservation

Case $\Gamma' \vdash \forall x \in \Delta . \exists \alpha . \Gamma' \vdash \alpha : \Delta$

- By induction on the derivation of $\Gamma'$. 
  - Either there exists $\alpha / \Delta \vdash \forall x \in \Delta . \exists \alpha . \Gamma' \vdash \alpha : \Delta$
  - By Lemma 22 (Deep Evar Introduction) 
  - By Lemma 35 (Extension Weakening (Sorts))

Lemma 66 (Decidability of Instantiation). If $\Gamma = \Gamma_0[\alpha : \kappa']$ and $\Gamma \vdash t : \kappa$ such that $\Gamma[t] = t$ and $\alpha \not\in FV(t)$, then:

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

Proof. By induction on the derivation of $\Gamma \vdash t : \kappa$.

- Case $\Gamma, \beta : \kappa' \vdash u : \kappa$
  - By inversion, $\beta : \kappa' \in \Gamma$, but $\Gamma[\beta] = \beta$ is given, so this case is impossible.

- Case $\Gamma \vdash \tau_1 : \kappa \vdash \tau_2 : \kappa$
  - By definition of substitution, $\Gamma[\tau_1 \vdash \tau_2] = \tau_1 \vdash \tau_2$. Since $\Gamma[\tau_1 \vdash \tau_2] = \tau_1 \vdash \tau_2$, we have $[\Gamma] \tau_1 = \tau_1$ and $[\Gamma] \tau_2 = \tau_2$. 
  - By weakening, $\Gamma_1, \delta_2 : \kappa, \delta_1 : \kappa, \delta : \kappa \vdash \delta_1 \vdash \delta_2 : \kappa$. 
  - Since $\Gamma \vdash \tau_1 : \kappa$ and $\Gamma \vdash \tau_2 : \kappa$, we have $\delta_1, \delta_2 \not\in FV(\Gamma(\tau_1)) \cup FV(\Gamma(\tau_2))$. 
  - By i.h., either there exists $\Theta$ s.t. $\Gamma_1, \delta_2 : \kappa, \delta_1 : \kappa, \delta : \kappa \vdash \Theta$, or not. 
  - If not, then no derivation by $\text{InstBin}$ exists.
Proof. By induction on then \( \text{Lemma 68 (Separation of Auxiliary Judgments)} \) exists.

Otherwise, there exists such a \( \Delta \). By rule \text{InstBin} \) we have \( \Gamma \vdash \alpha := t : \kappa \top \Delta \).

- Case \( \Gamma \vdash \text{zero} : \mathbb{N} \) \( \text{ZeroSort} \)

If \( \kappa' \neq \mathbb{N} \), then no rule matches and no derivation exists. Otherwise, apply rule \text{InstSolve}.

- Case \( \Gamma \vdash t_0 : \mathbb{N} \) \( \text{SuccSort} \)

If \( \kappa' \neq \mathbb{N} \), then no rule matches and no derivation exists. Otherwise:

If \( \Gamma, \vdash \text{succ}(t_0) : \mathbb{N} \), then we have a derivation by \text{InstSolve}.

If not, the only other rule whose conclusion matches \text{succ}(t_0) \) is \text{InstSucc}. The remainder of this case is similar to the \text{BinSort} case, but shorter. \( \square \)

H’ Separation

Lemma 67 (Transitivity of Separation).
If \( (\Gamma_L \ast \Gamma_R) \xrightarrow{w} (\Theta_L \ast \Theta_R) \) and \( (\Theta_L \ast \Theta_R) \xrightarrow{w} (\Delta_L \ast \Delta_R) \)
then \( (\Gamma_L \ast \Gamma_R) \xrightarrow{w} (\Delta_L \ast \Delta_R) \).

Proof.

\[
(\Gamma_L \ast \Gamma_R) \xrightarrow{w} (\Theta_L \ast \Theta_R) \quad \text{Given}
\]
\[
(\Gamma_L, \Gamma_R) \rightarrow (\Theta_L, \Theta_R) \quad \text{By Definition 5}
\]
\[
\Gamma_L \subseteq \Theta_L \quad \text{and} \quad \Gamma_R \subseteq \Theta_R
\]

\[
(\Theta_L \ast \Theta_R) \xrightarrow{w} (\Delta_L \ast \Delta_R) \quad \text{Given}
\]
\[
(\Theta_L, \Theta_R) \rightarrow (\Delta_L, \Delta_R) \quad \text{By Definition 5}
\]
\[
\Theta_L \subseteq \Delta_L \quad \text{and} \quad \Theta_R \subseteq \Delta_R
\]

\[
(\Gamma_L, \Gamma_R) \rightarrow (\Delta_L, \Delta_R) \quad \text{By Lemma 32 (Extension Transitivity)}
\]
\[
\Gamma_L \subseteq \Delta_L \quad \text{and} \quad \Gamma_R \subseteq \Delta_R
\]

\[
(\Gamma_L \ast \Gamma_R) \xrightarrow{w} (\Delta_L \ast \Delta_R) \quad \text{By Definition 5} \]

\( \square \)

Lemma 68 (Separation Truncation).
If \( H \) has the form \( \alpha : \kappa \) or \( \triangleright \alpha \) or \( \triangleright \gamma \)
and \( (\Gamma_L \ast (\Gamma_R, H)) \xrightarrow{w} (\Delta_L \ast \Delta_R) \)
then \( (\Gamma_L \ast \Gamma_R) \xrightarrow{w} (\Delta_L \ast \Delta_R) \) where \( \Delta_R = (\Delta'_R, H, \Delta) \).

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

If \( \Delta_R = (\ldots, H) \), we have \( (\Gamma_L \ast \Gamma_R, H) \xrightarrow{w} (\Delta_L \ast (\Delta, H)) \), and inversion on \( \xrightarrow{w} \text{Uvar} \) (if \( H \) is \( \alpha : \kappa \), or the corresponding rule for other forms) gives the result (with \( \Theta = : \)).

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 \( \Theta = (\Theta', \Delta') \). \( \square \)

Lemma 69 (Separation for Auxiliary Judgments).

(i) If \( \Gamma_L \ast \Gamma_R \vdash \sigma = \tau : \kappa \top \Delta \)
and \( \text{FEV} (\sigma) \cup \text{FEV} (\tau) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \xrightarrow{w} (\Delta_L \ast \Delta_R) \).
\( (i) \) If \( \Gamma_L \Gamma_R \vdash P \) true \( \vdash \Delta \)
and \( FEV(P) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

\( (ii) \) If \( \Gamma_L \Gamma_R \vdash P \) \( \vdash \Delta \)
and \( FEV(\sigma) \cap FEV(\tau) = \emptyset \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

\( (iii) \) If \( \Gamma_L \Gamma_R \vdash P \) \( \vdash \Delta \)
and \( FEV(\sigma, \tau) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

\( (iv) \) If \( \Gamma_L \Gamma_R \vdash P \) \( \vdash \Delta \)
and \( FEV(P) = \emptyset \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

\( (v) \) If \( \Gamma_L \Gamma_R \vdash P \) \( \vdash \Delta \)
and \( FEV(P) \cup FEV(Q) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

\( (vi) \) If \( \Gamma_L \Gamma_R \vdash A \) \( \vdash \Delta \)
and \( FEV(A) \cup FEV(B) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

\( (vii) \) If \( \Gamma_L \Gamma_R \vdash A \) \( \vdash \Delta \)
and \( FEV(A) \cup FEV(B) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \Delta_R) \) and \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R \).

Proof. For case \( (i) \), By induction on the derivation of the given checkeq judgment. Cases \( \text{CheckeqVar} \) and \( \text{CheckeqZero} \) are immediate \( (\Delta_L = \Gamma_L \wedge \Delta_R = \Gamma_R) \). For case \( \text{CheckeqSucc} \) apply the i.h. For cases \( \text{CheckeqInstL} \) and \( \text{CheckeqInstR} \) use the i.h. \( (v) \). For case \( \text{CheckeqBin} \) use reasoning similar to that in the \( \Delta \) case (transitivity of separation, and applying \( \Theta \) in the second premise).

Part (ii), checkprop: Use the i.h. \( (i) \).

Part (iii), elimprop: Cases \( \text{ElimeqUVarRef} \), \( \text{ElimeqUnit} \), and \( \text{CheckeqZero} \) are immediate \( (\Delta_L = \Gamma_L \wedge \Delta_R = \Gamma_R) \). Cases \( \text{ElimeqUVarL} \), \( \text{ElimeqUVarR} \), \( \text{ElimeqBot} \), and \( \text{ElimeqClash} \) are impossible (we have \( \Delta \), not \( \bot \)). For case \( \text{ElimeqSucc} \) apply the i.h. The case for \( \text{ElimeqBin} \) is similar to the case \( \text{CheckeqBin} \) in part \( (i) \). For cases \( \text{ElimeqUVarL} \) and \( \text{ElimeqUVarR} \) \( \Delta = (\Gamma_L, \Gamma_R, \alpha : \tau) \) which, since \( FEV(\tau) \subseteq \text{dom}(\Gamma_R) \), ensures that \( (\Gamma_L \Gamma_R) \rightarrow^L \Delta_L \Delta_R, \alpha = \tau \).

Part (iv), elimprop: Use the i.h. \( (iii) \).

Part (v), instjudg:

- Case \( \text{InstSolve} \) Here, \( \Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1) \) and \( \Delta = (\Gamma_0, \alpha : \kappa = \tau, \Gamma_1) \). We have \( \alpha \in \text{dom}(\Gamma_R) \), so the declaration \( \alpha : \kappa \) is in \( \Gamma_R \). Since \( FEV(\tau) \subseteq \text{dom}(\Gamma_R) \), the context \( \Delta \) maintains the separation.

- Case \( \text{InstReach} \) Here, \( \Gamma = \Gamma_0[\beta : \kappa] \) and \( \Delta = \Gamma_0[\beta : \kappa] \). We have \( \alpha \in \text{dom}(\Gamma_R) \), 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 \( FEV(\beta) \subseteq \text{dom}(\Gamma_R) \). Both declarations are in \( \Gamma_R \), so the context \( \Delta \) maintains the separation.

- Case \( \text{InstZero} \) In this rule, \( \Delta \) is the same as \( \Gamma \) except for a solution zero, which doesn’t violate separation.

- Case \( \text{InstSucc} \) The result follows by i.h., taking care to keep the declaration \( \alpha_1 : N \) on the right when applying the i.h., even if \( \alpha : N \) is the leftmost declaration in \( \Gamma_R \), ensuring that \( \text{succ}(\alpha_1) \) does not violate separation.

- Case \( \text{InstBin} \) As in the \( \text{InstSucc} \) case, the new declarations should be kept on the right-hand side of the separator. Otherwise the case is straightforward (using the i.h. twice and transitivity).

Part (vi), propequivjudg: Similar to the \( \text{CheckeqBin} \) case of part \( (i) \), using the i.h. \( (i) \).

Part (vii), equivjudg:

- Cases \( \equiv \text{Var} \equiv \text{Exvar} \equiv \text{Unit} \) Immediate \( (\Delta_L = \Gamma_L \wedge \Delta_R = \Gamma_R) \).

- Case \( \equiv \) Similar to the case \( \text{CheckeqBin} \) in part \( (i) \).

- Cases \( \equiv \) Similar to the case \( \text{CheckeqBin} \) in part \( (i) \).
Lemma 70 (Separation for Subtyping). If \( \Gamma_L \ast \Gamma_R \vdash A \equiv_\pm B \vdash \Delta \)
and \( \text{FEV}(A) \subseteq \text{dom}(\Gamma_R) \)
and \( \text{FEV}(B) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash \) \( (\Delta_L \ast \Delta_R) \).

Proof. By induction on the given derivation. In the \( \equiv_\pm \) case, use Lemma 69 (Separation for Auxiliary Judgments) (vii). Otherwise, the reasoning needed follows that used in the proof of Lemma 71 (Separation—Main).

Lemma 71 (Separation—Main).

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

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

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

(Match) If \( \Gamma_L \ast \Gamma_R \vdash \Pi :: \vec{A} \ll C \vdash \Delta \)
and \( \text{FEV}(\vec{A}) = \emptyset \)
and \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \)
then \( \Delta = (\Delta_L \ast \Delta_R) \) and \( (\Gamma_L \ast \Gamma_R) \vdash (\Delta_L \ast \Delta_R) \).

(Match Elim.) If \( \Gamma_L \ast \Gamma_R \vdash \Pi :: \vec{A} \ll C \vdash \Delta \)
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 \( (\Gamma_L \ast \Gamma_R) \vdash (\Delta_L \ast \Delta_R) \).

Proof. By induction on the given derivation.
First, the (Match) judgment part, giving only the cases that motivate the side conditions:

- **Case MatchBase**: Here we use the i.h. (Checking), for which we need \( \text{FEV}(C) \subseteq \text{dom}(\Gamma_R) \).

- **Case Match\( \neg \)**: Here we use the i.h. (Match Elim.), which requires that \( \text{FEV}(P) = \emptyset \), which motivates \( \text{FEV}(\vec{A}) = \emptyset \).

- **Case MatchNeg**: In its premise, this rule appends a type \( A \in \vec{A} \) to \( \Gamma_R \) and claims it is principal \( (z : A?) \), which motivates \( \text{FEV}(\vec{A}) = \emptyset \).

Similarly, (Match Elim.):

- **Case MatchUnify**: Here we use Lemma 69 (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]** \[ \subseteq \]**: 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 71 (Separation—Main) lem:separation-main
Proof of Lemma 71 (Separation—Main)

Case
\[ \Gamma_L \cup \Gamma_R \vdash e : A \quad \vdash \frac{A \vdash \emptyset}{\emptyset} \quad \text{EmptySpine} \]

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 \( \text{FEV}(C) \subseteq \text{dom}(\Delta_R) \).

Case
\[ \Gamma_L \cup \Gamma_R \vdash e \Rightarrow A \quad \vdash \frac{A \vdash \emptyset}{\emptyset} \quad \text{Sub} \]

By i.h., \( \Theta = (\Theta_L \cup \Theta_R) \) and \( (\Gamma_L \cup \Gamma_R) \sim \Theta \).
By Lemma 70 (Separation for Subtyping), \( \Delta = (\Delta_L \cup \Delta_R) \) and \( (\Theta_L \cup \Theta_R) \sim \Theta \).
By Lemma 67 (Transitivity of Separation), \( (\Gamma_L \cup \Gamma_R) \sim \Theta \).

Case
\[ \Gamma \vdash \text{At type} \quad \vdash \frac{[\Gamma]A ! \vdash \Delta}{\emptyset} \quad \text{Anno} \]

By i.h.; since \( \text{FEV}(A) = \emptyset \), the condition on the (Checking) part is trivial.

Case
\[ \Gamma[\varepsilon : \ast] \vdash () \Leftarrow \varepsilon \Rightarrow \emptyset \quad \text{H\varepsilon} \]

Adding a solution with a ground type cannot destroy separation.

Case
\[ \forall \text{chk-I} \quad \Gamma_L, \Gamma_R, \alpha : \kappa \vdash \forall \alpha : \kappa \quad \vdash \alpha_0 \quad \vdash \Delta, \alpha : \kappa, \Theta \quad \emptyset \quad \text{VI} \]

\text{FEV}(\forall \alpha : \kappa, \alpha_0) \subseteq \text{dom}(\Gamma_R) \quad \text{Given}
\text{FEV}(\alpha_0) \subseteq \text{dom}(\Gamma_R, \alpha : \kappa) \quad \text{From definition of FEV}
(\Delta, \alpha : \kappa, \Theta) = (\Delta_L \cup \Delta_R) \quad \text{By i.h.}
(\Gamma_L \cup \Gamma_R) \sim \Theta \quad \text{By Lemma 68 (Separation Truncation)}
\Delta_L = (\Delta_L, \alpha : \kappa, \Theta) \quad \text{By definition of } \sim
\Delta_R = (\Delta_R, \alpha : \kappa, \Theta) \quad \text{By above equation}
\Delta = (\Delta_L, \Delta_R) \quad \alpha \text{ not multiply declared}

Case
\[ \Gamma_L, \Gamma_R, \varepsilon : s \vdash [\varepsilon / \alpha]A_0 \Rightarrow C \quad \vdash \Delta \quad \emptyset \quad \text{VSpine} \]

\text{FEV}(\forall \alpha : \kappa, \alpha_0) \subseteq \text{dom}(\Gamma_R) \quad \text{Given}
\text{FEV}([\varepsilon / \alpha]A_0) \subseteq \text{dom}(\Gamma_R, \varepsilon : \kappa) \quad \text{From definition of FEV}
\Delta = (\Delta_L \cup \Delta_R) \quad \text{By i.h.}
(\Gamma_L \cup \Gamma_R) \sim \Theta \quad \text{By Lemma 68 (Separation Truncation)}
\text{FEV}(C) \subseteq \text{dom}(\Delta_R) \quad \text{By definition of } \sim
\text{dom}(\Gamma_R, \varepsilon : \kappa) \subseteq \text{dom}(\Delta_R) \quad \text{By Definition 5}
\text{dom}(\Gamma_R) \subseteq \text{dom}(\Delta_R) \quad \text{By Definition 5}
\text{dom}(\Gamma_R) \cup (\varepsilon) \subseteq \text{dom}(\Delta_R) \quad \text{By definition of dom(-)}
\text{Property of } \subseteq
(\Gamma_L, \Gamma_R) \sim \Theta \quad \text{By Lemma 50 (Typing Extension)}
(\Gamma_L \cup \Gamma_R) \sim (\Delta_L \cup \Delta_R) \quad \text{By Definition 5}
• Case $e$ not a case

\[ \Gamma_L \vdash _I \Gamma_R, P \vdash \Theta \quad \Theta \vdash \lx{\Theta}{A_0} \vdash \Delta \]

\[ \Gamma_L \vdash e \iff A_0 \land P \vdash \Delta \]

- $\Gamma_L \vdash \lx{A_0 \land P} p \text{ type}$
  - Given
- $\Gamma_L \vdash \lx{P \prop}$
  - By inversion
- $\Gamma_L \vdash \lx{A_0} p \text{ type}$
  - By inversion

\[ \text{FEV}(A_0 \land P) \subseteq \text{dom}(\Gamma_R) \]

\[ \text{FEV}(P) \subseteq \text{dom}(\Gamma_R) \]

\[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \]

\[ \Theta = (\Theta_L \bullet \Theta_R) \]

\[ (\Gamma_L \bullet \Gamma_R) \not\vdash (\Theta_L \bullet \Theta_R) \]

\[ \text{FEV}(A_0) \subseteq \text{dom}(\Gamma_R) \]

\[ \text{dom}(\Gamma_R) \subseteq \text{dom}(\Theta_R) \]

\[ \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R) \]

\[ \text{FEV}(\Theta A_0) \subseteq \text{dom}(\Theta_R) \]

\[ \Theta \vdash \lx{\Theta}{A_0} \vdash \Delta \]

\[ \Theta \vdash |\Theta| A_0 \vdash \Delta \]

\[ \Delta = (\Delta_L \bullet \Delta_R) \]

\[ (\Theta_L \bullet \Theta_R) \not\vdash (\Delta_L \bullet \Delta_R) \]

\[ (\Gamma_L \bullet \Gamma_R) \not\vdash (\Delta_L \bullet \Delta_R) \]

- $\triangleright$

\[ \text{By Characterization of Separation} \quad \text{(ii)} \]

\[ \triangleright \]

\[ \text{By Inversion} \quad \text{(2)} \]

\[ \Delta = (\Delta_L \bullet \Delta_R) \]

\[ (\Gamma_L \bullet \Gamma_R) \not\vdash (\Delta_L \bullet \Delta_R) \]

\[ (\Gamma_L \bullet \Gamma_R) \not\vdash (\Delta_L \bullet \Delta_R) \]

\[ \Gamma_L, \Gamma_R, \bullet p \vdash \Theta \]

\[ \text{Blurring separation} \]

\[ \Gamma_L \vdash \lx{P \supset A_0} ! \text{ type} \]

\[ \Gamma_L, \Gamma_R \vdash A_0 ! \text{ type} \]

\[ \Theta \vdash |\Theta| A_0 ! \text{ type} \]

\[ \text{FEV}(A_0) = \emptyset \]

\[ \text{FEV}(A_0) \subseteq \text{dom}(\Theta_R) \]

\[ (\Delta, \bullet p, \bullet p') = (\Delta_L \bullet \Delta_R') \]

\[ (\Theta_L \bullet \Theta_R) \not\vdash (\Delta_L \bullet \Delta_R') \]

\[ (\Gamma_L \bullet \Gamma_R, \bullet p) \not\vdash (\Delta_L \bullet \Delta_R') \]

\[ \Delta_L' = (\Delta_L, \bullet p, \ldots) \]

\[ \Delta' = (\Delta_L, \Delta_R) \]

\[ \Delta = (\Delta_L, \Delta_R) \]

\[ \Delta = (\Delta_L, \Delta_R) \]

\[ \triangleright \text{ Similar to the \triangleright \text{ case} \]

• Case $\Gamma_L \vdash P \true \vdash \Theta \quad \Theta \vdash \lx{\Theta}{A_0} \vdash P \gg C \vdash \Delta$
Proof of Lemma 71 (Separation—Main)

\[ \Gamma \vdash R \vdash \{P \supset A \} \vdash \text{p type} \quad \text{Given} \]
\[ \Gamma \vdash R \vdash \{P \dashv \vdash \} \vdash \text{prop} \quad \text{By inversion} \]
\[ \Gamma, \Delta \vdash R \vdash \{P \} \vdash \text{Subderivation} \]
\[ \Theta = (\Theta \vdash R \vdash \{P \}) \quad \text{By Lemma 66} \]
\[ (\Gamma \vdash R \vdash \{P \}) \vdash \Theta \quad \text{Proof for Auxiliary Judgments} \]
\[ \Theta \vdash e : \{Q \} \vdash \text{Subderivation} \]
\[ (\Delta \vdash \{P \}, \{Q \}) = (\Delta \vdash \{Q \}) \quad \text{By i.h.} \]
\[ (\Theta \vdash \{P \}) \vdash \Theta \quad \text{Substitution} \]
\[ (\Gamma \vdash R \vdash \{P \}) \vdash \Gamma \quad \text{Substitution} \]
\[蜥蜴 (\Gamma \vdash R \vdash \{P \}) \vdash \Gamma \quad \text{Substitution} \]

Case
\[ \Gamma, \Delta \vdash \{P \} \vdash \text{p type} \quad \text{By weakening and Definition 4} \]
\[ \Gamma \vdash \{P \} \vdash \text{p type} \quad \text{Subderivation} \]
\[ \text{FEV}(\Delta \vdash \{P \}) \subseteq \text{dom}(\Gamma) \quad \text{By i.h.} \]

\[ \Gamma \vdash \{P \} \vdash \{Q \} \vdash \text{p type} \quad \text{By weakening and Definition 4} \]
\[ \Gamma \vdash \{P \} \vdash \text{p type} \quad \text{Subderivation} \]
\[ \text{FEV}(\Delta \vdash \{P \}) \subseteq \text{dom}(\Gamma) \quad \text{By i.h.} \]

Case
\[ \Gamma \vdash \{P \} \vdash \text{p type} \quad \text{Subderivation} \]
\[ \Gamma \vdash \{P \} \vdash \text{p type} \quad \text{Subderivation} \]
\[ \text{FEV}(\Delta \vdash \{P \}) \subseteq \text{dom}(\Gamma) \quad \text{By i.h.} \]

We have \((\Gamma \vdash R \vdash \{P \}) = \Gamma \vdash \{P \} \vdash \{Q \} \quad \text{Substitution} \]

\[ \Gamma \vdash \{P \} \vdash \{Q \} \vdash \text{p type} \quad \text{Subderivation} \]
\[ \text{FEV}(\Delta \vdash \{P \}) \subseteq \text{dom}(\Gamma) \quad \text{By i.h.} \]

Then the input context in the premise has the following form:
\[ \Gamma \vdash \{P \} \vdash \{Q \} \vdash \{R \} \vdash \{S \} \]

Let us separate this context at the same point as \(\Gamma \vdash \{P \} \vdash \{Q \} \vdash \{R \} \vdash \{S \} \), that is, after \(\Gamma \) and before \(\Gamma \), and call the resulting right-hand context \(\Gamma' \). That is,
\[ \Gamma \vdash \{P \} \vdash \{Q \} \vdash \{R \} \vdash \{S \} \]

\[ \text{FEV}(\Delta \vdash \{P \}) \subseteq \text{dom}(\Gamma) \quad \text{Substitution} \]
\[ \Gamma \vdash \{P \} \vdash \{Q \} \vdash \{R \} \vdash \{S \} \quad \text{Substitution} \]
\[ \text{FEV}(\Delta \vdash \{P \}) \subseteq \text{dom}(\Gamma) \quad \text{Substitution} \]

\[ \Delta \vdash \{P \} \vdash \{Q \} \vdash \{R \} \vdash \{S \} \quad \text{Substitution} \]

\[ \Delta \vdash \{P \} \vdash \{Q \} \vdash \{R \} \vdash \{S \} \quad \text{Substitution} \]
Proof of Lemma 71 (Separation—Main) lem:separation-main

- **Case**
  \[ \Gamma \vdash e \Rightarrow A \triangleright p \downarrow \Theta \quad \Theta \vdash s : [\Theta] A \triangleright p \Rightarrow C [q] \downarrow \Delta \]

  Use the i.h. and Lemma 57 (Transitivity of Separation), with Lemma 88 (Well-formedness of Algorithmic Typing) and Lemma 12 (Right-Hand Substitution for Typing).

- **Case**
  \[ \Gamma \vdash s : A \triangleright C \triangleright [f] \downarrow \Delta \quad FEV([\Delta][C]) = \emptyset \]

  Use the i.h.

- **Case**
  \[ \Gamma \vdash s : A \triangleright C [q] \downarrow \Delta \quad ((p = f) \text{ or } (q = f) \text{ or } (FEV([\Delta][C]) \neq \emptyset)) \]

  Use the i.h.

- **Case**
  \[ \Gamma_L + \Gamma_R \vdash e \Leftarrow A_1 \triangleright p \downarrow \Theta \quad \Theta \vdash s : [\Theta] A_2 \triangleright p \Rightarrow C [q] \downarrow \Delta \]

  Use the i.h. (inverting \(\Gamma \vdash (A_1 + A_2) \triangleright p \downarrow \Delta\)).

- **Case**
  \[ \Gamma \vdash e \Leftarrow A_k \triangleright p \downarrow \Delta \]

  Use the i.h. (inverting \(\Gamma \vdash (A_1 + A_2) \triangleright p \downarrow \Delta\)).

- **Case**
  \[ \Gamma \vdash e_1 \Leftarrow A_1 \triangleright p \downarrow \Theta \quad \Theta \vdash e_2 : [\Theta] A_2 \triangleright p \Rightarrow \Delta \]

  Use the i.h. (inverting \(\Gamma \vdash \langle e_1, e_2 \rangle \Leftarrow A_1 \times A_2 \triangleright p \downarrow \Delta\)).
Proof of Lemma 71 (Separation—Main)

- **Case** \( \Gamma[\alpha_2: \star, \alpha_1: \star, \alpha: = \alpha_1 \times \alpha_2] \vdash e_1 \equiv \alpha_1 \vdash \Theta \quad \Theta \vdash e_2 \equiv [\Theta] \alpha_2 \vdash \Delta \)
  \( \times \)

We have \( \Gamma_L \ast \Gamma_R \) = \( \Gamma_0[\alpha : \star] \). We also have \( \text{FEV}(\alpha) \subseteq \text{dom}(\Gamma_R) \). Therefore \( \alpha \in \text{dom}(\Gamma_R) \) and

\[
\Gamma_0[\alpha : \star] = \Gamma_L, \Gamma_2, \alpha : \star, \Gamma_3
\]

where \( \Gamma_R = (\Gamma_2, \alpha : \star, \Gamma_3) \).

Then the input context in the premise has the following form:

\[
\Gamma_0[\alpha_1: \star, \alpha_2: \star, \alpha: = \alpha_1 \times \alpha_2] = (\Gamma_L, \Gamma_2, \alpha_1: \star, \alpha_2: \star, \alpha: = \alpha_1 \times \alpha_2, \Gamma_3)
\]

Let us separate this context at the same point as \( \Gamma_0[\alpha : \star] \), that is, after \( \Gamma_L \) and before \( \Gamma_2 \), and call the resulting right-hand context \( \Gamma' \):

\[
\Gamma_0[\alpha_1: \star, \alpha_2: \star, \alpha: = \alpha_1 \times \alpha_2] = \Gamma_L \ast (\Gamma_2, \alpha_1: \star, \alpha_2: \star, \alpha: = \alpha_1 \times \alpha_2, \Gamma_3)
\]

- **Case** \( \Gamma[\alpha_2: \star, \alpha_1: \star, \alpha: = \alpha_1 \times \alpha_2] \vdash e \equiv \alpha_1 \vdash \Delta \)
  \( \times \)

By Lemma 22 (Deep Evar Introduction) (i), (ii) and the definition of separation, we can show

\[
\Gamma_L \ast (\Gamma_2, \alpha : \star, \Gamma_3) \quad \Gamma_L \ast (\Gamma_2, \alpha_1: \star, \alpha_2: \star, \alpha : = \alpha_1 \times \alpha_2, \Gamma_3)
\]

\( \times \)

By above equalities

**Case** \( \Delta = (\Delta_1, \Delta_R) \)

By i.h.

\( \Gamma_R = (\Gamma_2, \alpha : \star, \Gamma_3) \)

Above

\( \Gamma'_R = (\Gamma_2, \alpha_1: \star, \alpha_2: \star, \alpha : = \alpha_1 \times \alpha_2, \Gamma_3) \)

Above

By Lemma 67 (Transitivity of Separation) twice

- **Case** \( \Gamma[\alpha_2: \star, \alpha_1: \star, \alpha: = \alpha_1 \times \alpha_2] \vdash e \equiv \alpha_2 \vdash \Delta \)

Similar to the \( \times \) case, but simpler.

- **Case** \( \Gamma[\alpha_2: \star, \alpha_1: \star, \alpha: = \alpha_1 \times \alpha_2] \vdash e \equiv s_0 : (\alpha_1 \rightarrow \alpha_2) \quad \Delta \quad \Gamma[\alpha : \star] \vdash e \equiv s_0 : \alpha \quad \Delta \quad \Gamma[\alpha : \star] \vdash e \equiv s_0 : \alpha \rightarrow \Delta \)

Similar to the \( \times \) and \( \ast \) 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** \( \Gamma \vdash e \rightarrow A \quad \Theta \vdash \Pi : A \equiv [\Theta] C \vdash [\Delta] \) \( \Delta \vdash \Pi \quad \text{covers} \quad [\Delta] \) \( \) \( \times \)Case \( \)

Use the i.h. and Lemma 67 (Transitivity of Separation) .
I’ Decidability of Algorithmic Subtyping

I’.1 Lemmas for Decidability of Subtyping

Lemma 72 (Substitution Isn’t Large). For all contexts Θ, we have \( \#\text{large}(\Theta A) = \#\text{large}(A) \).

**Proof.** By induction on A, following the definition of substitution.

Lemma 73 (Instantiation Solves). If \( \Gamma \vdash \delta : \kappa \rightarrow \Delta \) and \( \Gamma \tau = \tau \) and \( \delta \notin \text{FV}([\tau]) \) then \( |\text{unsolved}(\Gamma)| = |\text{unsolved}(\Delta)| + 1 \).

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

- **Case**
  \( \Gamma, \kappa, \delta \vdash \tau : \kappa \rightarrow \Delta \)

  \( \Gamma, \kappa, \delta := \tau : \kappa \rightarrow \Gamma, \kappa, \delta \rightarrow \kappa, \Gamma, \delta := \tau, \kappa, \delta := \tau \rightarrow \Delta \)

  It is evident that \( |\text{unsolved}(\Gamma, \kappa, \delta, \Gamma)| = |\text{unsolved}(\Gamma, \kappa, \delta, \kappa)| + 1 \).

- **Case**
  \( \beta \in \text{unsolved}(\Gamma[\delta : \kappa][\beta : \kappa]) \)

  \( \Gamma[\delta : \kappa][\beta : \kappa] \vdash \delta \rightarrow \beta : \kappa \rightarrow \Gamma[\delta : \kappa][\beta : \kappa] \rightarrow \kappa \rightarrow \Delta \)

  Similar to the previous case.

- **Case**
  \( \Gamma[\delta : \kappa][\beta : \kappa] \vdash \beta : \kappa \rightarrow \Gamma[\delta : \kappa][\beta : \kappa] \rightarrow \kappa \rightarrow \Delta \)

  \( |\text{unsolved}(\Gamma[\delta : \kappa][\beta : \kappa])| = |\text{unsolved}(\Gamma[\delta : \kappa][\beta : \kappa])| + 1 \) By i.h.

  \( |\text{unsolved}(\Gamma[\delta : \kappa][\beta : \kappa])| = |\text{unsolved}(\Gamma[\delta : \kappa][\beta : \kappa])| + 1 \) Subtracting 1

  \( |\text{unsolved}(\Delta)| + 1 \) By i.h.

- **Case**
  \( \Gamma[\delta : \kappa][\beta : \kappa] \vdash \beta : \kappa \rightarrow \Gamma[\delta : \kappa][\beta : \kappa] \rightarrow \kappa \rightarrow \Delta \)

  \( \text{By definition of unsolved}(\Delta) \) □

Lemma 74 (Checkeq Solving). If \( \Gamma \vdash s \triangleleft t : \kappa \rightarrow \Delta \) then either \( \Delta = \Gamma \) or \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \).

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

- **Case**
  \( \Gamma \vdash u \triangleleft u : \kappa \rightarrow \Gamma \)

  Here \( \Delta = \Gamma \).

- **Cases** CheckeqUnit, CheckeqZero Similar to the CheckeqVar case.

- **Case**
  \( \Gamma \vdash \sigma \triangleleft t : N \rightarrow \Delta \)

  \( \Gamma \vdash \text{succ}(\sigma) \triangleleft \text{succ}(t) : N \rightarrow \Delta \)

  Follows by i.h.
Proof of Lemma 74 (Checkeq Solving).

If \( \Gamma \vdash \alpha \equiv \alpha \vdash \Gamma \) then either \( \Delta = \Gamma \) or \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \).

Proof. Only one rule can derive the judgment:

- Case

  \[
  \Gamma \vdash \sigma_1 \equiv t_1 : \tau_1 \vdash \Theta \quad \Theta \vdash [\Theta] \sigma_2 \equiv [\Theta] t_2 : \tau_2 \vdash \Delta
  \]

  \[
  \Gamma \vdash (\sigma_1 \equiv \sigma_2) \equiv (t_1 \equiv t_2) \vdash \Delta
  \]

  By Lemma 74 (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 74 (Checkeq Solving) on the second premise.

  In the latter case, applying Lemma 74 (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 \( |\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)| \).  

  \[\square\]

Lemma 75 (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 76 (Equiv Solving)

Case $\equiv \text{Exvar} \equiv \text{Unit}$ Similar to the $\equiv \text{Var}$ case.

Case

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

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

By i.h., either $\Theta = \Gamma$ or $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$.

In the former case, apply the i.h. to the second premise. Now either $\Delta = \Theta$—and therefore $\Delta = \Gamma$—or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$. Since $\Theta = \Gamma$, we have $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

In the latter case, we have $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$. By i.h. on the second premise, either $\Delta = \Theta$, and substituting $\Delta$ for $\Theta$ gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$—or $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Theta)|$, which combined with $|\text{unsolved}(\Theta)| < |\text{unsolved}(\Gamma)|$ gives $|\text{unsolved}(\Delta)| < |\text{unsolved}(\Gamma)|$.

Case

\[ \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \vdash \Delta_1, \alpha : \kappa, \Delta' \]

\[ \Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \vdash \Delta \]

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 21 (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

\[ \Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta \]

\[ \Gamma \vdash P \supset A_0 \equiv Q \supset B_0 \vdash \Delta \]

Similar to the $\equiv \Theta$ case, but using Lemma 75 (Prop Equip Solving) on the first premise instead of the i.h.

Case

\[ \Gamma \vdash P \equiv Q \vdash \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \vdash \Delta \]

\[ \Gamma \vdash A_0 \land P \equiv B_0 \land Q \vdash \Delta \]

Similar to the $\equiv \land$ case.

Case

\[ \Gamma_0[\bar{\alpha}] \vdash \& := \tau : \star \vdash \Delta \quad \& \notin \text{FV}(\tau) \]

\[ \Gamma_0[\bar{\alpha}] \vdash \& \equiv \tau \vdash \Delta \]

By Lemma 73 (Instantiation Solves), $|\text{unsolved}(\Delta)| = |\text{unsolved}(\Gamma)| - 1$.

Case

\[ \Gamma_0[\bar{\alpha}] \vdash \& := \tau : \star \vdash \Delta \quad \& \notin \text{FV}(\tau) \]

\[ \Gamma_0[\bar{\alpha}] \vdash \tau \equiv \bar{\alpha} \vdash \Delta \]

Similar to the $\equiv \text{Instantiatel}$ case.

Lemma 77 (Decidability of Propositional Judgments).

The following judgments are decidable, with $\Delta$ as output in (1)–(3), and $\Delta^\perp$ as output in (4) and (5).

We assume $\sigma = |\Gamma^\sigma|$ and $t = |\Gamma^t|$ in (1) and (4). Similarly, in the other parts we assume $P = |\Gamma^P|$ and (in part (3)) $Q = |\Gamma^Q|$.

1. $\Gamma \vdash \sigma = t : \kappa \vdash \Delta$
2. $\Gamma \vdash P \vdash \Delta$
3. $\Gamma \vdash P \equiv Q \vdash \Delta$
4. $\Gamma / \sigma = t : \kappa \vdash \Delta^\perp$

Proof of Lemma 77 (Decidability of Propositional Judgments).
Decidability of $\Gamma / P \vdash \Delta^\perp$

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: CheckeqVar, CheckeqUnit, CheckeqZero. No premises.
   - Case CheckeqSucc: Both $\sigma$ and $t$ get smaller in the premise.
   - Cases: CheckeqInstL, CheckeqInstR. Follows from Lemma 66 (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, 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, ≡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: ElimeqUvarRefL, ElimeqUvarRefR. No premises, and the output is the same as the input.
   - Case ElimeqClash: The only premise is the clash judgment, which is clearly decidable. There is no output.
   - Case ElimeqBin: In the first premise, we have the same $\Gamma$ but both $\sigma$ and $t$ are smaller. By i.h., the first premise is decidable; moreover, either some variables in $\Theta$ were solved, or no additional variables were solved.
     If some variables in $\Theta$ were solved, the second premise is smaller than the conclusion according to our lexicographic measure, so by i.h., the second premise is decidable.
     If no additional variables were solved, then $\Theta = \Gamma$. Therefore $|\Theta|\tau_2 = |\Gamma|\tau_2$. It is given that $\sigma = |\Gamma|\sigma$ and $t = |\Gamma|t$, so $|\Gamma|\tau_2 = |\Gamma|\tau_2$. Likewise, $|\Theta|\tau_2' = |\Gamma|\tau_2'$, so we are making a recursive call on a strictly smaller subterm.
     Regardless, $\Delta^\perp$ is either $\perp$, or is a $\Delta$ which has no more unsolved variables than $\Theta$, which in turn has no more unsolved variables than $\Gamma$.
   - Case ElimeqBinBot: The premise is invoked on subterms, and does not yield an output context.
   - Case ElimeqSucc: Both $\sigma$ and $t$ get smaller. By i.h., the output context has fewer unsolved variables, if it exists.
   - Cases ElimeqInstL, ElimeqInstR. Follows from Lemma 66 (Decidability of Instantiation). Furthermore, by Lemma 73 (Instantiation Solves), instantiation solves a variable in the output.
   - Cases ElimeqUvarL, ElimeqUvarR: These rules have no nontrivial premises, and $\alpha$ is solved in the output context.
   - Cases ElimeqUvarL, ElimeqUvarR: These rules have no nontrivial premises, and produce the output context $\perp$.

5. Decidability of $\Gamma / P \vdash \Delta^\perp$: By induction on $P$. But we have only one rule deriving this judgment form, ElimpropEq, for which decidability follows from part (4).

Lemma 78 (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) $\#\text{large}(A) + \#\text{large}(B)$;
Proof of Lemma 78 (Decidability of Equivalence) lem:equiv-decidable

\[ |\text{unsolved}(\Gamma)|, \text{the number of unsolved existential variables in } \Gamma; \]

\[ |A| + |B|. \]

- **Cases**
  - \[ \equiv \text{Var} \equiv \text{Exvar} \equiv \text{Unit} \]
    - No premises.
  - **Case**
    - \[ \Gamma \vdash A_1 \equiv B_1 \vdash \Theta, \Theta \vdash [\Theta] A_2 \equiv [\Theta] B_2 \vdash \Delta \]
      - \[ \Gamma \vdash A_1 \odot A_2 \equiv B_1 \odot B_2 \vdash \Delta \equiv \odot \]
    - 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**
    - \[ \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \vdash \Delta, \alpha : \kappa, \Delta' \]
      - \[ \Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \vdash \Delta \equiv \forall \]
    - 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**
    - \[ \Gamma \vdash P \equiv Q \vdash \Theta, \Theta \vdash [\Theta] A_0 \equiv [\Theta] B_0 \vdash \Delta \]
      - \[ \Gamma \vdash P \supset A_0 \equiv Q \supset B_0 \vdash \Delta \equiv \supset \]
    - The first premise is decidable by Lemma 77 (Decidability of Propositional Judgments) (3).
    - For the second premise, by Lemma 72 (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**
    - \[ \Gamma \vdash P \equiv Q \vdash \Theta, \Theta \vdash [\Theta] A_0 \equiv [\Theta] B_0 \vdash \Delta \]
      - \[ \Gamma \vdash A_0 \land P \equiv B_0 \land Q \vdash \Delta \equiv \land \]
    - Similar to the \( \equiv \supset \) case.
  - **Case**
    - \[ \Gamma[\check{\alpha}] \vdash \check{\alpha} : \tau \vdash \Delta \]
      - \[ \check{\alpha} \notin \text{FV}(\tau) \]
    - \[ \Gamma[\check{\alpha}] \vdash \check{\alpha} \equiv \tau \vdash \Delta \equiv \text{Instantiate} \]
    - Follows from Lemma 66 (Decidability of Instantiation).
  - **Case**
    - \[ \equiv \text{InstantiateL} \]
      - Similar to the \( \equiv \text{InstantiateR} \) case.

\[ \square \]

\section{1.2 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 \( \Gamma \vdash A \ll A \; B \vdash \Delta \).

**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 \( \Gamma \vdash A \ll A \; B \vdash \Delta \).
Proof of Theorem 1 (Decidability of Subtyping)

Theorem 1: Decidability of Subtyping

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

\( \vdash A <: \pm B \vdash \Delta \)

In this case, we appeal to Lemma 78 (Decidability of Equivalence).

\begin{itemize}
  \item Case A not headed by \( \forall / \exists \\ B not headed by \( \forall / \exists \\)
  \[ \Gamma \vdash A \equiv B \vdash \Delta \]
  \[ \vdash A <: \pm B \vdash \Delta \]

  The premise has one fewer quantifier.

  \item Case B not headed by \( \forall \\)
  \[ \Gamma, \alpha : \kappa \vdash A <: B \vdash \Delta, \alpha, \Theta \]
  \[ \vdash \forall \alpha : \kappa. A <: B \vdash \Delta \]

  The premise has one fewer quantifier.

  \item Case \( \Gamma, \alpha : \kappa \vdash A <: B \vdash \Delta, \alpha, \Theta \)
  \[ \vdash \exists \alpha : \kappa. A <: B \vdash \Delta \]

  The premise has one fewer quantifier.

  \item Case A not headed by \( \exists \\)
  \[ \Gamma, \beta : \kappa \vdash A <: B \vdash \Delta, \beta, \Theta \]
  \[ \vdash \exists \beta : \kappa. A <: B \vdash \Delta \]

  The premise has one fewer quantifier.

  \item Case \( \Gamma, \beta : \kappa \vdash A <: B \vdash \Delta, \beta, \Theta \)
  \[ \vdash \neg A \]
  \[ \vdash \neg B \]

  Consider whether \( B \) is negative.

  \begin{itemize}
    \item Case \( \neg B \):
      \[ \Gamma, \beta : \kappa \vdash A <: B \vdash \Delta, \beta, \Theta \]
      \[ \vdash \neg \beta : \kappa. B \]

      Inversion on the premise

      There is one fewer quantifier in the subderivation.

      \item Case \( \nonneg B \):
        \[ \Gamma, \beta : \kappa \vdash A <: B \vdash \Delta, \beta, \Theta \]
        \[ \vdash \nonneg B \]

        Inversion on the premise

        There is one fewer quantifier in the subderivation.

  \end{itemize}

\end{itemize}
Proof of Theorem 1 (Decidability of Subtyping) \(\text{thm:subtyping-decidable}\)

\[ \Gamma \vdash A <:^+ B \rightarrow^+ \Delta \quad \text{pos}(B) \]

\[ \Gamma \vdash A <:^+ B \rightarrow^+ \Delta \quad \text{nonneg}(A) \]

This case is similar to the \(L\) case.

\[ \square \]

### I’.3 Decidability of Matching and Coverage

**Lemma 79 (Decidability of Expansion Judgments).**

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

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

**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 \(\wedge\) connectives appearing in \(\vec{A}\), and then (2) the size of \(\vec{A}\), considered to be the sum of the sizes \(|\vec{A}|\) of each type \(A\) in \(\vec{A}\).

(For \(\text{CoversVar}\), \(\text{Covers}\times\) and \(\text{Covers}^+\) we also use the appropriate part of Lemma 79 (Decidability of Expansion Judgments).)

- **Case C\overline{ov}\underline{ers}\underline{Em}\overline{pty}:** No premises.
- **Case C\overline{ov}\underline{ers}\underline{Var}:** The number of \(\wedge\) connectives does not grow, and \(\vec{A}\) gets smaller.
- **Case C\overline{ov}\underline{ers}\underline{1}:** The number of \(\wedge\) connectives does not grow, and \(\vec{A}\) gets smaller.
- **Case C\overline{ov}\underline{ers}\times\text{ }:\text{ }|\vec{A}_1| + |\vec{A}_2| < |\vec{A}_1 \times \vec{A}_2|\): The number of \(\wedge\) connectives does not grow, and \(\vec{A}\) gets smaller, since \(|\vec{A}_1| + |\vec{A}_2| < |\vec{A}_1 \times \vec{A}_2|\).
- **Case C\overline{ov}\underline{ers}\times\text{ }:\text{ }\vec{A} = (A_1 + A_2, \vec{B})\): Here we have \(\vec{A} = (A_1 + A_2, \vec{B})\). In the first premise, we have \((A_1, \vec{B})\), which is smaller than \(\vec{A}\), and in the second premise we have \((A_2, \vec{B})\), which is likewise smaller. (In both premises, the number of \(\wedge\) connectives does not grow.)
- **Case C\overline{ov}\underline{ers}\underline{\exists}:** The number of \(\wedge\) connectives does not grow, and \(\vec{A}\) gets smaller.
- **Case C\overline{ov}\underline{ers}\underline{Eq}:** The first premise is decidable by Lemma 77 (Decidability of Propositional Judgments) (4). The number of \(\wedge\) connectives in \(\vec{A}\) gets smaller (note that applying \(\Delta\) as a substitution cannot add \(\wedge\) connectives).
- **Case C\overline{ov}\underline{ers}\underline{Eq}\underline{Bot}:** Decidable by Lemma 77 (Decidability of Propositional Judgments) (4).

\[ \square \]

### I’.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

\[ \Gamma \vdash e : A \rightarrow p \rightarrow \Delta. \]

(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

\[ \Gamma \vdash s : A \rightarrow B \rightarrow q \rightarrow \Delta. \]
(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
\[
\Gamma \vdash e \Rightarrow - - \iff - -
\]
\[
\Gamma \vdash e \ll B \ p \iff - -
\]
\[
\Gamma \vdash s : B \ p \gg - - \iff - -
\]
\[
\Gamma \vdash \Pi :: \vec{A} \ll C \ p \iff - -
\]
(where we write “-” for parts of the judgments that are outputs), the following induction measure on such judgments is adequate to prove decidability:
\[
\langle \frac{e/s/\Pi, \Rightarrow \iff / \gg, \#\text{large}(B), B}{\text{Match}, \vec{A}}, \text{match judgment form} \rangle
\]
where \( \langle \ldots \rangle \) denotes lexicographic order, and where (when comparing two judgments typing terms of the same size) the synthesis judgment (top line) is considered smaller than the checking judgment (second line). That is,
\[
\Rightarrow \prec \iff / \gg / \text{Match}
\]
Two match judgments are compared according to, first, the list of branches \( \Pi \) (which is a subterm of the containing case expression, allowing us to invoke the i.h. for the \textbf{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** \textbf{EmptySpine} No premises.
- **Case** \textbf{→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** \textbf{Var} No nontrivial premises.
- **Case** \textbf{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** \textbf{Anno} It is easy to show that the judgment \( \Gamma \vdash A \! \text{ type} \) is decidable. The second premise types \( e \), but the conclusion types \( (e : A) \), so the first part of the measure gets smaller.
- **Cases** \textbf{1I} [\textbf{1I \^}\(\alpha\)] No premises.
- **Case** \textbf{∀I} Both the premise and conclusion type \( e \), and both are checking; however, \( \#\text{large}(A_0) < \#\text{large}(\forall \alpha : \kappa. A_0) \), so the premise is smaller.
- **Case** \textbf{∀Spine} Both the premise and conclusion type \( e \cdot s \), and both are spine judgments; however, \( \#\text{large}(-) \) decreases.
- **Case** \textbf{∧I} By Lemma 77 (Decidability of Propositional Judgments) (2), the first premise is decidable. For the second premise, \( \#\text{large}(\Theta A_0) = \#\text{large}(A_0) < \#\text{large}(A_0 \land P) \).
Proof of [Lemma 80](Decidability of Auxiliary Judgments) (lem:aux-det)

We now consider the match rules:

- **Case MatchEmpty**: No premises.
- **Case MatchSeg**: In each premise, the list of branches is properly contained in \( \Pi \), making each premise smaller by the first part (“e/s/\( \Pi \)”) of the measure.
- **Case MatchBase**: The term \( e \) in the premise is properly contained in \( \Pi \).
- **Cases Match\( \times \) [Match\( + \), MatchNeg, MatchWild]**: Smaller by part (2) of the measure.
- **Case Match\( \land \)**: The premise has a smaller \( \tilde{A} \), so it is smaller by the \( \tilde{A} \) part of the measure. (The premise is the other judgment form, so it is larger by the “match judgment form” part, but \( \tilde{A} \) lexicographically dominates.)
- **Case MatchI**: For the premise, use Lemma [77](Decidability of Propositional Judgments) (4).
- **Case MatchUnify**: Lemma [77](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.

\( J' \) Determinacy

**Lemma 80** (Determinacy of Auxiliary Judgments).

1. **Elimqe**: Given \( \Gamma, \sigma, t, \kappa \) such that \( \text{FEV}(\sigma) \cup \text{FEV}(t) = \emptyset \) and \( \mathcal{D}_1 : \Gamma \vdash \sigma \Downarrow t : \kappa \vdash \Delta^+_1 \) and \( \mathcal{D}_2 : \Gamma \vdash \sigma \Downarrow t : \kappa \vdash \Delta^+_2 \), it is the case that \( \Delta^+_1 = \Delta^+_2 \).

2. **Instantiation**: Given \( \Gamma, \tilde{\alpha}, t, \kappa \) such that \( \tilde{\alpha} \in \text{unsolved}(\Gamma) \) and \( \Gamma \vdash t : \kappa \) and \( \tilde{\alpha} \notin \text{FV}(t) \) and \( \mathcal{D}_1 : \Gamma \vdash \tilde{\alpha} := t : \kappa \vdash \Delta_1 \) and \( \mathcal{D}_2 : \Gamma \vdash \tilde{\alpha} := t : \kappa \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

3. **Symmetric instantiation**: Given \( \Gamma, \tilde{\alpha}, \tilde{\beta}, \kappa \) such that \( \tilde{\alpha}, \tilde{\beta} \in \text{unsolved}(\Gamma) \) and \( \tilde{\alpha} \neq \tilde{\beta} \) and \( \mathcal{D}_1 : \Gamma \vdash \tilde{\alpha} := \tilde{\beta} : \kappa \vdash \Delta_1 \) and \( \mathcal{D}_2 : \Gamma \vdash \tilde{\beta} := \tilde{\alpha} : \kappa \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

4. **Checkeq**: Given \( \Gamma, \sigma, t, \kappa \) such that \( \mathcal{D}_1 : \Gamma \vdash \sigma \Downarrow t : \kappa \vdash \Delta_1 \) and \( \mathcal{D}_2 : \Gamma \vdash \sigma \Downarrow t : \kappa \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

5. **Elimprop**: Given \( \Gamma, \mathcal{P} \) such that \( \mathcal{D}_1 : \Gamma \vdash \mathcal{P} \vdash \Delta_1 \) and \( \mathcal{D}_2 : \Gamma \vdash \mathcal{P} \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).

6. **Checkprop**: Given \( \Gamma, \mathcal{P} \) such that \( \mathcal{D}_1 : \Gamma \vdash \mathcal{P} \text{ true } \vdash \Delta_1 \) and \( \mathcal{D}_2 : \Gamma \vdash \mathcal{P} \text{ true } \vdash \Delta_2 \), it is the case that \( \Delta_1 = \Delta_2 \).
Proof.

Proof of Part (1) (Elimeq).

Rule ElimeqZero applies if and only if σ = t = zero.

Rule ElimeqSucc applies if and only if σ and t are headed by succ.

Now suppose σ = α.

• Rule ElimeqUvarK applies if and only if t = α. (Rule ElimeqClash cannot apply; rules ElimeqUvarL and ElimeqUvarR have a free variable condition; rules ElimeqUvarL and ElimeqUvarR have a condition that σ ≠ t.)

In the remainder, assume t ≠ alpha.

• If α ∈ FV(t), then rule ElimeqUvarL applies, and no other rule applies (including ElimeqUvarR and ElimeqClash).

In the remainder, assume α ∉ FV(t).

• Consider whether ElimeqUvarR applies. The conclusion matches if we have t = β for some β ≠ α (that is, σ = α and t = β). But ElimeqUvarR has a condition that β ∈ FV(σ), and σ = α, 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 t₁ ⊕ t₂.

Rule InstZero applies if and only if t has the form zero.

Rule InstSucc applies if and only if t has the form succ(t₂).

If t has the form β, then consider whether β is declared to the left of α in the given context:

• If β is declared to the left of α, then rule InstReach cannot be used, which leaves only InstSolve.

• If β is declared to the right of α, then InstSolve cannot be used because β is not well-formed under Γ₀ (the context to the left of α in InstSolve). That leaves only InstReach.

• â cannot be β, because it is given that â ∉ FV(t) = FV(β) = (β).

Proof of Part (3) (Symmetric instantiation).

InstBin, InstZero, and InstSucc cannot have been used in either derivation.

Suppose that InstSolve concluded D₁. Then Δ₁ is the same as Γ with â solved to β. Moreover, β is declared to the left of α in Γ. Thus, InstSolve cannot conclude D₂. However, InstReach can conclude D₂, but produces a context Δ₂ which is the same as Γ but with â solved to β. Therefore Δ₁ = Δ₂.

The other possibility is that InstReach concluded D₁. Then Δ₁ is the same as Γ with β solved to â, with â declared to the left of β in Γ. Thus, InstSolve cannot conclude D₂. However, InstSolve can conclude D₂, producing a context Δ₂ which is the same as Γ but with β solved to â. Therefore Δ₁ = Δ₂.

Proof of Part (4) (Checkeq).

Rule CheckeqVar applies if and only if σ = t = â or σ = t = α (note the free variable conditions in CheckeqInstL and CheckeqInstR).

Rule CheckeqUnit applies if and only if σ = t = 1.

Rule CheckeqBin applies if and only if σ and t are both headed by the same binary connective.

Rule CheckeqZero applies if and only if σ = t = zero.

Rule CheckeqSucc applies if and only if σ and t are headed by succ.

Now suppose σ = â. 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 β (distinct from â), and is unsolved, then both CheckeqInstL and CheckeqInstR apply, but by part (3), we get the same output context from each.

The t = â 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 81 (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 80 (Determinacy of Auxiliary Judgments) (4).

Proof of Part (2) (type equivalence). If neither \( A \) nor \( B \) is an existential variable, they must have the same head connectives, and the same rule must conclude both derivations.

If \( A \) and \( B \) are the same existential variable, then only \( \equiv \text{Exvar} \) applies (due to the free variable conditions in \( \equiv \text{InstantiateL} \) and \( \equiv \text{InstantiateR} \)).

If \( A \) and \( B \) are different unsolved existential variables, the judgment matches the conclusion of both \( \equiv \text{InstantiateL} \) and \( \equiv \text{InstantiateR} \) but by part (3) of Lemma 80 (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 \( \Gamma \vdash A :<: \pm B \vdash \Delta_1 \) and \( \Gamma \vdash A :<: \pm B \vdash \Delta_2 \), then we know the last rule ending the derivation of \( D_1 \) and \( D_2 \) must be:

  \[
  \begin{array}{cccc}
  \forall & \exists & \text{other} \\
  \forall & <: \text{R} & <: \text{L} & <: \equiv \\
  \exists & <: \equiv & <: \text{L} & <: \equiv \\
  \text{other} & <: \text{R} & <: \equiv & <: \equiv \\
  \end{array}
  \]

  The only case in which there are two possible final rules is in the \( \forall/\forall \) case. In this case, regardless of the choice of rule, by inversion we get subderivations \( \Gamma \vdash A :<: \pm B \vdash \Delta_1 \) and \( \Gamma \vdash A :<: \pm B \vdash \Delta_2 \).

- If \( \Gamma \vdash A :<: \pm B \vdash \Delta_1 \) and \( \Gamma \vdash A :<: \pm B \vdash \Delta_2 \), then we know the last rule ending the derivation of \( D_1 \) and \( D_2 \) must be:

  \[
  \begin{array}{cccc}
  \forall & \exists & \text{other} \\
  \forall & <: \equiv & <: \text{L} & <: \equiv \\
  \exists & <: \equiv & <: \text{L} & <: \equiv \\
  \text{other} & <: \equiv & <: \equiv & <: \equiv \\
  \end{array}
  \]

  The only case in which there are two possible final rules is in the \( \forall/\forall \) case. In this case, regardless of the choice of rule, by inversion we get subderivations \( \Gamma \vdash A :<: \pm B \vdash \Delta_1 \) and \( \Gamma \vdash A :<: \pm B \vdash \Delta_2 \).

As a result, the result follows by a routine induction.

Theorem 5 (Determinacy of Typing).

March 2, 2015
(1) Checking: Given $\Gamma$, $e$, $A$, $p$ such that $\mathcal{D}_1 \vdash \Gamma \vdash e \iff A \quad p \vdash \Delta_1$ and $\mathcal{D}_2 \vdash \Gamma \vdash e \iff A \quad p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

(2) Synthesis: Given $\Gamma$, $e$ such that $\mathcal{D}_1 \vdash \Gamma \vdash e \Rightarrow B_1 \quad p_1 \vdash \Delta_1$ and $\mathcal{D}_2 \vdash \Gamma \vdash e \Rightarrow B_2 \quad p_2 \vdash \Delta_2$, it is the case that $B_1 = B_2$ and $p_1 = p_2$ and $\Delta_1 = \Delta_2$.

(3) Spine judgments:

Given $\Gamma$, $e$, $A$, $p$ such that $\mathcal{D}_1 \vdash \Gamma \vdash e : A \quad p \gg C_1 \quad q_1 \vdash \Delta_1$ and $\mathcal{D}_2 \vdash \Gamma \vdash e : A \quad p \gg C_2 \quad q_2 \vdash \Delta_2$, it is the case that $C_1 = C_2$ and $q_1 = q_2$ and $\Delta_1 = \Delta_2$.

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

(4) Match judgments:

Given $\Gamma$, $\Pi$, $\tilde{A}$, $p$, $C$ such that $\mathcal{D}_1 \vdash \Gamma \vdash \Pi \vdash \tilde{A} \iff C \quad p \vdash \Delta_1$ and $\mathcal{D}_2 \vdash \Gamma \vdash \Pi \vdash \tilde{A} \iff C \quad p \vdash \Delta_2$, it is the case that $\Delta_1 = \Delta_2$.

Proof.

Proof of Part (1) (checking).

The rules with a checking judgment in the conclusion are: $\mathcal{D}_1 \vdash \Gamma \vdash \Pi \vdash \tilde{A} \iff C \quad p \vdash \Delta_1$ and $\mathcal{D}_2 \vdash \Gamma \vdash \Pi \vdash \tilde{A} \iff C \quad p \vdash \Delta_2$.

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 $\to$ only these forms. For example, given $e = x$ and $A = \forall \alpha : k \cdot A_0$, we need not choose between $\text{Sub}$ and $\land I$; the latter is ruled out by its chk-I premise.

\[
\begin{array}{cccccccc}
\lambda x. e_0 & \text{chk-I} & \land I & \lor I & \to I & + I & \times I & 1 & \alpha \\
\text{in}_k e_0 & \text{chk-I} & \land I & \lor I & \to I & + I & \times I & 1 & \alpha \\
\langle e_1, e_2 \rangle & \text{chk-I} & \land I & \lor I & \to I & + I & \times I & 1 & \alpha \\
\text{case}(e_0, \Pi) & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} & \text{Case} \\
x & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} \\
(e_0 : A) & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} \\
e_1 e_2 & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} & \text{Sub} \\
\end{array}
\]

Notes:

- **Note 1:** The choice between $\land I$ and $\lor I$ is resolved by Lemma [80](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 Case rule: only the 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: $\text{Var}$, $\text{Anno}$, $\text{E}$ and $\text{E}$. Rule $\text{Var}$ applies if and only if $e$ has the form $x$. Rule $\text{Anno}$ applies if and only if $e$ has the form $(e_0 : A)$.

Otherwise, the judgment can be derived only if $e$ has the form $e_1 e_2$, by $\text{E}$ or $\text{E}$. If $\mathcal{D}_1$ and $\mathcal{D}_2$ both end in $\text{E}$ or $\text{E}$, we are done. Suppose $\mathcal{D}_1$ ends in $\text{E}$ and $\mathcal{D}_2$ 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, $q = f$. However, from the condition in $\text{E}$ it must be the case that $\text{FEV}(\Delta C) = \emptyset$ in $\text{E}$.
Proof of Theorem 5 \textbf{(Determinacy of Typing)}

\textbf{Proof of Part (3) (spine judgments).} For the ordinary spine judgment, rule \texttt{EmptySpine} applies if and only if the given spine is empty. Otherwise, the choice of rule is determined by the head constructor of the input type: $\rightarrow$ \texttt{Spine} \lor \texttt{\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lor\lop...
\[ \Delta \rightarrow \Omega \]
\[ \Gamma' \vdash \Delta_1 := \tau_1 : \star \rightarrow \Theta \]
\[ \Theta \rightarrow \Delta \]
\[ \Theta \rightarrow \Omega \]
\[ [\Omega] \Delta_1 = [\Omega] \tau_1 \]
\[ \Theta \vdash \Delta_2 := [(\Theta) \tau_2 : \star \rightarrow \Delta \]
\[ [\Omega] \Delta_2 = [\Omega] [\Theta] \tau_2 \]
\[ = [\Omega] [\Theta] \tau_2 \]
\[ (\tau_1) \oplus (\tau_2) = (\tau_3) \]
\[ \forall \tau \in \Delta \]
\[ \forall \sigma \in \Delta \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
\[ \forall \theta \in \Omega \]
Proof of Lemma 83 (Soundness of Checkeq)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash \sigma_0 \equiv t_0 : \mathbb{N} \rightarrow \Delta$</td>
<td>Subderivation</td>
</tr>
<tr>
<td>$\Theta \vdash [\Theta]\sigma_1 \equiv [\Theta]t_1 : \star \rightarrow \Delta$</td>
<td>Subderivation</td>
</tr>
<tr>
<td>$\Delta \rightarrow \Omega$</td>
<td>Given</td>
</tr>
<tr>
<td>$\Theta \rightarrow \Delta$</td>
<td>By Lemma 45 (Checkeq Extension)</td>
</tr>
<tr>
<td>$\Theta \rightarrow \Omega$</td>
<td>By Lemma 32 (Extension Transitivity)</td>
</tr>
<tr>
<td>$[\Omega]\sigma_0 = [\Omega]t_0$</td>
<td>By i.h. on first subderivation</td>
</tr>
<tr>
<td>$[\Omega]\Theta\sigma_1 = [\Omega]\Theta t_1$</td>
<td>By i.h. on second subderivation</td>
</tr>
<tr>
<td>$[\Omega]\Theta\sigma_1 = [\Omega]\Theta t_1$</td>
<td>By Lemma 28 (Substitution Monotonicity)</td>
</tr>
<tr>
<td>$[\Omega]\Theta\sigma_1 = [\Omega]t_1$</td>
<td>By Lemma 28 (Substitution Monotonicity)</td>
</tr>
<tr>
<td>$\sigma_0 \equiv \sigma_1$</td>
<td>By transitivity of equality</td>
</tr>
<tr>
<td>$\sigma_0 \equiv [\Omega]t_0 \oplus [\Omega]t_1$</td>
<td>By congruence of equality</td>
</tr>
<tr>
<td>$\equiv [\Omega]t_0 \oplus [\Omega]t_1$</td>
<td>By definition of substitution</td>
</tr>
</tbody>
</table>

- **Case**
  - $\Gamma[\alpha] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta$  
    - $\hat{\alpha} \notin \text{FV}(t)$
  - $\Gamma[\alpha] \vdash \hat{\alpha} := t : \kappa \rightarrow \Delta$  
    - $\hat{\alpha} \notin \text{FV}(t)$  
    - By Lemma 82 (Soundness of Instantiation)

Similar to the CheckeqInstL case.

Lemma 84 (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.

- **Case**
  - $\Gamma \vdash \sigma_1 \equiv t_1 : \mathbb{N} \rightarrow \Theta$  
    - $\Theta \vdash [\Theta]\sigma_2 \equiv [\Theta]t_2 : \mathbb{N} \rightarrow \Delta$  
    - $\Gamma \vdash (\sigma_1 = \sigma_2) \equiv (t_1 = t_2) \rightarrow \Delta$  
    - $\Delta \rightarrow \Omega$  
    - $\Theta \rightarrow \Delta$  
    - $\Theta \rightarrow \Omega$  
    - $\Gamma \vdash \sigma_1 \equiv t_1 : \mathbb{N} \rightarrow \Theta$  
      - $[\Omega]\sigma_1 = [\Omega]t_1$  
      - $[\Omega]\Theta\sigma_1 = [\Omega]\Theta t_1$  
      - $[\Omega]\Theta\sigma_1 = [\Omega]t_1$  
      - $[\Omega]\Theta\sigma_1 = [\Omega]t_1$  
      - $\sigma_0 \equiv \sigma_1$  
      - $\sigma_0 \equiv [\Omega]t_0 \oplus [\Omega]t_1$  
      - $\equiv [\Omega]t_0 \oplus [\Omega]t_1$  
      - By reflexivity of equality

Lemma 85 (Soundness of Algorithmic Equivalence).

If $\Gamma \vdash \Lambda \equiv B \rightarrow \Delta$ where $\Delta \rightarrow \Omega$ then $[\Omega]\Lambda = [\Omega]B$.

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

- **Case**  
  - $\Gamma \vdash \alpha \equiv \alpha \rightarrow \Gamma$  
    - $\equiv \text{Var}$  
    - $[\Omega]\alpha = [\Omega]\alpha$  
      - By reflexivity of equality

- Cases $\equiv \text{Exvar} \equiv \text{Unit}$  
  - Similar to the $\equiv \text{Var}$ case.
Proof of Lemma 85: (Soundness of Algorithmic Equivalence)

- Case
  \[ \Gamma \vdash A_1 \equiv B_1 \rightarrow \Theta \quad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \rightarrow \Delta \]
  \[ \Gamma \vdash A_1 \oplus A_2 \equiv B_1 \oplus B_2 \rightarrow \Delta \]
  \[ \Delta \rightarrow \Omega \quad \text{Given} \]
  \[ \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \rightarrow \Delta \quad \text{Subderivation} \]
  \[ \Theta \rightarrow \Delta \quad \text{By Lemma 48 (Equivalence Extension)} \]
  \[ \Theta \rightarrow \Omega \quad \text{By Lemma 32 (Extension Transitivity)} \]
  \[ \Gamma \vdash A_1 \equiv B_1 \rightarrow \Theta \quad \text{Subderivation} \]
  \[ [\Omega]A_1 = [\Omega]B_1 \quad \text{By i.h.} \]
  \[ \Delta \rightarrow \Omega \quad \text{Given} \]
  \[ [\Omega][\Theta]A_2 = [\Omega][\Theta]B_2 \quad \text{By i.h.} \]
  \[ [\Omega]A_2 = [\Omega]B_2 \quad \text{By Lemma 28 (Substitution Monotonicity)} \]
  \[ ([\Omega]A_1) \oplus ([\Omega]A_2) = ([\Omega]B_1) \oplus ([\Omega]B_2) \]
  \[ \text{By above equations} \]
  \[ \equiv \]

- Case
  \[ \Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \rightarrow \Delta, \alpha : \kappa, \Delta' \]
  \[ \Gamma \vdash \forall \alpha : \kappa. A_0 \equiv \forall \alpha : \kappa. B_0 \rightarrow \Delta \]
  \[ \equiv \]
  \[ \Delta \rightarrow \Omega \quad \text{Given} \]
  \[ \Gamma, \alpha : \kappa, \Delta \rightarrow \Delta, \alpha : \kappa, \Delta' \quad \text{Subderivation} \]
  \[ \Delta' \quad \text{soft} \]
  \[ \Delta, \alpha : \kappa, \Delta' \rightarrow \Omega, \alpha : \kappa, \Omega_Z \quad \text{By Lemma 23 (Soft Extension)} \]
  \[ \Gamma, \alpha : \kappa \vdash A_0 \quad \text{type} \]
  \[ \Gamma, \alpha : \kappa \vdash B_0 \quad \text{type} \]
  \[ \text{By validity on subderivation} \]
  \[ \text{By validity on subderivation} \]
  \[ \text{By well-typing of } A_0 \]
  \[ \text{By well-typing of } B_0 \]
  \[ \Gamma, \alpha : \kappa \rightarrow \Omega, \alpha : \kappa \quad \text{By } \rightarrow \text{Uvar} \]
  \[ \text{By Lemma 19 (Declaration Order Preservation)} \]
  \[ \text{By Lemma 19 (Declaration Order Preservation)} \]
  \[ [\Omega, \alpha : \kappa, \Omega_Z]A_0 = [\Omega, \alpha : \kappa]A_0 \quad \text{By definition of substitution, since } \text{FV}(A_0) \cap \text{dom}(\Omega_Z) = \emptyset \]
  \[ [\Omega, \alpha : \kappa, \Omega_Z]B_0 = [\Omega, \alpha : \kappa]B_0 \quad \text{By definition of substitution, since } \text{FV}(B_0) \cap \text{dom}(\Omega_Z) = \emptyset \]
  \[ [\Omega, \alpha : \kappa]A_0 = [\Omega, \alpha : \kappa]B_0 \quad \text{By transitivity of equality} \]
  \[ [\Omega]A_0 = [\Omega]B_0 \quad \text{From definition of substitution} \]
  \[ \forall \alpha : \kappa. [\Omega]A_0 = \forall \alpha : \kappa. [\Omega]B_0 \quad \text{Adding quantifier to each side} \]
  \[ [\Omega][\forall \alpha : \kappa. A_0] = [\Omega][\forall \alpha : \kappa. B_0] \quad \text{By definition of substitution} \]

- Case
  \[ \Gamma \vdash P \equiv Q \rightarrow \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \]
  \[ \Gamma \vdash P \triangleright A_0 \equiv Q \triangleright B_0 \rightarrow \Delta \]
  \[ \Delta \rightarrow \Omega \quad \text{Given} \]
  \[ \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \quad \text{Subderivation} \]
  \[ \Theta \rightarrow \Delta \quad \text{By Lemma 48 (Equivalence Extension)} \]
  \[ \Theta \rightarrow \Omega \quad \text{By Lemma 32 (Extension Transitivity)} \]
  \[ \Gamma \vdash P \equiv Q \rightarrow \Theta \quad \text{Subderivation} \]
  \[ [\Omega][\Theta]A_0 = [\Omega][\Theta]B_0 \quad \text{By i.h.} \]
  \[ [\Omega]A_0 = [\Omega]B_0 \quad \text{By Lemma 28 (Substitution Monotonicity)} \]
  \[ \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \quad \text{Subderivation} \]
  \[ [\Omega][\Theta]A_0 = [\Omega][\Theta]B_0 \quad \text{By i.h.} \]
  \[ [\Omega]A_0 = [\Omega]B_0 \quad \text{By Lemma 28 (Substitution Monotonicity)} \]
  \[ \text{Symmetry of Propositional Equivalence} \]
  \[ \equiv \]

- Case
  \[ \Gamma \vdash P \equiv Q \rightarrow \Theta \quad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \rightarrow \Delta \]
  \[ \Gamma \vdash A_0 \land P \equiv B_0 \land Q \rightarrow \Delta \]
  \[ \equiv \]
Proof of [Lemma 85](Soundness of Algorithmic Equivalence)

Similar to the \(\equiv\) case.

- **Case**
  \[ \Gamma[\alpha] \vdash \alpha := \tau : \neg \Delta \quad \alpha \notin \text{FV}(\tau) \]
  \[ \Gamma[\alpha] \vdash \alpha \equiv \tau \quad \neg \Delta \]  
  **Instantiate**

  \[ \Gamma[\alpha] \vdash \alpha := \tau : \neg \Delta \]  
  Subderivation

  \[ \sigma \]  
  Similar to the **Instantiate** case.

\[ \Box \]

L’.2 Soundness of Checkprop

**Lemma 86** (Soundness of Checkprop).

If \(\Gamma \vdash P \text{ true} \quad \neg \Delta \) and \(\Delta \longrightarrow \Omega\) then \(\Psi \vdash (\Omega)P \text{ true}\).

**Proof.** By induction on the derivation of \(\Gamma \vdash P \text{ true} \quad \neg \Delta\).

- **Case**
  \[ \Gamma \vdash \sigma = t : N \quad \neg \Delta \]
  **CheckpropEq**

  \[ \Gamma \vdash \sigma = t \text{ true} \quad \neg \Delta \]  
  Subderivation

  \[ [\Omega]\sigma = [\Omega]t \]  
  By Lemma 83 (Soundness of Checkprop)

  \[ \Psi \vdash [\Omega]\sigma = [\Omega]t \text{ true} \]  
  By [DeclCheckpropEq]

  \[ \Psi \vdash [\Omega](\sigma = t) \text{ true} \]  
  By def. of subst.

  \[ \Box \]

L’.3 Soundness of Eliminations (Equality and Proposition)

**Lemma 87** (Soundness of Equality Elimination).

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

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

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

**Proof.** First, we need to recall a few properties of term unification.

1. If \(\sigma\) is a term, then \(\text{mgu}(\sigma, \sigma) = \text{id}\).

2. If \(f\) is a unary constructor, then \(\text{mgu}(f(\sigma), f(t)) = \text{mgu}(\sigma, t)\), supposing that \(\text{mgu}(\sigma, t)\) exists.

3. 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 \(\sigma_2 = \text{mgu}(\sigma_2, [\sigma_1][t_2])\), then \(\sigma = \sigma_2 \circ \sigma_1 = \sigma_1 \circ \sigma_2\).

4. If \(\alpha \notin \text{FV}(t)\), then \(\text{mgu}(\alpha, t) = (\alpha = t)\).

5. 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 \(\Gamma / \sigma \vdash t : \kappa \quad \neg \Delta\), proving both parts with a single induction.
Proof of Lemma 87 (Soundness of Equality Elimination) lem:elimeq-soundness

• Case
  \[ \Gamma / \alpha \equiv \alpha : \kappa \vdash \Gamma \]  
  ElimeqUvarRefl

Here we have \( \Delta = \Gamma \), so we are in part (1).
Let \( \theta = \text{id} \) (which is mgu(\( \sigma, \sigma \))).
We can easily show \([\text{id}]\{\Omega\} \alpha = [\Omega, \alpha] = [\Omega, -] \alpha \).

• Case
  \[ \Gamma / \text{zero} \equiv \text{zero} : \mathbb{N} \vdash \Gamma \]  
  ElimeqZero

Similar to the ElimeqUvarRefl case.

• Case
  \[ \Gamma / t_1 \equiv t_2 : \mathbb{N} \vdash \Delta^\bot \]  
  ElimeqSucc

We distinguish two subcases:

  – Case \( \Delta^\bot = \Delta \):
    Since we have the same output context in the conclusion and premise, the “for all \( t' \ldots \)” part follows immediately from the i.h. (1).
The i.h. also gives us \( \theta_0 = \text{mgu}(t_1, t_2) \).
Let \( \theta = \theta_0 \). By property (ii), mgu\((t_1, t_2) = \text{mgu}(\text{succ}(t_1), \text{succ}(t_2)) = \emptyset \).

  – Case \( \Delta^\bot = \bot \):
    \[ \Gamma / t_1 \equiv t_2 : \mathbb{N} \vdash \bot \]  
    Subderivation
    \[ \text{mgu}(t_1, t_2) = \bot \]  
    By i.h. (2)
    \[ \text{mgu}(\text{succ}(t_1), \text{succ}(t_2)) = \bot \]  
    By contrapositive of property (ii)

• Case
  \[ \alpha \not\in \text{FV}(t) \quad (\alpha = -) \not\in \Gamma \]  
  ElimeqUvarL

Here \( \Delta \neq \bot \), so we are in part (1).
\[ [\Omega, \alpha = t]t' = \left([\Omega]t/\alpha\right)[\Omega]t' \]  
By a property of substitution
\[ = [\Omega][\theta][\Omega]t' \]  
By a property of substitution
\[ = [\Omega][\theta][\Omega]t' \]  
By mgu\((\alpha, t) = (\alpha/t) \)
\[ = [\theta][\Omega]t' \]  
By a property of substitution (\( \theta \) creates no evars)

• Case
  \[ \alpha \not\in \text{FV}(t) \quad (\alpha = -) \not\in \Gamma \]  
  ElimeqUvarR

Similar to the ElimeqUvarL case.

• Case
  \[ \Gamma / 1 \equiv 1 : \ast \vdash \Gamma \]  
  ElimeqUnit

Similar to the ElimeqUvarRefl case.

• Case
  \[ \Gamma / \tau_1 \equiv \tau'_1 : \ast \vdash \Theta \quad \Theta / [\Theta]\tau_1 \equiv [\Theta]\tau'_1 : \ast \vdash \Delta^\bot \]  
  ElimeqBin

Either \( \Delta^\bot \) is some \( \Delta \), or it is \( \bot \).

  – Case \( \Delta^\bot = \Delta \):
Proof of Lemma 87 (Soundness of Equality Elimination)

\[ \Gamma \vdash \tau_1 \equiv \tau'_1 : \star \rightarrow \Theta \]

Subderivation

\[ \Theta = (\Gamma, \Delta_1) \]

By i.h. (1)

(H-1st) \[ [\Omega, \Delta_1]u_1 = [\theta_1][\Omega]u_1 \]

\[ \theta_1 = \text{mgu}(\tau_1, \tau'_1) \]

\[ \Theta \vdash [\Theta]\tau_1 \equiv [\Theta]\tau'_1 : \star \rightarrow \Delta \]

Subderivation

\[ \Delta = (\Theta, \Delta_2) \]

By i.h. (1)

(H-2nd) \[ [\Omega, \Delta_1, \Delta_2]u_2 = [\theta_2][\Omega, \Delta_1]u_2 \]

\[ \theta_2 = \text{mgu}(\tau_2, \tau'_2) \]

Suppose \( \Omega \vdash u : \kappa' \).

\[ [\Omega, \Delta_1, \Delta_2]u = [\theta_2][\Omega, \Delta_1]u \]

By (IH-2nd), with \( u_2 = u \)

\[ [\Omega, \Delta_1]u = [\theta_1][\Omega]u \]

By (IH-1st), with \( u_1 = u \)

\[ \Theta \vdash [\Theta]u \]

By a property of substitution

\[ \Theta = [\theta_2 \circ \theta_1]u \]

By property (iii) of substitution

\[ Z = [\Omega]u \]

By a property of substitution

\[ Z = \text{mgu}((\tau_1 \oplus \tau_2), (\tau'_1 \oplus \tau'_2)) \]

By property (iii) of substitution

– Case \( \Delta \perp = \perp \):

Use the i.h. (2) on the second premise to show \( \text{mgu}(\tau_2, \tau'_2) = \perp \), then use property (v) of unification to show \( \text{mgu}((\tau_1 \oplus \tau_2), (\tau'_1 \oplus \tau'_2)) = \perp \).

– Case \( \Gamma / \tau_1 \vdash \tau'_1 : \star \rightarrow \perp \)

Similar to the \( \perp \) subcase for ElimeqSucc, but using property (v) instead of property (ii).

– Case \( \sigma \# t \)

Since \( \sigma \# t \), we know \( \sigma \) and \( t \) have different head constructors, and thus no unifier. \( \square \)

Theorem 6 (Soundness of Algorithmic Subtyping).

If \( [\Gamma]A = A \) and \( [\Gamma]B = B \) and \( \Gamma \vdash A \) type and \( \Gamma \vdash B \) type and \( \Delta \rightarrow \Omega \) and \( \Gamma \vdash A \lesssim B \rightarrow \Delta \) then \( [\Omega]A \lesssim B \).

Proof. By induction on the given derivation.

– Case \( B \) not headed by \( \forall \)

\[ \Gamma, [\bar{A}, \Theta] \vdash \kappa \vdash [\bar{A}/\alpha]A_0 \lesssim B \rightarrow \Delta, [\bar{A}, \Theta] \]

\[ \Gamma \vdash \forall [\alpha]A_0 \lesssim B \rightarrow \Delta \]

Let \( \Omega' = (\Omega, [\bar{A}, \Theta]) \).
Proof of Theorem 6 (Soundness of Algorithmic Subtyping)  thm:subtyping-soundness

Given

\[ \Gamma, \beta : \kappa \vdash A \vdash B \Rightarrow \Delta, \beta : \kappa, \Theta \]

Subderivation

\[ \Delta \rightarrow \Omega \]

\[ (\Delta, \beta : \kappa, \Theta) \rightarrow \Omega' \]

\[ \Gamma, \beta : \kappa \vdash A \vdash B \Rightarrow \Delta, \beta : \kappa, \Theta \]

By Lemma 24 (Filling Completes)

Subderivation

\[ \Gamma, \beta : \kappa \vdash A \vdash B \Rightarrow \Delta, \beta : \kappa, \Theta \]

By Lemma 21 (Extension Inversion) (i)

Given

\[ \Theta \text{ is soft} \]

\[ \Delta, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 21 (Extension Inversion) (ii)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 53 (Completing Subderivation)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 51 (Context Partitioning) + thinning

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By def. of substitution

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By def. of substitution

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 16 (Substitution Stability)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 49 (Subtyping Extension)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 46 (Filling Completes)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 21 (Extension Inversion) (i)

Given

\[ \Theta \text{ is soft} \]

\[ \Delta, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 21 (Extension Inversion) (ii)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 53 (Completing Subderivation)

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By Lemma 51 (Context Partitioning) + thinning

\[ \Omega, \beta : \kappa \vdash A \vdash B \Rightarrow \Omega' \]

By def. of substitution
Proof of Theorem 6 (Soundness of Algorithmic Subtyping)

\[ \Gamma \vdash A \equiv B \rightarrow \Delta \quad \text{Subderivation} \]
\[ \Delta \rightarrow \Omega \quad \text{Given} \]
\[ [\Omega]A = [\Omega]B \quad \text{By Lemma 85 (Soundness of Algorithmic Equivalence)} \]
\[ \Gamma \rightarrow \Delta \quad \text{By Lemma 48 (Equivalence Extension)} \]
\[ \Gamma \vdash A \text{ type} \quad \text{Given} \]
\[ [\Omega] \Gamma \vdash [\Omega]A \leq [\Omega]B \quad \text{By induction} \]
\[ \implies [\Omega] \Gamma \vdash [\Omega]A \leq [\Omega]B \quad \text{By } \leq \text{Reflexivity} \]

- **Case**  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{nonpos}(B) \]
  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{nonpos}(A) \]
  By inversion
  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{nonneg}(B) \]
  By inversion
  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{nonneg}(A) \]
  Since \text{neg}(A)
  \[ [\Omega] \Gamma \vdash [\Omega]A \leq^+ [\Omega]B \quad \text{By induction} \]

- **Case**  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{pos}(B) \]
  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{pos}(A) \]
  By inversion
  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{nonneg}(B) \]
  By inversion
  \[ \Gamma \vdash A <^+ B \rightarrow \Delta \quad \text{nonneg}(A) \]
  Similar to the \[ <^+ \] case.

L’.4 Soundness of Typing

**Theorem 7** (Soundness of Match Coverage).

*If* \( \Gamma \vdash \Pi \text{ covers } \vec{A} \) and \( \Gamma \rightarrow \rightarrow \Omega \) and \( \Gamma \vdash \vec{A} \text{ types and } [\Gamma] \vec{A} = \vec{\tilde{A}} \), then \( [\Omega] \Gamma \vdash \Pi \text{ covers } \vec{\tilde{A}} \).

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

- **Case**  \( \Gamma \vdash \cdot \Rightarrow e_1 \mid \ldots \text{ covers } \cdot \quad \text{CoversEmpty} \)
  \[ [\Omega] \Gamma \vdash \cdot \Rightarrow e_1 \mid \ldots \text{ covers } \cdot \quad \text{By DeclCoversEmpty} \]

- **Cases** \[ \text{CoversVar, Covers1, Covers+, Covers-} \]
  Use the i.h. and apply the corresponding declarative rule.
Proof of Theorem 7 (Soundness of Match Coverage)

Proof.

1. Suppose \( \Gamma \vdash \mathsf{ctx} \)

2. \( \Delta \vdash [\Delta] \Pi \text{ covers } [\Delta] A_0, [\Delta] \overline{A} \)

3. By i.h.

\[ \Delta = (\Gamma', \Theta) \]

By Lemma 87 (Soundness of Equality Elimination) (1)

\[ \text{Subderivation} \]

- Case \( \Gamma \vdash [\Gamma] t_1 \mod {\mathsf{ctx}} \)

- Case \( \Gamma \vdash [\Gamma] t_2 \mod {\mathsf{ctx}} \)

- Subderivation

\[ \text{By Lemma 87 (Soundness of Equality Elimination) (2)} \]

\[ \therefore [\Omega] \Gamma \vdash [\Omega] \Pi \text{ covers } [\Theta] A_0, [\Theta] \overline{A} \]

By above equalities

- \( \text{By } \text{DeclCoversEq} \)

- \( \text{By } \text{DeclCoversEqBot} \)

- \( \text{By } \text{DeclCoversEqBot} \)

\[ \Box \]

Lemma 88 (Well-formedness of Algorithmic Typing).

Given \( \Gamma \vdash \text{ctx} \):

- (i) If \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \) then \( \Delta \vdash A \ p \text{ type} \).

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

Proof.

1. Suppose \( \Gamma \vdash e \Rightarrow A \ p \vdash \Delta \):

- Case \( (x : A \ p) \in \Gamma \)

- Case \( \Gamma \vdash A ! \text{ type} \)

- Case \( \Gamma \vdash A ! \text{ type} \)

- Case \( \Gamma \vdash A ! \text{ type} \)

- Case \( \Gamma \vdash e \Rightarrow A \ p \vdash \Theta \)

- Case \( \Theta \vdash s : \{\Theta\} A \ p \gg C \ q \vdash \Delta \) or \( \text{FEV}(\{\Delta\} C) \neq \emptyset \)

\[ \therefore \]

- \( \text{By inversion} \)

- \( \text{By induction} \)

- \( \text{By implicit assumption} \)

- \( \text{By mutual induction} \)
Proof of Lemma 88 (Well-formedness of Algorithmic Typing)

• Case
  \[ \Gamma \vdash e \Rightarrow A ! \Theta \quad \Theta \vdash s : [\Theta] A \Rightarrow C : \Delta \quad \text{FEV}(\Delta|C) = \emptyset \]

  \[ \Gamma \vdash e \Rightarrow A \quad \vdash \Theta \vdash s : \Gamma \quad \vdash C \quad \vdash \Delta \]

  \[ \vdash \text{FEV}(\Delta|C) = \emptyset \]

  \[ \vdash \text{E-I} \]

  \[ \Gamma \vdash e \Rightarrow A \quad \vdash \Theta \vdash s : \Gamma \quad \vdash C \quad \vdash \Delta \]

  \[ \vdash \text{By inversion} \]

• Case
  \[ \Gamma \vdash e \Leftarrow A \quad \Theta \vdash s : [\Theta] A \Rightarrow C : \Delta \]

  \[ \Gamma \vdash e \Leftarrow A \quad \vdash \Theta \vdash s : \Gamma \quad \vdash C \quad \vdash \Delta \]

  \[ \vdash \text{By mutual induction} \]

  \[ \vdash \text{PrinicipalWF} \]

2. Suppose \( \Gamma \vdash s : A \Rightarrow B \ q \ : \Delta \) and \( \Gamma \vdash A \ p \ : \Delta \):

• Case
  \[ \Gamma \vdash \cdot : A \Rightarrow A \ p \ : \Delta \]

  \[ \Gamma \vdash A \ p \ : \Delta \quad \text{Given} \]

• Case
  \[ \Gamma \vdash e \Leftarrow A \quad \Theta \vdash s : [\Theta] B \Rightarrow C : \Delta \]

  \[ \vdash \Theta \vdash s : \Gamma \quad \vdash C \quad \vdash \Delta \]

  \[ \vdash \text{By induction} \]

  \[ \vdash \text{Inversion of Principal Typing} \]

  \[ \vdash \text{Extension Weakening for Principal Typing} \]

  \[ \vdash \text{Right-Hand Subst. for Principal Typing} \]

• Case
  \[ \Gamma, \alpha : k \vdash e \Leftarrow s : [\alpha/\alpha] A \Rightarrow C : \Delta \]

  \[ \vdash \Gamma, \alpha : k \vdash e \Leftarrow s : [\alpha/\alpha] A \Rightarrow C : \Delta \]

  \[ \vdash \text{By weakening} \]

  \[ \vdash \text{Inversion} \]

  \[ \vdash \text{Inversion} \]

• Case
  \[ \Gamma \vdash P \quad \text{true} \ : \Delta \quad \Theta \vdash s : [\Theta] A \Rightarrow C : \Delta \]

  \[ \vdash \Theta \vdash s : \Gamma \quad \vdash C \quad \vdash \Delta \]

  \[ \vdash \text{By substitution} \]

  \[ \vdash \text{Inversion} \]

• Case
  \[ \Theta \vdash \alpha : k \Rightarrow \Delta \]

  \[ \vdash \Theta \vdash s : \alpha \Rightarrow C : \Delta \]

  \[ \vdash \Theta \vdash s : [\alpha/\alpha] A \Rightarrow C : \Delta \]

  \[ \vdash \Delta \vdash C \ q \ : \Delta \quad \text{By induction} \]

  \[ \vdash \text{Checkprop Extension} \]

  \[ \vdash \text{Extension Weakening for Principal Typing} \]

  \[ \vdash \text{Right-Hand Subst. for Principal Typing} \]

  \[ \vdash \text{By induction} \]

• Case
  \[ \Theta \vdash \alpha : k \Rightarrow \Delta \quad \Gamma \vdash e \Rightarrow s : \alpha \Rightarrow C : \Delta \]

  \[ \vdash \Theta \vdash e \Rightarrow s : \alpha \Rightarrow C : \Delta \]

  \[ \vdash \Theta \vdash e \Rightarrow s : \alpha \Rightarrow C : \Delta \]

  \[ \vdash \Theta \vdash e \Rightarrow s : \alpha \Rightarrow C : \Delta \]

  \[ \vdash \Delta \vdash C \ q \ : \Delta \quad \text{By induction} \]

  \[ \vdash \text{By rules} \]

  \[ \vdash \text{By induction} \]

\[ \square \]
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 \( [\Omega] \Delta \vdash [\Omega]e \leftarrow [\Omega]A \ p \).

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

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

(iv) If \( \Gamma \vdash s : A \ p \vdash \Delta \) and \( \Gamma \vdash A \ p \) 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 \( [\Omega] \Delta / [\Omega] \Pi : [\Omega] \tilde{A} \leftarrow [\Omega]C \ p \).

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

Proof. By induction, using the measure in Definition 7.

\[ \begin{align*}
\text{Case} & \quad \{x : A \ p\} \in \Gamma \\
\Gamma \vdash x & \Rightarrow [\Gamma] A \ p \vdash \Gamma \text{ Var} \\
\{x : A \ p\} \in \Gamma & \quad \text{Premise} \\
\{x : A \ p\} \in \Delta & \quad \Gamma = \Delta \\
\Delta \rightarrow \Omega & \quad \text{Given} \\
\{x : [\Omega]A \ p\} \in [\Omega] \Gamma & \quad \text{By Lemma 8 (Uvar Preservation) (ii)} \\
[\Omega] \Gamma \vdash [\Omega] x & \Rightarrow [\Omega] A \ p & \text{By DeciVar} \\
\Delta \rightarrow \Omega & \quad \text{Given} \\
\Gamma \rightarrow \Omega & \quad \Gamma = \Delta \\
[\Omega] A = [\Omega] [\Gamma] A & \quad \text{By Lemma 28 (Substitution Monotonicity) (iii)} \\
[\Omega] \Gamma \vdash [\Omega] x & \Rightarrow [\Omega] [\Gamma] A & \text{By above equality}
\end{align*} \]

\[ \begin{align*}
\text{Case} & \quad \Gamma \vdash e \leftarrow A \ q \vdash \Theta \vdash A \vdash ; : B \vdash \Delta \\
\Gamma \vdash e & \leftarrow B \ p \vdash \Delta & \text{Sub} \\
\Gamma \vdash e & \leftarrow A \ q \vdash \Theta & \text{Subderivation} \\
\Theta \vdash A \vdash ; : B \vdash \Delta & \text{Subderivation} \\
\Theta \rightarrow \Delta & \quad \text{By Lemma 50 (Typing Extension)} \\
\Delta \rightarrow \Omega & \quad \text{Given} \\
\Theta \rightarrow \Omega & \quad \text{By Lemma 32 (Extension Transitivity)} \\
[\Omega] \Theta \vdash [\Omega] e & \Rightarrow [\Omega] A \ q & \text{By i.h.} \\
[\Omega] \Theta = [\Omega] \Delta & \quad \text{By Lemma 55 (Confluence of Completeness)} \\
[\Omega] \Delta \vdash [\Omega] e & \Rightarrow [\Omega] A \ q & \text{By above equality} \\
\Theta \vdash A \vdash ; : B \vdash \Delta & \text{Subderivation} \\
[\Omega] \Delta \vdash [\Omega] A \vdash ; : [\Omega] B & \text{By Theorem 6 (Soundness of Algorithmic Subtyping)} \\
[\Omega] \Delta \vdash [\Omega] e & \Rightarrow [\Omega] B & \text{By DeciSub} \\
\end{align*} \]

\[ \begin{align*}
\text{Case} & \quad \Gamma \vdash A_0 \vdash \text{type} \\
\Gamma \vdash e_0 & \leftarrow [\Gamma] A_0 \vdash \Delta & \text{Anno} \\
\end{align*} \]

\[ \begin{align*}
\Gamma \vdash e_0 & \leftarrow [\Gamma] A_0 \vdash \Delta & \text{Subderivation} \\
[\Omega] \Delta \vdash [\Omega] e_0 & \leftarrow [\Omega] [\Gamma] A_0 & \text{By i.h.} \\
\end{align*} \]

\[ \begin{align*}
\Gamma \vdash A_0 & \vdash \text{type} & \text{Subderivation} \\
\Gamma \vdash A_0 & \text{type} & \text{By inversion} \\
\text{FEV}(A_0) & = \emptyset \\
\end{align*} \]
Proof of Theorem 8 (Soundness of Algorithmic Typing) thm:typing-soundness

\[ \Gamma \rightarrow \Delta \]
\[ \Delta \rightarrow \Omega \]
\[ \Gamma \rightarrow \Omega \]
\[ \Omega \vdash A_0 \text{ type} \]
\[ \Omega \vdash [\Omega]A_0 = [\Omega]A_0 \]
\[ \Omega \vdash [\Omega]A_0 \]
\[ \Omega \vdash (\forall \alpha. \forall \alpha. \Theta) \]
\[ \Theta \text{ is soft} \]
\[ \Theta \text{ is soft} \]
\[ \Omega \vdash [\Omega]A_0 \]
\[ \Omega \vdash (\Omega \forall \alpha. \forall \alpha. A_0) \]

**Case**

\[ \Gamma \vdash () \iff 1 \ p \rightarrow \Gamma \]
\[ [\Omega] \vdash () \iff 1 \ p \]
\[ \Gamma_0[\alpha : \times = 1] \]
\[ \Gamma_0[\alpha : \times = 1] \rightarrow \Omega \]
\[ \Gamma_0[\alpha : \times = 1] \rightarrow \Omega \]
\[ \Gamma_0[\alpha : \times = 1] \rightarrow \Omega \]
\[ \Gamma_0[\alpha : \times = 1] \rightarrow \Omega \]

**Case**

\[ v \text{ chk-I} \]
\[ \Gamma_\alpha : \kappa : v \iff A_0 \ p \rightarrow \Delta, \alpha : \kappa, \Theta \]
\[ \Gamma_\alpha : \kappa : v \iff A_0 \ p \rightarrow \Delta, \alpha : \kappa, \Theta \]

**Case**

\[ \Delta \rightarrow \Omega \]
\[ \Delta, \alpha \rightarrow \Omega, \alpha \]
\[ \Gamma_\alpha : \kappa : v \iff A_0 \ p \rightarrow \Delta, \alpha : \kappa, \Theta \]
\[ \Gamma_\alpha : \kappa : v \iff A_0 \ p \rightarrow \Delta, \alpha : \kappa, \Theta \]

\[ \Gamma_\alpha : \kappa : v \iff A_0 \ p \rightarrow \Delta, \alpha : \kappa, \Theta \]

\[ \Theta \text{ soft} \]
\[ \Theta \text{ soft} \]

\[ \Omega \vdash [\Omega]A_0 = [\Omega]A_0 \]
\[ \Omega \vdash [\Omega]A_0 \]
\[ \Omega \vdash [\Omega]A_0 \]

\[ \Omega \vdash [\Omega]A_0 \]
\[ \Omega \vdash [\Omega](\forall \alpha. \forall \alpha. A_0) \]

\[ [\Omega] \vdash [\Omega]A_0 \]
\[ [\Omega] \vdash [\Omega]A_0 \]

\[ [\Omega] \vdash [\Omega]A_0 \]
\[ [\Omega] \vdash [\Omega]A_0 \]

\[ [\Omega] \vdash [\Omega]A_0 \]

Proof of Theorem 8 (Soundness of Algorithmic Typing) thm:typing-soundness
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[ \text{Case } \Gamma, \delta : \kappa \vdash e \cdot s_0 : [\delta/\alpha]A_0 \not\gg C \not\vdash \Delta \]
\[ \Gamma \vdash e \cdot s_0 : \forall \alpha : \kappa. A_0 \not\gg C \not\vdash \Delta \]

\[ \text{Subderivation} \]

\[ \text{Case } e \text{ chk-I} \]
\[ \Gamma \vdash P \text{ true } \not\vdash \Theta \]
\[ \Theta \vdash e \leftarrow [\Theta]A_0 \not\vdash \Delta \]
\[ \Gamma \vdash e \leftarrow A_0 \land P \not\vdash \Delta \]

\[ \text{Subderivation} \]

\[ \Delta \rightarrow \Omega \]
\[ \Theta \rightarrow \Delta \]
\[ \Theta \rightarrow \Omega \]
\[ [\Omega] \Theta \vdash [\Omega]P \text{ true} \]
\[ [\Omega] \Delta \vdash [\Omega]P \text{ true} \]
\[ [\Theta] e \leftarrow [\Theta]A_0 \not\vdash \Delta \]
\[ [\Omega] \Delta \vdash [\Omega]e \leftarrow ([\Omega]A_0) \land [\Omega]P \not\vdash \Delta \]
\[ [\Omega] \Delta \vdash [\Omega]e \leftarrow ([\Omega]A_0) \land [\Omega]P \not\vdash \Delta \]

\[ \text{Subderivation} \]

\[ [\Omega] \Delta \vdash [\Omega]A_0 = [\Omega]A_0 \]
\[ [\Omega] \Delta \vdash [\Omega]e \leftarrow ([\Omega]A_0) \land [\Omega]P \not\vdash \Delta \]

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

\[ \text{By Lemma 28 (Substitution Monotonicity) (iii)} \]

\[ \text{By above equality} \]

\[ \text{By def. of substitution} \]
Proof of Theorem 8 (Soundness of Algorithmic Typing)

**Case** \( \nu \text{ chk-I} \)

\[
\begin{align*}
\Gamma, \rho_{\nu} & \vdash P, \Theta^+ & \Theta^+ & \vdash \nu \in (\Theta^+)A_0 & \vdash \Delta, \rho_{\nu}, A' \\
\Gamma & \vdash \nu \in P \supset A_0 & \vdash \Delta & \vdash
\end{align*}
\]

- **Given**
  \[\Theta^+ = (\Gamma, \rho_{\nu}, \Theta)\]
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'

- **Subderivation**
  \[\Theta^+ \vdash \nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]
  \[\Gamma \rightarrow \Theta^+\]
  \[\Theta^+ \rightarrow \Delta, \rho_{\nu}, A'\]

- **Subderivation**
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]
  \[\Theta^+ \vdash \nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]

- **By i.h.**
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]

- **By Lemma 32 (Extension Transitivity)**
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]

- **By Lemma 28 (Substitution Monotonicity)**
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]

- **By Lemma 92 (Substitution Upgrade) (ii)**
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]

- **By Lemma 92 (Substitution Upgrade) (iii)**
  \[\nu \in (\Theta^+)A_0 \]
  \[\vdash \Delta, \rho_{\nu}, A'\]

**Case** \( \nu \text{ chk-I} \)

\[
\begin{align*}
\Gamma, \rho_{\nu} & \vdash P & \vdash \nu \in P \supset A_0 & \vdash \Delta & \vdash
\end{align*}
\]

- **Subderivation**
  \[\nu \in P \supset A_0 \]
  \[\vdash \Delta \]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]

- **By inversion**
  \[\nu \in P \supset A_0 \]
  \[\vdash \Delta \]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]

- **As in (i) case (above)**
  \[\nu \in P \supset A_0 \]
  \[\vdash \Delta \]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]

- **By Lemma 87 (Soundness of Equality Elimination)**
  \[\nu \in P \supset A_0 \]
  \[\vdash \Delta \]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]
  \[\Gamma, \rho_{\nu} \rightarrow \Gamma, \rho_{\nu}, A'\]
Proof of **Soundness of Algorithmic Typing**

[Proof of Theorem 8](#)

**Case**

\[\Gamma \vdash P \text{ true} \vdash \Theta \quad \Theta \vdash e \cdot s_0 : [\Theta]A_0 \gg C q \vdash \Delta \]

\[\begin{align*}
\Theta & \vdash e \cdot s_0 : P \gg A_0 \gg C q \vdash \Delta \\
\Theta & \longrightarrow \Delta \\
\Delta & \longrightarrow \Omega \\
\Theta & \longrightarrow \Omega 
\end{align*}\]

\[\begin{align*}
[\Omega]\Delta \vdash [\Omega][e \cdot s_0] : [\Omega][\Theta]A_0 \gg [\Omega]C q \\
[\Omega][\Theta]A_0 = [\Omega]A_0 \\
[\Omega]\Delta \vdash [\Omega][e \cdot s_0] : [\Omega]A_0 \gg [\Omega]C q
\end{align*}\]

Subderivation

By **Lemma 50** *(Typing Extension)*

By **i.e.**

By **Lemma 28** *(Substitution Monotonicity)* (iii)

By above equality

Subderivation

By **Lemma 94** *(Completeness of Checkprop)*

By **Lemma 55** *(Confluence of Completeness)*

By above equality

\[\begin{align*}
[\Omega]\Delta \vdash [\Omega][e \cdot s_0] : [\Omega][P \gg A_0] \gg [\Omega]C q
\end{align*}\]

By **Dec \gg Spine**

By **def. of subst.**

**Case**

\[\Gamma, x : A_1 \vdash e_0 \ll A_2 p \vdash \Delta, x : A_1 p, \Theta \]

\[\Gamma \vdash \lambda x. e_0 \ll A_1 \rightarrow A_2 p \vdash \Delta \]

\[\begin{align*}
\Delta & \longrightarrow \Omega \\
\Delta, x : A_1 p \vdash \Omega, x : [\Omega]A_1 p \\
\Theta & \text{ soft}
\end{align*}\]

Subderivation

Given

By **Var**

By **Lemma 50** *(Typing Extension)*

By **Lemma 21** *(Extension Inversion)* (v)

(with \(\Gamma_{x} = \cdots\), which is soft)

By **Lemma 24** *(Filling Completes)*

\[\Delta, x : A_1 p, \Theta \longrightarrow \Omega, x : [\Omega]A_1 p, [\Theta]

\begin{align*}
\Theta & \text{ soft} \\
[\Omega']\Delta' \vdash [\Omega']e_0 \ll [\Omega']A_2 p
\end{align*}\]

Subderivation

By **Lemma 52** *(Softness Goes Away)*

\[\begin{align*}
[\Omega']\Delta' & = ([\Omega]\Delta, x : [\Omega]A_1 p) \\
[\Omega]\Delta, x : [\Omega]A_1 p \vdash [\Omega]e_0 \ll [\Omega]A_2 p \\
[\Omega]\Delta \vdash \lambda x. e_0 \ll [\Omega]A_1 p \rightarrow [\Omega]A_2 p
\end{align*}\]

By **Decl \gg \ll**

By definition of substitution

\[\begin{align*}
\Theta & \text{ soft} \\
[\Omega']\Delta' & \vdash [\Omega']\lambda x. e_0 \ll [\Omega'](A_1 \rightarrow A_2) p
\end{align*}\]

**Case**

\[\begin{align*}
\Gamma[\hat{\alpha}_1, x, \hat{\alpha}_2, x] \vdash x : \hat{\alpha}_1 \rightarrow \hat{\alpha}_2, x : \hat{\alpha}_1 J \vdash e_0 \ll \hat{\alpha}_2 J \vdash \Delta, x : \hat{\alpha}_1 J, \Theta \\
\Gamma[\hat{\alpha}_1, x] \vdash \lambda x. e_0 \ll \hat{\alpha} J \vdash \Delta
\end{align*}\]
Proof of \textit{Theorem 8 (Soundness of Algorithmic Typing)}

We show the quantified premise of \textit{DeclSpineRecover}:

\[
\begin{align*}
\Gamma[\alpha_1:*; \alpha_2:*; \alpha:* = \alpha_1 \rightarrow \alpha_2], \ x: \alpha \not\in \Delta, \ x: \alpha \not\in \Theta & \quad \text{By Lemma 50 (Typing Extension)} \\
\Delta \rightarrow \Omega & \quad \text{(with } \Gamma_R = \emptyset, \text{ which is soft)} \\
\end{align*}
\]

By \textit{DeclSpineRecover}.

By Lemma 21 (Extension Inversion) (v)

Given

By \textit{Var}

By Lemma 24 (Filling Completes)

\[
\begin{align*}
\Delta, \ x: \alpha_1 \not\in \Delta, \ x: \alpha_1 \not\in \Theta & \quad \text{Subderivation} \\
\Gamma[\alpha_1:*; \alpha_2:*; \alpha:* = \alpha_1 \rightarrow \alpha_2], \ x: \alpha \not\in \Delta, \ x: \alpha \not\in \Theta & \quad \text{Subderivation} \\
\end{align*}
\]

\[
\begin{align*}
\Omega[\alpha'] \Gamma' & \vdash [\Omega'][\alpha_2] \\
\Omega[\alpha'] \Gamma' & \vdash [\Omega][\alpha_1] \not\alpha_2 \\
\Omega[\alpha'] \Gamma' & \vdash [\Omega][\alpha_1] \not\alpha_2 & \quad \text{By definition of substitution} \\
\Omega[\alpha'] \Gamma' & \vdash [\Omega][\alpha_1] \not\alpha_2 & \quad \text{By definition of context substitution} \\
\Omega[\alpha'] \Gamma' & \vdash [\Omega][\alpha_1] \not\alpha_2 & \quad \text{By above equals} \\
\end{align*}
\]

Above and Lemma 32 (Extension Transitivity)

\[
\begin{align*}
[\Omega][\Delta \vdash \lambda x. [\Omega][e_0] \not\alpha_1 \not\alpha_1] & \quad \text{By above equality} \\
\end{align*}
\]

• Case

\[
\begin{align*}
\Gamma \vdash e_0 & \Rightarrow A \ q \not\Theta & \quad \text{\textbf{Case}} \\
\Theta & \vdash s_0: A \ q \not\Delta & \quad \text{\textbf{Case}} \\
\Gamma & \vdash s_0 & \Rightarrow C \ p \not\Delta \quad \text{By rule } \text{Decl} \rightarrow \text{E} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash e_0 & \Rightarrow A \ q \not\Theta & \quad \text{Subderivation} \\
\Theta & \vdash s_0: A \ q \not\Delta & \quad \text{Subderivation} \\
\Gamma & \rightarrow \Theta \text{ and } \Theta \rightarrow \Delta & \quad \text{By Lemma 50 (Typing Extension)} \\
\Delta & \rightarrow \Omega & \quad \text{Given} \\
\Theta & \rightarrow \Omega & \quad \text{By Lemma 32 (Extension Transitivity)} \\
\Gamma & \rightarrow \Omega & \quad \text{By Lemma 32 (Extension Transitivity)} \\
[\Omega][\Gamma] & = [\Omega][\Theta] & \quad \text{By Lemma 32 (Extension Transitivity)} \\
[\Omega][\Gamma] & \vdash [\Omega][\Delta] & \quad \text{By above equality} \\
[\Omega][\Gamma] & \vdash [\Omega][\Delta] & \quad \text{By Lemma 55 (Confluence of Completeness)} \\
[\Omega][\Gamma] & \vdash [\Omega][\Gamma][e_0] & \Rightarrow [\Omega][\Gamma][A] & \quad \text{By i.h.} \\
[\Omega][\Delta] & \vdash [\Omega][\Gamma][e_0] & \Rightarrow [\Omega][\Gamma][A] & \quad \text{By above equality} \\
[\Omega][\Gamma] & \vdash [\Omega][\Gamma][s_0: [\Omega][A] \Rightarrow [\Omega][C] [p]} & \quad \text{By i.h.} \\
\end{align*}
\]

\[
\begin{align*}
\lfloor [\Omega][\Gamma][e_0, s_0] \Rightarrow [\Omega][\Gamma][s_0: [\Omega][A] \Rightarrow [\Omega][C] [p]} & \quad \text{By rule } \text{Decl} \rightarrow \text{E} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash s: A \ ! \Rightarrow C \ f \not\Delta & \quad \text{FEV}(C) = \emptyset & \quad \text{SpineRecover} \\
\Gamma \vdash s: A \ ! \Rightarrow C \ f \not\Delta & \quad \text{Subderivation} \\
[\Omega][\Gamma] & \vdash [\Omega][s: [\Omega][A] \ ! \Rightarrow [\Omega][C] q] & \quad \text{By i.h.} \\
\end{align*}
\]

We show the quantified premise of \textit{DeclSpineRecover} namely,

\[
\begin{align*}
\quad \text{for all } C', & \quad \text{if } [\Omega][\Theta] \vdash s: [\Omega][A] \ ! \Rightarrow C' \ f \text{ then } C' = [\Omega][C] \\
\end{align*}
\]

Suppose we have \textit{C'} such that \textit{[\Omega][\Gamma] \vdash s: [\Omega][A] \ ! \Rightarrow C' \ f}. To apply \textit{DeclSpineRecover} we need to show \textit{C' = [\Omega][C]}. 

Proof of Theorem 8 (Soundness of Algorithmic Typing) thm:typing-soundness

We have thus shown the above “for all $\Gamma \vdash$ ...” statement.

- Case $\Gamma \vdash s : [\Gamma] A \vdash [\Gamma] C \downarrow$ By $\text{DeclSpinePass}$

  - Case $\Gamma \vdash s : A \vdash C \downarrow \Delta \quad ((p = f) \text{ or } (q = t) \text{ or } (FEV(C) \neq \emptyset))$

  \[
  \Gamma \vdash s : A \vdash C \downarrow \Delta \\
  \]

  - By $\text{SpinePass}$

  - Case $\Gamma \vdash s : A \vdash C \downarrow \Delta$

    - Subderivation $\Gamma \vdash [\Gamma] A \vdash [\Gamma] C q \vdash \Delta$

    \[
    [\Gamma] \vdash [\Gamma] s : [\Gamma] A \vdash [\Gamma] C q \vdash \Delta \\
    \]

    - By $\text{DeclSpinePass}$

- Case $\Gamma \vdash \cdot : A \vdash A \vdash \Gamma$

  \[
  \]

  - By $\text{DeclEmptySpine}$

- Case $\Gamma \vdash e_0 \leftarrow A_1 \vdash \Theta \quad \Theta \vdash s_0 : [\Theta] A_2 \vdash C q \downarrow \Delta$

  \[
  \Gamma \vdash e_0 \cdot s_0 : A_1 \rightarrow A_2 \vdash C q \downarrow \Delta \\
  \]

  - By $\text{Spine}$

  \[
  \Delta \rightarrow \Omega \\
  \Theta \rightarrow \Delta \\
  \Theta \rightarrow \Omega \\
  \]

  - By $\text{DeclSpineRecover}$

  - By Lemma 50 (Typing Extension)

  - By Lemma 32 (Extension Transitivity)

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

- Case $\Gamma \vdash e_0 \leftarrow A_1 \vdash \Theta$

  \[
  \]

  - Subderivation $\Gamma \vdash [\Theta] A_2 \vdash [\Theta] C q \vdash \Delta$

  \[
  [\Gamma] \Theta \vdash [\Gamma] e_0 \leftarrow [\Gamma] A_1 \vdash [\Gamma] C q \\
  [\Gamma] \Theta = [\Gamma] A_2 \\
  [\Gamma] \Theta \rightarrow [\Gamma] A_2 \rightarrow [\Gamma] C q \\
  [\Gamma] \Theta \vdash s_0 : [\Theta] A_2 \vdash [\Theta] C q \\
  [\Gamma] \Theta \vdash [\Theta] e_0 \cdot s_0 : [\Theta] A_1 \rightarrow A_2 \vdash [\Theta] C q \\
  [\Gamma] \Theta \vdash [\Theta] e_0 \cdot s_0 : (A_1 \rightarrow A_2) \vdash C q \\
  [\Gamma] \Theta \vdash \cdot : A \vdash A \vdash \Gamma \\
  \]

  - By Lemma 28 (Substitution Monotonicity)

  - By $\text{DeclSpine}$

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]

  \[
  \]
Proof of Theorem 8 (Soundness of Algorithmic Typing)

\[\Gamma \vdash e_0 \leftarrow A_k \ M \vdash \alpha \vdash \Delta \]

Subderivation

\[\Gamma \Delta \vdash \Gamma \vdash \Delta \]

By i.h.\[\text{Decl+1}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma[\Delta ; \vdash \alpha ] \vdash \Delta \]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash e_1 \leftarrow A_1 \ M \vdash \alpha \vdash \Delta \]

Subderivation

\[\Theta \vdash e_2 \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\Theta \vdash \Delta \]

By Lemma 32 (Extension Transitivity)

\[\text{Confluence of Completeness}\]

\[\Gamma \vdash e_1 \leftarrow A_1 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By i.h.

\[\text{Decl+1}\]

\[\text{Definition of substitution}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash e_2 \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\text{Extension Transitivity}\]

\[\text{Confluence of Completeness}\]

\[\Gamma \vdash e_1 \leftarrow A_1 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By i.h.

\[\Theta \vdash \Delta \]

By Lemma 55 (Confluence of Completeness)

\[\text{Substitution Monotonicity}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash e_2 \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\text{Extension Transitivity}\]

\[\text{Confluence of Completeness}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash (e_1, e_2) \Rightarrow \Delta \]

Similar to the +1\rightarrow k case, but using Lemma 50 (Typing Extension) and Lemma 55 (Confluence of Completeness) to show \(\Theta \vdash = \Delta \).

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash (e_1, e_2) \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\Theta \vdash \Delta \]

By Lemma 32 (Extension Transitivity)

\[\text{Confluence of Completeness}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash (e_1, e_2) \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\Theta \vdash \Delta \]

By Lemma 32 (Extension Transitivity)

\[\text{Confluence of Completeness}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash (e_1, e_2) \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\Theta \vdash \Delta \]

By Lemma 32 (Extension Transitivity)

\[\text{Confluence of Completeness}\]

\[\vdash \text{Def. of substitution}\]

\[\Gamma \vdash (e_1, e_2) \leftarrow \Theta \vdash A_2 \ p \vdash \Delta \]

Subderivation

\[\Theta \vdash \Delta \]

By Lemma 50 (Typing Extension)

\[\Theta \vdash \Delta \]

By Lemma 32 (Extension Transitivity)

\[\text{Confluence of Completeness}\]

\[\vdash \text{Def. of substitution}\]
Proof of Theorem 8 (Soundness of Algorithmic Typing)

Part (v):

- Case \( \Gamma \vdash \text{MatchEmpty} \)  
  Apply rule \( \text{DeclMatchEmpty} \)

- Case \( \Gamma \vdash e : C \dashv \Delta \)
  Apply the i.h. and \( \text{DeclMatchBase} \)

- Case \( \Gamma \vdash \text{MatchUnit} \)  
  Apply the i.h. and \( \text{DeclMatchUnit} \)

\[
\begin{align*}
[\Omega]\Delta \vdash \langle [\Omega]e_1, [\Omega]e_2 \rangle & \iff ([\Omega]\Delta_1) \times [\Omega]\Delta_2 \not\triangleright F \\
\text{By Decl} \times \text{I}

([\Omega]\Delta_1) \times [\Omega]\Delta_2 &= [\Omega]\Delta \\
\text{By above equality}

[\Omega]\Delta \vdash \langle [\Omega]e_1, [\Omega]e_2 \rangle & \iff [\Omega]\Delta \not\triangleright F \\
\text{By above equality}

\hline
\text{Case} \quad \Gamma[\Delta_2 : *, \Delta_1 : *] \vdash e_0 : [\Delta_1 \rightarrow \Delta_2] & \quad [\Delta_1 \rightarrow \Delta_2] \not\triangleright F \\
\text{Similar to the } \lambda \text{ case (above)} \\
\text{Subderivation}

\hline
\Gamma[\Delta] \not\triangleright \Gamma[\epsilon_0 \cdot s_0 : \Delta] \not\triangleright \Delta & \quad \text{By i.h.} \\
\text{Similar to the } \lambda \text{ case}

\hline
\text{Case} \quad \Gamma \vdash e_0 \Rightarrow B \dashv \Theta \quad \Theta \vdash \Pi : [\Theta]B & \iff [\Theta]C p \dashv \Delta \\
\Delta \vdash \Pi \text{ covers } [\Delta]B & \quad \text{By above equalities}

\begin{align*}
\Gamma & \vdash e_0 \Rightarrow B \dashv \Theta & \quad \text{Subderivation} \\
\Theta & \Rightarrow \Delta & \quad \text{By Lemma 50 \ (Typing Extension)} \\
\Theta & \Rightarrow \Omega & \quad \text{By Lemma 32 \ (Extension Transitivity)}

\text{[\Theta]B} & \Rightarrow [\Omega]B & \quad \text{By i.h.} \\
\text{[\Theta]C p} & \Rightarrow [\Omega]C p & \quad \text{By i.h. (v)}

\text{[\Theta]C} & \Rightarrow [\Omega]C & \quad \text{By Lemma 28 \ (Substitution Monotonicity)}

\text{[\Theta]C p} & \Rightarrow [\Omega]C p & \quad \text{By above equalities}

\begin{align*}
\text{[\Delta]\Delta} & \Rightarrow \Pi \text{ covers } [\Delta]B & \quad \text{Subderivation} \\
\text{[\Delta]\Delta} & \Rightarrow \Pi \text{ covers } [\Delta]B & \quad \text{By idempotence of substitution} \\
\Theta & \Rightarrow B \dashv \text{type} & \quad \text{By Lemma 62 \ (Well-Formed Outputs of Typing) \ (Synthesis)}

\text{[\Pi]C} & \Rightarrow [\Omega]C & \quad \text{By Lemma 39 \ (Right-Hand Subst. for Principal Typing)}

\end{align*}

\begin{align*}
\text{[\Delta]\Delta} & \Rightarrow \Pi \text{ covers } [\Delta]B & \quad \text{By above equalities} \\
\text{[\Delta]\Delta} & \Rightarrow [\Pi]C p & \Rightarrow [\Omega]C p & \quad \text{By above equalities}

\text{[\Delta]\Delta} & \Rightarrow [\Pi]C p & \Rightarrow [\Omega]C p & \quad \text{By above equalities}

\begin{align*}
\text{[\Delta]\Delta} & \Rightarrow [\Pi]C p & \Rightarrow [\Omega]C p & \quad \text{By DeclCase}

\end{align*}

Part (v):

- Case \( \text{MatchEmpty} \)  
  Apply rule \( \text{DeclMatchEmpty} \)

- Case \( \text{MatchUnit} \)  
  Apply the i.h. and \( \text{DeclMatchBase} \)

- Case \( \text{MatchUnit} \)  
  Apply the i.h. and \( \text{DeclMatchUnit} \)
Proof of Theorem 8 (Soundness of Algorithmic Typing)

• Case
  \[
  \Gamma \vdash \pi :: \vec{A} \leftarrow C \vdash \Theta \quad \Theta \vdash \Pi' :: \vec{A} \leftrightarrow C \vdash \Delta
  \]
  \[
  \Gamma \vdash \pi \mid \Pi' :: \vec{A} \leftarrow C \vdash \Delta
  \]
  MatchSeq

Apply the i.h. to each premise, using lemmas for well-formedness under \(\Theta\); then apply \(\text{DeclMatchSeq}\).

• Cases [Match] Match\^\wedge [MatchWild]

Apply the i.h. and the corresponding declarative match rule.

• Cases [Match\times] Match\_\times

We have \(\Gamma \vdash \vec{A}!\) types, so the first type in \(\vec{A}\) has no free existential variables.

Apply the i.h. and the corresponding declarative match rule.

• Case
  \[
  A \text{ not headed by } \land \text{ or } \exists \quad \Gamma, z : A! \vdash \rho \Rightarrow e' :: \vec{A} \leftarrow C \vdash \Delta, z : A!, \Delta'
  \]
  \[
  \Gamma \vdash z, \rho \Rightarrow e :: A, \vec{A} \leftarrow C \vdash \Delta
  \]
  MatchNeg

Construct \(\Omega'\) and show \(\Delta, z : A!, \Delta' \rightarrow \Omega'\) as in the \(\land\) case.

Use the i.h., then apply rule \(\text{DeclMatchNeg}\).

Part (vi):

• Case
  \[
  \Gamma / \sigma :: \tau : \kappa \vdash \perp
  \]
  \[
  \Gamma / \sigma :: \tau : \kappa \vdash \perp
  \]
  Subderivation

  \[
  [\Gamma](\sigma = \tau) = (\sigma = \tau)
  \]
  Given

  \[
  (\sigma = \tau) = [\Gamma](\sigma = \tau)
  \]
  Given

  \[
  = [\Omega](\sigma = \tau)
  \]
  By Lemma 28 (Substitution Monotonicity) (i)

  \[
  \text{mgu}(\sigma, \tau) = \perp
  \]
  By Lemma 87 (Soundness of Equality Elimination)

  \[
  \text{mgu}([\Omega]\sigma, [\Omega]\tau) = \perp
  \]
  By above equality

\(\Rightarrow [\Omega]/[\Omega](\sigma = \tau) \vdash [\Omega](\rho \Rightarrow e) :: [\Omega]\vec{A} \leftarrow [\Omega]C \vdash \Gamma \)

By \(\text{DeclMatch}\perp\).

• Case
  \[
  \Gamma, p / \sigma :: \tau : \kappa \vdash \Gamma'
  \]
  \[
  \Gamma' \vdash \rho \Rightarrow e :: \vec{A} \leftarrow C \vdash \Delta, \Gamma_p, \Delta'
  \]
  MatchUnify

  \[
  \Gamma, p / \sigma :: \tau : \kappa \vdash \Gamma'
  \]
  Subderivation

  \[
  (\sigma = \tau) = [\Gamma](\sigma = \tau)
  \]
  Given

  \[
  = [\Omega](\sigma = \tau)
  \]
  By Lemma 28 (Substitution Monotonicity) (i)

  \[
  \theta = \text{mgu}([\Omega]\sigma, [\Omega]\tau)
  \]
  By Lemma 87 (Soundness of Equality Elimination)

  \[
  [\Omega, \Gamma_p, \Theta]t' = [\theta][\Omega, \Gamma_p]t'
  \]
  \(\Rightarrow \) for all \(\Omega, \Gamma_p \vdash t' : \kappa'

  \[
  \Gamma, p, \Theta \vdash \rho \Rightarrow e \Rightarrow \vec{A} \leftarrow C \vdash \Delta, \Gamma_p, \Delta'
  \]
  Subderivation

  \[
  [\Omega, \Gamma_p, \Theta](\Delta, \Gamma_p, \Delta') \vdash [\Omega, \Gamma_p, \Theta](\rho \Rightarrow e) :: [\Omega, \Gamma_p, \Theta]\vec{A} \leftarrow [\Omega, \Gamma_p, \Theta]C \vdash \Gamma \]
  By i.h.

  \[
  (\Omega, \Gamma_p, \Theta) = [\theta][\Omega, \Gamma_p]
  \]
  By Lemma 92 (Substitution Upgrade) (iii)

  \[
  [\Omega, \Gamma_p, \Theta]\vec{A} = [\theta][\Omega, \Gamma_p]\vec{A}
  \]
  By Lemma 92 (Substitution Upgrade) (i)

  \[
  [\Omega, \Gamma_p, \Theta]C = [\theta][\Omega, \Gamma_p]C
  \]
  By Lemma 92 (Substitution Upgrade) (i)

  \[
  [\Omega, \Gamma_p, \Theta](\rho \Rightarrow e) = [\theta][\Omega, \Gamma_p](\rho \Rightarrow e)
  \]
  By Lemma 92 (Substitution Upgrade) (iv)

  \[
  \theta([\Omega, \Gamma_p]t') = [\theta][\Omega, \Gamma_p]t'
  \]
  By above equalities

  \[
  \text{Subst. not affected by } \Gamma_p
  \]

\(\Rightarrow [\Omega]/[\Omega](\sigma = \tau) \vdash [\Omega](\rho \Rightarrow e) :: [\Omega]\vec{A} \leftarrow [\Omega]C \vdash \Gamma \)

By \(\text{DeclMatchUnify}\).
M’ Completeness

M’.1 Completeness of Auxiliary Judgments

Lemma 89 (Completeness of Instantiation).
Given \( \Gamma \rightarrow \Omega \) and \( \text{dom}(\Gamma) = \text{dom}(\Omega) \) and \( \Gamma \vdash \tau : \kappa \) and \( \tau = [\Gamma]\tau \) and \( \hat{\alpha} \in \text{unsolved}(\Gamma) \) and \( \hat{\alpha} \notin \text{FV}(\tau) \):
If \( [\Omega]\hat{\alpha} = [\Omega]\tau \) then there are \( \Delta, \Omega' \) such that \( \Omega \rightarrow \Omega' \) and \( \Delta \rightarrow \Omega' \) and \( \text{dom}(\Delta) = \text{dom}(\Omega') \) and \( \Gamma \vdash \hat{\alpha} := \tau : \kappa \rightarrow \Delta \).

Proof. By induction on \( \tau \).
We have \([\Omega]\Gamma \vdash [\Omega]\hat{\alpha} \leq^* [\Omega]A\). We now case-analyze the shape of \( \tau \).

- Case \( \tau = \hat{\beta} \):
  \[ \hat{\alpha} \notin \text{FV}(\hat{\beta}) \]
  Given
  \[ \hat{\alpha} \neq \hat{\beta} \]
  From definition of \( \text{FV}(\cdot) \)
  \[ \hat{\beta} \in \text{unsolved}(\Gamma) \]
  From \([\Gamma]\hat{\beta} = \hat{\beta}\)
  Let \( \Omega' = \Omega \).
  \(\Rightarrow\) \(\Omega \rightarrow \Omega'\)
  By Lemma 31 (Extension Reflexivity)

Now consider whether \( \hat{\alpha} \) is declared to the left of \( \hat{\beta} \), or vice versa.

- Case \( \Gamma = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \):
  Let \( \Delta = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}] \).
  \(\Gamma \vdash \hat{\alpha} := \hat{\beta} : \kappa \rightarrow \Delta\)
  By \text{InstReach}
  \( [\Omega]\hat{\alpha} = [\Omega]\hat{\beta} \)
  Given
  \( \Gamma \rightarrow \Omega \)
  Given
  \(\Rightarrow\) \(\Delta \rightarrow \Omega\)
  By Lemma 26 (Parallel Extension Solution)
  \(\Rightarrow\) \(\text{dom}(\Delta) = \text{dom}(\Omega') \)
  \(\text{dom}(\Delta) = \text{dom}(\Gamma) \) and \(\text{dom}(\Omega') = \text{dom}(\Omega) \)

- Case \( \Gamma = \Gamma_0[\hat{\beta} : \kappa][\hat{\alpha} : \kappa] \):
  Similar, but using \text{InstSolve} instead of \text{InstReach}.

- Case \( \tau = \alpha \):
  We have \([\Omega]\hat{\alpha} = \alpha\), so (since \( \Omega \) is well-formed), \( \alpha \) is declared to the left of \( \hat{\alpha} \) in \( \Omega \).
  We have \( \Gamma \rightarrow \Omega \).
  By Lemma 20 (Reverse Declaration Order Preservation), we know that \( \alpha \) is declared to the left of \( \hat{\alpha} \) in \( \Gamma \); that is, \( \Gamma = \Gamma_1[\hat{\alpha} : \kappa][\hat{\alpha} : \kappa] \).
  Let \( \Delta = \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa = \alpha] \) and \( \Omega' = \Omega \).
  By \text{InstSolve} \( \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \alpha : \kappa \rightarrow \Delta \).
  By Lemma 26 (Parallel Extension Solution), \( \Gamma_1[\alpha : \kappa][\hat{\alpha} : \kappa = \alpha] \rightarrow \Omega \).
  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 \( \tau = \text{zero} \):
  Similar to the \( \tau = 1 \) case.

- Case \( \tau = \tau_1 \oplus \tau_2 \):

Proof of Lemma 89 (Completeness of Instantiation) lem:instantiation-completeness
Proof of Lemma 89 (Completeness of Instantiation)

\[ \Omega \hat{\alpha} = [\Omega](\tau_1 \oplus \tau_2) \]

Given

\[ = (\Omega|\tau_1) \oplus (\Omega|\tau_2) \]

By definition of substitution

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

Given

\[ \tau_1 = [\Gamma]\tau_1 \]

By definition of substitution and congruence

\[ \tau_2 = [\Gamma]\tau_2 \]

Similarly

\[ \hat{\alpha} \notin \text{FV}(\tau_1 \oplus \tau_2) \]

Given

\[ \hat{\alpha} \notin \text{FV}(\tau_1) \]

From definition of FV(−)

\[ \hat{\alpha} \notin \text{FV}(\tau_2) \]

Similarly

\[ \Gamma = \Gamma_0[\hat{\alpha} : \hat{s}] \]

By \( \hat{\alpha} \in \text{unsolved}(\Gamma) \)

\[ \Gamma \rightarrow \Omega \]

Given

\[ \Gamma_0[\hat{\alpha} : \hat{s}] \rightarrow \Gamma_0[\hat{\alpha}_2 : \hat{s}, \hat{\alpha}_1 : \hat{\alpha}, \hat{\alpha}_2 : \hat{\alpha}] \]

By Lemma 22 (Deep Evar Introduction) (i) twice

\[ \ldots, \hat{\alpha}_2, \hat{\alpha}_1 \vdash \hat{\alpha}_1 \oplus \hat{\alpha}_2 : \hat{\alpha} \]

Straightforward

\[ \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}] \rightarrow \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_2] \]

By Lemma 22 (Deep Evar Introduction) (ii)

\[ \Gamma_0[\hat{\alpha}] \rightarrow \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}_2] \]

By Lemma 32 (Extension Transitivity)

(In the last few lines above, and the rest of this case, we omit the "\( \hat{s} \)" annotations in contexts.)

Since \( \hat{\alpha} \in \text{unsolved}(\Gamma) \) and \( \Gamma \rightarrow \Omega \), we know that \( \Omega \) has the form \( \Omega_0[\hat{\alpha} = \tau_0] \).

To show that we can extend this context, we apply Lemma 22 (Deep Evar Introduction) (iii) twice to introduce \( \hat{\alpha}_2 = \tau_2 \) and \( \hat{\alpha}_1 = \tau_1 \), and then Lemma 27 (Parallel Variable Update) to overwrite \( \tau_0 \):

\[ \Omega_0[\hat{\alpha} = \tau_0] \rightarrow \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \]

We have \( \Gamma \rightarrow \Omega \), that is,

\[ \Gamma_0[\hat{\alpha}] \rightarrow \Omega_0[\hat{\alpha} = \tau_0] \]

By Lemma 25 (Parallel Admissibility) (i) twice, inserting unsolved variables \( \hat{\alpha}_2 \) and \( \hat{\alpha}_1 \) on both contexts in the above extension preserves extension:

\[ \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}] \rightarrow \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \tau_0] \]

By Lemma 25 (Parallel Admissibility) (ii) twice

\[ \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \rightarrow \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \]

By Lemma 27 (Parallel Variable Update)

Since \( \hat{\alpha} \notin \text{FV}(\tau_1) \), it follows that \( [\Gamma_1]|\tau = [\Gamma]|\tau = \tau \).

Therefore \( \hat{\alpha}_1 \notin \text{FV}(\tau_1) \) and \( \hat{\alpha}_1, \hat{\alpha}_2 \notin \text{FV}(\tau_2) \).

By Lemma 54 (Completing Completeness) (i) and (iii), \( [\Omega_1]|\Gamma_1 = [\Omega]|\Gamma \) and \( [\Omega_1]|\hat{\alpha}_1 = \tau_1 \).

By i.h., there are \( \Delta _2 \) and \( \Omega_2 \) such that \( \Gamma_1 \vdash \hat{\alpha}_1 : \tau_1 : \kappa \vdash \Delta_2 \) and \( \Delta_2 \rightarrow \Omega_2 \) and \( \Omega_1 \rightarrow \Omega_2 \).

Next, note that \( [\Omega_2]|\Delta_2 \tau_2 = [\Delta_2]\tau_2 \).

By Lemma 63 (Left Unsolvedness Preservation), we know that \( \hat{\alpha}_2 \in \text{unsolved}(\Delta_2) \).

By Lemma 64 (Left Free Variable Preservation), we know that \( \hat{\alpha}_2 \notin \text{FV}(\Delta_2)\tau_2 \).

By Lemma 32 (Extension Transitivity), \( \Omega \rightarrow \Omega_2 \).

We know \( [\Omega_2]|\Delta_2 = [\Omega]|\Gamma \) because:

\[ [\Omega_2]|\Delta_2 = [\Omega]|\Omega_2 \]

By Lemma 53 (Completing Stability)

\[ = [\Omega]|\Omega \]

By Lemma 54 (Completing Completeness) (iii)

\[ = [\Omega]|\Gamma \]

By Lemma 53 (Completing Stability)

By Lemma 54 (Completing Completeness) (i), we know that \( [\Omega_2]|\hat{\alpha}_2 = [\Omega_1]|\hat{\alpha}_2 = \tau_2 \).

By Lemma 54 (Completing Completeness) (i), we know that \( [\Omega_2]|\tau_2 = [\Omega]|\tau_2 \).

Hence we know that \( [\Omega_2]|\Delta_2 \vdash [\Omega_2]|\hat{\alpha}_2 \leq [\Delta_2]|\tau_2 \).

By i.h., we have \( \Lambda \) and \( \Omega ' \) such that \( \Delta_2 \vdash \hat{\alpha}_2 := [\Delta_2]|\tau_2 : \kappa \vdash \Lambda \) and \( \Delta_2 \rightarrow \Omega ' \) and \( \Lambda \rightarrow \Omega ' \).

By rule \text{InstBin}, \( \Gamma \vdash \hat{\alpha} := \tau : \kappa \vdash \Lambda \).

By Lemma 32 (Extension Transitivity), \( \Omega \rightarrow \Omega ' \).

- **Case** \( \tau = \text{succ}(\tau_0) \):

  Similar to the \( \tau = \tau_1 \oplus \tau_2 \) case, but simpler.
Lemma 90 (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 $\Gamma \vdash [\Omega] \sigma \downarrow [\Omega] \tau : \kappa \downarrow \Delta$

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

Proof. By mutual induction on the sizes of $[\Omega] \sigma$ and $[\Omega] \tau$.

We distinguish cases of $[\Omega] \sigma$ and $[\Omega] \tau$.

- **Case** $[\Omega] \sigma = [\Omega] \tau = 1$:

  $\Gamma \vdash 1 \downarrow 1 : * \darrow \Gamma$

  By CheckeqUnit

  Let $\Omega = \Omega$.

  $\Gamma \rightarrow \Omega$

  Given

  $\Delta \rightarrow \Omega'$

  $\Delta = \Gamma$ and $\Omega' = \Omega$

  $\Omega \rightarrow \Omega'$

  By Lemma 31 (Extension Reflexivity)

- **Case** $[\Omega] \sigma = [\Omega] \tau = 0$:

  Similar to the case for 1, applying CheckeqZero instead of CheckeqUnit

- **Case** $[\Omega] \sigma = [\Omega] \tau = \alpha$:

  Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit

- **Case** $[\Omega] \sigma = \alpha$ and $[\Omega] \tau = \beta$:

  - If $\alpha = \beta$: Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit

  - If $\alpha \neq \beta$:

    $\Gamma \rightarrow \Omega$

    Given

    $\alpha \notin \text{FV}(\beta)$

    By definition of FV(−)

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

    Given

    $[\Omega] \alpha = [\Omega] \beta$

    By Lemma 28 (Substitution Monotonicity) (i) twice

    $\Gamma \vdash \alpha \darrow [\Omega] \tau : \kappa \downarrow \Delta$

    By Lemma 89 (Completeness of Instantiation)

- **Case** $[\Omega] \sigma = \alpha$ and $[\Omega] \tau = 1$ or zero or $\alpha$:

  Similar to the previous case, except:

  $\alpha \notin \text{FV}(\frac{1}{\tau})$ By definition of FV(−)

  and similarly for 1 and $\alpha$.

- **Case** $[\Omega] \tau = \alpha$ and $[\Omega] \sigma = 1$ or zero or $\alpha$: Symmetric to the previous case.

- **Case** $[\Omega] \sigma = \alpha$ and $[\Omega] \tau = \text{succ}([\Omega] t_0)$:

  If $\alpha \notin \text{FV}([\Omega] t_0)$, then $\alpha \notin \text{FV}([\Omega] \tau)$. Proceed as in the previous several cases.

  The other case, $\alpha \in \text{FV}([\Omega] t_0)$, is impossible.
We have \( \delta \subseteq [\Gamma]t_0 \).
Therefore \( \delta \bowtie \text{succ}(\Gamma)t_0 \), that is, \( \delta \bowtie [\Gamma]t \).
By a property of substitutions, \([\Omega]\delta \bowtie [\Omega][\Gamma]t\).
Since \( [\Gamma] \rightarrow [\Omega] \), by Lemma 28 (Substitution Monotonicity) (i), \([\Omega][\Gamma]t = [\Omega]t\), so \([\Omega]\delta \bowtie [\Omega]t\).
But it is given that \([\Omega]\delta = [\Omega]t\), a contradiction.

- **Case** \([\Gamma]t = \delta \) and \([\Gamma]t = \text{succ}(\Gamma)t_0\): Symmetric to the previous case.

- **Case** \([\Gamma]t = \delta \) and \([\Gamma]t = \text{succ}(\Gamma)t_1 \oplus \text{succ}(\Gamma)t_2\):

  \[
  \begin{align*}
  \Gamma & \rightarrow [\Omega] \\
  [\Gamma]t & \vdash [\Gamma]t_1 \oplus [\Gamma]t_2 & \text{By i.h.} \\
  \Theta & \rightarrow [\Omega] & \text{By i.h.} \\
  \Omega & \rightarrow [\Omega] & \text{By i.h.} \\
  \text{dom}(\Theta) & = \text{dom}(\Omega) & \text{By Lemma 32 (Extension Transitivity)} \\
  \Theta & \vdash [\Theta][\Gamma]t_2 : \gamma \Delta & \text{By \text{CheckeqSucc}}
  \end{align*}
  \]

- **Case** \([\Gamma]t = \delta \) and \([\Gamma]t = \sigma_1 \oplus \text{succ}(\Gamma)t_2\):

  \[
  \begin{align*}
  \Gamma & \vdash [\Gamma]t_1 \oplus [\Gamma]t_2 : \gamma \Delta & \text{By \text{CheckeqSucc}}
  \end{align*}
  \]

- **Case** \([\Gamma]t = \delta \) and \([\Gamma]t = \sigma_1 \oplus \sigma_2\):

  Symmetric to the previous case. \(\Box\)

**Lemma 91** (Completeness of Elimeq).
If \([\Gamma]t = \sigma \) and \([\Gamma]t = t_1 \oplus t_2 \) 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 \(\Gamma/\sigma \vdash \kappa \lambda (\Gamma, \Delta)\)
   - where \(\Delta\) has the form \(\alpha_1 = t_1, \ldots, \alpha_n = t_n\)
   - and for all \(u\) such that \(\Gamma/\sigma \vdash u : \kappa\), 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 \(\Gamma/\sigma \vdash \kappa \lambda \perp\).

**Proof.** By induction on the structure of \([\Gamma]t\) and \([\Gamma]t\):

- **Case** \([\Omega\sigma] = \text{zero}\):

  \[
  \begin{align*}
  \text{mgu} & (\text{zero}, \text{zero}) = \cdot & \text{By properties of unification} \\
  \Gamma/\text{zero} \vdash \text{zero} : [\Gamma] & \text{By rule \text{ElimeqZero}} \\
  \Gamma/\text{zero} \vdash \text{zero} : [\Gamma, \Delta] & \text{where } \Delta = \cdot
  \end{align*}
  \]

- **Suppose** \(\Gamma/\text{u} : \kappa'\):

  \[
  \begin{align*}
  [\Gamma, \Delta]u & = [\Gamma]u & \text{where } \Delta = \cdot \\
  = \emptyset([\Gamma]u) & \text{where } \emptyset \text{ is the identity}
  \end{align*}
  \]

- **Case** \(\sigma = \text{succ}(\sigma')\) and \(t = \text{succ}(t')\):
  - **Case** \(\text{mgu}([\Gamma]t, \text{succ}(t')) = \emptyset\):

    \[
    \begin{align*}
    \text{mgu}([\Gamma]t, \text{succ}(t')) & = \text{mgu}([\Gamma]t, \text{succ}(t')) & \text{By properties of unification} \\
    \text{mgu} & = \text{mgu}([\Gamma]t, \text{succ}(t')) & \text{By definition of substitution} \\
    \text{mgu} & = \text{succ}([\Gamma]t') & \text{By injectivity of successor} \\
    \text{mgu} & = \text{succ}([\Gamma]t') & \text{By definition of substitution} \\
    \text{mgu} & = \text{succ}([\Gamma]t') & \text{By injectivity of successor} \\
    \text{mgu} & = \text{succ}([\Gamma]t') & \text{By i.h.} \\
    \text{mgu} & = \text{succ}([\Gamma]t') & \text{By \text{ElimeqSucc}}
    \end{align*}
    \]
Proof of Lemma 91 (Completeness of Elimeq)

\[
\begin{align*}
\text{Case} & \quad \Gamma / t \\
\text{Subcase} & \quad \ast = 1 \\
\text{By properties of unification} & \quad \text{Given} \\
\text{By definition of substitution} & \quad \text{By injectivity of successor} \\
\text{By definition of substitution} & \quad \text{By i.h.} \\
\text{By rule ElimeqSucc} & \\
\end{align*}
\]

1. Case \( \sigma = \sigma_1 \uplus \sigma_2 \) and \( t = t_1 \uplus t_2 \):

First we establish some properties of the subterms:

\[
\begin{align*}
\sigma_1 \uplus \sigma_2 & = [\Gamma](\sigma_1 \uplus \sigma_2) & \text{Given} \\
& = [\Gamma]\sigma_1 \uplus [\Gamma]\sigma_2 & \text{By definition of substitution} \\
\end{align*}
\]

\[
\begin{align*}
[\Gamma]\sigma_1 & = \sigma_1 & \text{By injectivity of } \uplus \\
[\Gamma]\sigma_2 & = \sigma_2 & \text{By injectivity of } \uplus \\
\end{align*}
\]

\[
\begin{align*}
t_1 \uplus t_2 & = [\Gamma](t_1 \uplus t_2) & \text{Given} \\
& = [\Gamma]\sigma_1 \uplus [\Gamma]\sigma_2 & \text{By definition of substitution} \\
\end{align*}
\]

\[
\begin{align*}
\text{By i.h.} & \\
\end{align*}
\]

- Subcase \( \text{mgu}(\sigma, t) = \perp \):

\[
\begin{align*}
\text{Subcase} & \quad \text{mgu}(\sigma_1, t_1) = \perp \\
\text{By i.h.} & \\
\text{Subcase} & \quad \text{mgu}(\sigma_1, t_1) = \theta_1 \text{ and } \text{mgu}(\sigma_2, \theta_1(t_2)) = \perp \\
\text{By i.h.} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\end{align*}
\]

1. Subcase \( \text{mgu}(\sigma_1, t_1) = \theta \):

\[
\begin{align*}
\text{mgu}(\sigma_1 \uplus \sigma_2, t_1 \uplus t_2) & = \theta \circ \theta_1 & \text{By properties of unifiers} \\
\text{mgu}(\sigma_1, t_1) & = \theta_1 & \text{By i.h.} \\
\end{align*}
\]

\[
\begin{align*}
[\Gamma, \Delta_1]u & = \theta_1([\Gamma]u) \text{ for all } u \text{ such that } \ldots & \" \\
[\Gamma, \Delta_1]\sigma_2 & = \theta_1([\Gamma]\sigma_2) & \text{Above line with } \sigma_2 \text{ as } u \\
& = \theta_1(\sigma_2) & \text{Since } [\Gamma]\sigma_2 = \sigma_2 \\
[\Gamma, \Delta_1]t_2 & = \theta_1([\Gamma]t_2) & \text{Above line with } t_2 \text{ as } u \\
& = \theta_1(t_2) & \text{Since } [\Gamma]\sigma_2 = \sigma_2 \\
\text{By Lemma 28 (Substitution Monotonicity)} & \\
\text{By Lemma 28 (Substitution Monotonicity)} & \\
\text{By i.h.} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\text{transitivity of equality} & \\
\end{align*}
\]

Proof of Lemma 91 (Completeness of Elimeq) lem:elimeq-completeness
Proof of Lemma 91 (Completeness of Elimeq). Let \( \Delta \) be a typing context and let \( \Gamma \vdash \theta \Delta \) for all \( u \) such that \( \Gamma \vdash u : \kappa \). Then there exists \( \tilde{\Theta} \) such that \( \tilde{\Theta} \Delta \) is provable.

Proof. By induction on the given derivations. There is only one possible case:

- Case \( \sigma = \alpha \):
  - Subcase \( \alpha \in \text{FV}(t) \):
    - \( \text{mg}(\alpha, t) = \perp \) By properties of unification
      - \( \Gamma \vdash \alpha \vdash t : \kappa \vdash \perp \) By rule \text{ElimeqUvarL}.
    - Subcase \( \alpha \notin \text{FV}(t) \):
      - \( \text{mg}(\alpha, t) = [t/\alpha] \) By properties of unification
        - \( (\alpha = t') \notin \Gamma \)
      - \( \Gamma \vdash \alpha \vdash t : \kappa \vdash \Gamma, \alpha = t \) By rule \text{ElimeqUvarL}.
  - \( \Gamma, \alpha = t \vdash u : \kappa' \).

Proof of Lemma 92 (Substitution Upgrade). If \( \Delta \) has the form \( \alpha_1 = t_1, \ldots, \alpha_n = t_n \)
and, for all \( u \) such that \( \Gamma \vdash u : \kappa \), it is the case that \( \Gamma, \Delta \vdash u = \theta(\Gamma)u \),
then:

(i) If \( \Gamma \vdash A \) type then \( \Gamma, \Delta \vdash A = \theta(\Gamma)A \).
(ii) If \( \Gamma \vdash \Omega \) then \( \overline{\Omega} \vdash \theta(\overline{\Omega}) \).
(iii) If \( \Gamma \vdash \overline{\Omega} \) then \( \overline{\Omega}, \Delta \vdash \theta(\overline{\Omega}) \).
(iv) If \( \Gamma \vdash \overline{\Omega} \) then \( \overline{\Omega}, \Delta \vdash \theta(\overline{\Omega}) \).

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 \( \overline{\text{Var}} \) case.
Part (iii): By induction on \( \Delta \). In the base case (\( \Delta = - \)), use the i.h. and the definition of context substitution.
Part (iv): By induction on \( e \), using part (i) in the \( e = (e_0 : A) \) case.

Lemma 93 (Completeness of Propequiv). Given \( \Omega \vdash \Omega \)
and \( \Gamma \vdash P \) prop and \( \Gamma \vdash Q \) prop
and \( [\Omega]P \equiv [\Omega]Q \)
then \( \Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \) \( \text{Var} \)
where \( \Delta \vdash \Omega' \) and \( \Omega \vdash \Omega' \).

Proof. By induction on the given derivations. There is only one possible case:

- Case \( \Gamma \vdash \sigma_1 : N \quad \Gamma \vdash \sigma_2 : N \quad \Gamma \vdash \tau_1 : N \quad \Gamma \vdash \tau_2 : N \quad \text{EqProp} \quad \text{EqProp} \)
Proof of Lemma 94 (Completeness of Checkprop)

\[
[\Omega](\sigma_1 = \sigma_2) = [\Omega](\tau_1 = \tau_2)
\]

\[
[\Omega]\sigma_1 = [\Omega]\tau_1
\]

\[
[\Omega]\sigma_2 = [\Omega]\tau_2
\]

Given

Definition of substitution

\[
\Gamma \vdash \sigma_1 : N
\]

Subderivation

\[
\Gamma \vdash \tau_1 : N
\]

\[
\Gamma \vdash [\Gamma]\sigma_1 \equiv [\Gamma]\sigma_2 : N \dashv \Theta
\]

\[
\Theta \rightarrow \Omega_0
\]

\[
\Omega \rightarrow \Omega_0
\]

\[
\Gamma \vdash \sigma_2 : N
\]

\[
\Theta \vdash \sigma_2 : N
\]

\[
\Theta \vdash [\Theta]\tau_1 \equiv [\Theta]\tau_2 : N \dashv \Delta
\]

\[
\Delta \rightarrow \Omega_0
\]

\[
\Omega_0 \rightarrow \Omega'
\]

\[
[\Theta]\tau_1 = [\Theta][[\Gamma]\tau_1
\]

\[
[\Theta]\tau_2 = [\Theta][[\Gamma]\tau_2
\]

\[
\Theta \vdash [\Theta][\Gamma]\tau_1 \equiv [\Theta][\Gamma]\tau_2 : N \dashv \Delta
\]

\[
\Omega \rightarrow \Omega'
\]

\[
\Gamma \vdash ([\Gamma]\sigma_1 = [\Theta]\sigma_2) \equiv ([\Gamma]\tau_1 = [\Theta]\tau_2) \dashv \Gamma
\]

\[
\Gamma \vdash ([\Gamma]\sigma_1 = [\Gamma]\sigma_2) \equiv ([\Gamma]\tau_1 = [\Gamma]\tau_2) \dashv \Gamma
\]

\[
\Gamma \vdash ([\Gamma]\sigma_1 \equiv [\Gamma]\sigma_2) \equiv ([\Gamma]\tau_1 \equiv [\Gamma]\tau_2) \dashv \Gamma
\]

\[
\Gamma \vdash ([\Gamma]t_1 = [\Gamma]t_2) \true \dashv \Delta.
\]

Lemma 94 (Completeness of Checkprop).

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

Proof. Only one rule, \textsf{DecCheckpropEd}, 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 90 (Completeness of Checkprop), \(\Gamma \vdash (t_1 = t_2) \true \dashv \Gamma\).
By \textsf{CheckpropEq}, \(\Gamma \vdash (t_1 = t_2) \true \dashv \Delta\).

M’.2 Completeness of Equivalence and Subtyping

Lemma 95 (Completeness of Equiv).

If \(\Gamma \rightarrow \Omega\) and \(\Gamma \vdash A\) type and \(\Gamma \vdash B\) type
and \([\Omega] A = [\Omega] B\)
then there exist \(\Delta\) and \(\Omega'\) such that \(\Delta \rightarrow \Omega'\) and \(\Omega \rightarrow \Omega'\) and \(\Gamma \vdash [\Gamma] A \equiv [\Gamma] B \dashv \Delta\).

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, \textsf{impliesWF} \textsf{WithWF} \textsf{ForallWF} \textsf{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 \textsf{impliesWF} \textsf{WithWF} \textsf{ForallWF} or \textsf{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 \textit{impos} (impossible), \textit{fully written out}, and \textit{similar to fully-written-out cases}. For the lower-right case, where both \([\Gamma] A\) and \([\Gamma] B\) have a binary connective \(+\), it must be the same connective.
Proof of Lemma 95 (Completeness of Equiv)

\[ \Gamma \vdash \emptyset \land \forall \beta. B' \lor \exists \beta. B' \lor \top \alpha \beta \lor B_1 \lor B_2 \]

- **Case** \( \Gamma \vdash P \text{ prop} \quad \Gamma \vdash A_0 \text{ type} \)
  \( \text{ImpliesWF} \)

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, \( \{\Omega\}(P \supset A_0) = \{\Omega\}B \).

Because \( \Omega \) is predicative, \( B \) must have the form \( Q \supset B_0 \), where \( \{\Omega\}P = \{\Omega\}Q \) and \( \{\Omega\}A_0 = \{\Omega\}B_0 \).

\[ \Gamma \vdash P \text{ prop} \quad \text{Subderivation} \]
\[ \Gamma \vdash A_0 \text{ type} \quad \text{Subderivation} \]
\[ \Gamma \vdash Q \supset B_0 \text{ type} \quad \text{Given} \]
\[ \Gamma \vdash Q \text{ prop} \quad \text{By inversion on rule ImpliesWF} \]
\[ \Gamma \vdash B_0 \text{ type} \quad \text{"} \]
\[ \Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \leftarrow \Theta \quad \text{By Lemma 93 (Completeness of Propequiv)} \]
\[ \Theta \rightarrow \Omega_0 \quad \text{"} \]
\[ \Omega \rightarrow \Omega_0 \quad \text{"} \]

\[ \Gamma \rightarrow \Theta \quad \text{By Lemma 47 (Prop Equivalence Extension)} \]
\[ \Gamma \vdash A_0 \text{ type} \quad \text{Above} \]
\[ \Gamma \vdash B_0 \text{ type} \quad \text{Above} \]
\[ [\Omega]A_0 = [\Omega]B_0 \quad \text{Above} \]
\[ [\Omega]B_0 = [\Omega]B_0 \quad \text{By Lemma 54 (Completing Completeness) (ii) twice} \]
\[ \Gamma \vdash [\Gamma]A_0 \equiv [\Gamma]B_0 \leftarrow \Delta \quad \text{By i.h.} \]
\[ \Rightarrow \Delta \rightarrow \Omega' \quad \text{"} \]
\[ \Omega_0 \rightarrow \Omega' \quad \text{"} \]

\[ \Omega \rightarrow \Omega' \quad \text{By Lemma 32 (Extension Transitivity)} \]
\[ \Gamma \vdash ([\Gamma]P \supset [\Gamma]A_0) \equiv ([\Gamma]Q \supset [\Gamma]B_0) \leftarrow \Delta \quad \text{By \( \equiv \rightarrow \)} \]
\[ \Rightarrow \Gamma \vdash [\Gamma](P \supset A_0) \equiv [\Gamma](Q \supset B_0) \leftarrow \Delta \quad \text{By definition of substitution} \]

- **Case** **WithWF** Similar to the ImpliesWF case, coinciding with the **With** entry in the table.

- **Case** \( \Gamma, \alpha : \kappa \vdash A_0 \text{ type} \quad \text{ForallWF} \)

This case coincides with the **Forall** entry in the table.
Proof of Lemma 95 (Completeness of Equiv)  lem:equiv-completeness

\[ \Gamma \longrightarrow \Omega \]
\[ \Gamma, \alpha : \kappa \longrightarrow \Omega, \alpha : \kappa \]
\[ \Gamma, \alpha : \kappa \vdash \Lambda_0 \quad \text{type} \]
\[ B = \forall \alpha : \kappa. B_0 \]
\[ [\Omega]A_0 = [\Omega]B_0 \]
\[ \Gamma, \alpha : \kappa \vdash [\Gamma]A_0 \equiv [\Gamma]B_0 \thicksim \Delta_0 \]
\[ \Delta_0 \longrightarrow \Omega_0 \]
\[ \Omega, \alpha : \kappa \longrightarrow \Omega_0 \]
\[ \equiv \quad \Omega \longrightarrow \Omega' \quad \text{and} \quad \Omega_0 = \{\Omega', \alpha : \kappa, \ldots\} \]
\[ \Delta_0 = \{\Delta, \alpha : \kappa, \Delta'\} \]
\[ \equiv \quad \Delta \longrightarrow \Omega' \]
\[ \equiv \quad \Gamma \vdash \forall \alpha : \kappa. [\Gamma]A_0 \equiv \forall \alpha : \kappa. [\Gamma]B_0 \thicksim \Delta \]
\[ \equiv \quad \forall \alpha : \kappa. [\Gamma]A_0 \equiv [\Gamma]B_0 \thicksim \Delta \]
\[ \equiv \quad \Gamma \vdash [\Gamma]B = 1 \]
\[ \equiv \quad \Gamma \vdash \alpha \equiv \alpha \thicksim \Gamma \]
\[ \equiv \quad \Delta \longrightarrow \Omega \]
\[ \equiv \quad \Omega \longrightarrow \Omega' \]
\[ \equiv \quad \Gamma \vdash [\Gamma]B = 1 \]
\[ \equiv \quad \Gamma \vdash \alpha \equiv \alpha \thicksim \Gamma \]
\[ \equiv \quad \Delta \longrightarrow \Omega \]
\[ \equiv \quad \Omega \longrightarrow \Omega' \]
\[ \equiv \quad \Gamma \vdash [\Gamma]B = 1 \]
\[ \equiv \quad \Gamma \vdash \alpha \equiv \alpha \thicksim \Gamma \]
\[ \equiv \quad \Delta \longrightarrow \Omega \]
\[ \equiv \quad \Omega \longrightarrow \Omega' \]

- **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 case, but on the first premise, using the i.h. instead of Lemma 93 (Completeness of Propequiv). This is the 2.Bins entry in the lower right corner of the table.

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
- **Case 2.Unvars**: [\Gamma]A = [\Gamma]B = \alpha
- **Case 2.AExUnit**: [\Gamma]A = \atilde and [\Gamma]B = 1.

Given

Subderivation

\[ \Delta = \Gamma \]

By Lemma 31 (Extension Reflexivity) and \( \Omega' = \Omega \)

By definition of substitution

By definition of substitution

By i.h.
Proof of Lemma 95 (Completeness of Equiv).

If \( \Gamma \) and \( \Omega \) are type systems and \( \Delta \) is a type, then there exist \( \Delta_{\text{dom}} \) and \( \Delta' \) such that:

- \( \Delta \rightarrow \Delta' \)
- \( \Delta_{\text{dom}} \rightarrow \Delta' \)
- \( \Delta' \rightarrow \Delta' \)

Symmetric to the 2.2ExUnit case.

Case 2.2ExUvar: \( [\Gamma]A = \alpha \) and \( [\Gamma]B = \beta \)

Similar to the 2.2ExUnit case, using \( \beta = [\Omega]\beta = [\Gamma]\beta \) and \( \beta \notin \text{FV}([\Gamma]B) \).

Case 2.2ExSameEx: \( [\Gamma]A = \alpha \) and \( [\Gamma]B = \beta \)

Given \( \Gamma \rightarrow \Omega \)

Let \( \Omega' = \Omega \).

Case 2.2ExOtherEx: \( [\Gamma]A = \alpha \) and \( [\Gamma]B = \beta \) and \( \alpha \neq \beta \)

Either \( \alpha \in \text{FV}([\Gamma]B) \), or \( \alpha \notin \text{FV}([\Gamma]B) \).

- \( \omega \in \text{FV}([\Gamma]B) \):
  - We have \( \alpha \leq [\Gamma]\beta \).
  - Therefore \( \alpha = \beta \), or \( \alpha < [\Gamma]\beta \).
  - But we are in Case 2.2ExOtherEx, so the former is impossible.
  - Therefore, \( \alpha < [\Gamma]\beta \).
  - By a property of substitutions, \( [\Omega]\alpha < [\Omega][\Gamma]\beta \).
  - Since \( \Gamma \rightarrow \Omega \), by Lemma 28 (Substitution Monotonicity) (iii), \( [\Omega][\Gamma]\beta = [\Omega]\beta \), so \( [\Omega]\alpha < [\Omega]\beta \).
  - But it is given that \( [\Omega]\alpha = [\Omega]\beta \), a contradiction.

- \( \alpha \notin \text{FV}([\Gamma]B) \):
  - \( \Gamma \vdash \alpha : \ast \rightarrow \Delta \) By Lemma 89 (Completeness of Instantiation)
  - \( \Delta \rightarrow \Omega' \) "
  - \( \Omega \rightarrow \Omega' \) "

Case 2.2ExBin: \( [\Gamma]A = \alpha \) and \( [\Gamma]B = \Gamma_{1} \oplus \Gamma_{2} \)

Since \( [\Gamma]B \) is an arrow, it cannot be exactly \( \alpha \). By the same reasoning as in the previous case (2.2ExOtherEx), \( \alpha \notin \text{FV}([\Gamma]B) \).

- \( \Gamma \vdash \alpha : \ast \rightarrow \Delta \) By Lemma 89 (Completeness of Instantiation)
  - \( \Delta \rightarrow \Omega' \) "
  - \( \Omega \rightarrow \Omega' \) "
  - \( \Gamma \vdash [\Gamma]A = [\Gamma]B \) by \( \alpha \) \( B_{1} \oplus B_{2} \) by \( \equiv_{\text{Instantiatel}} \)

Case 2.2ExBin: \( [\Gamma]A = A_{1} \oplus A_{2} \) and \( [\Gamma]B = \beta \)

Symmetric to the 2.2ExBin case, applying \( \equiv_{\text{Instantiatel}} \) instead of \( \equiv_{\text{Instantiatel}} \).

\( \square \)

Theorem 9 (Completeness of Subtyping).

If \( \Gamma \rightarrow \Omega \) and \( \text{dom}(\Gamma) = \text{dom}(\Omega) \) and \( \Gamma \vdash A \) type and \( \Gamma \vdash B \) type

and \( [\Omega][\Gamma]A \leq_{\text{sub}} [\Omega][\Gamma]B \)

then there exist \( \Delta \) and \( \Omega' \) such that:

- \( \Delta \rightarrow \Omega' \)
- \( \text{dom}(\Delta) = \text{dom}(\Omega') \)
- \( \Omega \rightarrow \Omega' \)
- \( \Gamma \vdash [\Gamma]A <_{\text{sub}} [\Gamma]B \rightarrow \Delta \).

Proof of Theorem 9 (Completeness of Subtyping)
Proof of Theorem 9 (Completeness of Subtyping)  

Proof. By induction on the number of \( \forall / \exists \) quantifiers in \([\Omega]A\) and \([\Omega]B\).

It is straightforward to show \( \text{dom}(\Delta) = \text{dom}(\Omega') \); for examples of the necessary reasoning, see the proof of Theorem 11 (Completeness of Algorithmic Typing).

We have \([\Omega]A \leq^* [\Omega]B\).

- **Case** \( [\Omega] \vdash [\Omega]A \leq [\Omega]B \)
  
  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 \([\Omega]A = (\forall \alpha : \kappa. [\Omega]A_\alpha) = (\forall \alpha : \kappa. [\Omega]B_0) = [\Omega]B\).
    
    Let \( \Gamma_0 = (\Gamma, \alpha : \kappa, \tau : \chi, \delta : \kappa) \).
    
    Let \( \Omega_0 = (\Omega, \alpha : \kappa, \tau : \chi, \delta : \kappa = \alpha) \).
    
    * If \( \operatorname{pol}(A_0) \in \{-, 0\} \), then:
      
      (We elide the straightforward use of lemmas about context extension.)

      \[
      \begin{align*}
      [\Omega_0] \vdash [\Omega]A_0 & \leq [\Omega]A_0 & \text{By} \leq \text{RefL} \cr
      [\Omega_0] \vdash [\Omega]A_0 & \leq A_0 & \text{By def. of subst.} \cr
      \Delta_0 \rightarrow \Omega_0 & \quad & \text{By i.h. (fewer quantifiers)} \cr
      \Omega_0 \rightarrow \Omega & \quad & \" \cr
      \Gamma_0 \vdash [\Gamma_0][\delta/\alpha]A_0 & \leq [\Gamma]B_0 \rightarrow \Delta_0 & \text{By subst, unsolved in} \Gamma_0 \cr
      \Gamma_0 \vdash [\delta/\alpha][\Gamma]A_0 & \leq [\Gamma]B_0 \rightarrow [\Gamma]B_0 & \text{By subst, unsolved in} \Gamma_0 \cr
      \Gamma, \alpha : \kappa \vdash [\forall \alpha : \kappa. [\Gamma]A_\alpha] & \leq [\forall \alpha : \kappa. [\Gamma]B_0 \rightarrow [\Gamma]B_0] \rightarrow \Delta \quad & \text{By} \leq \forall L \cr
      \Gamma \vdash [\forall \alpha : \kappa. [\Gamma]A_\alpha] & \leq [\forall \alpha : \kappa. [\Gamma]B_0 \rightarrow [\Gamma]B_0] \rightarrow \Delta \quad & \text{By} \leq \forall R \cr
      \exists & \Delta \rightarrow \Omega & \text{By def. of subst.} \cr
      \exists & \Omega \rightarrow \Omega_0 & \text{By def. of subst.} \cr
      \end{align*}
\]

  - If \( \operatorname{pol}(A_0) = + \), then proceed as above, but apply \( \leq \text{RefL} \) instead of \( \leq \text{RefL} \) and apply \( \leq L \) after applying the i.h. (Rule \( \leq R \) also works.)

  - If \( A \) is not headed by \( \forall \): We have \( \operatorname{nonneg}([\Omega]A) \). Therefore \( \operatorname{nonneg}(A) \), and thus \( A \) is not headed by \( \exists \). Since the same quantifiers must also head \( B \), the conditions in rule \( \leq \text{R} \) are satisfied.

    \[
    \begin{align*}
    \Gamma & \rightarrow \Omega \quad \text{Given} \cr
    \Gamma \vdash [\Gamma]A & \equiv [\Gamma]B \rightarrow \Delta & \text{By Lemma 95 (Completeness of Equiv)} \cr
    \exists & \Delta \rightarrow \Omega' & \text{By Lemma 95 (Completeness of Equiv)} \cr
    \exists & \Omega \rightarrow \Omega' & \text{By Lemma 95 (Completeness of Equiv)} \cr
    \exists & \Gamma \vdash [\Gamma]A & \leq [\Gamma]B \rightarrow \Delta & \text{By} \leq \text{Equiv} \cr
    \end{align*}
\]

  - **Case** \( \leq \text{RefL} \)
    
    Symmetric to the \( \leq \text{RefL} \) case, using \( \leq L \) (or \( \leq R \)), and \( \leq : R \leq : L \) instead of \( \leq : R \leq : L \).

  - **Case** \( \leq \forall L \)
    
    We begin by considering whether or not \([\Omega]B\) is headed by a universal quantifier.

    - \([\Omega]B = (\forall \beta : \kappa'. B')\):
      
      \( [\Omega] \vdash \beta : \kappa' \rightarrow [\Omega]A \leq [\Omega]B \)  
      
      By Lemma 4 (Subtyping Inversion).
      
      The remaining steps are similar to the \( \leq \forall \text{Case} \).
Proof of **Theorem 9** (Completeness of Subtyping) thm:subtyping-completeness

- \([\Omega]B\) not headed by \(\forall\):
  \[
  [\Omega]\Gamma \vdash \tau : \kappa \quad \text{Subderivation}
  \]
  \[
  \Gamma \rightarrow \Omega \quad \text{Given}
  \]
  \[
  \Gamma, \alpha \vdash \Omega, \beta : \kappa \rightarrow \Omega, \alpha, \beta : \kappa = \tau \quad \text{Subderivation}
  \]
  \[
  [\Omega]\Gamma = [\Omega_0](\Gamma, \alpha, \beta : \kappa) \quad \text{By definition of context application (lines 16, 13)}
  \]
  \[
  [\Omega]\Gamma \vdash [\tau/\alpha]|\Omega|A_0 \leq [\Omega]|B \quad \text{Subderivation}
  \]
  \[
  [\Omega_0](\Gamma, \alpha, \beta : \kappa) \vdash [\tau/\alpha]|\Omega|A_0 \leq [\Omega]|B \quad \text{By above equality}
  \]
  \[
  [\Omega_0](\Gamma, \alpha, \beta : \kappa) \vdash [\Omega]|\alpha/\alpha|\Omega|A_0 \leq [\Omega]|B \quad \text{By definition of substitution}
  \]
  \[
  [\Omega_0](\Gamma, \alpha, \beta : \kappa) \vdash [\Omega]|\alpha/\alpha|\Omega|A_0 \leq [\Omega]|B \quad \text{By definition of substitution}
  \]
  \[
  [\Omega_0](\Gamma, \alpha, \beta : \kappa) \vdash [\Omega_0]|\alpha/\alpha|A_0 \leq [\Omega]|B \quad \text{By distributivity of substitution}
  \]

\[
\Gamma, \alpha, \beta : \kappa \vdash \Gamma, \alpha, \beta : \kappa [\alpha/\alpha]|A_0 \leq [\Gamma]|B + \Delta_0 \quad \text{By i.h. (A lost a quantifier)}
\]
\[
\Delta_0 \quad \rightarrow \quad \Omega''
\]
\[
\Omega_0 \quad \rightarrow \quad \Omega''
\]

\[
\Gamma, \alpha, \beta : \kappa \vdash \Gamma, \alpha, \beta : \kappa [\alpha/\alpha]|A_0 \leq [\Gamma]|B + \Delta_0 \quad \text{By definition of substitution}
\]

\[
\Gamma, \alpha, \beta : \kappa \rightarrow \Delta_0 \quad \text{By Lemma 49 (Subtyping Extension)}
\]
\[
\Delta_0 = (\Delta, \alpha, \beta, \Theta) \quad \text{By Lemma 21 (Extension Inversion) (ii)}
\]
\[
\Gamma \rightarrow \Delta \quad \text{"}
\]
\[
\Omega'' = (\Omega', \alpha, \beta, \Theta) \quad \text{By Lemma 21 (Extension Inversion) (ii)}
\]
\[
\Delta \rightarrow \Omega' \quad \text{"}
\]
\[
\Omega_0 \rightarrow \Omega'' \quad \text{Above}
\]
\[
\Omega, \alpha, \beta : \kappa \rightarrow \Omega', \alpha, \beta \quad \text{By above equalities}
\]
\[
\Omega \rightarrow \Delta' \quad \text{By Lemma 21 (Extension Inversion) (ii)}
\]
\[
\Gamma, \alpha, \beta : \kappa \vdash \Gamma, \alpha, \beta : \kappa [\alpha/\alpha]|A_0 \leq [\Gamma]|B + \Delta_0 \quad \text{By above equality} \Delta_0 = (\Delta, \alpha, \beta, \Theta)
\]
\[
\Gamma, \alpha, \beta : \kappa \vdash \Gamma, \alpha, \beta : \kappa [\alpha/\alpha]|A_0 \leq [\Gamma]|B + \Delta, \alpha, \beta, \Theta \quad \text{By def. of subst. ([\Gamma]|\alpha = \Delta \text{ and } [\Gamma]|\alpha = \alpha)}
\]
\[
[\Gamma]|\beta : \kappa \text{ not headed by } \forall \quad \text{From the case assumption}
\]
\[
\Gamma \vdash \forall \alpha : \kappa, [\Gamma]|A_0 \leq [\Gamma]|B + \Delta \quad \text{By } [\exists : \forall]L
\]
\[
\Gamma \vdash \forall \alpha : \kappa, \forall \beta : \kappa, [\Omega]|A_0 \leq [\Gamma]|B + \Delta \quad \text{By definition of substitution}
\]

**Case**

\[
([\Omega]|\Gamma \beta : \kappa \vdash [\Omega]|A \leq [\Omega]|B_0 \quad \text{\(\forall\beta : \kappa, [\Omega]|B_0 \leq [\Omega]|B\)}
\]

\[
B = \forall \beta : \kappa, B_0 \quad \Omega \text{ predicate}
\]
\[
[\Omega]\Gamma \vdash [\Omega]|A \leq [\Omega]|B \quad \text{Given}
\]
\[
[\Omega]\Gamma \vdash [\Omega]|A \leq \forall \beta, [\Omega]|B_0 \quad \text{By above equality}
\]
\[
[\Omega]\Gamma \beta : \kappa \vdash [\Omega]|A \leq [\Omega]|B_0 \quad \text{Subderivation}
\]
\[
[\Omega] \beta : \kappa \vdash [\Omega]|A \leq [\Omega]|B_0 \quad \text{By definitions of substitution}
\]
\[
[\Omega] \beta : \kappa \vdash [\Omega]|B_0 \quad \text{By i.h. (B lost a quantifier)}
\]
\[
[\Gamma]|\beta : \kappa \rightarrow [\Gamma]|A \leq [\Gamma]|B_0 \quad \text{By definition of substitution}
\]
\[
[\Gamma]|\beta : \kappa \rightarrow [\Gamma]|B_0 \quad \text{By Lemma 42 (Instantiation Extension)}
\]
\[
[\Gamma]|\beta : \kappa \rightarrow \Delta' \quad \text{By Lemma 21 (Extension Inversion) (i)}
\]
\[
\Delta' = (\Delta, \beta : \kappa, \Theta) \quad \text{By Lemma 21 (Extension Inversion) (i)}
\]
\[
\Delta \rightarrow \Delta' \quad \text{"}
\]
\[
\Delta, \beta : \kappa, \Theta \rightarrow \Omega_0' \quad \text{By } [\Delta' \rightarrow \Omega' \text{ and above equality}}
\]
\[
\Delta, \beta : \kappa, \Theta \rightarrow \Omega_0' \quad \text{By Lemma 21 (Extension Inversion) (i)}
\]
Proof of Theorem 9 (Completeness of Subtyping)

\[ \Gamma, \beta : \kappa \vdash [\Gamma]\lambda < : [\Gamma]B_0 \rightarrow \Delta, \beta : \kappa, \Theta \]

By above equality

\[ \Delta \rightarrow \Omega' \]

By Lemma 32 (Extension Transitivity)

\[ \Gamma \vdash [\Gamma]{\lambda} < : \forall \beta : \kappa, [\Gamma]B_0 \rightarrow \Delta \]

By \( \subseteq : \forall R \)

---

Case

\[ \Omega, \alpha : \kappa \vdash [\Omega]{\lambda} \leq [\Omega]B \]

By definition of substitution

\[ \| \rightarrow \exists \alpha : \kappa, [\Omega]{\lambda} \leq [\Omega]B \]

Ω predicative

\[ [\Omega] \Gamma \vdash [\Omega]A \leq^+ [\Omega]B \]

Given

\[ [\Omega] \Gamma \vdash [\Omega] \exists \alpha : \kappa, A_0 \leq^+ [\Omega]B \]

By above equality

\[ [\Omega] \Gamma, \alpha : \kappa \vdash [\Omega]{\lambda}A_0 \leq^+ [\Omega]B \]

Subderivation

\[ [\Omega, \alpha : \kappa] \Gamma, \alpha : \kappa \vdash [\Omega, \alpha : \kappa]A_0 \leq^+ [\Omega, \alpha : \kappa]B \]

By definitions of substitution

\[ \Gamma, \alpha : \kappa \vdash [\Gamma, \alpha : \kappa]A \leq^+ [\Gamma, \alpha : \kappa]B \rightarrow \Delta' \]

By i.h. (A lost a quantifier)

\[ \Delta' \rightarrow \Omega' \]

By definition of substitution

\[ \Omega, \alpha : \kappa \rightarrow \Omega' \]

By \( \subseteq : \forall R \)

\[ [\Omega, \alpha : \kappa] \Gamma \vdash [\Gamma]{\lambda}A \leq^+ [\Gamma]B_0 \rightarrow \Delta' \]

Case

\[ \Psi \vdash \tau : \kappa \]

By above equality

\[ \Psi \vdash [\Omega]{\lambda} \leq^+ [\tau/\beta]B_0 \]

By definition of substitution

\[ \psi \vdash [\Omega]{\lambda} \leq^+ [\tau/\beta]B_0 \]

We consider whether \( [\Omega]{\lambda} \) is headed by an existential.

If \( [\Omega]{\lambda} = \exists \alpha : \kappa', A' \):

\[ [\Omega] \Gamma \vdash \exists \alpha : \kappa', A' \leq^+ [\Omega]B \]

By Lemma 4 (Subtyping Inversion)

The remaining steps are similar to the \( \subseteq : \forall R \) case.

If \( [\Omega]{\lambda} \) not headed by \( \exists \):

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

Subderivation

\[ \Gamma \rightarrow \Omega \]

Given

\[ \Gamma \vdash \phi \rightarrow \Omega, \psi \]

By Marker

\[ \Gamma \vdash \phi, \beta : \kappa \rightarrow \Omega, \psi, \beta : \kappa \]

By Solve

\[ [\Omega] \Gamma = [\Omega] \Gamma \vdash \phi, \beta : \kappa \]

By definition of context application (lines 16, 13)

\[ [\Omega] \Gamma \vdash [\Omega]{\lambda} \leq^+ [\tau/\beta]B_0 \]

Subderivation

\[ [\Omega_0] \Gamma \vdash [\Omega_0]{\lambda} \leq^+ [\tau/\beta]B_0 \]

By above equality

\[ [\Omega_0] \Gamma \vdash [\Omega_0]{\lambda} \leq^+ [\tau/\beta]B_0 \]

By definition of substitution

\[ [\Omega_0] \Gamma \vdash [\Omega_0]{\lambda} \leq^+ [\tau/\beta]B_0 \]

By distribution of substitution
Proof of Theorem 9 (Completeness of Subtyping)

\[ \Gamma, [\varphi, \sigma, \xi : \kappa] \vdash A : R \quad [\Gamma, \varphi, \sigma, \xi : \kappa] \vdash B : S \quad \Delta \vdash C \]

By i.h. (B lost a quantifier)
Proof of Theorem 10 (Completeness of Match Coverage). Given $\Gamma \rightarrow \Omega$ such that $\text{dom}(\Gamma) = \text{dom}(\Omega)$:

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

(ii) If $\Gamma \vdash \text{A p type}$ and $[\Omega] \Gamma \vdash [\Omega]e \Rightarrow A \text{ p}$
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 $\Gamma \vdash e \Rightarrow A' \text{ p' } - \Delta$ and $A' = [\Delta]A'$ and $A = [\Omega']A'$.

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

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

(v) If $\Gamma \vdash -1 \text{ types}$ and $\Gamma \vdash C \text{ p type}$ and $[\Omega] \Gamma \vdash [\Omega] \overline{A} \leftarrow [\Omega]C \text{ p and } p' \subseteq p$
then there exist $\Delta$, $\Omega'$, and $C$
such that $\Delta \rightarrow \Omega'$ and $\text{dom}(\Delta) = \text{dom}(\Omega')$ and $\Omega \rightarrow \Omega'$
and $\Gamma \vdash \overline{\Pi} : [\Gamma] \overline{A} \leftarrow [\Gamma]C \text{ p' } - \Delta$.

(vi) If $\Gamma \vdash -1 \text{ types}$ and $\Gamma \vdash \text{P prop and } \text{FEV(P) = } \emptyset$ and $\Gamma \vdash C \text{ p type}$
and $[\Omega] \Gamma \vdash [\Omega] \overline{P} \vdash [\Omega] \overline{A} \leftarrow [\Omega]C \text{ p}$
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 $\Gamma \vdash \overline{\Pi} \vdash \overline{\Pi} : [\Gamma] \overline{A} \leftarrow [\Gamma]C \text{ p' } - \Delta$.

Proof. By induction, using the measure in Definition[7]

- Case $[x : \text{A p}] \in [\Omega] \Gamma$

  $[\Omega] \Gamma \vdash x \Rightarrow A \text{ p}$

  Premise

  $[x : \text{A' p}] \in \Gamma$ where $[\Omega]A' = A$

  From definition of context application

  Let $\Delta = \Gamma$.

  Let $\Omega' = \Omega$.

  $\Gamma \rightarrow \Omega$

  Given

  $\Omega \rightarrow \Omega$

  By Lemma 31 (Extension Reflexivity)

  $\Gamma \vdash x \Rightarrow [\Gamma]A' \text{ p } - \Gamma$

  By Var

  $[\Gamma]A' = [\Gamma][\Gamma]A'$

  By idempotence of substitution

  $\text{dom}(\Gamma) = [\Gamma] \text{ dom}(\Omega)$

  Given

  $\Gamma \rightarrow \Omega$

  Given

  $[\Omega][\Gamma]A' = [\Omega]A'$

  By Lemma 28 (Substitution Monotonicity) (iii)

  $= A$

  By above equality
Case \(\Omega \vdash (\Omega)A \rightarrow 1\ p\) \hspace{1cm} \text{DeclI1}

We have \(\Omega A = 1\). Either \([\Gamma]A = 1\), or \([\Gamma]A = \alpha\) where \(\alpha \in \text{unsolved}(\Gamma)\).

In the former case:

Let \(\Delta = \Gamma\).

Let \(\Omega' = \Omega\).

\(\Gamma \rightarrow \Omega\) \hspace{1cm} \text{Given}

\(\Omega \rightarrow \Omega'\) \hspace{1cm} \text{By Lemma 31 (Extension Reflexivity)}

\(\text{dom}(\Gamma) = \text{dom}(\Omega)\) \hspace{1cm} \text{Given}

\(\Gamma \vdash \Omega 1 \rightarrow \Gamma\) \hspace{1cm} \text{By 11}

\(\Gamma \vdash \Omega 1 \rightarrow \Gamma\) \hspace{1cm} \(1 = [\Gamma]1\)
In the latter case, since \( A = \hat{\alpha} \) and \( \Gamma \vdash \hat{\alpha} p \) type is given, it must be the case that \( p = f \).

\[
\Gamma_0[\hat{\alpha} : \star] \vdash (\hat{\alpha} \eta f) \Gamma_0[\hat{\alpha} : \star] = 1
\]

By \ref{lem:typing-substitution}

\[
\Gamma_0[\hat{\alpha} : \star] \vdash (\hat{\alpha}) \eta f \Gamma_0[\hat{\alpha} : \star] = 1
\]

By def. of subst.

\[
\Gamma_0[\hat{\alpha} : \star] \rightarrow \Omega \quad \text{Given}
\]

By Lemma \ref{lem:typing-completeness} (Parallel Extension Solution)

\[
\Omega \rightarrow \Omega \quad \text{By Lemma} \ref{lem:typing-completeness} \text{ (Extension Reflexivity)}
\]

**Case** \( \text{chk-I} \)

\[
[\Omega] \Gamma \vdash \alpha : \kappa \vdash [\Omega] \nu \equiv A_0 p \quad \text{Declvi}
\]

\[
\begin{align*}
[\Omega] \Lambda &= \forall \alpha : \kappa. A_0 \\

&= \forall \alpha : \kappa. [\Omega] A'
\end{align*}
\]

Given

\[
A_0 = [\Omega] A'
\]

By def. of subst. and predicativity of \( \Omega \)

\[
[\Omega] \Gamma, \alpha : \kappa \vdash [\Omega] \nu \equiv [\Omega] A' p
\]

Subderivation and above equality

\[
\Gamma \rightarrow \Omega
\]

By \ref{def:typing-context}

\[
\Gamma, \alpha : \kappa \rightarrow \Omega, \alpha : \kappa
\]

By \ref{def:typing-context}

\[
[\Omega], \alpha : \kappa = [\Omega], \alpha : \kappa \quad \text{By definition of context substitution}
\]

\[
[\Omega], \alpha : \kappa \vdash [\Gamma], \alpha : \kappa \vdash [\Omega] \nu \equiv [\Omega] A' p
\]

By above equality

\[
[\Omega], \alpha : \kappa \vdash [\Gamma], \alpha : \kappa \vdash [\Omega] \nu \equiv [\Omega], \alpha : \kappa A' p
\]

By definition of substitution

\[
\Gamma, \alpha : \kappa \vdash \nu \equiv [\Gamma], \alpha : \kappa A' p \rightarrow \Delta'
\]

By i.h.

\[
\Delta' \rightarrow \Omega_0'
\]

\[
\Omega', \alpha : \kappa \rightarrow \Omega_0'
\]

\[
dom(\Delta') = \text{dom}(\Omega_0')
\]

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

By Lemma \ref{lem:typing-extension} (Typing Extension)

\[
\Delta' = (\Delta, \alpha : \kappa, \Theta)
\]

By Lemma \ref{lem:typing-extension} (Extension Inversion) \ref{item:i}

\[
\Delta, \alpha : \kappa, \Theta \rightarrow \Omega_0'
\]

By above equality

\[
\Omega_0' = (\Omega', \alpha : \kappa, \Omega_Z)
\]

By Lemma \ref{lem:typing-extension} (Extension Inversion) \ref{item:i}

\[
\Delta \rightarrow \Omega'
\]

By \ref{def:typing-context}

\[
\text{dom}(\Delta) = \text{dom}(\Omega_0')
\]

\[
\Omega \rightarrow \Omega'
\]

By Lemma \ref{lem:typing-extension} (Extension Inversion) on \( \Omega, \alpha : \kappa \rightarrow \Omega_0' \)

**Case** \( \text{chk-I} \)

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

[\Omega] \Gamma \vdash [\Omega][e \cdot s_0] : [\tau/\alpha][\Omega]A_0 f \gg B q
\]

\[
\begin{align*}
[\Omega] \Gamma \vdash [\Omega][e \cdot s_0] : \forall \alpha : \kappa. [\Omega]A_0 p \gg B q
\end{align*}
\]

\[
\text{DeclviSpine}
\]

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

Subderivation

\[
\Gamma \rightarrow \Omega
\]

Given

\[
\Gamma, \hat{\alpha} : \kappa \rightarrow \Omega, \hat{\alpha} : \kappa = \tau
\]

By \ref{def:typing-context}

\[
[\Omega] \Gamma \vdash [\Omega][e \cdot s_0] : [\tau/\alpha][\Omega]A_0 f \gg B q
\]

Subderivation

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

\[
[\tau/\alpha][\Omega]A_0 = [\tau/\alpha][\Omega], \hat{\alpha} : \kappa = \tau|A_0
\]

By def. of subst.

\[
[\tau/\alpha][\Omega]A_0 = [\tau|\alpha|][\Omega], \hat{\alpha} : \kappa = \tau|A_0
\]

By above equality

\[
[\tau|\alpha|][\Omega], \hat{\alpha} : \kappa = \tau|\hat{\alpha}/\alpha|A_0
\]

By distributivity of substitution

\[
[\Omega] \Gamma \vdash [\Omega], \hat{\alpha} : \kappa = \tau|\Gamma, \hat{\alpha} : \kappa
\]

By definition of context application
Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

\[ \Gamma \vdash [\sigma = \tau] \supset A_0 \]

By above equalities
\[ \Gamma, \delta : \kappa \vdash e - s_0 : [\Gamma, \delta : \kappa] [\delta/\alpha] A_0 \not\vdash B' \quad \Delta \]

By i.h.
\[ B = [\Gamma, \delta : \kappa] B' \]

\[ \Delta \rightarrow \Omega' \]

\[ \Omega \rightarrow \Omega' \]

\[ \Omega \rightarrow \Omega' \]

\[ B' \rightarrow [\Delta] B' \]

\[ B \rightarrow [\Omega'] B' \]

\[ [\Gamma, \delta : \kappa] [\delta/\alpha] A_0 = [\Gamma] [\delta/\alpha] A_0 \]

By def. of context application
\[ \Gamma, \delta : \kappa \vdash e - s_0 : [\delta/\alpha][\Gamma] A_0 \not\vdash B' \quad \Delta \]

By above equality
\[ \Gamma \vdash e - s_0 : \forall \alpha : \kappa. [\Gamma] A_0 \not\vdash B' \quad \Delta \]

By \text{vSpine}
\[ \Gamma \vdash e - s_0 : [\Gamma] [\forall \alpha : \kappa. A_0] \not\vdash B' \quad \Delta \]

By def. of subst.

\[ \text{Case} \]
\[ \var{v \text{chkI}} [\Gamma] \not\vdash [\Omega] P \vdash [\Omega] v \not\vdash [\Omega] A_0 ! \]

Subderivation

The concluding rule in this subderivation must be \text{DeclCheckL} or \text{DeclCheckUnify} In either case, \[ [\Omega] P \not\vdash [\Omega] A_0 ! \]

\[ \text{Case} \]
\[ \text{mgu}(\Omega) \sigma_0 [\Omega] \tau = \bot \]

\[ [\Omega] \Gamma / [\Omega] (\sigma = \tau) \vdash [\Omega] v \not\vdash [\Omega] A_0 ! \]

We have \text{mgu}(\Omega) \sigma_0 [\Omega] \tau = \bot. To apply Lemma 91 (Completeness of Elimeq) (2), we need to show conditions 1–5.

\[ \Gamma \vdash [\sigma = \tau] \supset A_0 ! \quad \text{Given} \]

\[ [\Omega] (\sigma = \tau) \supset A_0 = [\Gamma] (\sigma = \tau) \supset A_0 \]

By Lemma 38 (Principal Agreement) (i)

\[ [\Omega] \sigma = [\Gamma] \sigma \quad \text{By a property of subst.} \]

\[ [\Omega] \tau = [\Gamma] \tau \quad \text{Similar} \]

\[ \Gamma \vdash \sigma : \kappa \quad \text{By inversion} \]

\[ \Gamma \vdash [\Gamma] \sigma : \kappa \quad \text{By Lemma 10 (Right-Hand Substitution for Sorting)} \]

\[ \Gamma \vdash [\Gamma] \tau : \kappa \quad \text{Similar} \]

\[ \text{mgu}(\Omega) \sigma_0 [\Omega] \tau = \bot \quad \text{Given} \]

\[ \text{mgu}(\Gamma) \sigma_0 [\Gamma] \tau = \bot \quad \text{By above equalities} \]

\[ \text{FEV}(\sigma) \cup \text{FEV}(\tau) = \emptyset \quad \text{By inversion on ***} \]

\[ \text{FEV}([\Omega] \sigma) \cup \text{FEV}([\Omega] \tau) = \emptyset \quad \text{By a property of complete contexts} \]

\[ \text{FEV}([\Gamma] \sigma) \cup \text{FEV}([\Gamma] \tau) = \emptyset \quad \text{By above equalities} \]

\[ \Gamma / [\Gamma] \sigma \equiv [\Gamma] \tau : \kappa \theta \perp \quad \text{By Lemma 91 (Completeness of Elimeq) (2)} \]

\[ \Gamma, \var{p} / [\Gamma] \sigma = [\Gamma] \tau \theta \perp \quad \text{By ElimpropEq} \]

\[ \Gamma \vdash v \not\vdash ([\Gamma] \sigma = [\Gamma] \tau) \supset [\Gamma] A_0 ! \quad \Delta \]

By \text{\sigmaL}

\[ \Gamma \vdash v \not\vdash ([\Gamma] \sigma = [\Gamma] \tau) A_0 ! \quad \Delta \]

By def. of subst.

\[ \Gamma \rightarrow \Omega \quad \text{Given} \]

\[ \Omega \rightarrow \Omega \quad \text{By Lemma 31 (Extension Reflexivity)} \]

\[ \text{dom(\Gamma)} = \text{dom}(\Omega) \quad \text{Given} \]

\[ \text{Case} \]
\[ \text{mgu}(\Omega) \sigma_0 [\Omega] \tau = \emptyset \quad \theta([\Omega] \Gamma) \vdash \theta([\Omega] e) \not\vdash \theta([\Omega] A_0) ! \]

\[ [\Omega] \Gamma / ([\Omega] \sigma) = [\Omega] \tau \theta \perp \quad \text{By DeclCheckUnify} \]
We have \( \text{mgu}(\sigma, \tau) = \emptyset \), and will need to apply Lemma \( \text{(Completeness of Elimeq) } (1) \). That lemma has five side conditions, which can be shown exactly as in the \( \text{DeclCheck} \) case above.

\[
\begin{align*}
\text{mgu}(\sigma, \tau) &= \emptyset & \text{Premise} \\
\text{Let } \Omega_0 &= (\Omega, \bowtie p). \\
\Gamma &\rightarrow \Omega & \text{Given} \\
\Gamma, \bowtie p &\rightarrow \Omega_0 & \text{By } \text{Marker} \\
\text{dom}(\Gamma) &= \text{dom}(\Omega) & \text{Given} \\
\text{dom}(\Gamma, \bowtie p) &= \text{dom}(\Omega_0) & \text{By def. of } \text{dom}(-) \\
\Gamma, \bowtie p \vdash [\Gamma] \sigma = \Gamma \tau & \text{By Lemma } \text{(Completeness of Elimeq) } (1) \\
\Gamma, \bowtie p &\vdash \Gamma \sigma = \Gamma \tau & \text{By } \text{ElimPropEq} \\
\text{EQ0 } &\text{for all } \Gamma, \bowtie p \vdash u : \kappa. [\Gamma, \bowtie p, \Theta] u = \theta([\Gamma, \bowtie p] u) " \\
\Gamma &\vdash P \supset A_0 ! \text{ type} & \text{Given} \\
\Gamma &\vdash A_0 ! \text{ type} & \text{By inversion} \\
\Gamma &\rightarrow \Omega & \text{Given} \\
\text{EQa } \Gamma \vdash [\Omega] A_0 = [\Omega] A_0 & \text{By Lemma } \text{(Principal Agreement) } (i) \\
\text{Let } \Omega_1 &= (\Omega, \bowtie p, \Theta). \\
\theta([\Omega] \Gamma) &\vdash \theta(e) \iff \theta([\Omega] A_0) ! & \text{Subderivation} \\
\text{Subderivation } \Gamma, \bowtie p, \Theta &\rightarrow \Omega_1 & \text{By induction on } \Theta \\
\theta([\Omega] A_0) &= \theta([\Gamma] A_0) & \text{By above equality EQa} \\
&= [\Gamma, \bowtie p, \Theta] A_0 & \text{By Lemma } \text{(Substitution Upgrade) } (i) \text{ (with EQ0)} \\
&= [\Omega_1] A_0 & \text{By Lemma } \text{(Principal Agreement) } (i) \\
&= [\Omega_1][\Gamma, \bowtie p, \Theta] A_0 & \text{By Lemma } \text{(Substitution Monotonicity) } (iii) \\
\theta([\Omega] \Gamma) &= [\Omega_1](\Gamma, \bowtie p, \Theta) & \text{By Lemma } \text{(Substitution Upgrade) } (iii) \\
\theta([\Omega] e) &= [\Omega_1] e & \text{By Lemma } \text{(Substitution Upgrade) } (iv) \\
\text{[\Omega_1][\Gamma, \bowtie p, \Theta] } &\vdash [\Omega_1] e \iff [\Omega_1][\Gamma, \bowtie p, \Theta] A_0 ! & \text{By above equalities} \\
\text{dom}(\Gamma, \bowtie p, \Theta) &= \text{dom}(\Omega_1) & \text{dom}(\Gamma) = \text{dom}(\Omega) \\
\Gamma, \bowtie p, \Theta &\vdash e \iff [\Gamma, \bowtie p, \Theta] A_0 ! \vdash \Delta' & \text{By i.h.} \\
\Delta' &\rightarrow \Omega_2' & \text{"} \\
\Omega_1 &\rightarrow \Omega_2' & \text{"} \\
\text{dom}(\Delta') &= \text{dom}(\Omega_2') & \text{"} \\
\Delta' &= (\Delta, \bowtie p, \Delta'') & \text{By Lemma } \text{(Extension Inversion) } (ii) \\
\Omega_2' &= (\Omega', \bowtie p, \Omega_Z) & \text{By Lemma } \text{(Extension Inversion) } (ii) \\
\text{By } \text{def. of subst.} \\
\Gamma, \bowtie p, \Theta &\vdash e \iff [\Gamma, \bowtie p, \Theta] A_0 ! \vdash \Delta, \bowtie p, \Delta'' & \text{By above equality} \\
\Gamma &\vdash e \iff [\Gamma] \sigma = [\Gamma] \tau \supset [\Gamma] A_0 ! \vdash \Delta & \text{By } \text{impl} \\
\Gamma &\vdash e \iff [\Gamma] (P \supset A_0) ! \vdash \Delta & \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 & \text{Bq } q & \text{DeclCheck} \text{ Spine}
\end{align*}
\]
Proof of Theorem 11 (Completeness of Algorithmic Typing) [thm:typing-completeness]

\[ [\Omega] \Gamma \vdash [\Omega] P \text{ true} \]
Subderivation

\[ [\Omega] \Gamma \vdash [\Omega][\Gamma] P \text{ true} \]
By Lemma 28 (Substitution Monotonicity) (ii)

\[ \Gamma \vdash [\Gamma] P \text{ true} \rightarrow \Theta \]
By Lemma 94 (Completeness of Checkprop)

\[ \Theta \rightarrow \Omega \]

\[ \Omega \rightarrow \Omega \]

dom(\Theta) = dom(\Omega_1)

\[ \Gamma \rightarrow \Omega \]
Given

\[ [\Omega] \Gamma = [\Omega_1] \Theta \]
By Lemma 56 (Multiple Confluence)

\[ [\Omega] A_0 = [\Omega_1] A_0 \]
By Lemma 54 (Completing Completeness) (ii)

\[ [\Omega] \Gamma \vdash [\Omega] (e \cdot s_0) : [\Omega] A_0 \rightarrow B \ q \]
Subderivation

\[ [\Omega_1] \Theta \vdash [\Omega_1] (e \cdot s_0) : [\Omega_1] A_0 \rightarrow B \ q \]
By above equalities

\[ \Theta \vdash e \cdot s_0 : [\Theta] A_0 \rightarrow B' \ q \rightarrow \Delta \]
By i.h.

\[ \Gamma \vdash e \cdot s_0 : [\Gamma] P \rightarrow [\Gamma] A_0 \rightarrow B' \ q \rightarrow \Delta \]
By above equality

\[ \theta \vdash e \cdot s_0 : [\theta][\Gamma] A_0 \rightarrow B' \ q \rightarrow \Delta \]
By \text{Spine}

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \leftarrow A_k' \ p \]
Subderivation

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \leftarrow [\Omega] A_k \ p \]
\[ [\Omega] A_k = A_k' \]
\[ \Gamma \vdash e_0 \leftarrow [\Gamma] A_k \ p \rightarrow \Delta \]
By i.h.

\[ \Delta \rightarrow \Omega \]

\[ \Omega \rightarrow \Omega' \]

\[ \Omega \rightarrow \Omega' \]

\[ \theta \vdash e \cdot s_0 : [\theta][\Gamma] A_0 \rightarrow B' \ q \rightarrow \Delta \]
By above equalities

\[ \Gamma \vdash e \cdot s_0 : [\Gamma][\Gamma] A_0 \rightarrow B' \ q \rightarrow \Delta \]
By \text{Decl+I}_k

Case

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \leftarrow A_k' \ p \]
\[ [\Omega] \Gamma \vdash \text{inj}_{k} [\Omega] e_0 \leftarrow \Gamma A_1 + A_2' \ p \]

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:

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \leftarrow A_k' \ p \]
Subderivation

\[ [\Omega] \Gamma \vdash [\Omega] e_0 \leftarrow [\Omega] A_k \ p \]
\[ [\Omega] A_k = A_k' \]
\[ \Gamma \vdash e_0 \leftarrow [\Gamma] A_k \ p \rightarrow \Delta \]
By i.h.

\[ \Delta \rightarrow \Omega \]

\[ \Omega \rightarrow \Omega' \]

\[ \Gamma \vdash \text{inj}_k e_0 \leftarrow ([\Gamma] A_1) + ([\Gamma] A_2) \ p \rightarrow \Delta \]
By \text{Decl+I}_k

In the latter case, \( A = \alpha \) and \( [\Omega] \lambda = \alpha \rightarrow A_1 + A_2 = \tau_1 + \tau_2 \).

By inversion on \( \Gamma \vdash \alpha \ p \text{ type} \), it must be the case that \( p = f \).

\[ \Gamma \rightarrow \Omega \]
Given

\[ \Gamma = \Gamma_f[\alpha : \star] \]
\[ \alpha \in \text{unsolved}(\Gamma) \]

\[ \Omega = \Omega_0[\alpha : \star = \tau_0] \]
By Lemma 21 (Extension Inversion) (vi)

Let \( \Omega_2 = \Omega_0[\delta_1 : \star = \tau_1', \delta_1 : \star = \tau_1', \delta_2 : \star = \delta_1 \delta_2]. \)

Let \( \Gamma_2 = \Gamma_0[\delta_1 : \star, \delta_2 : \star, \delta : \star = \delta_1 + \delta_2]. \)

\[ \Gamma \rightarrow \Gamma_2 \]
By Lemma 22 (Deep Evar Introduction) (iii) twice

\[ \Omega \rightarrow \Omega_2 \]
By Lemma 22 (Deep Evar Introduction) (iii) twice

\[ \Gamma_2 \rightarrow \Omega_2 \]
By Lemma 25 (Parallel Admissibility) (ii), (ii), (iii)
Proof of [**Theorem 11**] (Completeness of Algorithmic Typing) thm:typing-completeness

\[\Gamma \vdash e_0 \iff \Omega \] Subd. and \( \Lambda' = \tau' = [\Omega_2]_k \)

\[\Omega \vdash \Omega_2 \Gamma \] By Lemma 56 (Multiple Confluence)

\[\Omega_2 \vdash e_0 \iff [\Omega_2]_k \] By above equality

\[\Gamma \vdash e_0 \iff [\Gamma]_k \] By i.h.

\[\Delta \rightarrow \Omega' \]
\[\text{dom}(\Delta) = \text{dom}(\Omega') \]
\[\Omega_2 \rightarrow \Omega' \]
\[\Omega \rightarrow \Omega' \] By Lemma 32 (Extension Transitivity)

\[\Gamma \vdash \text{inj}_k e_0 \rightarrow \check{\alpha} \div \Delta \]
\[\Gamma \vdash \text{inj}_k e_0 \rightarrow [\Gamma] \\check{\alpha} \div \Delta \]
\[\check{\alpha} \in \text{unsolved}(\Gamma)\]

**Case** \([\Omega] \Gamma, x : A_1 p \vdash [\Omega] e_0 \iff A_2 p\] Decl→I

We have \([\Omega]A = A'_1 \rightarrow A'_2\). Either \([\Gamma]A = A_1 \rightarrow A_2\) where \(A'_1 = [\Omega]A_1\) and \(A'_2 = [\Omega]A_2\)—or \([\Gamma]A = \check{\alpha}\) and \([\Omega]A = [\Omega]_k \rightarrow A'_1 \rightarrow A'_2\).

In the former case:

\[\Gamma \vdash e_0 \iff A'_2 p\] Subderivation

\[\Lambda'_1 = [\Omega]A_1\]
\[\Lambda'_1 = [\Omega][\Gamma]A_1\]
\[\Lambda'_1 = [\Omega][\Omega][\Gamma]A_1\]
\[\Lambda'_2 = [\Omega][\Gamma]A_1\] Known in this subcase

By Lemma 29 (Substitution Invariance)

Applying \(\Omega\) on both sides

By idempotence of substitution

\[\Omega_1 \vdash e_0 \iff A_2 p\] By definition of context application

\[\Omega, x : A_1 p \vdash [\Gamma] e_0 \iff A_2 p\] By above equality

\[\Gamma \rightarrow \Omega\]
\[\Gamma, x : [\Gamma] A_1 p \rightarrow \Omega, x : A'_1 p\] Given

\[\text{dom}(\Gamma) = \text{dom}(\Omega)\]
\[\text{dom}(\Gamma, x : p) = \Omega, x : A'_1 p\]
\[\Gamma, x : [\Gamma] A_1 p \vdash e_0 \iff A_2 p \div \Delta'\] By i.h.

\[\Delta' \rightarrow \Omega_0'\]
\[\text{dom}(\Delta') = \text{dom}(\Omega_0')\]
\[\Omega_0 \vdash A'_1 p \rightarrow \Omega_0'\]
\[\Omega, x : A'_1 p \rightarrow \Omega_0'\]
\[\Omega_0' = (\Omega', x : A'_1 p, \Theta)\] By Lemma 21 (Extension Inversion) (v)

\[\Omega \rightarrow \Omega'\]

\[\Gamma, x : [\Gamma] A_1 p \rightarrow \Delta'\] By Lemma 50 (Typing Extension)

\[\Delta' = (\Delta, x : \cdots, \Theta)\] By Lemma 21 (Extension Inversion) (v)

\[\Delta, x : \cdots, \Theta \rightarrow \Omega', x : A'_1 p, \Theta\] By above equalities

\[\Delta \rightarrow \Omega'\]

\[\text{dom}(\Delta') = \text{dom}(\Omega')\]

\[\Gamma, x : [\Gamma] A_1 p \vdash e_0 \iff [\Gamma] A_2 p \div \Delta, x : A'_1 p, \Theta\] By above equality

\[\Gamma \vdash \lambda x. e_0 \iff ([\Gamma] A_1) \rightarrow ([\Gamma] A_2) p \div \Delta\] By \(\rightarrow\)

\[\Gamma \vdash \lambda x. e_0 \iff [\Gamma](A_1 \rightarrow A_2) p \div \Delta\] By definition of substitution

In the latter case \([\Gamma]A = \check{\alpha}\in \text{unsolved}(\Gamma)\) and \([\Omega]\check{\alpha} = A'_1 \rightarrow A'_2 = \tau'_1 \rightarrow \tau'_2\):

By inversion on \(\Gamma \vdash \check{\alpha} p \text{ type}\), it must be the case that \(p = \check{\beta}\).

Since \(\check{\alpha} \in \text{unsolved}(\Gamma)\), the context \(\Gamma\) must have the form \(\Gamma_0[\check{\alpha} : \check{\beta}]\).

Let \(\Gamma_2 = \Gamma_0[\check{\alpha} : \check{\beta}_1 : \star, \check{\alpha}_2 : \star, \check{\alpha} : \check{\alpha}_1 \rightarrow \check{\alpha}_2]\).
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[
\begin{align*}
\Gamma & \longrightarrow \Gamma_2 & \text{By Lemma 22 (Deep Evar Introduction) (iii) twice} \\
\Omega & = \Omega_0[\bar{\alpha} : \star = \tau_0] & \text{By Lemma 21 (Extension Inversion) (vi)} \\
\text{Let } \Omega_2 = \Omega_0[\bar{\alpha} : \star = \tau_1', \bar{\alpha}_1 : \star = \tau_2', \bar{\alpha} : \star = \bar{\alpha}_1 \rightarrow \bar{\alpha}_2]. \\
\Gamma & \longrightarrow \Gamma_2 & \text{By Lemma 22 (Deep Evar Introduction) (iii) twice} \\
\Omega & \longrightarrow \Omega_2 & \text{By Lemma 22 (Deep Evar Introduction) (iii) twice} \\
\Gamma_2 & \longrightarrow \Omega_2 & \text{By Lemma 25 (Parallel Admissibility) (ii), (ii), (iii)} \\
\end{align*}
\]

\[
\begin{align*}
& \text{[Ω]}, x : \tau_1' \not\vdash [Ω]e_0 \leftrightarrow \tau_2' \not\vdash \text{Subderivation} \\
& \text{[Ω]}[\Gamma] = \{[Ω_2]\}, \tau_2' = [Ω]\bar{\alpha}_2 \text{ From above equality} \\
& \text{[Ω], } x : \tau_1' \not\vdash e_0 \iff [Γ_2', x : \bar{\alpha}_1 \not\vdash \Delta^+ \not\vdash \Omega^+ \text{ By i.h.} \\
& \text{dom}(\Delta^+) = \text{dom}(\Omega^+) \text{ Similar} \\
& \text{[Ω], } x : \bar{\alpha}_1 \not\vdash \Delta^+ \not\vdash \Omega^+ \text{ By Lemma 25 (Parallel Admissibility) (ii), (ii), (iii)} \\
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash \lambda x. e_0 \leftrightarrow \bar{\alpha} \not\vdash \Delta \not\vdash \text{By def. of context application} \\
\Delta^+ & = (\Delta, x : \bar{\alpha}_1 \not\vdash \Delta_Z) & \text{By Lemma 21 (Extension Inversion) (v)} \\
\Omega^+ & = (\Omega', x : \ldots \not\vdash \Omega_Z) & \text{By Lemma 21 (Extension Inversion) (v)} \\
\end{align*}
\]

\[
\begin{align*}
\text{[Ω]}[\Gamma]' & \vdash e_0 \Rightarrow A \text{ q} \text{ Subderivation} \\
\Theta & \longrightarrow \Omega_0 \text{ By i.h.} \\
\text{dom}(\Theta) & = \text{dom}(\Omega_0) \text{ "} \\
\Omega & \longrightarrow \Omega_0 \text{ "} \\
A & = [\Omega_0]A' \text{ "} \\
A' & = [\Theta]A' \text{ "} \\
\end{align*}
\]

\[
\begin{align*}
\text{[Ω]}[\Gamma]' & \vdash A \text{ q} \not\vdash \Theta & \text{Subderivation} \\
\Theta & \longrightarrow \Omega_0 & \text{By i.h.} \\
\text{dom}(\Theta) & = \text{dom}(\Omega_0) & \text{"} \\
\Omega & \longrightarrow \Omega_0 & \text{"} \\
A & = [\Omega_0]A' & \text{"} \\
A' & = [\Theta]A' & \text{"} \\
\end{align*}
\]

\[\text{Decl→E}\]
\[ \Gamma \rightarrow \Omega \]
\[ [\Omega]\Gamma = [\Omega\Theta] \]
\[ [\Omega]\Gamma \vdash [\Omega]s_0 : A \gg C [p] \]
\[ [\Omega\Theta] \vdash [\Omega]s_0 : [\Omega\Theta]A' \gg C [p] \]
\[ \Theta \vdash s_0 : [\Theta]A' \gg C' [p] \vdash \Delta \]
\[ \Rightarrow C' = [\Delta]C' \]
\[ \Rightarrow \Delta \rightarrow \Omega' \]
\[ \Rightarrow \text{dom}(\Delta) = \text{dom}(\Omega') \]
\[ \Rightarrow \Omega_{\Theta} \rightarrow \Omega' \]
\[ \Rightarrow \Theta \vdash s_0 : A' \gg C' [p] \vdash \Delta \]
\[ \Rightarrow \Omega \rightarrow \Omega' \]
\[ \Rightarrow \Gamma \vdash e_0 \cdots s_0 \Rightarrow C' p \vdash \Delta \]

Given

By Lemma 56 (Multiple Confluence)

Subderivation

By above equalities

By i.h.

By above equality

By Lemma 32 (Extension Transitivity)

By \text{\rightarrow E}
Proof of Theorem 11 (Completeness of Algorithmic Typing)

• Case for all $C_2$.

  $$\Gamma \vdash \Omega \vdash [\Omega]A ! \gg C \not\in C$$
  $$\Gamma \vdash [\Omega]s : [\Omega]A ! \gg C \not\in C$$
  $$\Gamma \vdash [\Omega]s : [\Omega]A ! \gg C \not\in C$$

  $\Gamma \rightarrow \Omega$$

  $[\Omega] \Gamma \vdash [\Omega]s : [\Omega]A ! \gg C \not\in C$ Subderivation

  $\Gamma \vdash s : [\Gamma]A ! \gg C \not\in \Delta$ By i.h.

  $\Delta \rightarrow \Omega'$

  $\Omega \rightarrow \Omega'$

  $\text{dom}(\Delta) = \text{dom}(\Omega')$

  $C = [\Omega']C'$

  $C' = [\Delta]C'$

  Suppose, for a contradiction, that $\text{FEV}([\Delta]C') \neq \emptyset$.
  That is, there exists some $\hat{\alpha} \in \text{FEV}([\Delta]C')$.

  $\Delta \rightarrow \Omega_2$ By Lemma 59 (Split Solutions)

  $\Omega_2' [\hat{\alpha} : \kappa = t_1] \rightarrow \Omega'$

  $\Omega_2 = \Omega_2' [\hat{\alpha} : \kappa = t_2]$ $t_2 \neq t_1$

  $[\Omega_2 ]\hat{\alpha} \neq [\Omega_2']\hat{\alpha}$ By def. of subst. ($t_2 \neq t_1$)

  $[\Omega_2 ]\hat{\beta} = [\Omega_2']\hat{\beta}$ for all $\hat{\beta} \neq \hat{\alpha}$

  $\text{FEV}([\Gamma]A) = \emptyset$ By inversion

  $\text{FEV}([\Gamma]A) \subseteq \text{dom}(-)$ Property of $\subseteq$

  $\Delta = (\Delta_1 \ast \Delta_R)$ By Lemma 71 (Seperation—Main) (Spines)

  $(\Gamma \ast \cdot) \rightarrow (\Delta_1 \ast \Delta_R)$

  $\text{FEV}(C') \subseteq \text{dom(\Delta_R)}$ Above

  $\hat{\alpha}_R \in \text{FEV}(C')$ Above

  $\hat{\alpha}_R \in \text{dom}(\Delta_R)$ Property of $\subseteq$

  $\text{dom}(\Delta_1) \cap \text{dom}(\Delta_R) = \emptyset$

  $\hat{\alpha}_R \notin \text{dom}(\Delta_1)$

  $\text{dom}(\Gamma) \subseteq \text{dom}(\Delta_1)$ By Definition 5

  $\hat{\alpha}_R \notin \text{dom}(\Gamma)$

  $[\Omega_2 ]\Gamma \vdash [\Omega_2 ]s : [\Omega_2 ][\Gamma]A ! \gg [\Omega_2 ]C' \not\in C$ Above

  $\Omega_2$ and $\Omega_1$ differ only at $\hat{\alpha}$ Above

  $\text{FEV}([\Gamma]A) = \emptyset$ Above

  $[\Omega_2 ][\Gamma]A = [\Omega_1 ][\Gamma]A$ By preceding two lines

  $\Gamma \vdash [\Gamma]A \text{ type}$ Above

  $\Gamma \rightarrow \Omega_2$

  $\Omega_2 \vdash [\Gamma]A \text{ type}$ By Lemma 32 (Extension Transitivity)

  $\text{dom}(\Omega_2) = \text{dom}(\Omega_1)$

  $\Omega_1 \vdash [\Gamma]A \text{ type}$ By Lemma 17 (Equal Domains)

  $\text{dom}(\Gamma) \subseteq \text{dom}(\Delta_1)$

  $\hat{\alpha}_R \notin \text{dom}(\Gamma)$

  $\text{FEV}([\Gamma]A) = \emptyset$ Above

  $[\Omega_2 ][\Gamma]A = [\Omega_1 ][\Gamma]A$ By preceding two lines

  $\Gamma \vdash [\Gamma]A \text{ type}$ Above

  $\Gamma \rightarrow \Omega_2$

  $\Omega_2 \vdash [\Gamma]A \text{ type}$ By Lemma 32 (Extension Transitivity)

  $\text{dom}(\Omega_2) = \text{dom}(\Omega_1)$

  $\Omega_1 \vdash [\Gamma]A \text{ type}$ By Lemma 17 (Equal Domains)
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ \Gamma \vdash [\Gamma] A \text{ type} \]
\[ \Omega \vdash [\Gamma] A \text{ type} \]
\[ [\Omega_1][\Gamma] A = [\Omega_1][\Gamma] A = [\Omega][\Gamma] A \]
\[ \Omega \Gamma = [\Omega] \Gamma \]
\[ [\Omega][\Gamma] = [\Omega_1][\Gamma] \]
\[ [\Omega_2][\Gamma] \]
\[ [\Omega_2][s] = [\Omega][s] \Omega \text{ and } \Omega \text{ differ only in } \alpha \]
\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A ! \gg [\Omega_2][\Gamma] C' \] 
\[ F E V ([\Delta][C']) = \emptyset \]
\[ \vdash \Gamma \vdash s : [\Gamma] A ! \gg C' [!] \rightarrow \Delta \]

**Case**

\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A \gg C q \]
\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A \gg C q \] 
\[ \Gamma \vdash s : [\Gamma][\Gamma] A \gg C' q \rightarrow \Delta \]
\[ \Delta \rightarrow \Omega' \]
\[ \text{Subderivation} \]
\[ \Gamma \vdash s : [\Gamma] A \gg C' q \rightarrow \Delta \]
\[ \text{By i.h.} \]

**Case**

\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A \gg C q \]
\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A \gg C q \]
\[ \Gamma \vdash s : [\Gamma][\Gamma] A \gg C' q \rightarrow \Delta \]
\[ \Delta \rightarrow \Omega' \]
\[ \text{Subderivation} \]
\[ \Gamma \vdash s : [\Gamma][\Gamma] A \gg C' q \rightarrow \Delta \]
\[ \text{By i.h.} \]

**Case**

\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A \gg [\Omega][\Gamma] A \]
\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s : [\Omega][\Gamma] A \gg [\Omega][\Gamma] A \]
\[ \Gamma \vdash s : [\Gamma][\Gamma] A \gg [\Gamma][\Gamma] A \rightarrow \Gamma \]
\[ \text{By EmptySpine} \]
\[ [\Gamma][\Gamma] A = [\Gamma][\Gamma] A \]
\[ \text{By idempotence of substitution} \]
\[ \Gamma \rightarrow \Omega \]
\[ \text{Given} \]
\[ \text{dom}(\Gamma) = \text{dom}(\Omega) \]
\[ \text{Given} \]
\[ [\Omega][\Gamma] A = [\Omega][\Gamma] A \]
\[ \text{By Lemma 28 (Substitution Monotonicity) (iii)} \]
\[ \Omega \rightarrow \Omega \]
\[ \text{By Lemma 31 (Extension Reflexivity)} \]

**Case**

\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s_0 : [\Omega][\Gamma] A_1 q \]
\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s_0 : [\Omega][\Gamma] A_2 q \gg B \]
\[ [\Omega][\Gamma] \vdash [\Omega][\Gamma] s_0 : [\Omega][\Gamma] A_1 \rightarrow [\Omega][\Gamma] A_2 q \gg B \]
\[ \text{Decl} \rightarrow \text{Spine} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)

\[ \Gamma \vdash [\Omega]e_0 \iff [\Omega]A_1 \ q \]
\[ \Gamma \vdash e_0 \iff A' \ q \vdash \Theta \]
\[ \Theta \rightarrow \Omega_\Theta \]
\[ \Omega \rightarrow \Omega_\Theta \]
\[ A = [\Omega_\Theta]A' \]
\[ A' = [\Theta]A' \]
\[ [\Omega]\varGamma \vdash [\Omega]s_0 : [\Omega]A_2 \ q \gg B \ p \]

Subderivation

By i.h.

**Proof**

\[ \Gamma \vdash s_0 : A_2 \ q \gg B \ p \vdash \Delta \]

By i.h.

\[ \Delta \rightarrow \Omega' \]

By \( \rightarrow \) Spine

\[ \text{dom} (\Delta) = \text{dom} (\Omega') \]
\[ \Omega \rightarrow \Omega' \]
\[ B' = [\Delta]B' \]
\[ B = [\Omega']B' \]

\[ \Gamma \vdash e_0 \cdot s_0 : A_1 \rightarrow A_2 \ q \gg B \ p \vdash \Delta \]

By \( \rightarrow \) Spine

\( \rightarrow \) Subderivation
• Case: \( |\Omega|^\Gamma \vdash (|\Omega|^\Pi \, \text{true}) \quad |\Omega|^\Gamma \vdash |\Omega|^e \Leftrightarrow (|\Omega|A_0 \, p) \)

If \( e \) is not a case, then:

\( [\Omega|\Gamma \vdash |\Omega|P \, \text{true} \quad |\Omega|^\Gamma \vdash |\Omega|^e \Leftrightarrow (|\Omega|A_0) \wedge |\Omega|P \, p \)

\( \text{Decl} \wedge \text{I} \)

Otherwise, we have \( e = \text{case}(e_0, \Pi) \). Let \( n \) be the height of the given derivation.

\[
\begin{align*}
\text{Subderivation} & \quad |\Omega|^\Gamma \vdash |\Omega|\left(\text{case}(e_0, \Pi)\right) \Leftrightarrow |\Omega|A_0 \, p \\
\text{Subderivation} & \quad |\Omega|^\Pi \vdash B \ \
\text{Subderivation} & \quad |\Omega|^\Pi \vdash B \Leftrightarrow |\Omega|A_0 \, p \\
\text{Subderivation} & \quad |\Omega|^\Pi \vdash B \\
\text{Subderivation} & \quad |\Omega|^\Pi \vdash B \\
\text{Case} & \quad |\Omega|\left(\text{case}(e_0, \Pi)\right) \Leftrightarrow |\Omega|A_0 \, p \\
\end{align*}
\]

By Lemma 61 (Case Invertibility) (1)

By Lemma 60 (Interpolating With and Exists) (1)

By def. of subst.

By i.h.

By i.h.

By Lemma 29 (Substitution Invariance)

By Lemma 56 (Multiple Confluence)

By Lemma 54 (Completing Completeness) (ii)

By above equalities

By above equalities

By Lemma 29 (Substitution Invariance)

By Lemma 54 (Completing Completeness) (iii)

By Lemma 55 (Confluence of Completeness)

By Lemma 32 (Extension Transitivity)

By Lemma 32 (Extension Transitivity)

By Case

By Case
Case \[|\Omega|\Gamma \vdash |\Omega|e_1 \leftrightarrow A_1^1 \ p \quad |\Omega|\Gamma \vdash |\Omega|e_2 \leftrightarrow A_2^1 \ p\]

Either |\Gamma|A = A_1 \times A_2 or |\Gamma|A = \hat{\alpha} \in \text{unsolved}(\Gamma).

- In the first case \(|\Gamma|A = A_1 \times A_2\), we have \(A_1^1 = |\Omega|A_1\) and \(A_2^1 = |\Omega|A_2\).

\[|\Omega|\Gamma \vdash |\Omega|e_1 \leftrightarrow A_1^1 \ p \quad \text{Subderivation} \]
\[|\Omega|\Gamma \vdash |\Omega|e_1 \leftrightarrow |\Omega|A_1 \ p \quad |\Omega|A_1 = A_1^1 \]
\[\Gamma \vdash e_1 \leftrightarrow |\Gamma|A_1 \ p \vdash \Theta \quad \text{By i.h.} \]
\[\Theta \rightarrow \Omega_\Theta \quad " \]
\[\text{dom}(\Theta) = \text{dom}(\Omega_\Theta) \quad " \]
\[\Omega \rightarrow \Omega_\Theta \quad " \]
\[|\Omega|\Gamma \vdash |\Omega|e_2 \leftrightarrow A_2^1 \ p \quad |\Omega|A_2 = A_2^1 \]
\[|\Omega|\Gamma \vdash |\Omega|e_2 \leftrightarrow |\Omega|A_2 \ p \quad |\Omega|A_2 = A_2^1 \]
\[\Gamma \rightarrow \Theta \quad \text{By Lemma 50 (Typing Extension)} \]
\[|\Omega|\Gamma = |\Omega_\Theta|\Theta \quad \text{By Lemma 56 (Multiple Confluence)} \]
\[|\Omega_\Theta|\Theta \vdash |\Omega_\Theta|A_2 \quad \text{By Lemma 54 (Completing Completeness) (ii)} \]
\[|\Omega_\Theta|\Theta \vdash |\Omega_\Theta|e_2 \leftrightarrow |\Omega_\Theta|A_2 \ p \quad \text{By above equalities} \]
\[\Theta \vdash e_2 \leftrightarrow |\Gamma|A_2 \ p \vdash \Delta \quad \text{By i.h.} \]
\[\Delta \rightarrow \Omega' \quad " \]
\[\text{dom}(\Delta) = \text{dom}(\Omega') \quad " \]
\[\Omega_\Theta \rightarrow \Omega' \quad " \]
\[\Omega \rightarrow \Omega' \quad \text{By Lemma 32 (Extension Transitivity)} \]
\[\Gamma \vdash \langle e_1, e_2 \rangle \leftrightarrow (|\Gamma|A_1) \times (|\Gamma|A_2) \ p \vdash \Delta \quad \text{By i.h.} \]
\[\Gamma \vdash \langle e_1, e_2 \rangle \leftrightarrow |\Gamma|(A_1 \times A_2) \ p \vdash \Delta \quad \text{By def. of subst.} \]

- In the second case, where \(|\Gamma|A = \hat{\alpha}\), combine the corresponding subcase for \(\text{Decl}+1_\Theta\) with some straightforward additional reasoning about contexts (because here we have two sub-derivations, rather than one).

Case \[|\Omega|\Gamma \vdash |\Omega|e_0 \Rightarrow C ! \quad |\Omega|\Gamma \vdash |\Omega|\Pi \Leftarrow C \Leftarrow |\Omega|A \ p \quad |\Omega|\Gamma \vdash |\Omega|\Pi \text{ covers } C \quad \text{DeclCase} \]

\[|\Omega|\Gamma \vdash |\Omega|e_0 \Rightarrow C ! \quad \text{Subderivation} \]
\[\Gamma \vdash e_0 \Rightarrow C' ! \vdash \Theta \quad \text{By i.h.} \]
\[\Theta \rightarrow \Omega_\Theta \quad " \]
\[\text{dom}(\Theta) = \text{dom}(\Omega_\Theta) \quad " \]
\[\Omega \rightarrow \Omega_\Theta \quad " \]
\[C = |\Omega_\Theta|C' \quad " \]
\[\Theta \vdash C' ! \text{ type} \quad \text{By Lemma 62 (Well-Formed Outputs of Typing)} \]
\[\text{FEV}(C') = \emptyset \quad \text{By inversion} \]
\[|\Omega_\Theta|C' = C' \quad \text{By a property of substitution} \]
Proof of Theorem 11 (Completeness of Algorithmic Typing)  

\[ \text{Either } A = A_1 \times A_2, \text{ or } A = A_1' \times A_2' \text{ where } A_1 = [\Omega]A_1 \text{ and } A_2 = [\Omega]A_2'. \]

In the former case (A = A):  
We have [\Omega]A = A_1 \times A_2. Therefore A_1 = [\Omega]A_1' \text{ and } A_2 = [\Omega]A_2'. Moreover, \( \Gamma = \Gamma_0 [\kappa : \kappa]. \)

\[ \text{Subderivation} \]

Let \( \Gamma' = \Gamma_0 [\kappa_1 : \kappa_1, \kappa_2 : \kappa_2, \kappa_1 + \kappa_2]. \)

\[ \text{Subderivation} \]

\[ \text{By def. of context substitution} \]

\[ \text{By above equality} \]

\[ \text{By i.h.} \]

\[ \text{Subderivation} \]
Proof of Theorem 11

**Completeness of Algorithmic Typing**

[151]

Proof of Theorem 11 (Completeness of Algorithmic Typing) thm:typing-completeness

In the latter case \((A = A'_1 \times A'_2)\):

\[
\begin{align*}
[\Omega] \Gamma & = \{\Omega_1, \Omega_2\} \Theta & \text{By Lemma 56 (Multiple Confluence)} \\
[\Omega] \Gamma \vdash [\Omega] e_1 \leftarrow [\Omega] A'_1 p & A_1 = [\Omega] A'_1 & \text{By Lemma 54 (Completing Completeness) (ii)} \\
[\Omega] \Gamma \vdash [\Omega] e_2 \leftarrow [\Omega] A'_2 p & A_2 = [\Omega] A'_2 & \text{By above equalities} \\
\end{align*}
\]

By i.h.

Subderivation

Given \((A = A'_1 \times A'_2)\)

By inversion

By Lemma 32 (Extension Transitivity)

By Lemma 37 (Extension Weakening (Types))

By Lemma 54 (Completing Completeness)

By Lemma 28 (Substitution Monotonicity) (iii)

By Lemma 56 (Multiple Confluence)

By i.h.

By Lemma 32 (Extension Transitivity)

By def. of substitution

Now we turn to parts (v) and (vi), completeness of matching.

- **Case** DeclMatchEmpty
  
  Apply rule MatchEmpty

- **Case** DeclMatchSeq
  
  Apply the i.h. twice, along with standard lemmas.

- **Case** DeclMatchBase
  
  Apply the i.h. (i) and rule MatchBase

- **Case** DeclMatchUnit
  
  Apply the i.h. and rule MatchUnit

- **Case** DeclMatch\(\leftarrow\)
  
  By i.h. and rule Match\(\leftarrow\)

- **Case** DeclMatch\(\times\)
  
  By i.h. and rule Match\(\times\)

- **Case** DeclMatch\(+,1\)
  
  By i.h. and rule Match\(+,1\)

- **Case** DeclMatch\(\rightarrow\)
  
  By i.h. and rule Match\(\rightarrow\)

<table>
<thead>
<tr>
<th>Case</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\Omega] \Gamma \vdash \bar{e} \rightarrow [\Gamma] A, [\Omega] \bar{A} \leftarrow [\Omega] C p</td>
<td>DeclMatch(\rightarrow)</td>
</tr>
</tbody>
</table>
Proof of Theorem 11 \(\text{(Completeness of Algorithmic Typing)}\)

\[\Gamma \vdash (A \wedge P, \tilde{A}) \not\vdash \ \text{types} \quad \text{Given}\]
\[\Gamma \vdash (A \wedge P) \not\vdash \ \text{type} \quad \text{By inversion on Principal TypevecWF}\]
\[\Gamma \vdash A \not\vdash \ \text{type} \quad \text{By Lemma 41 (Inversion of Principal Typing)}\]

(2) \(\Gamma \vdash P \prop\)  

(3) \(\text{FEV}(\Gamma) = \emptyset\)  

(1) \(\Gamma \vdash (A, \tilde{A}) \not\vdash \ \text{types} \quad \text{By inversion and Principal TypevecWF}\)

(4) \(\Gamma \vdash C \not\vdash \ \text{type} \quad \text{Given}\)

(5) \([\Omega] \Gamma / P \vdash \bar{p} \Rightarrow [\Omega] e : [\Omega] \bar{A} \iff [\Omega] C \not\vdash \ \text{Subderivation}\)

(6) \(p' \subseteq\ p\)  

\[\Gamma / [\Gamma] P \vdash \bar{p} \Rightarrow e : [\Gamma] (A, \tilde{A}) \iff [\Gamma] C p' \vdash \Delta\]  

\[\Delta \longrightarrow \Omega'\]  

\[\text{dom}(\Delta) = \text{dom}(\Omega')\]  

\[\Omega \longrightarrow \Omega'\]

\[\Gamma / [\Gamma] P \vdash \bar{p} \Rightarrow e : [\Gamma] (A \wedge P, \tilde{A}) \iff [\Gamma] C p' \vdash \Delta\]  

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

\[\Gamma \vdash \bar{p} \Rightarrow e : ([\Gamma] A \wedge [\Gamma] P), [\Gamma] \tilde{A} \iff [\Gamma] C p' \vdash \Delta\]  

\[\text{By MatchNeg}\]

\[\Gamma \vdash \bar{p} \Rightarrow e : [\Gamma] (A \wedge P), [\Gamma] \tilde{A} \iff [\Gamma] C p' \vdash \Delta\]  

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

\[\therefore\]  

\[\text{Case} \text{DeclMatchNeg}: \text{By i.h. and rule MatchNeg}\]

\[\text{Case} \text{DeclMatchWild}: \text{By i.h. and rule MatchWild}\]

\[\text{Case} \text{DeclMatchUnify}\]

\[\text{Given}\]

\[\text{Given}\]

\[\text{By above equalities}\]

\[\text{By Lemma 91 (Completeness of Elimeq)}\]

\[\text{By Lemma 91 (Completeness of Elimeq)}\]

\[\text{By Lemma 31 (Extension Reflexivity)}\]

\[\therefore\]

\[\text{Case}\]

\[\text{By above equalities}\]

\[\text{By Lemma 91 (Completeness of Elimeq)}\]

\[\text{By Lemma 31 (Extension Reflexivity)}\]

\[\text{By above equalities}\]
Proof of [Theorem 11] (Completeness of Algorithmic Typing) thm:typing-completeness

\[ \Gamma, \triangleright_p, \Theta \vdash (\overline{\rho} \Rightarrow e) : [\Gamma, \triangleright_p, \Theta] \overline{A} \leftarrow [\Gamma, \triangleright_p, \Theta] C p \vdash \Delta, \triangleright_p, \Delta' \] By i.h.

\[ \Delta, \triangleright_p, \Delta' \longrightarrow \Omega', \triangleright_p, \Omega'' \]

\[ \Omega, \triangleright_p, \Theta \longrightarrow \Omega', \triangleright_p, \Omega'' \]

\[ \text{dom}(\Delta, \triangleright_p, \Delta') = \text{dom}(\Omega', \triangleright_p, \Omega'') \] By Lemma 21 (Extension Inversion) (ii)

\[ \Delta \longrightarrow \Omega' \]

\[ \text{dom}(\Delta) = \text{dom}(\Omega') \] By Lemma 21 (Extension Inversion) (ii)

\[ \Omega \longrightarrow \Omega' \]

\[ \Gamma / [\Gamma] \sigma = [\Gamma] \tau \vdash \overline{\rho} \Rightarrow e : [\Gamma] \overline{A} \leftarrow [\Gamma] C p \vdash \Delta \] By MatchUnify