Bidirectional Type Checking for Relational Properties
(Appendix)

Ezgi Çiçek  Weihao Qu  Gilles Barthe  Marco Gaboardi  Deepak Garg

Contents

1 relSTLC  ................................................................. 5
  1.1 Syntax of relSTLC .............................................. 5
  1.2 relSTLC Lemmas .................................................. 7

2 RelRef  ........................................................................ 11
  2.1 Syntax of RelRef .................................................... 11
  2.2 RelRef Lemmas ...................................................... 23

3 RelInf  .......................................................................... 41
  3.1 Syntax of RelInf ..................................................... 41
  3.2 RelInf Lemmas ....................................................... 46

4 RelRefU  ..................................................................... 51
  4.1 Syntax of RelRefU ................................................... 51
  4.2 RelRefU Lemmas .................................................... 69

5 RelCost  ....................................................................... 89
  5.1 Metatheory ............................................................ 125

6 Experimental Evaluation ............................................. 150
  6.1 Experimental results ................................................ 156
List of Figures

<table>
<thead>
<tr>
<th></th>
<th>Syntax of values and expressions in relSTLC</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Syntax of values and expressions in BirelSTLC</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>relSTLC typing rules</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>relSTLC subtyping rules</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>relSTLC algorithmic typing rules</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>relSTLC algorithmic subtyping rules</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>relSTLC annotation erasure</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Syntax of values and expressions in RelRef</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>Syntax of values and expressions in RelRef Core</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Syntax of values and expressions in BiRelRef</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>RelRef typing rules (Part 1)</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>RelRef typing rules (Part 2)</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>RelRef Subtyping rules</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>RelRef Core type equivalence rules</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>RelRef Core typing rules (Part 1)</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>RelRef Core typing rules (Part 2)</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>RelRef embedding rules (Part 1)</td>
<td>18</td>
</tr>
<tr>
<td>17</td>
<td>RelRef embedding rules (Part 2)</td>
<td>19</td>
</tr>
<tr>
<td>18</td>
<td>RelRef algorithmic type equivalence rules</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>RelRef algorithmic typing rules (Part 1)</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>RelRef algorithmic typing rules (Part 2)</td>
<td>22</td>
</tr>
<tr>
<td>21</td>
<td>Syntax of values and expressions in RelInf</td>
<td>41</td>
</tr>
<tr>
<td>22</td>
<td>Syntax of values and expressions in RelInf Core</td>
<td>41</td>
</tr>
<tr>
<td>23</td>
<td>Syntax of values and expressions in RelInf U</td>
<td>41</td>
</tr>
<tr>
<td>24</td>
<td>RelInf unary typing rules</td>
<td>42</td>
</tr>
<tr>
<td>25</td>
<td>RelInf binary typing rules</td>
<td>42</td>
</tr>
<tr>
<td>26</td>
<td>RelInf unary subtyping rules</td>
<td>42</td>
</tr>
<tr>
<td>27</td>
<td>RelInf unary subtyping rules</td>
<td>42</td>
</tr>
<tr>
<td>28</td>
<td>RelInf binary subtyping rules</td>
<td>42</td>
</tr>
<tr>
<td>29</td>
<td>RelInf refinement removal operation</td>
<td>43</td>
</tr>
<tr>
<td>30</td>
<td>RelInf relational type equivalence rules</td>
<td>43</td>
</tr>
<tr>
<td>31</td>
<td>RelInf Core unary typing rules</td>
<td>43</td>
</tr>
<tr>
<td>32</td>
<td>RelInf Core binary typing rules</td>
<td>43</td>
</tr>
<tr>
<td>33</td>
<td>RelInf binary embedding rules</td>
<td>44</td>
</tr>
<tr>
<td>34</td>
<td>RelInf unary embedding rules</td>
<td>44</td>
</tr>
<tr>
<td>35</td>
<td>RelInf unary algorithmic typing rules</td>
<td>44</td>
</tr>
<tr>
<td>36</td>
<td>RelInf binary algorithmic typing rules</td>
<td>45</td>
</tr>
<tr>
<td>37</td>
<td>RelInf algorithmic subtyping rules</td>
<td>45</td>
</tr>
<tr>
<td>38</td>
<td>Syntax of values and expressions in RelInf U Core</td>
<td>51</td>
</tr>
<tr>
<td>39</td>
<td>Syntax of values and expressions in RelInf U Core</td>
<td>51</td>
</tr>
<tr>
<td>40</td>
<td>Syntax of values and expressions in BiRelInf U</td>
<td>52</td>
</tr>
<tr>
<td>41</td>
<td>RelInfU relational typing rules (Part 1)</td>
<td>53</td>
</tr>
<tr>
<td>42</td>
<td>RelInfU relational typing rules (Part 2)</td>
<td>54</td>
</tr>
<tr>
<td>43</td>
<td>RelInfU relational subtyping rules</td>
<td>55</td>
</tr>
<tr>
<td>44</td>
<td>RelInfU unary subtyping rules</td>
<td>56</td>
</tr>
<tr>
<td>45</td>
<td>RelInfU Core binary type equivalence rules</td>
<td>56</td>
</tr>
<tr>
<td>46</td>
<td>RelInfU Core unary typing rules</td>
<td>57</td>
</tr>
<tr>
<td>47</td>
<td>RelInfU Core binary typing rules (Part 1)</td>
<td>58</td>
</tr>
<tr>
<td>48</td>
<td>RelInfU Core binary typing rules (Part 2)</td>
<td>59</td>
</tr>
<tr>
<td>49</td>
<td>RelInfU unary embedding typing rules</td>
<td>60</td>
</tr>
<tr>
<td>Page</td>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>50</td>
<td>RelRefU relational embedding rules (Part 1)</td>
<td>61</td>
</tr>
<tr>
<td>51</td>
<td>RelRefU relational embedding rules (Part 2)</td>
<td>62</td>
</tr>
<tr>
<td>52</td>
<td>RelRefU unary algorithmic subtyping rules</td>
<td>63</td>
</tr>
<tr>
<td>53</td>
<td>RelRefU binary algorithmic equivalence rules</td>
<td>64</td>
</tr>
<tr>
<td>54</td>
<td>RelRefU unary algorithmic typing rules (Part 1)</td>
<td>65</td>
</tr>
<tr>
<td>55</td>
<td>RelRefU unary algorithmic typing rules (Part 2)</td>
<td>66</td>
</tr>
<tr>
<td>56</td>
<td>RelRefU binary algorithmic typing rules (Part 1)</td>
<td>67</td>
</tr>
<tr>
<td>57</td>
<td>RelRefU binary algorithmic typing rules (Part 2)</td>
<td>68</td>
</tr>
<tr>
<td>58</td>
<td>Syntax of types and contexts</td>
<td>90</td>
</tr>
<tr>
<td>59</td>
<td>Syntax of values and expressions in RelCost</td>
<td>90</td>
</tr>
<tr>
<td>60</td>
<td>Syntax of values and expressions in RelCostCore</td>
<td>91</td>
</tr>
<tr>
<td>61</td>
<td>Syntax of values and expressions in BiRelCost</td>
<td>91</td>
</tr>
<tr>
<td>62</td>
<td>Well-formedness of binary types</td>
<td>92</td>
</tr>
<tr>
<td>63</td>
<td>Well-formedness of unary types</td>
<td>93</td>
</tr>
<tr>
<td>64</td>
<td>Sorting of Substitutions</td>
<td>93</td>
</tr>
<tr>
<td>65</td>
<td>RelCost subtyping rules (part 1)</td>
<td>94</td>
</tr>
<tr>
<td>66</td>
<td>RelCost subtyping rules (Part 2)</td>
<td>95</td>
</tr>
<tr>
<td>67</td>
<td>RelCost unary subtyping rules</td>
<td>96</td>
</tr>
<tr>
<td>68</td>
<td>RelCost typing rules (Part 1)</td>
<td>97</td>
</tr>
<tr>
<td>69</td>
<td>RelCost typing rules (Part 2)</td>
<td>98</td>
</tr>
<tr>
<td>70</td>
<td>RelCost typing rules (Part 3)</td>
<td>99</td>
</tr>
<tr>
<td>71</td>
<td>Typing rules (Part 4)</td>
<td>100</td>
</tr>
<tr>
<td>72</td>
<td>RelCost typing rules (Part 6)</td>
<td>102</td>
</tr>
<tr>
<td>73</td>
<td>Embedding Rules (Part 1)</td>
<td>103</td>
</tr>
<tr>
<td>74</td>
<td>Embedding Rules (Part 2)</td>
<td>104</td>
</tr>
<tr>
<td>75</td>
<td>Embedding Rules (Part 3)</td>
<td>105</td>
</tr>
<tr>
<td>76</td>
<td>Embedding Rules (Part 4)</td>
<td>106</td>
</tr>
<tr>
<td>77</td>
<td>Embedding Rules (Part 5)</td>
<td>107</td>
</tr>
<tr>
<td>78</td>
<td>Embedding Rules (Part 6)</td>
<td>108</td>
</tr>
<tr>
<td>79</td>
<td>RelCostCore binary type equivalence rules</td>
<td>109</td>
</tr>
<tr>
<td>80</td>
<td>RelCostCore typing rules (Part 1)</td>
<td>110</td>
</tr>
<tr>
<td>81</td>
<td>RelCostCore typing rules (Part 2)</td>
<td>111</td>
</tr>
<tr>
<td>82</td>
<td>RelCostCore typing rules (Part 3)</td>
<td>112</td>
</tr>
<tr>
<td>83</td>
<td>RelCostCore typing rules (Part 4)</td>
<td>113</td>
</tr>
<tr>
<td>84</td>
<td>RelCostCore typing rules (Part 5)</td>
<td>114</td>
</tr>
<tr>
<td>85</td>
<td>RelCostCore typing rules (Part 6)</td>
<td>115</td>
</tr>
<tr>
<td>86</td>
<td>Algorithmic typing rules (part 1)</td>
<td>116</td>
</tr>
<tr>
<td>87</td>
<td>Algorithmic typing rules (part 2)</td>
<td>117</td>
</tr>
<tr>
<td>88</td>
<td>Algorithmic typing rules (part 3)</td>
<td>118</td>
</tr>
<tr>
<td>89</td>
<td>Algorithmic typing rules (part 4)</td>
<td>119</td>
</tr>
<tr>
<td>90</td>
<td>Algorithmic typing rules (part 5)</td>
<td>120</td>
</tr>
<tr>
<td>91</td>
<td>Algorithmic typing rules (part 6)</td>
<td>121</td>
</tr>
<tr>
<td>92</td>
<td>Algorithmic typing rules (part 7)</td>
<td>122</td>
</tr>
<tr>
<td>93</td>
<td>Algorithmic type equivalence rules</td>
<td>123</td>
</tr>
<tr>
<td>94</td>
<td>Annotation erasure</td>
<td>124</td>
</tr>
</tbody>
</table>

**List of Theorems and Lemmas**

1. Lemma (Reflexivity of algorithmic subtyping in relSTLC) ........................................ 7
2. Lemma (Transitivity of algorithmic subtyping in relSTLC) ........................................ 7
1 relSTLC

1.1 Syntax of relSTLC

Types $\tau ::= \text{bool} \mid \text{bool}_u \mid \tau_1 \to \tau_2$

Expressions $e ::= x \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid \lambda x. e \mid e_1 \cdot e_2$

Value $v ::= \text{true} \mid \text{false} \mid \lambda x. e$

Figure 1: Syntax of values and expressions in relSTLC

Types $\tau ::= \text{bool}_r \mid \text{bool}_u \mid \tau_1 \to \tau_2$

Expressions $e ::= x \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid \lambda x. e \mid e_1 \cdot e_2 \mid (e : \tau)$

Value $v ::= \text{true} \mid \text{false} \mid \lambda x. e$

Figure 2: Syntax of values and expressions in BirelSTLC

Γ ⊢ x :: τ  \quad \text{r-var}
Γ ⊢ b :: \text{bool}_r
b ∈ \{\text{true}, \text{false}\}
Γ ⊢ b :: \text{bool}_u
Gamma ⊢ b_1 :: b_2 :: \text{bool}_u
b_1, b_2 ∈ \{\text{true}, \text{false}\}
Γ ⊢ e :: \text{bool}_r
Γ ⊢ e_1 :: e_1' :: τ
Γ ⊢ e_2 :: e_2' :: τ
Γ ⊢ \text{if } e \text{ then } e_1 \text{ else } e_2 :: \tau
Γ ⊢ \lambda x. e :: \text{bool}_r
Γ ⊢ \lambda x. e_1 :: \lambda x. e_2 :: \tau_1 \to \tau_2
Γ ⊢ e_1 :: e_2 :: \tau_1
Γ ⊢ e_1' :: e_2' :: \tau_2
Γ ⊢ e_1 \cdot e_2 :: \tau_1 \to \tau_2
Γ ⊢ e_1' \cdot e_2' :: \tau_2
Γ ⊢ \tau :: \tau' \quad \text{r-⊑}
Gamma ⊢ \tau :: \tau'
Γ ⊢ e_1 :: e_2 :: \tau
Γ ⊢ e_1' :: e_2' :: \tau'
Γ ⊢ e_1 \cdot e_2 :: \tau'
Γ ⊢ e_1' \cdot e_2' :: \tau'

Figure 3: relSTLC typing rules
\begin{align*}
\Gamma(x) = \tau & \quad \text{alg-r-var} \\
\Gamma \vdash x \bowtie x \uparrow \tau & \\
\text{alg-r-bool} & \\
\Gamma, x : \tau_1 \vdash e_1 \bowtie e_2 \downarrow \tau_2 & \quad \text{alg-r-lam} \\
\Gamma \vdash \lambda x . e_1 \bowtie \lambda x . e_2 \downarrow \tau_1 \rightarrow \tau_2 & \\
\text{alg-r-app} & \\
\Gamma \vdash e \bowtie e' \uparrow \tau' & \quad \text{alg-\updownarrow} \\
\Gamma \vdash e \bowtie e' \downarrow \tau & \\
\Gamma \vdash e \bowtie e' \downarrow \tau & \\
\Gamma \vdash e \bowtie e' \downarrow \tau & \\
\Gamma \vdash (e : \tau) \bowtie (e' : \tau) \uparrow \tau & \\
\text{alg-r-anno-\uparrow} & \\
\Gamma \vdash e \bowtie e' \downarrow \tau & \\
\Gamma \vdash e \bowtie e' \downarrow \tau & \\
\Gamma \vdash e \bowtie e' \downarrow \tau & \\
\Gamma \vdash (e : \tau) \bowtie (e' : \tau) \uparrow \tau & \\
\text{alg-r-anno-\uparrow} & \\
\end{align*}

\begin{align*}
\vdash \text{bool}_r \leq \text{bool}_u & \quad \text{alg-bl-r} \\
\vdash \text{bool}_u \leq \text{bool}_u & \quad \text{alg-bl-u} \\
\vdash \text{bool}_r \leq \text{bool}_u & \quad \text{alg-boole} \\
\vdash \tau_1' \leq \tau_1 & \quad \text{alg-\rightarrow} \\
\vdash \tau_2 \leq \tau_2' & \quad \text{alg-\rightarrow} \\
\vdash \tau_1 \rightarrow \tau_2 \leq \tau_1' \rightarrow \tau_2' & \\
\end{align*}

\begin{align*}
|.| : & \text{Expression} \rightarrow \text{Expression} \\
|x| = x & \\
|e_1 \ e_2| = |e_1| \ |e_2| & \\
\vdots & \\
|(e : \tau)| = |e| & \\
\end{align*}

\begin{figure}
\begin{align*}
\Gamma \vdash \text{bool}_r \subseteq \text{bool}_u & \quad \text{relSTLC subtyping rules} \\
\vdash \text{bool}_r \leq \text{bool}_u & \quad \text{alg-bl-r} \\
\vdash \text{bool}_u \leq \text{bool}_u & \quad \text{alg-bl-u} \\
\vdash \text{bool}_r \leq \text{bool}_u & \quad \text{alg-boole} \\
\vdash \tau_1' \leq \tau_1 & \quad \text{alg-\rightarrow} \\
\vdash \tau_2 \leq \tau_2' & \quad \text{alg-\rightarrow} \\
\vdash \tau_1 \rightarrow \tau_2 \leq \tau_1' \rightarrow \tau_2' & \\
\end{align*}
\end{figure}
1.2 relSTLC Lemmas

Lemma 1 (Reflexivity of algorithmic subtyping in relSTLC)
The reflexivity property holds for algorithm subtyping for relSTLC.
\(\models \tau \leq \tau\).

Proof. By induction on the binary type.

Case \(\text{bool}_r\)
It is proved by \text{alg-bl-r}.

Case \(\text{bool}_u\)
It is proved by \text{alg-bl-u}.

Case \(\tau_1 \rightarrow \tau_2\)
By \text{alg-→}, TS: \(\models \tau_1 \leq \tau_1\) and \(\models \tau_2 \leq \tau_2\), which is obtained by IH on \(\tau_1\) and \(\tau_2\) respectively.

Lemma 2 (Transitivity of algorithmic subtyping in relSTLC)
The transitivity property holds for algorithm subtyping for relSTLC.
\(\models \tau_1 \leq \tau_2\) and \(\models \tau_2 \leq \tau_3\), then \(\models \tau_1 \leq \tau_3\).

Proof. By simultaneous induction on the first two subtyping derivation.

Case \(\models \text{bool}_r \leq \text{bool}_r\), \(\models \text{bool}_r \leq \text{bool}_r\)
TS: \(\models \text{bool}_r \leq \text{bool}_r\), which is proved by \text{alg-bl-r}.

Case \(\models \text{bool}_u \leq \text{bool}_u\), \(\models \text{bool}_u \leq \text{bool}_u\)
TS: \(\models \text{bool}_u \leq \text{bool}_u\), which is proved by \text{alg-bl-u}.

Case \(\models \text{bool}_r \leq \text{bool}_u\), \(\models \text{bool}_u \leq \text{bool}_u\)
TS: \(\models \text{bool}_r \leq \text{bool}_u\), which is proved by \text{alg-bool}.

Case \(\models \text{bool}_r \leq \text{bool}_u\), \(\models \text{bool}_u \leq \text{bool}_u\)
TS: TS: \(\models \text{bool}_r \leq \text{bool}_u\), which is proved by \text{alg-bool}.

Case \(\models \tau'_1 \leq \tau_1\), \(\models \tau_2 \leq \tau'_2\)
By \text{alg-→}, TS: \(\models \tau_1 \rightarrow \tau_2 \leq \tau'_1 \rightarrow \tau'_2\), \(\models \tau'_1 \rightarrow \tau'_2 \leq \tau''_1 \rightarrow \tau''_2\), \(\models \tau'_1 \rightarrow \tau'_2 \leq \tau'_1 \rightarrow \tau''_2\), \(\models \tau'_1 \rightarrow \tau''_2 \).
By IH on \(\models \tau''_1 \leq \tau_1\) and \(\models \tau'_1 \leq \tau_1\), \(\models \tau''_1 \leq \tau_1\).

By IH on \(\models \tau'_2 \leq \tau''_1\) and \(\models \tau_2 \leq \tau'_2\), \(\models \tau_2 \leq \tau''_2\).

It is proved by using the two statements and rule \text{alg-→}.

\[\square\]

Next, we can show that the algorithmic formulation of subtyping \((\tau_1 \leq \tau_2)\) coincides with the declarative formulation of subtyping \((\tau_1 \sqsubseteq \tau_2)\). We do this in two steps.
Lemma 3 (Soundness of algorithmic subtyping in relSTLC)
If \( \models \tau \leq \tau' \) then \( \models \tau \subseteq \tau' \).

Proof. Proof is by straightforward induction on the given algorithmic subtyping derivation.

Case \( \models \tau_1' \leq \tau_1 \) \( \models \tau_2 \leq \tau_2' \) \( \text{alg-}\to \)
\[ TS: \models \tau_1 \to \tau_2 \leq \tau_1' \to \tau_2' \]
By IH on \( \models \tau_1' \leq \tau_1 \), \( \models \tau_1' \subseteq \tau_1 \).
By IH on \( \models \tau_2 \leq \tau_2' \), \( \models \tau_2 \subseteq \tau_2' \).
It is proved using these two statements and subtyping rule \( \to \).

Case \( \models \text{bool}_r \leq \text{bool}_u \) \( \text{alg-bool} \)
\[ TS: \models \text{bool}_r \leq \text{bool}_u, \text{which is proved by subtyping rule bool} \]

Case \( \models \text{bool}_u \leq \text{bool}_u \) \( \text{alg-bl-u} \)
\[ TS: \models \text{bool}_u \leq \text{bool}_u, \text{which is proved by subtyping rule refl} \]

Case \( \models \text{bool}_r \leq \text{bool}_r \) \( \text{alg-bl-r} \)
\[ TS: \models \text{bool}_r \leq \text{bool}_r, \text{which is proved by subtyping rule refl} \]

Lemma 4 (Completeness of algorithmic subtyping in relSTLC)
If \( \models \tau \subseteq \tau' \) then \( \models \tau \leq \tau' \).

Proof. Proof is by induction on the given subtyping derivation.

Case \( \models \tau_1' \subseteq \tau_1 \) \( \models \tau_2 \subseteq \tau_2' \) \( \to \)
\[ TS: \models \tau_1 \to \tau_2 \subseteq \tau_1' \to \tau_2' \]
By IH on \( \models \tau_1' \subseteq \tau_1 \), \( \models \tau_1' \leq \tau_1 \).
By IH on \( \models \tau_2 \subseteq \tau_2' \), \( \models \tau_2 \leq \tau_2' \).
It is proved using these two statements and algorithmic subtyping rule \( \text{alg-}\to \).

Case \( \models \text{bool}_r \subseteq \text{bool}_u \) \( \text{bool} \)
\[ TS: \models \text{bool}_r \subseteq \text{bool}_u, \text{which is proved by algorithmic subtyping rule \text{alg-bool}} \]

Case \( \models \tau \subseteq \tau \) \( \text{refl} \)
\[ TS: \models \tau \subseteq \tau, \text{which is proved by Lemma 1} \]

Case \( \models \tau_1 \subseteq \tau_2 \) \( \models \tau_2 \subseteq \tau_3 \) \( \text{trans} \)
\[ TS: \models \tau_1 \subseteq \tau_3 \]
By IH on \( \models \tau_1 \subseteq \tau_2 \), \( \models \tau_1 \leq \tau_2 \).
By IH on \( \models \tau_2 \subseteq \tau_3 \), \( \models \tau_2 \leq \tau_3 \).
It is proved using these two statements and Lemma 2.

Lemma 5 (Soundness of algorithmic typing in relSTLC)
The following holds, \( |.| \) is the annotation erasure function.
1. If \( \Gamma \vdash e_1 \sim e_2 \downarrow \tau \) then \( \Gamma \vdash |e_1| \sim |e_2| : \tau \).
2. If \( \Gamma \vdash e_1 \sim e_2 \uparrow \tau \) then \( \Gamma \vdash |e_1| \sim |e_2| : \tau \).

Proof. Proof is by simultaneous induction on the given algorithmic checking and inference derivations and Lemma 3.

Case \( \Gamma(x) = \tau \quad \text{alg-r-var} \)
TS: \( \Gamma \vdash |x| \sim |x| : \tau \)
It is proved by using the premise \( \Gamma(x) = \tau \) and typing rule \( r\text{-var} \).

Case \( \Gamma \vdash e \sim e' \uparrow \text{bool} \quad \Gamma \vdash e_1 \sim e_1' \downarrow \tau \quad \Gamma \vdash e_2 \sim e_2' \downarrow \tau \quad \text{alg-r-if} \)
TS: \( \Gamma \vdash |e| \sim |e'| : \tau \)
By IH2 on \( \Gamma \vdash e \sim e' \uparrow \text{bool} \), \( \Gamma \vdash |e| \sim |e'| : \text{bool} \).
By IH on \( \Gamma \vdash e_1 \sim e_1' \downarrow \tau \), \( \Gamma \vdash |e_1| \sim |e_1'| : \tau \).
By IH on \( \Gamma \vdash e_2 \sim e_2' \downarrow \tau \), \( \Gamma \vdash |e_2| \sim |e_2'| : \tau \).
Because \( |e| \sim e \) and \( e \sim e' \), we get the conclusion.

Case \( \Gamma, x : \tau_1 \vdash e_1 \sim e_2 \downarrow \tau_2 \quad \text{alg-r-lam} \)
TS: \( \Gamma \vdash \lambda x.e_1 \sim \lambda x.e_2 \downarrow \tau_1 \rightarrow \tau_2 \)
By Lemma 3 on \( \|e| \sim |e'| \) and \( \|e| \sim |e'| \).
By IH on \( \Gamma, x : \tau_1 \vdash e_1 \sim e_2 \downarrow \tau_2 \), \( \Gamma, x : \tau_1 \vdash |e_1| \sim |e_2| : \tau_2 \).
It is proved by the above statement and typing rule \( r\text{-lam} \).

Case \( \Gamma \vdash e \sim e' \uparrow \tau' \quad \|\tau' \leq \tau \quad \text{alg-\updownarrow} \)
TS: \( \Gamma \vdash |e| \sim |e'| : \tau \).
By Lemma 3 on \( \|\tau \leq \tau' \), \( \|\tau \subseteq \tau' \).
By IH2 on \( \Gamma \vdash e \sim e' \uparrow \tau \), \( \Gamma \vdash |e| \sim |e'| : \tau \).
It is proved by the above statement and typing rule \( r\text{-\updownarrow} \).

Case \( \Gamma \vdash e \sim e' \downarrow \tau \quad \text{alg-r-anno-\uparrow} \)
TS: \( \Gamma \vdash |(e : \tau)| \sim |(e' : \tau)| : \tau \), which is simplified as \( \Gamma \vdash e \sim e' : \tau \).
By IH on \( \Gamma \vdash e \sim e' \downarrow \tau \), \( \Gamma \vdash e \sim e' : \tau \).
It is proved by the above statement.

\[
\text{Lemma 6 (Completeness of algorithmic typing in refSTLC)}
\]
If \( \Gamma \vdash e_1 \sim e_2 : \tau \) then there exist \( e_1' \) and \( e_2' \) such that \( \Gamma \vdash e_1' \sim e_2' \downarrow \tau \) and \( |e_i'| = e_i \) for \( i \in \{1, 2\} \).

Proof. Proof is by induction on the given typing derivation and Lemma 4.

Case \( \Gamma, x : \tau_1 \vdash e_1 \sim e_2 : \tau_2 \quad \text{r-lam} \)
By IH on \( \Gamma, x : \tau_1 \vdash e_1 \sim e_2 : \tau_2 \), \( \exists e_1', e_2', \Gamma, x : \tau_1 \vdash e_1' \sim e_2' \downarrow \tau_2 \) and \( |e_i'| = e_i \).
Then, using the above statement and algorithmic typing rule \( \text{alg-r-lam} \), we can construct the derivation where \( e_1' = \lambda x.e_1' \) and \( e_2' = \lambda x.e_2' \).

\[
\Gamma \vdash e_1' \sim e_2' \downarrow \tau_1 \rightarrow \tau_2
\]
And we show that \(|e''| = |\lambda x.e'_{\tau} = \lambda x.e'_{\tau} = \lambda x.e'_{\tau}|

\[\text{Case}\]
\[\Gamma \vdash e \triangleright e': \text{bool}, \quad \Gamma \vdash e_1 \triangleright e_1': \tau, \quad \Gamma \vdash e_2 \triangleright e_2': \tau\]

\text{r-if} \\
\[\Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 \triangleright e_3 \text{ if } e' \text{ then } e_1' \text{ else } e_2' \triangleright e_3' \text{ if } e'' \text{ then } e_1'' \text{ else } e_2'' \triangleright e_3''\]

By IH on \(\Gamma \vdash e \triangleright e': \text{bool}, \) \(\exists e'_1, e''_1, \Gamma \vdash e''_1 \triangleright e''_1 \downarrow \text{bool, and } |e''| = e, |e''_1| = e'.\)

By using algorithmic typing rule \text{alg-r-anno-\uparrow}, \(\Gamma \vdash (e''': \text{bool}) \triangleright (e''': \text{bool}) \uparrow \text{bool}..\)

Then, using the above statement and algorithmic typing rule \text{alg-r-if}, we can construct the derivation where \(e_3 = \text{if } (e''': \text{bool}) \) then \(e_3' \) else \(e_3'' \) and \(e_4 = \text{if } (e''': \text{bool}) \) then \(e_4' \) else \(e_4'' \).

\[\Gamma \vdash e_3 \triangleright e_4 \downarrow \tau\]

And we show that \(|e_3| = |(e''': \text{bool})| \) then \(|e_3''| = |(e''': \text{bool})|\) then \(|e_4''| = |e''''| = \text{if } e \text{ then } e_1 \text{ else } e_2, \) similarly for \(e_4.\)

\[\text{Case}\]
\[\Gamma \vdash e_1 \triangleright e_1': \tau_1 \rightarrow \tau_2, \quad \Gamma \vdash e_2 \triangleright e_2': \tau_1 \rightarrow \tau_2\]

\text{r-app} \\
\[\Gamma \vdash e_1 e_2 \triangleright e_1' e_2': \tau_1 \rightarrow \tau_2\]

By IH on \(\Gamma \vdash e_1 \triangleright e_1': \tau_1 \rightarrow \tau_2, \exists e''_1, e''_1', \Gamma \vdash e''_1' \triangleright e''_1' \downarrow \tau_1 \rightarrow \tau_2 \) and \(|e''_1'| = e_1, |e''_1''| = e_1'.\)

By using algorithmic typing rule \text{alg-r-anno-\uparrow}, \(\Gamma \vdash (e''_1': \tau_1 \rightarrow \tau_2) \triangleright (e''_1': \tau_1 \rightarrow \tau_2) \uparrow \tau_1 \rightarrow \tau_2.\)

By IH on \(\Gamma \vdash e_2 \triangleright e_2': \tau_1, \exists e''_2, e''_2', \Gamma \vdash e''_2' \triangleright e''_2' \downarrow \tau_1 \) and \(|e''_2'| = e_2, |e''_2''| = e_2'.\)

Then, using the above statement and algorithmic typing rule \text{alg-r-app}, we can construct the derivation where \(e_3 = e_1' e_2' \) and \(e_4 = e_1'' e_2'' \).

\[\Gamma \vdash e_3 \triangleright e_4 \downarrow \tau_2\]

And we show that \(|e_3| = |e_1' e_2'| = |e_1'' e_2''| = |e_1 e_2, \) similarly for \(e_4.\)

\[\text{Case}\]
\[\Gamma \vdash e_1 \triangleright e_2 : \tau \mid \tau \subseteq \tau' \]

\text{r-\subseteq} \\
\[\Gamma \vdash e_1 \triangleright e_2 : \tau'\]

By IH on \(\Gamma \vdash e_1 \triangleright e_2 : \tau, \exists e_1', e_2', \Gamma \vdash e_1' \triangleright e_1' \downarrow \tau \) and \(|e_1'| = e_1, |e_2'| = e_2.\)

By using algorithmic typing rule \text{alg-r-anno-\uparrow}, \(\Gamma \vdash (e_1' : \tau) \triangleright (e_1' : \tau) \uparrow \tau.\)

By using Lemma 3 on premise \(|\tau \subseteq \tau', |\tau \subseteq \tau'\).

Then, using the above statement and algorithmic typing rule \text{alg-r-\uparrow}, we can construct the derivation where \(e_3 = (e_1' : \tau) \) and \(e_4 = (e_2' : \tau).\)

\[\Gamma \vdash e_3 \triangleright e_4 \downarrow \tau\]

And we show that \(|e_3| = |(e_1' : \tau)| = |e_1| = e_1, \) similarly for \(e_4.\)
2 RelRef

2.1 Syntax of RelRef

Types \[ \tau ::= \text{bool} | \text{bool}_u | \tau_1 \to \tau_2 | \text{list}[n]^n \tau | \forall i::S. \tau | \exists i::S. \tau | \Box \tau | C \land \tau | C \lor \tau \]

Expressions \[ e ::= x | \text{true} | \text{false} | \text{if} \ e \text{ then } e_1 \text{ else } e_2 | \text{fix} \ f(x). e | e_1 \cdot e_2 | \text{nil} | \text{cons}(e_1, e_2) \]

Value \[ v ::= \text{true} | \text{false} | \text{fix} \ f(x). \text{v} | \text{nil} | \text{cons}(e_1, e_2) | \Lambda e | \text{pack} \text{v} \]

Index terms \[ I, n, \alpha ::= i | 0 | I + 1 | I \cdot I_2 | I_1 \cdot I_2 | \frac{1}{\alpha} | I_1 \cdot I_2 | I | J | \text{min}(I_1, I_2) | \text{max}(I_1, I_2). \]

Figure 8: Syntax of values and expressions, index terms in RelRef

\[ \tau ::= \text{bool} | \text{bool}_u | \tau_1 \to \tau_2 | \text{list}[n]^n \tau | \forall i::S. \tau | \exists i::S. \tau | \Box \tau | C \land \tau | C \lor \tau \]

Expressions \[ e ::= x | \text{true} | \text{false} | \text{if} \ e \text{ then } e_1 \text{ else } e_2 | \text{fix} \ f(x). e | \text{fix}_{NC} \ f(x). e | e_1 \cdot e_2 | \text{split}(e_1, e_2) \text{ with } C | \text{contra} \ e | \text{der} \ e | \Lambda i.e | e[I] | \text{pack} \ e \text{ with } I \]

Value \[ v ::= \text{true} | \text{false} | \text{fix} \ f(x). \text{v} | \text{fix}_{NC} \ f(x). e | \text{nil} | \text{cons}(e_1, e_2) | \text{pack} \ e \text{ with } I | \Lambda i.e | \text{cons}_{NC}(v_1, v_2) | \text{cons}_{C}(v_1, v_2) \]

Figure 9: Syntax of values and expressions in RelRef Core

\[ \tau ::= \text{bool} | \text{bool}_u | \tau_1 \to \tau_2 | \text{list}[n]^n \tau | \forall i::S. \tau | \exists i::S. \tau | \Box \tau | C \land \tau | C \lor \tau \]

Expressions \[ e ::= x | \text{true} | \text{false} | \text{if} \ e \text{ then } e_1 \text{ else } e_2 | \text{fix} \ f(x). e | \text{fix}_{NC} \ f(x). e | e_1 \cdot e_2 | \text{NC} \ e | \text{split}(e_1, e_2) \text{ with } C | \text{contra} \ e | \text{der} \ e | \Lambda i.e | e[I] | \text{pack} \ e \text{ with } I \]

Value \[ v ::= \text{true} | \text{false} | \text{fix} \ f(x). \text{v} | \text{fix}_{NC} \ f(x). e | \text{nil} | \text{cons}(e_1, e_2) | \text{pack} \ e \text{ with } I | \Lambda i.e | \text{cons}_{NC}(v_1, v_2) | \text{cons}_{C}(v_1, v_2) \]

Figure 10: Syntax of values and expressions in BiRelRef
\[
\begin{align*}
\Gamma(x) &= \tau \\
\Delta; \Phi_a; \Gamma \vdash x \sim x : \tau & \quad \text{rr-var} \\
\Delta; \Phi_a; \Gamma \vdash b \sim b : \text{bool} & \quad \text{rr-bool}
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \text{bool} & \quad \text{rr-u-bool} \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 : \tau & \Delta; \Phi_a; \Gamma \vdash e_2 \sim e'_2 : \tau & \quad \text{rr-if}
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash \text{fix}\ f(x).e_1 \sim \text{fix}\ f(x).e_2 : \tau_1 & \Rightarrow \tau_2 & \quad \text{rr-fix}
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 : \tau_1 \Rightarrow \tau_2 & \quad \Delta; \Phi_a; \Gamma \vdash e_2 \sim e'_2 : \tau_1 & \quad \text{rr-app}
\end{align*}
\]

\[
\Delta; \Phi_a; \Gamma \vdash \forall x \in \text{dom}(\Gamma). \Delta; \Phi_a \vdash \Gamma(x) \subseteq \Box \Gamma(x) & \quad \text{rr-nochange}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 : \tau & \quad \Delta; \Phi_a \vdash \tau \subseteq \tau' & \quad \text{rr-}\subseteq
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash \tau_1 \rightarrow \tau_2 & \quad \text{wwf}
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; x : \tau_1, f : \Box (\tau_1 \rightarrow \tau_2), \Gamma \vdash e \sim e : \tau_2 & \forall x \in \text{dom}(\Gamma), \Delta; \Phi_a \vdash \Gamma(x) \subseteq \Box \Gamma(x) & \quad \text{rr-fixNC}
\end{align*}
\]

\[
\begin{align*}
i : S, \Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau & \quad \text{rr-iLam} \\
i \notin \text{FIV}(\Phi_a, \Gamma) \\
\Delta; \Phi_a; \Gamma \vdash \Lambda e \sim \Lambda e' : \forall i : S. \tau & \quad \text{rr-iApp} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau\{I/i\} & \quad \text{rr-pack}
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash \text{pack}\ e \sim \text{pack}\ e' : \exists i : S. \tau & \quad \text{rr-unpack1}
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \land C; \Gamma \vdash e \sim e' : \tau & \quad \text{rr-c-implII} \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 : C \supset \tau & \Delta; \Phi_a \vdash C & \quad \text{rr-c-implE}
\end{align*}
\]

Figure 11: RelRef typing rules (Part 1)
\[\Delta; \Phi \vdash \tau \text{ wf} \]
\[\Delta; \Phi, a; \Gamma \vdash \text{n} \ni \text{n} : \text{list}[0]^{\tau} \quad \text{rr-nil}\]

\[\begin{align*}
\Delta; \Phi; \Gamma \vdash e_1 \ni e'_1 : \tau & \quad \Delta; \Phi; \Gamma \vdash e_2 \ni e'_2 : \text{list}[n]^{\tau} \\
\Delta; \Phi; \Gamma \vdash \text{cons}(e_1, e_2) \ni \text{cons}(e'_1, e'_2) : \text{list}[n+1]^{\tau} & \quad \text{rr-cons1}
\end{align*}\]

\[\begin{align*}
\Delta; \Phi; \Gamma \vdash e_1 \ni e'_1 : \Box \tau & \quad \Delta; \Phi; \Gamma \vdash e_2 \ni e'_2 : \text{list}[n]^{\tau} \\
\Delta; \Phi; \Gamma \vdash \text{cons}(e_1, e_2) \ni \text{cons}(e'_1, e'_2) : \text{list}[n+1]^{\tau} & \quad \text{rr-cons2}
\end{align*}\]

\[\Delta; \Phi, a; \Gamma \vdash \text{clet} e_1 \text{ as } x \in e_2 \ni e'_1 \ni e'_2 : \tau \\
\Delta; \Phi, \Gamma \vdash 
\]

\[\begin{align*}
\Delta; \Phi, \Gamma \vdash \text{n} \ni \text{n} & \quad \text{rr-contra}
\end{align*}\]

Figure 12: RelRef typing rules (Part 2)
\[
\begin{align*}
\Gamma \vdash \text{bool} \subseteq \text{bool} & \quad \Gamma \vdash \tau_1 \subseteq \tau_2 \quad \Gamma \vdash \tau_2 \subseteq \tau_1' \quad \Gamma \vdash \tau_1 \rightarrow \tau_2 \subseteq \tau_1' \rightarrow \tau_2' \quad \Gamma \vdash \tau \subseteq \text{refl} \\
\Gamma \vdash \Box (\tau_1 \rightarrow \tau_2) \subseteq \Box \tau_1 \rightarrow \Box \tau_2 & \quad \Gamma \vdash \Box \tau \subseteq \Box \text{diff} \\
\Delta; \Phi_a \models n \neq n' & \quad \Delta; \Phi_a \models \alpha \leq \alpha' \quad \Delta; \Phi_a \models \tau \subseteq \tau' \\
\Delta; \Phi_a \models \text{list}[\alpha^n \tau] \subseteq \text{list}[\alpha'^n \tau'] & \quad \Delta; \Phi_a \models \text{list}[\alpha^n \Box \tau] \subseteq \Box \text{list}[\alpha'n \tau] \\
\Delta; \Phi_a \models \text{int} \subseteq \Box \text{int} & \quad \Delta; \Phi_a \models \Box \tau \subseteq \Box \Box \tau \\
\Delta; \Phi_a \models \tau_1 \subseteq \tau_2 & \quad \Delta; \Phi_a \models \tau_2 \subseteq \tau_3 & \quad \Delta; \Phi_a \models \tau_1 \cap \tau_3 \quad \text{trans} \\
i :: S; \Delta; \Phi_a \models \tau \subseteq \tau' & \quad i \not\in \text{FV}(\Phi_a) \quad \forall \text{diff} \\
\Delta; \Phi_a \models \Box S. \tau \subseteq \Box \forall i :: S. \tau' & \quad \Delta; \Phi_a \models \Box \exists i :: S. \tau \subseteq \Box \exists i :: S. \tau' \quad \exists \text{diff} \\
\Delta; \Phi_a \models C \wedge \tau \subseteq \tau' & \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad \text{c-impl} \\
\Delta; \Phi_a \models C \wedge \tau \subseteq \tau' & \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad \text{c-and} \\
\Delta; \Phi_a \models C \cap \tau \subseteq \tau' & \quad \Delta; \Phi_a \models \Box C \cap \tau \subseteq \Box (C \cap \tau) \quad \text{c-and} \\
\Delta; \Phi_a \models C \cap \tau \subseteq \tau' & \quad \Delta; \Phi_a \models C \cap \tau \subseteq \tau' \quad \text{c-impl} \\
\Delta; \Phi_a \models \Box (i :: S. \tau) \subseteq \Box \forall i :: S. \Box \tau & \quad \Delta; \Phi_a \models \Box \exists i :: S. \Box \tau \subseteq \Box \exists i :: S. \Box \tau \quad \Box \text{diff} \\
\end{align*}
\]

Figure 13: RelRef Subtyping rules
\[ \Delta; \Phi \models \tau_1 \equiv \tau_2 \]
checks whether \( \tau_1 \) is equivalent to \( \tau_2 \).

\[
\begin{array}{c}
\Delta; \Phi \models \tau_1 \equiv \tau'_1 \\
\Delta; \Phi \models \tau_2 \equiv \tau'_2 \\
\Delta; \Phi \models \tau_1 \rightarrow \tau_2 \equiv \tau'_1 \rightarrow \tau'_2
\end{array}
\]

\[
\begin{array}{c}
\Delta; \Phi \models n \equiv n' \\
\Delta; \Phi \models \alpha \equiv \alpha'
\end{array}
\]

\[
\begin{array}{c}
\Delta; \Phi \models \text{list}[\eta]\tau \equiv \text{list}[\eta']\tau'
\end{array}
\]

\[
\Delta; \Phi \models \Box \tau \equiv \Box \tau'
\]

\[
\begin{array}{c}
i, \Delta; \Phi \models \tau \equiv \tau'
\end{array}
\]

\[
i \notin \text{FV}(\Phi_a)
\]

\[
\Delta; \Phi \models \exists i :: S. \tau \equiv \exists i :: S. \tau'
\]

\[
\Delta; C \land \Phi_a \models C \quad \Delta; C \land \Phi_a \models C' \\
\Delta; \Phi_a \models \tau \equiv \tau'
\]

\[
\Delta; C \land \Phi_a \models C' \quad \Delta; C' \land \Phi_a \models C \\
\Delta; \Phi_a \models \tau \equiv \tau'
\]

\[
i, \Delta; \Phi_a \models \tau \equiv \tau'
\]

\[
\Delta; \Phi_a \models \forall i :: S. \tau \equiv \forall i :: S. \tau'
\]

\[
\Delta; \Phi_a \models \tau \equiv \tau'
\]

Figure 14: RelRef Core type equivalence rules
\[
\Gamma(x) = \tau \\
\Delta; \Phi_\alpha; \Gamma \vdash x \sim x : \tau \quad \text{c-r-var}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \tau_1 \rightarrow \tau_2 \text{ wf} \quad \Delta; \Phi_\alpha; \Gamma \vdash x : \tau_1, f : \tau_1 \rightarrow \tau_2, \Gamma \vdash e_1 \sim e_2 : \tau_2
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \text{fix}(f(x), e_1 \sim \text{fix}(f(x), e_2) : \tau_1 \rightarrow \tau_2) \quad \text{c-r-fix}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \tau_1 \rightarrow \tau_2 \text{ wf} \quad \Delta; \Phi_\alpha; \Gamma \vdash x : \tau_1, f : \Box(\tau_1 \rightarrow \tau_2), \Box \Gamma \vdash e \sim e : \tau_2
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \Box, \Gamma' \vdash \text{fix}_{\Box \Gamma' \vdash f(x), e \sim \text{fix}_{\Box \Gamma' \vdash f(x), e : \Box(\tau_1 \rightarrow \tau_2)} \quad \text{c-r-fixNC}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash e_1 \sim e'_1 : \tau \quad \Delta; \Phi_\alpha; \Gamma \vdash e_2 \sim e'_2 : \text{list}[n]^{\alpha \tau}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \text{cons}(e_1, e_2) \sim \text{cons}(e'_1, e'_2) : \text{list}[n + 1]^{\alpha + 1 \tau} \quad \text{c-r-cons1}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash e_1 \sim e'_1 : \Box \tau \quad \Delta; \Phi_\alpha; \Gamma \vdash e_2 \sim e'_2 : \text{list}[n]^{\alpha \tau}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \text{cons}_{\Box \Gamma \vdash \text{list}[n]}(e_1, e_2) \sim \text{cons}_{\Box \Gamma \vdash \text{list}[n]}(e'_1, e'_2) : \text{list}[n + 1]^{\alpha \tau} \quad \text{c-r-cons2}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash e \sim e' : \text{list}[n]^{\alpha \tau} \quad \Delta; \Phi_\alpha; \Gamma \vdash e_1 \sim e'_1 : \tau' \quad i, \beta, \Delta; \Phi_\alpha; \Gamma \vdash n = i + 1 \land \alpha = \beta + 1 \land h : \Box \tau, t \vdash \text{list}[i]^{\alpha \tau}, \Gamma \vdash e_2 \sim e'_2 : \tau'
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \text{case } e \text{ of nil } \rightarrow e_1 \quad \text{case } e' \text{ of nil } \rightarrow e'_1 \quad \text{c-caseL}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash e_1 \sim e_2 : \tau \quad \Delta; \Phi_\alpha; \Gamma \vdash e' \sim e' : \tau
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \Box, \Gamma' \vdash e \sim e : \tau \quad \text{c-nochange}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash e \sim e' : \tau \quad \Delta; \Phi_\alpha; \Gamma \vdash \tau \equiv \tau' \quad \text{c-r-\equiv}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \tau \equiv \tau \quad \Delta; \Phi_\alpha; \Gamma \vdash \Box, \Gamma' \vdash e \sim e : \tau \quad \text{c-r-split}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \tau \equiv \tau \quad \Delta; \Phi_\alpha; \Gamma \vdash \text{split } (e_1, e'_1) \text{ with } C \vdash \text{split } (e_2, e'_2) \text{ with } C' : \tau \quad \text{c-r-split}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \bot \quad \Delta; \Phi_\alpha; \Gamma \vdash \text{wF} \quad \text{c-r-contra}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash e \sim e' : \forall i : S. \tau \quad \Delta \vdash I : S \quad \text{c-r-iApp}
\]
\[
\Delta; \Phi_\alpha; \Gamma \vdash \text{fix}(I/e, I/\tau) \quad \text{c-r-iLam}
\]

Figure 15: RelRef Core typing rules (Part 1)
\[
\begin{align*}
\Gamma: \Phi; C \vdash e \sim e' : \tau & \quad \text{c-r-cimpl} \\
\Delta; \Phi; \Gamma \vdash e \sim e' : C \supset \tau & \quad \text{c-r-cimpI} \\
\Delta; \Phi; \Gamma \vdash \text{celim } e \sim \text{celim } e' : \tau & \quad \text{c-r-cimpE} \\
\Delta; \Phi; \Gamma \vdash \text{celim } e \sim \text{celim } e' : \tau & \quad \text{c-r-candI} \\
\Delta; \Phi; \Gamma \vdash e_1 \sim e'_1 : C & \quad \text{c-r-candE} \\
\Delta; \Phi; \Gamma \vdash e_2 \sim e'_2 : \tau_2 & \quad \text{c-r-app} \\
\Delta; \Phi; \Gamma \vdash \text{let } x = e_1 \text{ in } e_2 \sim \text{let } x = e'_1 \text{ in } e'_2 : \tau_2 & \quad \text{c-r-let} \\
\Delta; \Phi; \Gamma \vdash \text{pack } e \text{ with } I \sim \text{pack } e' \text{ with } I : \exists i : S. \tau & \quad \text{c-r-pack} \\
\Delta; \Phi; \Gamma \vdash \text{unpack } e_1 \text{ as } (x, i) \text{ in } e_2 \sim \text{unpack } e'_1 \text{ as } (x, i) \text{ in } e'_2 : \tau_2 & \quad \text{c-r-unpack1}
\end{align*}
\]

Figure 16: RelRef Core typing rules (Part 2)
\[ \Gamma(x) = \tau \]
\[ \Delta; \Phi_a; \Gamma \vdash x \rightsquigarrow x : \tau \text{ e-r-var} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{fix } f(x).e_1 \rightsquigarrow \text{fix } f(x).e_2 \rightsquigarrow \text{fix } f(x).e_1^* \rightsquigarrow \text{fix } f(x).e_2^* : \tau_1 \rightarrow \tau_2 \text{ e-r-fix} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{fix } f(x).e_1 \rightsquigarrow \text{fix } f(x).e_2 \rightsquigarrow \text{fix } f(x).e_1^* \rightsquigarrow \text{fix } f(x).e_2^* : \tau_1 \rightarrow \tau_2 \text{ e-r-fixNC} \]
\[ \Delta; \Phi_a; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e^{**} : C \supset \tau \text{ e-r-implI} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{celim } e \rightsquigarrow \text{celim } e' \rightsquigarrow e^{*} \rightsquigarrow e^{**} : C \supset \tau \text{ e-r-implE} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{celim } e \rightsquigarrow \text{celim } e' \rightsquigarrow e^{*} \rightsquigarrow e^{**} : C \supset \tau \text{ e-r-andI} \]
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e_1' \rightsquigarrow e_1^* \rightsquigarrow e_1^{**} : C \& \tau_1 \text{ e-r-c-andE} \]
\[ \Delta; \Phi_a; \Gamma \vdash e_2 \rightsquigarrow e_2' \rightsquigarrow e_2^* \rightsquigarrow e_2^{**} : \tau_2 \text{ e-r-c-andE} \]

Figure 17: RelRef embedding rules (Part 1)
\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \]

\[ \Delta; \Phi; \Gamma \vdash \text{cons}(e_1, e_2) \rightsquigarrow \text{cons}(e_1', e_2') \rightsquigarrow \text{cons}_C(e_1', e_2') \rightsquigarrow \text{cons}_C(e_1^*, e_2^*) : \text{list}^n[i] \tau \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \square \tau \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \]

\[ \forall x_i \in \text{dom}(\Gamma), \ e_i = \text{coerce}_{\Gamma(x_i), \Box(x_i)}(x_i), \ \Delta; \Phi \models \Gamma(x_i) \subseteq \Box(\Gamma(x_i)) \]

\[ \Delta; \Phi; \Gamma; \Gamma' \vdash e \rightsquigarrow e \rightsquigarrow \text{let } y_i = e_i \text{ in } \text{NC} e^*[\text{der } y_i/x_i] \rightsquigarrow \text{let } y_i = e_i \text{ in } \text{NC} e^*[\text{der } y_i/x_i] : \Box \tau \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \]

\[ i, \Delta; \Phi \land n = i + 1; h : \Box \tau, tl : \text{list}^n[i] \tau, \Gamma \vdash e \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau' \]

\[ i, \beta, \Delta; \Phi \land n = i + 1 \land \alpha = \beta + 1; h : \tau, tl : \text{list}^n[i] \tau, \Gamma \vdash e \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau' \]

\[ \Delta; \Phi; \Gamma \vdash \text{case of nil } \rightarrow e_1 \quad \text{case of nil } \rightarrow e_1' \quad \text{case of nil } \rightarrow e_1^* \quad \text{case of nil } \rightarrow e_1^* \]

\[ \Delta; \Phi; \Gamma \vdash i : S, \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_2^* : \tau \quad \Delta; \Phi; \Gamma \vdash i \not\in \text{FIV}(\Phi; \Gamma) \]

\[ \Delta; \Phi; \Gamma \vdash e_1^* \rightsquigarrow e_2^* : \tau \]

\[ i \not\in \text{FIV}(\Phi; \Gamma) \]

\[ \Delta; \Phi; \Gamma \vdash h :: tl \rightarrow e_2 \rightsquigarrow h :: tl \rightarrow e_2' \rightsquigarrow h :: tl \rightarrow e_2^* \]

\[ \Delta; \Phi; \Gamma \vdash h :: tl \rightarrow e_2^* : \tau' \]

\[ \Delta; \Phi; \Gamma \vdash h :: C tl \rightarrow e_2^* : \tau' \]

\[ \Delta; \Phi; \Gamma \vdash h :: C tl \rightarrow e_2^* : \tau' \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau' \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e' \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau' \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \]

\[ \Delta; \Phi; \Gamma \vdash \text{pack } e \rightsquigarrow \text{pack } e' \rightsquigarrow \text{pack } e'' \text{ with } I \rightsquigarrow \text{pack } e'' \text{ with } I \rightsquigarrow \exists i : S, \tau \]

\[ \Delta; \Phi; \Gamma \vdash \text{pack } e \rightsquigarrow \text{pack } e' \rightsquigarrow \text{pack } e'' \text{ with } I \rightsquigarrow \text{pack } e'' \text{ with } I \rightsquigarrow \exists i : S, \tau \]

\[ \Delta; \Phi; \Gamma \vdash \text{unpack } e_1 \text{ as } x \text{ in } e_2 \text{ as } x \text{ in } e_2^* \text{ as } (x, i) \text{ in } e_2^* \text{ as } (x, i) \text{ in } e_2^* \]

\[ \Delta; \Phi; \Gamma \vdash e_1 \rightsquigarrow e_1' \rightsquigarrow e_1^* : \exists i : S, \tau_1 \]

\[ i \not\in \text{FV}(\Phi; \Gamma), \Delta; \Phi; \Gamma \vdash e_2 \rightsquigarrow e_2' \rightsquigarrow e_2^* : \tau_2 \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e_1 \rightsquigarrow e_2 \rightsquigarrow e_3 \rightsquigarrow e_4 \rightsquigarrow e_5 \rightsquigarrow \text{split}(e_1', e_1^*) \text{ with } C \rightsquigarrow \text{split}(e_2', e_2^*) \text{ with } C \rightsquigarrow \tau \]

\[ \Delta; \Phi; \Gamma \vdash e \rightsquigarrow e_1 \rightsquigarrow e_2 \rightsquigarrow e_3 \rightsquigarrow e_4 \rightsquigarrow e_5 \rightsquigarrow \text{split}(e_1', e_1^*) \text{ with } C \rightsquigarrow \text{split}(e_2', e_2^*) \text{ with } C \rightsquigarrow \tau \]

\[ \Delta; \Phi ; \Gamma \vdash \bot \]

\[ \Delta; \Phi ; \Gamma \vdash \text{contra } e_1 \text{ contra } e_2 : \tau \]

Figure 18: RefRef embedding rules (Part 2)
\[ \Delta; \psi_a; \Phi_a \models \tau_1 \equiv \tau'_1 \Rightarrow \Phi_1 \quad \Delta; \psi_a; \Phi_a \models \tau_2 \equiv \tau'_2 \Rightarrow \Phi_2 \]

\[ \Phi = \Phi_1 \land \Phi_2 \]

\[ \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \quad \text{alg-r-fun} \]

\[ i, \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \quad \text{alg-r-forall} \]

\[ \Delta; \psi_a; \Phi_a \models \forall i :: S. \tau \equiv \forall i :: S. \tau' \Rightarrow \forall i :: S. \Phi \quad \text{alg-r-list} \]

\[ i, \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \quad \text{alg-r-∃} \]

\[ \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \quad \text{alg-r-cprod} \]

\[ \Delta; \psi_a; \Phi_a \models \Box \tau_1 \equiv \Box \tau_2 \Rightarrow \Phi \quad \text{alg-r-□} \]

\[ \Delta; \psi_a; \Phi_a \models C \supset \tau \equiv C' \supset \tau' \Rightarrow C \leftrightarrow C' \land \Phi \quad \text{alg-r-cimpl} \]

Figure 19: RelRef algorithmic type equivalence rules
\[
\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e \downarrow \tau \Rightarrow \Phi \quad \text{alg-r-nochange-} \\
\Delta; \psi_a; \Phi_a; \Gamma; \Box \Gamma \vdash e \circ \Box \Gamma \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash \text{der} e_1 \circ \text{der} e_2 \uparrow \tau \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash x \circ x \uparrow \tau \Rightarrow \Gamma \quad \text{alg-r-var-} \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash f : \tau_1 \rightarrow \tau_2, x : \tau_1, \Gamma \vdash e \circ e' \downarrow \tau_2 \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash \text{fix} f(x).e \circ \text{fix} f(x).e' \downarrow \tau_1 \rightarrow \tau_2 \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a; \Gamma' \vdash \text{fix}_NC f(x).e \circ \text{fix}_NC f(x).e \downarrow \Box (\tau_1 \rightarrow \tau_2) \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_1' \uparrow \tau_1 \rightarrow \tau_2 \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e_2 \circ e_2' \downarrow \tau_1 \Rightarrow \Phi_2 \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_1' \uparrow \tau_2 \Rightarrow \Phi_1 \land \Phi_2 \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash \text{nil} \circ \text{nil} \downarrow \text{list}[n]^{\alpha} \tau \Rightarrow n = 0 \\
\Delta; i, \beta \in \text{fresh}(N) \\
\Delta; i, \beta, \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_1' \downarrow \tau \Rightarrow \Phi_1 \\
\Delta; i, \beta, \psi_a; \Phi_a; \Gamma \vdash e_2 \circ e_2' \downarrow \text{list}[i]^{\beta} \tau \Rightarrow \Phi_2 \\
\Delta; i, \psi_a; \Phi_a; \Gamma \vdash \text{cons}_C(e_1, e_2) \circ \text{cons}_C(e_1', e_2') \downarrow \text{list}[i]^{\alpha} \tau \Rightarrow \Phi_1 \land \exists i : N. \Phi'_2 \\
\Delta; i \in \text{fresh}(N) \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \Box \tau \Rightarrow \Phi_e \\
\Delta; i, \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_1' \downarrow \text{list}[i]^{\alpha} \tau \Rightarrow \Phi_e \\
\Delta; i : N, \Delta; \psi_a; \Phi_a; n = 1 + i \land \alpha = \beta + 1 \land \Phi_a : h : \tau, t : \text{list}[i]^{\beta} \tau, \Gamma \vdash e_2 \circ e_2' \downarrow \tau' \Rightarrow \Phi_2 \\
\Delta; i : N, \beta : N, \Delta; \psi_a; n \in i + 1 \land \alpha = \beta + 1 \land \Phi_a : h : \tau, t : \text{list}[i]^{\beta} \tau, \Gamma \vdash e_3 \circ e_3' \downarrow \tau' \Rightarrow \Phi_3 \\
\Phi_{\text{body}} = (n \geq 0 \vdash \Phi_1) \land (\forall i : N. (n \geq i + 1) \rightarrow (\Phi_2 \land \forall \beta : N. (\alpha = \beta + 1) \rightarrow \Phi_3)) \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \text{nil} \Rightarrow e_1 \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \text{nil} \Rightarrow e_1' \\
\Delta; h : \text{NC} t \downarrow \tau_2 \Rightarrow \tau_3' \rightarrow (\Phi_e \land \Phi_{\text{body}}) \\
\Delta; h : \text{NC} t \downarrow \tau_2 \Rightarrow e_2' \downarrow \tau_3' \Rightarrow (\Phi_e \land \Phi_{\text{body}}) \\
\Delta; h : \text{NC} t \downarrow \tau_2 \Rightarrow e_3' \downarrow \tau_3' \Rightarrow (\Phi_e \land \Phi_{\text{body}}) \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \text{NC} e \downarrow \Box \tau \Rightarrow \Phi \\
\Gamma(x) = \tau \\
\Delta; \psi_a; \Phi_a; \Gamma \vdash x \circ x \uparrow \tau \Rightarrow \Gamma .
\]

Figure 20: RelRef algorithmic typing rules (Part 1)
\[
\begin{align*}
i &:: S, \Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \downarrow \tau \Rightarrow \Phi \\
\Delta; \psi_a; \Phi; \Gamma \vdash \Lambda e \circ \Lambda e' \downarrow \forall i::S. \tau \Rightarrow (\forall i::S. \Phi) \quad \text{alg-r-iLam-\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \uparrow \forall i::S. \tau' \Rightarrow \Phi \\
\Delta; \psi_a; \Phi; \Gamma \vdash e[I] \circ e'[I] \uparrow \tau'[I/i] \Rightarrow \Phi \quad \text{alg-r-iApp-\uparrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \downarrow \tau[I/i] \Rightarrow \Phi \\
\Delta; \psi_a; \Phi; \Gamma \vdash I :: S \\
\Delta; \psi_a; \Phi; \Gamma \vdash \text{pack} e \text{ with } I \circ \text{pack } e' \text{ with } I \downarrow \exists i::S. \tau \Rightarrow \Phi \quad \text{alg-r-pack-\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e_1 \circ e_1' \uparrow \exists i::S. \tau_1 \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi; \Gamma \vdash e_2 \circ e_2' \downarrow \tau_2 \Rightarrow \Phi_2 \\
\Delta; \psi_a; \Phi; \Gamma \vdash (\forall i \notin FV(\Phi_a; \Gamma; \tau_2)) \quad \Phi = (\Phi_1 \land \forall i::S. \Phi_2) \quad \text{alg-r-unpack-\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash \text{celim } e \text{ as } (x, i) \text{ in } e_2 \circ \text{celim } e_1' \text{ as } (x, i) \text{ in } e_2' \downarrow \tau_2 \Rightarrow \Phi \quad \text{alg-r-c-andI-\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \downarrow \tau \Rightarrow \Phi \\
\Delta; \psi_a; \Phi; \Gamma \vdash e_1 \circ e_2 \downarrow C \circ \tau \Rightarrow C \land (C \Rightarrow \Phi) \quad \text{alg-r-c-implI-\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \uparrow C \circ \tau \Rightarrow \Phi \\
\Delta; \psi_a; \Phi; \Gamma \vdash \text{celim } e \circ \text{celim } e' \uparrow \tau \Rightarrow (C \land \Phi) \quad \text{alg-r-c-implE-\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \uparrow \tau' \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi; \Gamma \vdash \tau' \equiv \tau \Rightarrow \Phi_2 \quad \text{alg-r-\uparrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \downarrow \tau \Rightarrow \Phi \quad \text{alg-r-anno-\uparrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash (e: \tau) \circ (e': \tau) \uparrow \tau \Rightarrow \Phi \quad \text{alg-r-contrada} \\
\Delta; \psi_a; \Phi; \Gamma \vdash \text{split } (e_1, e_2) \text{ with } C \circ \text{split } (e_1', e_2') \text{ with } C \downarrow \tau \Rightarrow C \Rightarrow \Phi_1 \land \neg C \Rightarrow \Phi_2 \quad \text{alg-r-spli\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash e \circ e' \downarrow \tau \Rightarrow \Phi \quad \text{alg-r-contra\downarrow} \\
\Delta; \psi_a; \Phi; \Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 \text{ if } e' \text{ then } e_1' \text{ else } e_2' \downarrow \tau \Rightarrow \Phi_1 \land \Phi_2 \land \Phi_3 \quad \text{alg-r-if} \\
\end{align*}
\]

Figure 21: RelRef algorithmic typing rules (Part 2)
2.2 RelRef Lemmas

Lemma 7 (Substitution of RelRef Core)
Assume that

1. \( \Delta; \Psi; \Gamma \vdash e_1 \sim e_2 : \tau \) (1).
2. \( \Delta; \Psi; \Gamma, x : \tau \vdash e'_1 \sim e'_2 : \tau' \) (2).

then \( \Delta; \Psi; \Gamma \vdash e'_1[e_1/x] \sim e'_2[e_2/x] : \tau' \).

Proof. By induction on the typing derivation of the second assumption (2).

Case \( (\Gamma, x : \tau)(z) = \tau' \) c-r-var
Subcase: \( z = x \).
TS: \( \Delta; \Phi; \Gamma \vdash z[e_1/x] \sim z[e_2/x] : \tau' \).
Because \( x = z, z[e_1/x] = e_1, z[e_2/x] = e_2 \), STS: \( \Delta; \Phi; \Gamma \vdash e_1 \sim e_2 : \tau' \).
From \( (\Gamma, x : \tau)(z) = \tau' \) and \( z = x \), we know \( \tau = \tau' \).
By the above statements, STS: \( \Delta; \Phi; \Gamma \vdash e_1 \sim e_2 : \tau \), which is proved by the assumption.
Subcase: \( z \neq x \).
we know: \( \Gamma(z) = \tau' \),
TS: \( \Delta; \Phi; \Gamma \vdash z[e_1/x] \sim z[e_2/x] : \tau' \).
Because \( x \neq z, z[e_1/x] = z, z[e_2/x] = z \), STS: \( \Delta; \Phi; \Gamma \vdash z \sim z : \tau' \).
By the core rule c-r-vars and (\ast), we construct the derivation :
\[ \Gamma(z) = \tau' \]

Case \( \Delta; \Phi; \Gamma, x : \tau \vdash e''_1 \sim e''_2 : \square \tau' \) (3) c-r-der
\[ \Delta; \Phi; \Gamma, x : \tau \vdash \text{der } e''_1 \sim \text{der } e''_2 : \tau' \]
\( e'_1 = \text{der } e''_1, e'_2 = \text{der } e''_2 \).
TS: \( \Delta; \Phi; \Gamma \vdash \text{der } (e''_1[e_1/x]) \sim \text{der } (e''_2[e_2/x]) : \tau' \).
By IH on (1) and (3), \( \Delta; \Phi; \Gamma \vdash e''_1[e_1/x] \sim e''_2[e_2/x] : \tau' \) (4).
By core rule c-r-der and (4), we can derive:
\[ \Delta; \Phi; \Gamma, x : \tau \vdash e''_1[e_1/x] \sim e''_2[e_2/x] : \tau' \]

Case \( \Delta; \Phi; \Gamma, x : \tau \vdash \text{fix } f(z).e''_1[e_1/x] \sim \text{fix } f(z).e''_2[e_2/x] : \tau' \) c-r-fix
\[ \Delta; \Phi; \Gamma, x : \tau \vdash \text{fix } f(z).e''_1 \sim \text{fix } f(z).e''_2 : \tau_2 \] (3)
\[ \Delta; \Phi; \Gamma, x : \tau \vdash e''_1 \sim e''_2 : \tau_2 \]
\( e'_1 = \text{fix } f(z).e''_1, e'_2 = \text{fix } f(z).e''_2, \tau' = \square (\tau_1 \rightarrow \tau_2) \).
TS: \( \Delta; \Phi; \Gamma \vdash \text{fix } f(z).(e''_1[e_1/x]) \sim \text{fix } f(z).(e''_2[e_2/x]) : \tau' \).
By IH on (1) and (3), \( \Delta; \Phi; \Gamma, x : \tau \vdash e''_1 \sim e''_2 : \tau_2 \) (4).
By core rule c-r-fix and (4), we can derive:
\[ \Delta; \Phi; \Gamma, x : \tau \vdash \text{fix } f(z).(e''_1[e_1/x]) \sim \text{fix } f(z).(e''_2[e_2/x]) : \tau' \]

Case \( \Delta; \Phi; \Gamma, x : \tau \vdash \text{fix}_{\text{NC}} f(z).e \sim \text{fix}_{\text{NC}} f(z).e : \tau_2 \) c-r-fixNC
\( e'_1 = \text{fix } f(z).e, e'_2 = \text{fix } f(z).e, \tau' = \square (\tau_1 \rightarrow \tau_2) \).
TS: \( \Delta; \Phi; \Gamma \vdash \text{fix } f(z).(e[e_1/x]) \sim \text{fix } f(z).(e[e_2/x]) : \tau' \).
By IH on (1) and (3), \( \Delta; \Phi; \Gamma, x : \tau \vdash e \sim e : \tau_2 \) (4).
By core rule c-r-fixNC and (4), we can derive:
\[ \Delta; \Phi; \Gamma, x : \tau \vdash \text{fix } f(z).(e[e_1/x]) \sim \text{fix } f(z).(e[e_2/x]) : \tau' \]
Case \[\Delta; \Phi_a; \Gamma, x : \tau \vdash e_3 \rightsquigarrow e_3';\tau'' \quad (3) \quad \Delta; \Phi_a; \Gamma, x : \tau \vdash e_4 \rightsquigarrow e_4'; list[n]^\alpha \tau'' \quad (4)\]

\[e_1 = \text{cons}(e_3, e_4), \ e_2 = \text{cons}(e_3', e_4'), \tau' = \text{list}[n+1]^\alpha \tau''.\]

TS: \[\Delta; \Phi_a; \Gamma \vdash \text{cons}_C(e_3[e_1/x], e_4[e_1/x]) \sim \text{cons}_C(e_3'[e_2/x], e_4'[e_2/x]) : \tau'.\]

By IH on (1) and (3), \[\Delta; \Phi_a; \Gamma \vdash e_3[e_1/x] \sim e_3'[e_2/x] : \tau''.\]

By IH on (1) and (4), \[\Delta; \Phi_a; \Gamma \vdash e_4[e_1/x] \sim e_4'[e_2/x] : \text{list}[n]^\alpha \tau''.\]

By core rule c-r-cons1 and (5),(6), we can derive:

\[\Delta; \Phi_a; \Gamma \vdash \text{cons}_C(e_3[e_1/x], e_4[e_1/x]) \sim \text{cons}_C(e_3'[e_2/x], e_4'[e_2/x]) : \tau'.\]

Case \[\Delta; \Phi_a; \square \Gamma, x : \tau \vdash e \rightsquigarrow e : \tau'' \quad (3) \quad \text{c-nochange}\]

\[e_1 = \text{NC} e, \ e_2 = \text{NC} e, \tau' = \square \tau''.\]

TS: \[\Delta; \Phi_a; \Gamma \vdash \text{NC} e[e_1/x] \sim \text{NC} e[e_2/x] : \tau''.\]

By IH on (1) and (3), \[\Delta; \Phi_a; \Gamma \vdash e[e_1/x] \sim e[e_2/x] : \tau''.\]

By core rule c-nochange and (4), we can derive:

\[\Delta; \Phi_a; \Gamma \vdash \text{NC} e[e_1/x] \sim \text{NC} e[e_2/x] : \tau''.\]

\[\square\]

Lemma 8 (Reflexivity of Algorithmic Binary Type Equivalence in RelRef)

\[\forall \Delta, \psi_a, \Phi_a, \tau. \text{there exists } \Phi \text{ s.t } \Delta; \psi_a; \Phi_a \models \tau \equiv \tau \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]

\[\text{Proof.}\] By induction on the binary type.

Case \[\forall i::S. \tau\]

Assume \[\Delta; \psi_a, \Phi_a.\]

TS: \[\exists \Phi \text{ s.t } \Delta; \psi_a; \Phi_a \models \forall i::S. \tau \equiv \forall i::S. \tau \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]

By IH on \(\tau\) where \(\Delta' = i, \Delta, \exists \Phi' \text{ s.t } \Delta'; \psi_a; \Phi_a \models \tau \equiv \tau \Rightarrow \Phi' \text{ and } \Delta'; \psi_a; \Phi_a \models \Phi'.\)

By the above statement and algorithmic type eq rules alg-r-forall, we conclude the following statement where \(\Phi' = \forall i::S. \Phi'.\)

\[\Delta; \psi_a; \Phi_a \models \forall i::S. \tau \equiv \forall i::S. \tau \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]

Case \[\tau_1 \rightarrow \tau_2\]

Assume \[\Delta; \psi_a, \Phi_a.\]

TS: \[\exists \Phi \text{ s.t } \Delta; \psi_a; \Phi_a \models \tau_1 \rightarrow \tau_2 \equiv \tau_1 \rightarrow \tau_2 \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]

By IH on \(\tau_1, \exists \Phi_1 \text{ s.t } \Delta; \psi_a; \Phi_a \models \tau_1 \equiv \tau_1 \Rightarrow \Phi_1 \text{ and } \Delta; \psi_a; \Phi_a \models \Phi_1.\)

By IH on \(\tau_2, \exists \Phi_2 \text{ s.t } \Delta; \psi_a; \Phi_a \models \tau_2 \equiv \tau_2 \Rightarrow \Phi_2 \text{ and } \Delta; \psi_a; \Phi_a \models \Phi_2.\)

By the above statement and algorithmic type eq rules alg-r-fun, we conclude the following statement where \(\Phi = \Phi_1 \land \Phi_2.\)

\[\Delta; \psi_a; \Phi_a \models \tau_1 \rightarrow \tau_2 \equiv \tau_1 \rightarrow \tau_2 \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]

Case \[\text{list}[n]^\alpha \tau\]

Assume \[\Delta; \psi_a, \Phi_a.\]

TS: \[\exists \Phi \text{ s.t } \Delta; \psi_a; \Phi_a \models \text{list}[n]^\alpha \tau \equiv \text{list}[n]^\alpha \tau \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]

By IH on \(\tau, \exists \Phi_1 \text{ s.t } \Delta; \psi_a; \Phi_a \models \tau \equiv \tau \Rightarrow \Phi_1 \text{ and } \Delta; \psi_a; \Phi_a \models \Phi_1.\)

By the above statement and algorithmic type eq rules alg-r-list, we conclude the following statement where \(\Phi = \Phi_1 \land n = n \land \alpha = \alpha.\)

\[\Delta; \psi_a; \Phi_a \models \text{list}[n]^\alpha \tau \equiv \text{list}[n]^\alpha \tau \Rightarrow \Phi \text{ and } \Delta; \psi_a; \Phi_a \models \Phi.\]
Case $\exists i::S.\tau$
Assume $\Delta;\psi_a;\Phi_a$.
TS: $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models \exists i::S.\tau \equiv \exists i::S.\tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$
By IH on $\tau$, choose $i \notin FV(\Phi_a)$, $\exists \Phi s.t. i,\Delta;\psi_a;\Phi_a \models \tau \equiv \tau \Rightarrow \Phi_1$ and $i,\Delta;\psi_a;\Phi_a \models \Phi_1$.
By the above statement and algorithmic type eq rules $\text{alg-r-}$, we conclude the following statement
where $\Phi = \forall i::S.\Phi_1$.
$\Delta;\psi_a;\Phi_a \models \exists i::S.\tau \equiv \exists i::S.\tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$

Case $\Box \tau$
Assume $\Delta;\psi_a;\Phi_a$.
TS: $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models \Box \tau \equiv \Box \tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$
By IH on $\tau$, $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models \tau \equiv \tau \Rightarrow \Phi_1$ and $\Delta;\psi_a;\Phi_a \models \Phi_1$.
By the above statement and algorithmic type eq rules $\text{B-}$, we conclude the following statement
where $\Phi = \Phi_1$.
$\Delta;\psi_a;\Phi_a \models \Box \tau \equiv \Box \tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$

Case $C \& \tau$
Assume $\Delta;\psi_a;\Phi_a$.
TS: $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models C \& \tau \equiv C \& \tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$
By IH on $\tau$, $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models \tau \equiv \tau \Rightarrow \Phi_1$ and $\Delta;\psi_a;\Phi_a \models \Phi_1$.
By the above statement and algorithmic type eq rules $\text{alg-r-cprod}$, we conclude the following statement
where $\Phi = C \leftrightarrow C \& \Phi_1$.
$\Delta;\psi_a;\Phi_a \models C \& \tau \equiv C \& \tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$

Case $C \supset \tau$
Assume $\Delta;\psi_a;\Phi_a$.
TS: $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models C \supset \tau \equiv C \supset \tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$
By IH on $\tau$, $\exists \Phi s.t. \Delta;\psi_a;\Phi_a \models \tau \equiv \tau \Rightarrow \Phi_1$ and $\Delta;\psi_a;\Phi_a \models \Phi_1$.
By the above statement and algorithmic type eq rules $\text{alg-r-cimpl}$, we conclude the following statement
where $\Phi = C \leftrightarrow C \& \Phi_1$.
$\Delta;\psi_a;\Phi_a \models C \supset \tau \equiv C \supset \tau \Rightarrow \Phi$ and $\Delta;\psi_a;\Phi_a \models \Phi$

\begin{proof}

\textbf{Theorem 9 (Soundness of the Algorithmic Binary Type Equality in RelRef)}
Assume that
1. $\Delta;\psi_a;\Phi_a \models \tau \equiv \tau' \Rightarrow \Phi$
2. $\text{FIV}(\Phi_a, \tau, \tau') \subseteq \Delta, \psi_a$
3. $\Delta;\Phi_a[\theta_a] \models \Phi[\theta_a]$ is provable and $\Delta \vdash \theta_a : \psi_a$ is derivable.

Then $\Delta;\Phi_a[\theta_a] \models \tau[\theta_a] \equiv \tau'[\theta_a]$.

\textit{Proof.} By induction on the algorithmic binary type equivalence derivation.

\begin{align*}
\Delta;\psi_a;\Phi_a & \models \tau_1 \equiv \tau'_1 \Rightarrow \Phi_1 & \Delta;\psi_a;\Phi_a & \models \tau_2 \equiv \tau'_2 \Rightarrow \Phi_2 \\
\Phi & = \Phi_1 \& \Phi_2 & \text{alg-r-fun}
\end{align*}

Assume $\text{FIV}(\Phi_a, \tau_1 \rightarrow \tau_2, \tau'_1 \rightarrow \tau'_2) \subseteq \Delta, \psi_a$ and $\Delta;\Phi_a[\theta_a] \models \Phi[\theta_a]$ is provable and $\Delta \vdash \theta_a : \psi_a$ is
Case 

\[ \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \quad \Phi' = \Phi \land n \equiv n' \land \alpha \equiv \alpha' \]

\[ \text{Alg-r-list} \]

Assume FIV(\(\Phi_a, \text{list}[n^\alpha \tau] \equiv \text{list}[n'^{\alpha'} \tau'] \)) \subseteq \Delta, \psi_a \text{ and } \Delta; \Phi_a[\theta_a] \models \Phi'[\theta_a] \text{ is provable and } \Delta \not\supset \theta_a : \psi_a \text{ is derivable.}

TS: \Delta; \Phi_a[\theta_a] \models (\text{list}[n^\alpha \tau])[\theta_a] \equiv (\text{list}[n'^{\alpha'} \tau'])[\theta_a].

We know \(\Phi' = \Phi \land n \equiv n' \land \alpha \equiv \alpha'\) s.t.

From assumption \(\Delta; \Phi_a[\theta_a] \models \Phi'[\theta_a]\), we know FIV(\(\Phi_a, \tau, n, \alpha, \tau', n', \alpha'\)) \subseteq \Delta, \psi_a.

From assumption \(\Delta; \Phi_a[\theta_a] \models \Phi'[\theta_a]\), we get \(\Delta; \Phi_a[\theta_a] \models \Phi'[\theta_a]\) is provable and \(\Delta; \Phi_a[\theta_a] \models (n \equiv n')[\theta_a]\) and \(\Delta; \Phi_a[\theta_a] \models (\alpha \equiv \alpha')[\theta_a]\).

By IH on the first premise \(\Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi\) with the same substitution \(\theta_a\) and the above statements, \(\Delta; \Phi_a[\theta_a] \models \tau[\theta_a] \equiv \tau'[\theta_a]\).

By the above statements and RelRef type equivalence rule \text{eq-list}, we conclude that

\[ \Delta; \Phi_a[\theta_a] \models (\text{list}[n^\alpha \tau])[\theta_a] \equiv (\text{list}[n'^{\alpha'} \tau'])[\theta_a] \]

Case 

\[ i, \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \]

\[ \text{Alg-r-forall} \]

Assume FIV(\(\Phi_a, \forall i : S. \tau \equiv \forall i : S. \tau' \)) \subseteq \Delta, \psi_a \text{ and } \Delta; \Phi_a[\theta_a] \models \Phi'[\theta_a] \text{ is provable and } \Delta \not\supset \theta_a : \psi_a \text{ is derivable.}

TS: \Delta; \Phi_a[\theta_a] \models \forall i : S. \tau[\theta_a] \equiv \forall i : S. \tau'[\theta_a].

We know \(\Phi' = \forall i : S. \Phi\) s.t.

From assumption FIV(\(\Phi_a, \forall i : S. \tau, \forall i : S. \tau'\)) \subseteq \Delta, \psi_a, \text{ we know FIV(\(\Phi_a, \tau, \tau', i\)) \subseteq (i, \Delta), \psi_a\).

From assumption \(\Delta; \Phi_a[\theta_a] \models \forall i : S. \Phi[\theta_a]\), we get \(i, \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a]\) is provable.

By IH on the first premise \(i, \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi\) with the above statements, \(i, \Delta; \Phi_a[\theta_a] \models \tau[\theta_a] \equiv \tau'[\theta_a]\).

By the above statements and RelRef type equivalence rule \text{eq-V}, we conclude that

\[ \Delta; \Phi_a[\theta_a] \models \forall i : S. \tau[\theta_a] \equiv \forall i : S. \tau'[\theta_a] \]

Case 

\[ \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \]

\[ \text{Alg-r-cprod} \]

Assume FIV(\(\Phi_a, C \land \tau \equiv C' \land \tau' \Rightarrow C \leftrightarrow C' \land \Phi\)) \subseteq \Delta, \psi_a \text{ and } \Delta; \Phi_a[\theta_a] \models \Phi'[\theta_a] \text{ is provable and } \Delta \not\supset \theta_a : \psi_a \text{ is derivable.}
TS: \( \Delta; \Phi_a[\theta_a] \models (C \land \tau)[\theta_a] \). We know \( \Phi' = C \leftrightarrow C' \land \Phi \) s.t. 

From assumption FIV(\( \Phi_a, C, \tau, C', \tau' \)) \( \subseteq \Delta, \psi_a \), we know FIV(\( \Phi_a, \tau, \tau', C, C' \)) \( \subseteq \Delta, \psi_a \). 

From assumption \( \Delta; \Phi_a[\theta_a] \models (C' \land \Phi)[\theta_a] \), we get \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) is provable and \( \Delta; \Phi_a[\theta_a] \models C[\theta_a] \) and \( \Delta; \Phi_a[\theta_a] \models C'[\theta_a] \). 

By IH on the second premise \( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \) with the above statements, \( \Delta; \Phi_a[\theta_a] \models \tau[\theta_a] \equiv \tau'[\theta_a] \). 

By the above statements and RelRef type equivalence rule eq-c-prod, we conclude that 
\[
\Delta; \Phi_a[\theta_a] \models (C \land \tau)[\theta_a] \equiv (C' \land \tau'')[\theta_a]
\]

\[\blacksquare\]

**Theorem 10** (Completeness of the Binary Algorithmic Type Equivalence in RelRef) 
Assume that \( \Delta; \Phi_a \models \tau \equiv \tau' \). Then \( \exists \Phi \) such that \( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi \) and \( \Delta; \Phi_a \models \Phi \).

**Proof.** By induction on the binary equivalence typing derivation.

**Case** 
\[
\Delta; \Phi_a \models \tau \equiv \tau'
\]
By IH on the first premise \( \Delta; \Phi_a \models \tau \equiv \tau' \), there exists \( \Phi_1 \) s.t. \( \Delta; \Phi_a \models \tau_1 \equiv \tau'_1 \Rightarrow \Phi_1 \) and \( \Delta; \Phi_a \models \Phi_1 \).

By IH on the second premise \( \Delta; \Phi_a \models \tau_2 \equiv \tau'_2 \), there exists \( \Phi_2 \) s.t. \( \Delta; \Phi_a \models \tau_2 \equiv \tau'_2 \Rightarrow \Phi_2 \) and \( \Delta; \Phi_a \models \Phi_2 \).

By the above statements and algorithmic type equivalence rules alg-r-fun with an empty \( \Psi_a \), we conclude the following statement where \( \Phi = \Phi_1 \land \Phi_2 \).
\[
\Delta; \Phi_a \models \tau \equiv \tau'
\]

**Case** 
\[
\Delta; \Phi_a \models \text{list}[n]^\tau \equiv \text{list}[n']^\tau'
\]
By IH on the third premise \( \Delta; \Phi_a \models \tau \equiv \tau' \), there exists \( \Phi_1 \) s.t. \( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi_1 \) and \( \Delta; \Phi_a \models \Phi_1 \).

By the above statements, the first and second premises and algorithmic type equivalence rules alg-r-list with an empty \( \Psi_a \), we conclude the following statement where \( \Phi = \Phi_1 \land n \equiv n' \land \alpha \equiv \alpha' \).
\[
\Delta; \Phi_a \models \text{list}[n]^\tau \equiv \text{list}[n']^\tau' \Rightarrow \Phi \) and \( \Delta; \Phi_a \models \Phi \).

**Case** 
\[
\Delta; \Phi_a \models \tau \equiv \tau'
\]
By IH on the premise \( \Delta; \Phi_a \models \tau \equiv \tau' \), there exists \( \Phi_1 \) s.t. \( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi_1 \) and \( \Delta; \Phi_a \models \Phi_1 \).

By the above statements and algorithmic type equivalence rules alg-r-\( \Box \) with an empty \( \Psi_a \), we conclude the following statement where \( \Phi = \Phi_1 \).
\[
\Delta; \Phi_a \models \Box \tau \equiv \Box \tau' \Rightarrow \Phi \) and \( \Delta; \Phi_a \models \Phi \).

**Case** 
\[
i, \Delta; \Phi_a \models \tau \equiv \tau'
\]
By IH on the premise \( i, \Delta; \Phi_a \models \tau \equiv \tau' \), there exists \( \Phi_1 \) s.t. \( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi_1 \) and \( i, \Delta; \Phi_a \models \Phi_1 \).

By the above statements and algorithmic type equivalence rules alg-r-forall with an empty \( \Psi_a \), we
conclude the following statement where $\Phi = \forall i : S_1$.
$\Delta; \Phi_a \models \forall i : S, \tau \equiv \forall i : S, \tau' \Rightarrow \Phi$ and $\Delta; \Phi_a \models \Phi$.

Case $\Delta; C' \land \Phi_a \models C \quad \Delta; C \land \Phi_a \models C' \quad \Delta; \Phi_a \models \tau \equiv \tau'$

\[ \text{eq-c-prod} \]

By IH on the third premise $\Delta; \Phi_a \models \tau \equiv \tau'$, there exists $\Phi_1$ s.t $\Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi_1$ and $\Delta; \Phi_a \models \Phi_1$.
From the first two premises $\Delta; C' \land \Phi_a \models C$ and $\Delta; C \land \Phi_a \models C'$, we know that $\Delta; \Phi_a \models C \equiv C'$.

By the above statements and the algorithmic type equivalence rules $\text{alg-r-cprod}$ with an empty $\Psi_a$, we conclude the following statement where $\Phi = C \equiv C' \land \Phi_1$.
$\Delta; \Phi_a \models C \land \tau \equiv C' \land \Phi_1$.

\[ \square \]

**Lemma 11 (Existence of coercions for relational subtyping in RelRef)**

If $\Delta; \Phi_a \models \tau \equiv \tau'$ then there exists $e \in \text{RelRef Core}$ s.t. $\Delta; \Phi_a; \vdash e : \tau \equiv \tau'$.

**Proof.** Proof is by induction on the subtyping derivation. We denote the witness $e$ of type $\tau \rightarrow \tau'$ as $\text{coerce}_{\tau, \tau'}$ for clarity.

Case $\Delta; \Phi_a \models \tau'_1 \subseteq \tau_1 \quad (\ast) \quad \Delta; \Phi_a \models \tau_2 \subseteq \tau'_2 \quad (\circ) \quad \Delta; \Phi_a \models \tau_1 \rightarrow \tau_2 \rightarrow \tau'_1 \rightarrow \tau'_2$.

By IH on (\ast), $\exists \text{coerce}_{\tau_1, \tau_1}$. $\Delta; \Phi_a; \vdash \text{coerce}_{\tau_1, \tau_1} \sim \text{coerce}_{\tau_1, \tau_1} : \tau_1 \rightarrow \tau_1$
By IH on (\circ), $\exists \text{coerce}_{\tau_2, \tau_2}$. $\Delta; \Phi_a; \vdash \text{coerce}_{\tau_2, \tau_2} \sim \text{coerce}_{\tau_2, \tau_2} : \tau_2 \rightarrow \tau_2$

Then, using these two statements, we can construct the following derivation where $e = \text{fix } f(x).\text{fix } f(y).\text{coerce}_{\tau_2, \tau'_2} (x \circ y)$
$\Delta; \Phi_a; \vdash e : \tau_1 \rightarrow \tau_2 \rightarrow \tau'_1 \rightarrow \tau'_2$

Case $\Delta; \Phi_a \models \text{int}_r \subseteq \Box \text{int}_r$

Then, we can construct the derivation using the primitive function $\text{box}_{\text{int}_r} : \text{int}_r \rightarrow \Box \text{int}_r$

$\Delta; \Phi_a; \vdash \text{fix } f(x).\text{box}_{\text{int}_r} x : \text{fix } f(x).\text{box}_{\text{int}_r} x : \text{int}_r \rightarrow \Box \text{int}_r$

Case $\Delta; \Phi_a \models \Box \tau \subseteq \tau$

Then, we can immediately construct the derivation using the rule $\text{c-der}$.

$\Delta; \Phi_a; \vdash \text{fix } f(x).\text{der } x : \text{fix } f(x).\text{der } x : \Box \tau \rightarrow \tau$

Case $\Delta; \Phi_a \models \Box \tau \subseteq \Box \Box \tau$

Then, we can immediately construct the derivation using the rule $\text{c-nochange}$.

$\Delta; \Phi_a; \vdash \text{fix } f(x).\text{NC } x : \text{fix } f(x).\text{NC } x : \Box \tau \rightarrow \Box \Box \tau$

Case $\Delta; \Phi_a \models \tau_1 \subseteq \tau_2 (\ast)$

By IH on (\ast), $\exists \text{coerce}_{\tau_1, \tau_2}$. $\Delta; \Phi_a; \vdash \text{coerce}_{\tau_1, \tau_2} \sim \text{coerce}_{\tau_1, \tau_2} : \tau_1 \rightarrow \tau_2$

28
Then, using (*)& the rules c-der and c-nochange, we can construct the derivation
\[
\Delta; \Phi_a; \vdash \fix f(x).\NC (\coerce_{\tau_1,\tau_2} (\der x)) \rightsquigarrow \fix f(x).\NC (\coerce_{\tau_1,\tau_2} (\der x)) : \square \tau_1 \rightarrow \square \tau_2
\]

Case \(\Delta; \Phi_a \models \tau \subseteq \tau\)

Then, we can immediately construct the derivation
\[
\Delta; \Phi_a; \vdash \fix f(x).x \rightsquigarrow \fix f(x).x : \tau \rightarrow \tau
\]

Case \(\Delta; \Phi_a \models \tau_1 \subseteq \tau_2 \quad (*) \quad \Delta; \Phi_a \models \tau_2 \subseteq \tau_3 \quad (\o)\) trans

By IH on (*)& \(\exists \coerce_{\tau_1,\tau_2} \cdot i :: S,\Delta; \Phi_a ; \vdash \coerce_{\tau_1,\tau_2} \rightsquigarrow \coerce_{\tau_1,\tau_2} : \tau_1 \rightarrow \tau_2\)

By IH on (\o) \(\exists \coerce_{\tau_2,\tau_3} \cdot i :: S,\Delta; \Phi_a ; \vdash \coerce_{\tau_2,\tau_3} \rightsquigarrow \coerce_{\tau_2,\tau_3} : \tau_2 \rightarrow \tau_3\)

Then, using (*)& (\o), we can construct the derivation simply by function composition
\[
\Delta; \Phi_a; \vdash \fix f(x).\coerce_{\tau_2,\tau_3} (\coerce_{\tau_1,\tau_2} x) \rightsquigarrow \fix f(x).\coerce_{\tau_2,\tau_3} (\coerce_{\tau_1,\tau_2} x) : \tau_1 \rightarrow \tau_3
\]

Case \(\Delta; \Phi_a \models \diff(\tau_1) \subseteq \square \tau_1 \rightarrow \square \tau_2\)

Then, we can immediately construct the derivation where \(e = \fix f(x) . \fix (y).\NC (\der x) (\der y)\)
\[
\Delta; \Phi_a; \vdash e \rightsquigarrow e : \square (\tau_1 \rightarrow \tau_2) \rightarrow \square \tau_1 \rightarrow \square \tau_2
\]

Case \(i :: S,\Delta; \Phi_a \models \tau \subseteq \tau' \quad (*) \quad i \notin \FV(\Phi_a) \quad \forall \diff\)

By IH on (*)& \(\exists \coerce_{\tau,\tau'} \cdot i :: S,\Delta; \Phi_a ; \vdash \coerce_{\tau,\tau'} \rightsquigarrow \coerce_{\tau,\tau'} : \tau \rightarrow \tau'\)

Then, using this, and the c-r-iLam and c-r-iApp rules, we can construct the following derivation:
\[
\Delta; \Phi_a; \vdash \fix f(x).\lambda i.\coerce_{\tau,\tau'} (x [i]) \rightsquigarrow \fix f(x).\lambda i.\coerce_{\tau,\tau'} (x [i]) : \forall i :: S. \tau \rightarrow \forall i :: S. \tau'
\]

Case \(\Delta; \Phi_a \models \diff(\forall i :: S. \tau) \subseteq \square \forall i :: S. \square \tau\)

Then, we can immediately construct the following derivation using the c-der, c-nochange, c-r-iLam and c-r-iApp rules.
\[
\Delta; \Phi_a; \vdash \fix f(x).\lambda i.\NC ((\der x) [i]) \rightsquigarrow \fix f(x).\lambda i.\NC ((\der x) [i]) : \square (\forall i :: S. \tau) \rightarrow \forall i :: S. \square \tau
\]

Case \(\Delta; \Phi_a \models n \equiv n' \quad (*) \quad \Delta; \Phi_a \models \alpha \leq \alpha' \quad (\o) \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\dagger)\)

By IH on (\dagger) \(\exists \coerce_{\tau,\tau'} \cdot \Delta; \Phi_a ; \vdash \coerce_{\tau,\tau'} \rightsquigarrow \coerce_{\tau,\tau'} : \tau \rightarrow \tau'\)

We first construct the more generic term for type:
\[
\text{unit}_\tau \rightarrow \forall n :: \mathbb{N}. \forall n' :: \mathbb{N}. \forall \alpha :: \mathbb{N}. \forall \alpha' :: \mathbb{N}. ((n = n' \land \alpha \leq \alpha') \rightarrow \text{list}[n]^{\alpha'} \tau') \quad (2.1)
\]
and then instantiate the term for eq. (5.1) later. It can be shown that such a derivation can be constructed for expression
\[ e' = \text{fix } f\text{List}(\_).\Lambda n.\Lambda n'.\Lambda \alpha.\Lambda \alpha'.\text{fix } f(x).\text{clet } x \text{ as } e \text{ in } \]
case e of
  \[ \text{nil } \rightarrow \text{nil} \]
  \[ h :: N \ t l \rightarrow \text{let } r = f\text{List } (n - 1)[n'] [\alpha][\alpha'] t l \text{ in } \]
  \[ \text{cons}_{NC}(\text{NC}(\text{coerce}_{\tau}, \tau') \text{ der } h, r) \]
  \[ h :: C \ t l \rightarrow \text{let } r = f\text{List } (n - 1)[n'] [\alpha][\alpha'] t l \text{ in } \]
  \[ \text{cons}_{C}(\text{coerce}_{\tau}, \tau' h, r) \]
Then, we can instantiate fList using (\*\*) and (\*\*\*) as follows where
\[ e'' = \text{fix } f(x).f\text{List } (n' \alpha) t \rightarrow \text{list}[n' \alpha' \tau' \]
\[ \Delta; \Phi_a; \vdash e'' \vdash e'' : \text{list}\{n\alpha\tau \rightarrow \text{list}[n'\alpha'\tau' \]

\[ \Delta; \Phi_a; \vdash \text{list}[n\alpha\tau \rightarrow \Box (\text{list}[n\alpha\tau) \]

\[ \Delta; \Phi_a; \vdash \text{unit}_{\tau} \rightarrow \forall \alpha::N.\forall \alpha::N.\text{list}[n\alpha\tau \rightarrow \Box (\text{list}[n\alpha\tau) \]

and then instantiate the term for eq. (5.2) later. It can be shown that such a derivation can be constructed for expression
\[ e' = \text{fix } f\text{List}(\_).\Lambda n.\Lambda \alpha.\text{fix } f(x).\text{clet } x \text{ as } e \text{ in } \]
case e of
  \[ \text{nil } \rightarrow \text{nil} \]
  \[ h :: N \ t l \rightarrow \text{let } r = f\text{List } (n - 1)[n'] [\alpha][\alpha'] t l \text{ in } \]
  \[ \text{cons}_{NC}(\text{NC}(\text{der } h, \text{der } r)) \]
  \[ h :: C \ t l \rightarrow \text{let } r = f\text{List } (n - 1)[n'] [\alpha][\alpha'] t l \text{ in } \]
  \[ \text{cons}_{C}(\text{der } h, \text{der } r)) \]
Then, we can instantiate fList with a concrete n and \( \alpha \) as follows where \( e'' = \text{fix } f(x).f\text{List } (n\alpha) x \)
\[ \Delta; \Phi_a; \vdash e'' \vdash e'' : \text{list}[n\alpha\tau \rightarrow \Box (\text{list}[n\alpha\tau) \]

\[ \Delta; \Phi_a; \vdash \alpha = 0 \rightarrow \Box \text{list}[n\alpha\tau \rightarrow \Box (\text{list}[n\alpha\tau) \]

and then instantiate the term for eq. (5.3) later. It can be shown that such a derivation can be constructed for expression
\[ e' = \text{fix } f\text{List}(\_).\Lambda n.\Lambda \alpha.\text{fix } f(x).\text{clet } x \text{ as } e \text{ in } \]
case e of
  \[ \text{nil } \rightarrow \text{nil} \]
  \[ h :: N \ t l \rightarrow \text{let } r = f\text{List } (n - 1)[n'] [\alpha][\alpha'] t l \text{ in } \]
  \[ \text{cons}_{NC}(\text{NC}(\text{h, r}) \text{ der } h, r)) \]
  \[ h :: C \ t l \rightarrow \text{contra} \]
Then, we can instantiate fList with a concrete n and \( \alpha \) (note the premise \( \alpha = 0 \)) as follows where
\[ e'' = \text{fix } f(x).f\text{List } (n\alpha) x \]
\[ \Delta; \Phi_a; \vdash e'' \vdash e'' : \text{list}[n\alpha\tau \rightarrow \Box (\text{list}[n\alpha\tau) \]

30
Case
\[ i :: S, \Delta; \Phi_a \models \tau \subseteq \tau' \quad (*) \quad i \notin FV(\Phi_a) \]

By IH on \((*)\), \(\exists \text{coerce}_{\tau,\tau'} \quad i :: S, \Delta; \Phi_i \vdash \text{coerce}_{\tau,\tau'} \quad \vdash \tau \rightarrow \tau'\)

Then, using this and the c-r-pack and c-r-unpack rules, we can construct the following derivation where \(e = \text{fix } f(x).\text{unpack } x\) as \((y, i)\) in \(\text{pack } (\text{coerce}_{\tau,\tau'}\ y)\) with \(i\)

\[ \Delta; \Phi_u ; \vdash e \leadsto e : \vdash (\exists i :: S. \tau) \rightarrow \exists i :: S. \tau' \]

Case
\[ \frac{\Delta; \Phi_a \models C \vdash C \quad (*) \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\cdot)}{\Delta; \Phi_a \models C \supset \tau \supset C' \supset \tau'} \text{c-impl} \]

By IH on \((*)\), \(\exists \text{coerce}_{\tau,\tau'} \quad \Delta; \Phi_u ; \vdash \text{coerce}_{\tau,\tau'} \quad \vdash \tau \rightarrow \tau'\)

Then, using this and the premise \((*)\) along with the c-r-c-implI and c-r-c-implE rules, we can construct the following derivation where \(e = \text{fix } f(x).\text{coerce}_{\tau,\tau'}\) (celim \(x\))

\[ \Delta; \Phi_u ; \vdash e \leadsto e : \vdash (C \supset \tau) \rightarrow C' \supset \tau' \]

Case
\[ \frac{\Delta; \Phi_a \models \Box (C \supset \tau) \subseteq C \supset \Box \tau}{\text{c-impl-\Box}} \]

Then, we can immediately construct the following derivation using the c-der, c-nochange, c-r-pack, and c-r-unpack rules in in Figures 80 and 83 where \(e = \text{fix } f(x).\text{unpack } x\) as \((y, i)\) in \(\text{NC} (\text{pack } \text{der } y\) with \(i\).

\[ \Delta; \Phi_u ; \vdash e \leadsto e : \vdash \Box (C \supset \tau) \rightarrow (C \supset \Box \tau) \]

Case
\[ \frac{\Delta; \Phi_a \models C \vdash C' \quad (*) \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\cdot)}{\Delta; \Phi_a \models C \& \tau \subseteq C' \& \tau'} \text{c-and} \]

By IH on \((*)\), \(\exists \text{coerce}_{\tau,\tau'} \quad \Delta; \Phi_i ; \vdash \text{coerce}_{\tau,\tau'} \quad \vdash \tau \rightarrow \tau'\)

Then, using this and the premise \((*)\) along with the c-r-c-prodI and c-r-c-prodE rules, we can construct the following derivation where \(e = \text{fix } f(x).\text{clet } x\) as \(y\) in \(\text{coerce}_{\tau,\tau'}\ y\)

\[ \Delta; \Phi_u ; \vdash e \leadsto e : \vdash (C \& \tau) \rightarrow C' \& \tau' \]

Case
\[ \frac{\Delta; \Phi_a \models C \& \Box \tau \subseteq \Box (C \& \tau)}{\text{c-and-\Box}} \]

Then, we can immediately construct the following derivation using the c-der, c-nochange, c-r-c-prodI and c-r-c-prodE rules where \(e = \text{fix } f(x).\text{clet } x\) as \(y\) in \(\text{NC} (\text{der } y)\).
\( \Delta; \Phi_a; \vdash e \rightsquigarrow e : (C \& \Box \tau) \rightarrow \Box (C \& \tau) \)

\[\square\]

**Theorem 12** (Types are preserved by embedding for RelRef)

1. If \( \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e_2 \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \), then \( \Delta; \Phi_a; \Gamma \vdash e_1^* \rightsquigarrow e_2^* : \tau \) and \( \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e_2 : \tau \).

**Proof.** Proof is by induction on the embedding derivations.

**Case** 
\( \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e_2 \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau \)  \( (\ast) \)

\( \Delta; \Phi_a \models \tau \sqsubseteq \tau' \)  \( (\circ) \)

\( e' = \text{coerce}_{\tau', \tau}^\ast \)  \( (\dagger) \)

By applying c-r-fixNC.

By IH on (\ast), \( \Delta; \Phi_a; \Gamma \vdash e_1^* \rightsquigarrow e_2^* : \tau \)  \( (\ast \ast \ast) \)

By Lemma 11 using (\circ), we know that \( \Delta; \Phi_a; \vdash e' \rightsquigarrow e' : \tau \rightarrow \tau' \)  \( (\infty) \).

By applying c-r-app rule and (\ast \ast \ast) and (\infty), we get \( \Delta; \Phi_a; \Gamma \vdash e' e_1^* \rightsquigarrow e' e_2^* : \tau' \)  \( (\bullet) \).

By reflexivity of binary type equivalence, we know \( \Delta; \Phi_a \models \tau' \equiv \tau' \)  \( (\bullet \bullet) \).

Then, we conclude as follows:

\( \Delta; \Phi_a; \Gamma \vdash e' e_1^* \rightsquigarrow e' e_2^* : \tau' \)  \( (\bullet) \)

\( \Delta; \Phi_a \models \tau' \equiv \tau' \)  \( (\bullet \bullet) \)

By (\ast \ast \ast) and (\circ) and rr-\( \subseteq \), we also conclude \( \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e_2 : \tau' \).

**Case** 
\( \Delta; \Phi_a \vdash \tau_1 \rightarrow \tau_2 \)  \( \text{uf} \)

\( \Delta; \Phi_a; x : \tau_1 ; f : \tau_1 \rightarrow \tau_2 ; \Gamma \vdash e_1 \rightsquigarrow e_2 \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau_2 \)

By applying c-r-fixNC.

By IH on the premise \( \Delta; \Phi_a; x : \tau_1 ; f : \tau_1 \rightarrow \tau_2 ; \Gamma \vdash e_1 \rightsquigarrow e_2 \rightsquigarrow e_1^* \rightsquigarrow e_2^* : \tau_2 ; \Delta; \Phi_a; x : \tau_1 ; f : \tau_1 \rightarrow \tau_2 , \Gamma \vdash e_1^* \rightsquigarrow e_2^* : \tau_2 \)  \( (\ast) \).

By applying c-r-fix rule and (\ast), we get \( \Delta; \Phi_a; \Gamma \vdash f(x).e_1^* \rightsquigarrow f(x).e_2^* : \tau_1 \rightarrow \tau_2 \)  \( (\bullet) \).

Similarly, using rr-fix, we conclude \( \Delta; \Phi_a; \Gamma \vdash f(x).e_1 \rightsquigarrow f(x).e_2 : \tau_1 \rightarrow \tau_2 \).

\( \Delta; \Phi_a \vdash \tau_1 \rightarrow \tau_2 \)  \( \text{uf} \)

\( \Delta; \Phi_a; x : \tau_1 ; f : \Box (\tau_1 \rightarrow \tau_2) ; \Gamma \vdash e \rightsquigarrow e \rightsquigarrow e^* \rightsquigarrow e^* : \tau_2 \)

\[ \forall x_i \in \text{dom}(\Gamma) , \quad e_i = \text{coerce}_{\Gamma(x_i) \sqsubseteq \Box \Gamma(x_i)}^* \]

\( \Delta; \Phi_a \models \Box \Gamma(x) \subseteq \Box \Gamma(x) \)

\( e^* = \text{let } y_i = e_i x_i \text{ in } \text{fix}_{\text{NC}} f(x).e^* \text{[der } y_i/x_i] \)

By applying c-r-let rule on \( e^* \).

\( \Delta; \Phi_a; \Gamma \vdash \text{fix}_{\text{NC}} f(x).e \rightsquigarrow \text{fix}_{\text{NC}} f(x).e \rightsquigarrow e^* \rightsquigarrow e^* : \Box (\tau_1 \rightarrow \tau_2) \)

Set \( \Gamma = x_i : \Gamma(x_i) , \Gamma_1 = \Gamma x_i \).

STS: \( \Delta; \Phi_a ; \Box \Gamma_1 , \Gamma \vdash \text{fix}_{\text{NC}} f(x).e^* \text{[der } y_i/x_i] \rightsquigarrow \text{fix}_{\text{NC}} f(x).e^* \text{[der } y_i/x_i] : \Box (\tau_1 \rightarrow \tau_2) \)

By c-r-fixNC.

\( \Delta; \Phi_a ; x : \tau_1 , f : \Box (\tau_1 \rightarrow \tau_2) , \Box \Gamma_1 \vdash e \text{[der } y_i/x_i] \rightsquigarrow e \text{[der } y_i/x_i] : \tau_2 \)

\( \Delta; \Phi_a ; \Box \Gamma_1 , \Gamma \vdash \text{fix}_{\text{NC}} f(x).e \text{[der } y_i/x_i] \rightsquigarrow \text{fix}_{\text{NC}} f(x).e \text{[der } y_i/x_i] : \Box (\tau_1 \rightarrow \tau_2) \)

By c-r-fixNC.
STS: $\Delta; \Phi_a; x : \tau_1, f : \Box (\tau_1 \rightarrow \tau_2), \Box \Gamma_1 \vdash e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \tau_2$.

From (⋆), we extend the context and get $\Delta; \Phi_a; x : \tau_1, f : \Box (\tau_1 \rightarrow \tau_2), \Box \Gamma_1, \Gamma \vdash e^{*} \sim e^{*} : \tau_2$ (⋆⋆).

By Lemma 7 on (⋆⋆) for the number of $x_i$ in $\Gamma$ times, we conclude:

$\Delta; \Phi_a; x : \tau_1, f : \Box (\tau_1 \rightarrow \tau_2), \Box \Gamma_1, \Gamma \vdash e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \tau_2$.

Similarly, using (⋆⋆) and the third premise and $\mathbf{rr}$-$\mathbf{fixNC}$, we conclude $\Delta; \Phi_a; \Gamma \vdash \mathbf{fix} f(x). e \sim \mathbf{fix} f(x). e : \Box \tau_1 \rightarrow \tau_2$.

Case: $\Delta; \Phi_a; \Gamma \vdash e^{*} \sim e^{*} : \tau$ (⋆)

By Lemma 7 on (⋆) for the number of $x_i$ in $\Gamma$ times, we extend the context and get $\Delta; \Phi_a; \Gamma, \Gamma' \vdash e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \Box \tau$.

By IH on the premise $\Delta; \Phi_a; \Gamma \vdash e^{*} \sim e^{*} : \tau$, we conclude $\Delta; \Phi_a; \Gamma, \Gamma' \vdash e^{*} \sim e^{*} : \tau$ (⋆⋆).

By applying $\mathbf{c-r-let}$ rule using ** and ∞, we get $\Delta; \Phi_a; \Gamma \vdash e^{*} \sim e^{*} : \tau$ (⋆⋆).

By applying $\mathbf{c-r-let}$ rule.

$$\Delta; \Phi_a; \Gamma \vdash e^{*} \sim e^{*} : \tau$$

Set $\Gamma = \tau_1 : \Gamma(x_i), \Gamma_1 = y_i : \Gamma_i$.

STSS: $\Delta; \Phi_a; \Box \Gamma_1, \Gamma' \vdash \mathbf{c-r-let} e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \Box \tau$.

By applying $\mathbf{c-r-let}$.

$$\Delta; \Phi_a; \Box \Gamma_1, \Gamma' \vdash \mathbf{c-r-let} e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \Box \tau$$

STSS: $\Delta; \Phi_a; \Box \Gamma_1, \Gamma' \vdash e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \Box \tau$.

From (⋆), we extend the context and get $\Delta; \Phi_a; \Box \Gamma_1 \vdash e^{*} \sim e^{*} : \tau$. (⋆⋆)

By Lemma 7 on (⋆⋆) for the number of $x_i$ in $\Gamma$ times, we conclude:

$\Delta; \Phi_a; \Box \Gamma_1 \vdash e^{*}[\text{der } y_i/x_i] \sim e^{*}[\text{der } y_i/x_i] : \tau$.

Similarly, using (⋆⋆) and the third premise and $\mathbf{rr}$-$\mathbf{fixNC}$, we conclude $\Delta; \Phi_a; \Gamma \vdash e^{*} : \Box \tau$.

Theorem 13 (Completeness of embedding in RelRef)

1. If $\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 : \tau$, then there exist $e_1', e_2'$ such that $\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 \sim e_1' \sim e_2' : \tau$.

Proof. By induction on the typing derivations.

33
\[\forall x \in \text{dom}(\Gamma), \ (\ast) \Delta; \Phi_a |\!| \Gamma(x) \subseteq \Box \Gamma(x) \ (\ast) \quad \text{rr-nochange}\]

By IH on \((\ast)\), we get \(\exists e^*\) such that \(\Delta; \Phi_a; \Gamma \vdash e \leftarrow e^* \leftarrow e^* : \tau\) (††).

By Lemma 11 on \((\ast)\), we get \(\exists e_i = \text{coerce}_{\Gamma(x), \Box \Gamma(x)}\) for all \(x_i \in \text{dom}(\Gamma)\) (†††).

By e-nochange embedding rule using \((\dagger)\) and \((\dagger\dagger)\), we can conclude as follows:

\[
\forall x_i \in \text{dom}(\Gamma), \ e_i = \text{coerce}_{\Gamma(x_i), \Box \Gamma(x_i)} \ (\dagger\dagger) \quad \forall x_i \in \text{dom}(\Gamma), \ \Delta; \Phi_a |\!| \Gamma(x_i) \subseteq \Box \Gamma(x_i) \quad \text{e-nochange.}
\]

Case

By e-caseL embedding rule using \((\ast\ast)\), \((\infty)\), and \((\bigstar\bigstar)\), we can conclude as follows:

\[
\Delta; \Phi_a; \Gamma \vdash \text{case } e \text{ of nil } \to e_1 \mid h :: tl \to e_2 \quad \text{case } e^* \text{ of nil } \to e_1^* \mid h :: tl \to e_2^* \quad \text{case } e^* \text{ of nil } \to e_1^{**} \mid h :: tl \to e_2^{**} \quad \text{e-r-caseL}
\]

Case

By e-fixNC embedding rule using \((\bigstar)\) and \((\infty)\), we can conclude as follows:

\[
\Delta; \Phi_a; \Gamma \vdash f :: \Box(\tau_1 \to \tau_2), \Gamma \vdash e \Leftarrow e_2 \quad \forall x \in \text{dom}(\Gamma), \ \Delta; \Phi_a |\!| \Gamma(x) \subseteq \Box \Gamma(x) \ (\ast) \quad \text{rr-fixNC}
\]

By by IH on \((\ast)\), we get \(\exists e^*\) such that \(\Delta; \Phi_a; x : x, f :: \Box(\tau_1 \to \tau_2), \Gamma \vdash e \Leftarrow e^* \Leftarrow e^* : \tau_2\) (\(\bigstar\)).

By Lemma 11 on \((\ast)\), we get \(\exists e_i = \text{coerce}_{\Gamma(x), \Box \Gamma(x)}\) for all \(x_i \in \text{dom}(\Gamma)\) (\(\infty\)).

Case

By e-fixNC embedding rule using \((\bigstar)\) and \((\infty)\), we can conclude as follows:

\[
\Delta; \Phi_a; \Gamma \vdash f(x).e :: \Box(\tau_1 \to \tau_2) \quad \text{e-r-fixNC.}
\]

Case

By IH on \((\ast)\), we get \(\exists e^*\) and \(\exists e'^*\) such that \(i :: S, \Delta; \Phi_a; \Gamma \vdash e \Leftarrow e^* \Leftarrow e^* : \tau\) (\(\bigstar\)).

By e-lam embedding rule using \((\bigstar)\), we can conclude as follows:

\[
i :: S, \Delta; \Phi_a; \Gamma \vdash e \Leftarrow e^* \Leftarrow e^* : \tau \quad \text{e-r-lam.}
\]

34
Example of a proof in the document:

\[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' : \forall i : S. \tau \quad (\ast) \]
\[ \Delta \vdash I : S \quad (\ast) \]

**Case** \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' : [I/\tau] \]

By IH on (\ast), we get \( \exists e^* \) such that \( \Delta; \Phi; \Gamma \vdash e \rightarrow e^* \rightarrow e'' : \forall i : S. \tau \quad (\ast\ast) \).

By e-iApp, \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e^* \rightarrow e'' : \forall i : S. \tau \]
\[ \Delta \vdash I : S \quad \text{e-iApp} \]

**Case** \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' : \tau[I/i] \quad (\ast) \]
\[ \Delta \vdash I :: S \quad (\ast) \]

By IH on (\ast), we get \( \exists e^* \) such that \( \Delta; \Phi; \Gamma \vdash e \rightarrow e^* \rightarrow e'' : \tau[I/i] \quad (\ast\ast) \).

By e-pack, \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e^* \rightarrow e'' : \tau[I/i] \]
\[ \Delta \vdash I :: S \quad \text{e-pack} \]

**Case** \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' : \tau[I/i] \quad (\ast) \]
\[ \Delta \vdash I :: S \quad (\ast) \]

By IH on (\ast), we get \( \exists e^* \) and \( \exists e'' \) such that \( \Delta; \Phi; \Gamma \vdash e \rightarrow e' \rightarrow e'' : \tau[I/i] \quad (\ast\ast) \).

By e-pack, \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' \rightarrow e'' : \tau[I/i] \]
\[ \Delta \vdash I :: S \quad \text{e-pack} \]

**Case** \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' : \tau[I/i] \quad (\ast) \]
\[ \Delta \vdash e :: S \quad (\ast) \]

By IH on (\ast), we get \( \exists e' \) and \( \exists e'' \) such that \( \Delta; \Phi; \Gamma \vdash e \rightarrow e' \rightarrow e'' : \tau[I/i] \quad (\ast\ast) \).

By e-pack, \[ \Delta; \Phi; \Gamma \vdash e \rightarrow e' \rightarrow e'' : \tau[I/i] \]
\[ \Delta \vdash e :: S \quad \text{e-pack} \]

**Theorem 14** (Soundness of algorithmic typechecking in RelRef)

1. Assume that \( \Delta; \psi_\theta; \Phi ; \Gamma \vdash e \circ e \downarrow \tau \Rightarrow \Phi \) and \( \text{FIV}(\Phi, \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_\theta) \) and \( \theta_\psi \) is a valid substitution for \( \psi_\theta \) such that \( \Delta; \Phi; \theta_\psi \mid \theta \vdash \Phi[\theta_\psi] \) holds. Then, \( \Delta; \Phi; \theta_\psi[\theta_\psi] \vdash e \circ e' \downarrow \tau \Rightarrow \Phi \).

2. Assume that \( \Delta; \psi_\theta; \Phi ; \Gamma \vdash e \circ e \downarrow \tau \Rightarrow \Phi \) and \( \text{FIV}(\Phi, \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_\theta) \) and \( \theta_\psi \) is a valid substitution for \( \psi_\theta \) such that \( \Delta; \Phi; \theta_\psi \mid \theta \vdash \Phi[\theta_\psi] \) holds. Then, \( \Delta; \Phi; \theta_\psi[\theta_\psi] \vdash e \circ e' \downarrow \tau \Rightarrow \Phi \).

**Proof.** By simultaneous induction on the given algorithmic typing derivations.

**Proof of Theorem 14.1:**

**Case** \[ \Delta; \psi_\theta; \Phi ; \Gamma \vdash e \circ e \downarrow \tau \Rightarrow \Phi \quad \text{alg-r-nochange-down} \]
\[ \Delta; \psi_\theta; \Phi ; \Gamma \vdash e \circ e \downarrow \tau \Rightarrow \Phi \]
\[ \text{TS: } \Delta; \Phi[\theta_\psi]; \Gamma[\theta_\psi] \vdash e \circ e \downarrow \tau \Rightarrow \Phi \]
\[ \Delta; \Phi[\theta_\psi]; \Gamma[\theta_\psi] \vdash e \circ e \downarrow \tau \Rightarrow \Phi \]

TS: \[ \Delta; \Phi[\theta_\psi]; \Gamma[\theta_\psi] \vdash e \circ e \downarrow \tau \Rightarrow \Phi \]
\[ \Delta; \Phi[\theta_\psi]; \Gamma[\theta_\psi] \vdash e \circ e \downarrow \tau \Rightarrow \Phi \]

35
By the main assumptions, we have \( \text{FIV}(\Phi_a, \Gamma, \Box \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_a) \) (⋆) and 
\[ \Delta; \Phi_a[\theta_a] \vdash \Phi[\theta_a] \quad (**) \]
Using (⋆), we can show that

a) \( \text{FIV}(\Phi_a, \Box \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_a). \)

By IH1 on the premise using a) and (**), we can show that

\[ \Delta; \Phi_a[\theta_a]; \Box \Gamma[\theta_a] \vdash [e] \otimes [e'] : \tau[\theta_a] \]  
(2.4)

By the c-nochange rule using eq. (2.4), we obtain \( \Delta; \Phi_a[\theta_a]; \Gamma'[\theta_a], \Box \Gamma[\theta_a] \vdash \text{NC}[e] \otimes \text{NC}[e'] : \Box \tau[\theta_a] \).

\[
\begin{array}{ll}
\Delta; \psi_a; \Phi_a; \Gamma \vdash e \ominus e' & \text{list}[n] \tau \Rightarrow \Phi_e \\
\Delta; \psi_a; n \equiv 0 \wedge \Phi_a; \Gamma \vdash e_1 \ominus e'_1 \downarrow \tau' \Rightarrow \Phi_1 \\
i : \mathbb{N}, \Delta; \psi_a; n \equiv i + 1 \wedge \Phi_a; h : \Box \tau, \tau \vdash \text{list}[i] \tau, \Gamma \vdash e_2 \ominus e'_2 \downarrow \tau' \Rightarrow \Phi_2 \\
i : \mathbb{N}, \beta : \mathbb{N}, \Delta; \psi_a; n \equiv i + 1 \wedge \alpha \equiv \beta + 1 \wedge \Phi_a; h : \tau, \tau \vdash \text{list}[i] \beta, \tau, \Gamma \vdash e_3 \ominus e'_3 \downarrow \tau' \Rightarrow \Phi_3 \\
\Phi_{\text{body}} = (n \equiv 0 \rightarrow \Phi_1) \wedge (\forall i : \mathbb{N}. (n \equiv i + 1 \rightarrow (\Phi_2 \wedge \forall \beta : \mathbb{N}. (\alpha \equiv \beta + 1 \rightarrow \Phi_3)))
\end{array}
\]

\[\begin{array}{ll}
\text{Case} & \text{alg-r} \\
\text{case e of nil} \rightarrow e_1 & \text{case e' of nil} \rightarrow e'_1 \\
| h : \text{NC} \tau l \rightarrow e_2 & \text{TS:} \\
| h : \text{C} \tau l \rightarrow e_3 \\
\text{case e of nil} \rightarrow [e_1] & \text{case e' of nil} \rightarrow [e'_1] \\
\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash | h : \text{N} \tau l \rightarrow e_2 \ominus | h : \text{N} \tau l \rightarrow e'_2 \downarrow \tau' \Rightarrow (\Phi_e \wedge \Phi_{\text{body}}) \\
| h : \text{C} \tau l \rightarrow e_3 & \| h : \text{C} \tau l \rightarrow e'_3 \\
\end{array}\]

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Gamma, \tau') \subseteq \text{dom}(\Delta, \psi_a) \) (⋆) and 
\[ \Delta; \Phi_a[\theta_a] \vdash (\Phi_e \wedge \Phi_{\text{body}})[\theta_a] \quad (**) \]
Using (⋆), (**’s derivation must be in a form such that we have

a) \( \Delta; \Phi_a[\theta_a] \vdash \Phi_e[\theta_a] \)

b) \( \Delta; n \equiv 0 \wedge \Phi_a[\theta_a] \vdash \Phi_1[\theta_a] \)

c) \( i : S, \Delta; n \equiv i + 1 \wedge \Phi_a[\theta_a] \vdash \Phi_2[\theta_a] \)

d) \( i : S, \beta : S, \Delta; n \equiv i + 1 \wedge \Phi_a[\theta_a] \equiv \beta + 1 \wedge \Phi_a[\theta_a] \vdash \Phi_3[\theta_a] \)

By IH2 on the first premise using a) and (⋆), we can show that

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e] \otimes [e'] : \text{list}[n[\theta_a]]^\alpha[\theta_a] \tau[\theta_a] \]  
(2.5)

By IH1 on the second premise using b) and (⋆), we can show that

\[ \Delta; n \equiv 0 \wedge \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e_1] \otimes [e'_1] : \tau'[\theta_a] \]  
(2.6)
By IH1 on the third premise using c) and (⋆), we can show that

\[ i :: S, \Delta; n[\theta_a] \equiv i + 1 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_2| \bigcirc |e'_2| : \tau'[\theta_a] \]  

(2.7)

By IH1 on the fourth premise using d) and (⋆), we can show that

\[ i :: S, \beta :: S, \Delta; n[\theta_a] \equiv i + 1 \land \alpha[\theta_a] \equiv \beta + 1 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_3| \bigcirc |e'_3| : \tau'[\theta_a] \]  

(2.8)

Then by \textbf{c-r-caseL} rule using eqs. (2.5) to (2.8), we can show that

\[
\begin{align*}
\text{Case} & \quad \Delta; i; \psi_a; \Phi_a; \Gamma[\theta_a] \vdash \text{list}[|i|^\alpha \tau \Rightarrow \Phi_2] \\
\text{alg-r-consNC-↓} & \quad \Delta; \psi_a; \Phi_a; \Gamma[\theta_a] \vdash \text{cons}_{NC}(e_1, e_2) \circ \text{cons}_{NC}(e_1', e_2') \Downarrow \text{list}[n]^\alpha \tau \Rightarrow \Phi_1 \land \exists \beta :: \text{N}. \Phi_2' \\
\text{TS} & \quad \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{cons}_{NC}([e_1], [e_2]) \circ \text{cons}_{NC}([e_1'], [e_2']) : \text{list}[n[\theta_a]]^{\alpha[\theta_a]} \tau[\theta_a].
\end{align*}
\]

By the main assumptions, we have \( FIV(\Phi_a, \Gamma, \text{list}[n]^\alpha \tau) \subseteq \text{dom}(\Delta, \psi_a) \) (⋆) and

\[ \Delta; \Phi_a[\theta_a] \equiv (\Phi_1 \land \exists \beta :: \text{N}. \Phi_2')[\theta_a] \]  

(⋆⋆)

Using (⋆), (⋆⋆)'s derivation must be in a form such that we have

a) \( \Delta; \Phi_a[\theta_a] \equiv \Phi_1[\theta_a] \)
b) \( \Delta \vdash I :: \text{N} \)
c) \( \Delta; \Phi_a[\theta_a] \equiv \Phi_2[\theta_a, i \mapsto I] \)
d) \( \Delta; \Phi_a[\theta_a] \equiv (I + 1) \equiv n[\theta_a] \)

By IH1 on the second premise using (⋆) and a), we can show that

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_1| \bigcirc |e'_1| : \square \tau[\theta_a] \]  

(2.9)

By IH1 on the third premise using (⋆) and b), we can show that

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_2| \bigcirc |e'_2| : \text{list}[I]^\alpha[\theta_a] \tau[\theta_a] \]  

(2.10)

By \textbf{c-r-cons2} typing rule using eqs. (2.9) and (2.10), we obtain

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{cons}_{NC}([e_1], [e_2]) \circ \text{cons}_{NC}([e_1'], [e_2']) : \text{list}[I + 1]^\alpha[\theta_a] \tau[\theta_a]. \]

We conclude by applying \textbf{c-r-c} rule to this using d).

\textbf{Proof of Theorem 14.2:}
Case: \[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e \otimes e' : \tau \Rightarrow \Phi \quad \Delta; \Phi_a \vdash \tau \text{ wf} \quad \text{FIV}(\tau) \in \Delta \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash (e : \tau) \odot (e' : \tau) : \tau \Rightarrow \Phi \]

TS: \[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \tau[\theta_a] \]

Since by definition, \( |e : \cdot| = |e| \), STS: \[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \tau[\theta_a] \]

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Gamma) \subseteq \text{dom}(\Delta, \psi_a) \) (\( \ast \)) and \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) (\( \ast \ast \))

Using the third premise, we can show that

a) \( \text{FIV}(\Phi_a, \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_a) \).

By IH2 on the first premise using (\( \ast \ast \)) and a), we can conclude that

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \tau[\theta_a] \].

Case: \[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \odot e_2 \uparrow \Box \tau \Rightarrow \Phi \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash \text{der} e_1 \odot \text{der} e_2 \uparrow \tau \Rightarrow \Phi \]

TS: \[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{der} |e| \odot \text{der} |e'| : \tau[\theta_a] \]

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Gamma) \subseteq \text{dom}(\Delta, \psi_a) \) (\( \ast \)) and \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) (\( \ast \ast \)) such that \( \Delta \triangleright \theta_a : \psi_a \) are derivable.

By IH2 on the first premise using (\( \ast \)) and (\( \ast \ast \)), we obtain

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \Box \tau[\theta_a] \] (2.11)

Then, by c-der rule using eq. (5.30), we can conclude that

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{der} |e| \odot \text{der} |e'| : \tau[\theta_a] \].

\[ \square \]

**Theorem 15** (Completeness of algorithmic typechecking in RelRef)

1. Assume that \( \Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 : \tau \). Then, there exist \( e_1', e_2' \) such that \( \Delta; \vdash \Phi_a; \Gamma \vdash e_1' \odot e_2' \downarrow \tau \Rightarrow \Phi \) and \( \Delta; \Phi_a \models \Phi \) and \( |e_1'| = e_1 \) and \( |e_2'| = e_2 \).

*Proof.* By simultaneous induction on the given Core typing derivations.

Case: \[ \Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 : \Box \tau \]

\[ \Delta; \Phi_a; \Gamma \vdash \text{der} e_1 \sim \text{der} e_2 : \tau \]

By IH on the premise, \( \exists e_1', e_2' \) such that

a) \( \Delta; \vdash \Phi_a; \Gamma \vdash e_1' \odot e_2' \downarrow \tau, t \Rightarrow \Phi \)

b) \( \Delta; \Phi_a \models \Phi \)

c) \( |e_1'| = e_1 \) and \( |e_2'| = e_2 \)

Then, we can conclude by using a), b) and c) as follows:

\[ \Delta; \vdash \Phi_a; \Gamma \vdash \text{der} e_1' \odot \text{der} e_2' \downarrow \Box \tau \Rightarrow \Phi \]

\[ \Delta; \vdash \Phi_a; \Gamma \vdash \text{der} e_1' \odot \text{der} e_2' \downarrow \tau \Rightarrow \Phi \]

\[ \text{alg-r-der-\downarrow} \]

\[ 38 \]
Then, we can conclude as follows

By IH on the second premise, \( \exists \)

By IH on the first premise, \( \exists \)

Then, we can conclude as follows

1. By using a) and d)

2. By c), \(|(\text{der } e'_1 : \tau)| = \text{der } |e'_1|, \)

3. By b) and Lemma 8.

Case \( \Delta; \Phi_a; \Gamma \vdash e \rightsquigarrow e' : \tau \)

By IH on the first premise, \( \exists e'_1, e'_2 \) such that

a) \( \Delta; \Phi_a; \Gamma \vdash e'_1 \sqcup e'_2 \downarrow \tau \Rightarrow \Phi_1 \)

b) \( \Delta; \Phi_a \models \Phi_1 \)

c) \(|e'_1| = e\) and \(|e'_2| = e'\)

By IH on the second premise,

\( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi_2 \)

e) \( \Delta; \Phi_a \models \Phi_2 \).

Then, we can conclude as follows

1. By using a) and d)

2. By b), c), we can show that \( \Delta; \Phi_a \models \Phi_1 \land \Phi_2 \)

3. By c), \(|(e'_1 : \tau)| = e'_1\) and \((e'_2 : \tau) = e'_2\)

\( \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e'_1 : \tau_1 \Rightarrow \tau_2 \)

\( \Delta; \Phi_a; \Gamma \vdash e_2 \rightsquigarrow e'_2 : \tau_1 \)

Case \( \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e'_1 \mid e'_2 : \tau_1 \)

By IH on the first premise, \( \exists e_1, e'_1 \) such that

a) \( \Delta; \Phi_a; \Gamma \vdash \tau_1 \sqcup \tau'_1 \downarrow \tau_1 \Rightarrow \Phi_1 \)

b) \( \Delta; \Phi_a \models \Phi_1 \)

c) \(|\tau_1| = e_1\) and \(|\tau'_1| = e'_1\)

By IH on the second premise, \( \exists \tau_2, \tau'_2 \) such that

\( \Delta; \Phi_a; \Gamma \vdash \tau_2 \sqcup \tau'_2 \downarrow \tau_1 \Rightarrow \Phi_2 \)

e) \( \Delta; \Phi_a \models \Phi_2 \)

f) \(|\tau_2| = e_2\) and \(|\tau'_2| = e'_2\)

Then, we can conclude as follows

39
1.

\[
\Delta;::\Phi_a;\Gamma\vdash \varphi_1 \downarrow \varphi_2 \Rightarrow \Phi_1 \quad \text{alg-r-anno} \uparrow
\]

\[
\Delta;::\Phi_a;\Gamma\vdash (\varphi_1: \tau_1 \rightarrow \tau_2) \varphi_1 \uparrow \varphi_2 \Rightarrow \Phi_1
\]

\[
\Delta;::\Phi_a;\Gamma\vdash (\varphi_2: \tau_2 \rightarrow \tau_3) \varphi_2 \downarrow \varphi_3 \Rightarrow \Phi_2
\]

\[
\Delta;::\Phi_a;\Gamma\vdash E_1 \circ E_2 \Rightarrow \Phi_1 \land \Phi_2 \quad \text{c-r-app}
\]

\[
\Delta;::\Phi_a;\Gamma\vdash E_1 \downarrow \varphi_2 \Rightarrow \Phi_1 \land \Phi_2 \quad \text{alg-r} \downarrow
\]

where \( E_1 = (\varphi_1: \tau_1 \rightarrow \tau_2) \) \( \varphi_1 \) and \( E_2 = (\varphi_2: \tau_1 \rightarrow \tau_2) \) \( \varphi_2 \).

2. By using b) and c).

3. Using c) and f), \(|\varphi_1: \tau_1 \rightarrow \tau_2| = e_1 \) \( e_2 \) and \(|\varphi_2: \tau_1 \rightarrow \tau_2| = e'_1 \) \( e'_2 \).

Case i). \( \exists \iota: \tau_1 \Rightarrow \Phi_1 \)

By IH on the first premise, \( \exists \varphi_1, \varphi_1' \) such that

a) \( \Delta;::\Phi_a;\Gamma\vdash \varphi_1 \downarrow \varphi_1' \Rightarrow \exists \iota: \tau_1 \Rightarrow \Phi_1 \)

b) \( \Delta;::\Phi_a;\Gamma\vdash \varphi_1 \Rightarrow \Phi_1 \)

c) \( |\varphi_1| = e_1 \) \( |\varphi_1'| = e'_1 \)

By IH on the second premise, \( \exists \varphi_2, \varphi_2' \) such that

d) \( i::\Delta;::\Phi_a;\Gamma\vdash \varphi_2 \downarrow \varphi_2' \Rightarrow \Phi_2 \)

e) \( i::\Delta;::\Phi_a;\Gamma\vdash \varphi_2 \Rightarrow \Phi_2 \)

Then, we can conclude as follows

1.

\[
\Delta;::\Phi_a;\Gamma\vdash \varphi_1 \downarrow \varphi_1' \Rightarrow \exists \iota: \tau_1 \Rightarrow \Phi_1 \quad \text{alg-r-anno} \uparrow
\]

\[
i::\Delta;::\Phi_a;\Gamma\vdash E_1 \circ E_2 \Rightarrow \exists \iota: \tau_1 \Rightarrow \Phi_1
\]

\[
\Delta;::\Phi_a;\Gamma\vdash \varphi_2 \downarrow \varphi_2' \Rightarrow \Phi_2 \quad \Phi' = \Phi_1 \land \Phi_2'
\]

\[
\Delta;::\Phi_a;\Gamma\vdash E_1 \circ E_2 \Rightarrow \Phi_1 \land \Phi_2 \quad \text{alg-r-anno} \downarrow
\]

where \( E_1 = (\varphi_1: \exists \iota: \tau_1) \) and \( E_2 = (\varphi_2: \exists \iota: \tau_1) \).

2. By using b) and c).

3. Using c) and f), \(|\varphi_1: \exists \iota: \tau_1| = \) \( e_1 \) \( (x, i) \) in \( \varphi_2 \) \( = \) \( e_2 \) \( (x, i) \) in \( \varphi_1 \) \( = \) \( e'_1 \) \( (x, i) \) in \( \varphi_1' \) \( = \) \( e'_2 \).

\( \square \)
3 RelInf

3.1 Syntax of RelInf

Unary types  \( A ::= \) bool \( | \) int \( | A_1 \rightarrow A_2 \)

Relational types  \( \tau ::= \) bool \( r \) \( | \) int \( r \) \( | \tau_1 \rightarrow \tau_2 \) \( | U (A_1, A_2) \)

Expressions  \( e ::= x \mid n \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid \text{fix } f(x).e \mid e_1 \ e_2 \)

Value  \( v ::= n \mid \text{true} \mid \text{false} \mid \text{fix } f(x).e \)

Figure 22: Syntax of values and expressions in RelInf

Unary types  \( A ::= \) bool \( | \) int \( | A_1 \rightarrow A_2 \)

Relational types  \( \tau ::= \) bool \( r \) \( | \) int \( r \) \( | \tau_1 \rightarrow \tau_2 \) \( | U (A_1, A_2) \)

Expressions  \( e ::= x \mid n \mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e_1 \text{ else } e_2 \mid \text{fix } f(x).e \mid e_1 \ e_2 \mid \text{switch } e \)

Value  \( v ::= n \mid \text{true} \mid \text{false} \mid \text{fix } f(x).e \)

Figure 23: Syntax of values and expressions in RelInf Core

Unary types  \( A ::= \) bool \( | \) int \( | A_1 \rightarrow A_2 \)

Relational types  \( \tau ::= \) bool \( r \) \( | \) int \( r \) \( | \tau_1 \rightarrow \tau_2 \) \( | U (A_1, A_2) \)

Expressions  \( e ::= \ldots \mid \text{switch } e \mid (e : \tau) \mid (e : A) \)

Value  \( v ::= n \mid \text{true} \mid \text{false} \mid \text{fix } f(x).e \)

Figure 24: Syntax of values and expressions in BiRelInf
\[
\frac{\Omega(x) = A}{\Omega \vdash x : A} \quad \text{var} \quad b \in \{\text{true, false}\} \quad \text{bool} \quad \frac{\Omega \vdash b : \text{bool}}{\Omega \vdash b : \text{bool}}
\]

\[
\frac{\vdash^A A_1 \to A_2 \text{ wf}}{\vdash^A A_1, f : A_1 \to A_2, \Omega \vdash e : A_2}{\Omega \vdash \text{fix} \ f(x).e : A_1 \to A_2} \quad \text{fix}
\]

\[
\frac{\Omega \vdash \text{if} \ e \ \text{then} \ e_1 \ \text{else} \ e_2 : \tau}{\Omega \vdash \text{if} \ e \ \text{then} \ e_1 \ \text{else} \ e_2 : \tau} \quad \text{if}
\]

\[
\frac{\Omega \vdash e : A}{\Omega \vdash e : A \vdash^A A \subseteq A'} \quad \text{exec}
\]

Figure 25: RelInf unary typing rules

\[
\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \text{r-var} \quad b \in \{\text{true, false}\} \quad \text{r-bool} \quad \frac{\Gamma \vdash b : \text{bool}}{\Gamma \vdash b : \text{bool}_r} \quad \frac{\| \Gamma \|_1 \vdash e_1 : A_1}{\| \Gamma \|_1 \vdash e_1 : A_1} \quad \frac{\| \Gamma \|_2 \vdash e_2 : A_2}{\| \Gamma \|_2 \vdash e_2 : A_2} \quad \text{r-switch}
\]

\[
\frac{\Gamma \vdash e \sim e' : \text{bool}_r, \Gamma \vdash e_1 \sim e_1' : \tau, \Gamma \vdash e_2 \sim e_2' : \tau}{\Gamma \vdash \text{if} \ e \ \text{then} \ e_1 \ \text{else} \ e_2 \sim e'_1 \ \text{else} \ e'_2 : \tau} \quad \text{r-if}
\]

\[
\frac{\Delta; \Phi; x : \tau_1, f : \tau_1 \to \tau_2, \Gamma \vdash e_1 \sim e_2 : \tau_2}{\Delta; \Phi; \Gamma \vdash \text{fix} \ f(x).e_1 \sim \text{fix} \ f(x).e_2 : \tau_1 \to \tau_2} \quad \text{r-fix}
\]

\[
\frac{\Gamma \vdash e_1 \sim e_1' : \tau_1 \to \tau_2, \Gamma \vdash e_2 \sim e_2' : \tau_1}{\Gamma \vdash e_1 \sim e_2 \sim e_1' \sim e_2' : \tau_1} \quad \text{r-app}
\]

\[
\frac{\| \Gamma \|_1 \vdash e_1 : A \vdash^A A \subseteq A'}{\| \Gamma \|_1 \vdash \tau \subseteq \tau'} \quad \frac{\| \Gamma \|_2 \vdash e_2 : A \vdash^A A \subseteq A'}{\| \Gamma \|_2 \vdash \tau \subseteq \tau'} \quad \frac{\| \Gamma \| \vdash e_1 \sim e_2 : \tau}{\| \Gamma \| \vdash \tau \subseteq \tau'} \quad \text{r-\subseteq}
\]

Figure 26: RelInf binary typing rules

\[
\frac{\vdash^A A_1' \subseteq A_1}{\vdash^A A_1 \to A_2 \subseteq A'_1 \to A'_2} \quad \text{exec} \quad \frac{\vdash^A A_1 \subseteq A_2}{\vdash^A A_1 \subseteq A_3} \quad \text{u-refl} \quad \frac{\vdash^A A_1 \subseteq A_2}{\vdash^A A_1 \subseteq A_3} \quad \text{u-tran}
\]

Figure 27: RelInf unary subtyping rules

\[
\vdash U(A_1 \to A_2, A'_1 \to A'_2) \subseteq U(A_1, A'_1) \to U(A_2, A'_2) \quad \text{execdiff}
\]

\[
\frac{\vdash^A A_1 \subseteq A_3}{\vdash U(A_1, A_2) \subseteq U(A'_1, A'_2)} \quad \frac{\| \tau \| \subseteq U(|\tau_1|, |\tau_2|)}{\| \tau \| \subseteq U(|\tau_1|, \tau_2)} \quad \frac{\| \tau \| \subseteq U(|\tau_1|, \tau_2)}{\| \tau \| \subseteq U(|\tau_1|, |\tau_2|)}
\]

\[
\frac{\vdash \tau \subseteq \tau_1 \quad \vdash \tau_2 \subseteq \tau_2}{\vdash \tau \subseteq \tau_1 \to \tau_2 \to \tau_1 \to \tau_2} \quad \text{trans}
\]

Figure 28: RelInf binary subtyping rules

42
<table>
<thead>
<tr>
<th></th>
<th>$i \in {1,2}$</th>
<th>Binary type $\rightarrow$ Unary type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>\mathtt{int}</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>\mathtt{bool}</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>\tau_1 \rightarrow \tau_2</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>U(A_1,A_2)</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>\Gamma, x : \tau</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>\emptyset</td>
</tr>
</tbody>
</table>

Figure 29: RelInf refinement removal operation

$\vdash \mathtt{bool} \equiv \mathtt{bool}$ \hspace{2cm} $\vdash \mathtt{int} \equiv \mathtt{int}$ \hspace{2cm} $\vdash \tau \equiv \tau'$

$\vdash ^A A_1 \sqsubseteq A_1'$ \hspace{2cm} $\vdash ^A A_1' \sqsubseteq A_1$ \hspace{2cm} $\vdash ^A A_2 \sqsubseteq A_2'$ \hspace{2cm} $\vdash ^A A_2' \sqsubseteq A_2$

Figure 30: RelInf relational type equivalence rules

\[
\frac{x : A_1, f : A_1 \rightarrow A_2, \Omega \vdash e :^c A_2}{\Omega \vdash \mathtt{fix}\ f(x).e :^c A_1 \rightarrow A_2} \quad \frac{\Omega(x) = A}{\Omega \vdash x :^c A} \quad \frac{\Omega \vdash n :^c \mathtt{int}}{\Omega \vdash n :^c \mathtt{int}} \quad \frac{\Omega \vdash e :^c A}{\Omega \vdash e :^c A}
\]

$\frac{\Omega \vdash e \vdash e' :^c A' \vdash A \sqsubseteq A'}{\Omega \vdash e \vdash e' :^c A'}$

Figure 31: RelInf Core unary typing rules

\[
\frac{|\Gamma|_1 \vdash e_1 :^c A_1}{\Gamma \vdash \mathtt{switch} e_1 \vdash \mathtt{switch} e_2 :^c U(A_1,A_2)} \quad \frac{|\Gamma|_2 \vdash e_2 :^c A_2}{\Gamma \vdash e_2 :^c \tau_2}
\]

$\vdash \tau_1 \rightarrow \tau_2$ \hspace{2cm} $\vdash \tau_1, f : \tau_1 \rightarrow \tau_2, \Gamma \vdash e_1 \rightarrow e_2 :^c \tau_2$

$\Gamma \vdash \mathtt{fix}\ f(x).e_1 \vdash \mathtt{fix}\ f(x).e_2 :^c \tau_1 \rightarrow \tau_2$

Figure 32: RelInf Core binary typing rules

43
Figure 33: RelInf binary embedding rules

Figure 34: RelInf unary embedding rules

Figure 35: RelInf unary algorithmic typing rules
\[
\begin{array}{ll}
\Gamma(x) = \tau & \text{alg-r-var} \\
\Gamma \vdash x \odot x \uparrow \tau & \text{b} \in \{\text{true}, \text{false}\} \quad \text{alg-r-bool} \\
\end{array}
\]

\[
\begin{array}{ll}
\frac{\mid \Gamma \mid_1 \vdash e_1 \uparrow A_1 \quad \mid \Gamma \mid_2 \vdash e_2 \uparrow A_2}{\Gamma \vdash \text{switch } e_1 \odot \text{switch } e_2 \uparrow U(A_1, A_2)} & \text{alg-r-switch}\uparrow \\
\frac{\mid \Gamma \mid_1 \vdash e_1 \downarrow A_1 \quad \mid \Gamma \mid_2 \vdash e_2 \downarrow A_2}{\Gamma \vdash \text{switch } e_1 \odot \text{switch } e_2 \downarrow U(A_1, A_2)} & \text{alg-r-switch}\downarrow \\
\end{array}
\]

\[
\begin{array}{ll}
\frac{\Gamma \vdash e \odot e' \uparrow \text{bool} \quad \Gamma \vdash e_1 \odot e'_1 \downarrow \tau \quad \Gamma \vdash e_2 \odot e'_2 \downarrow \tau}{\Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 \downarrow \tau} & \text{alg-r-if} \\
\end{array}
\]

\[
\begin{array}{ll}
\frac{\Gamma, x : \tau_1, f : \tau_1 \rightarrow \tau_2 \vdash e_1 \odot e_2 \downarrow \tau_2}{\Gamma \vdash \text{fix } f(x).e_1 \odot \text{fix } f(x).e_2 \downarrow \tau_1 \rightarrow \tau_2} & \text{alg-r-fix} \\
\end{array}
\]

\[
\begin{array}{ll}
\frac{\Gamma \vdash e_1 \odot e'_1 \uparrow \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 \odot e'_2 \downarrow \tau_1}{\Gamma \vdash e_1 e_2 \odot e'_1 e'_2 \uparrow \tau_2} & \text{alg-r-app} \\
\end{array}
\]

\[
\begin{array}{ll}
\frac{\Gamma \vdash e \odot e' \downarrow \tau}{\Gamma \vdash (e : \tau) \odot (e : \tau) \uparrow \tau} & \text{alg-r-anno}\uparrow \\
\end{array}
\]

Figure 36: RelInf binary algorithmic typing rules

\[
\begin{array}{ll}
\models \text{bool} \equiv \text{bool} & \text{alg-bool} \\
\models \text{int}_r \equiv \text{int}_r & \text{alg-intr} \\
\mid A \subset A' \models^A \quad \mid A' \subset A \models^A & \text{alg-u} \\
\mid \tau_1' \equiv \tau_1 \quad \mid \tau_2 \equiv \tau_2' & \text{alg-}\rightarrow \\
\models \tau_1 \rightarrow \tau_2 \equiv \tau_1' \rightarrow \tau_2' \\
\end{array}
\]

Figure 37: RelInf algorithmic subtyping rules
3.2 RelInf Lemmas

Lemma 16 (Reflexivity of Algorithmic Binary Type Equivalence in RelInf)
\[ \vdash \tau \equiv \tau. \]

Proof. By induction on the binary type.

Case bool,
It is proved by algorithmic binary type equivalence rule alg-boolr.

Case \( \tau_1 \to \tau_2 \)
By IH on \( \vdash \tau_1 \vdash \tau_1 \equiv \tau_1. \)
By IH on \( \vdash \tau_2 \vdash \tau_2 \equiv \tau_2. \)
By the above statements and rule alg-\( \to \), we conclude
\[ \vdash \tau_1 \to \tau_2 \equiv \tau_1 \to \tau_2. \]

Case \( U (A_1, A_2) \)
TS: \( \vdash U (A_1, A_2) \equiv U (A_1, A_2). \)
By unary subtyping rule u-refl, \( \vdash^A A_1 \subseteq A_1 \) and \( \vdash^A A_2 \subseteq A_2. \)
By the above statements and rule alg-u, we conclude
\[ \vdash U (A_1, A_2) \equiv U (A_1, A_2). \]

Lemma 17 (Existence of coercions for relational subtyping in RelInf)
If \( \vdash \tau \subseteq \tau' \) then there exists \( \text{coerce}_{\tau, \tau'} \in \text{Core s.t. } \vdash \text{coerce}_{\tau, \tau'} \vdash \text{coerce}_{\tau, \tau'} : \tau \to \tau'. \)

Proof. Proof is by induction on the subtyping derivation. We denote the witness \( e \) of type \( \tau \to \tau' \)
also \( \text{coerce}_{\tau, \tau'} \) for clarity.

Case \[ \vdash \tau_1 \subseteq \tau_1 \hspace{1cm} \vdash \tau_2 \subseteq \tau_2 \]
By IH on \( \vdash \tau_1 \vdash \exists \text{coerce}_{\tau_1, \tau_1} \vdash \text{coerce}_{\tau_1, \tau_1} \vdash \tau_1' \to \tau_1 \)
By IH on \( \vdash \tau_2 \vdash \exists \text{coerce}_{\tau_2, \tau_2} \vdash \text{coerce}_{\tau_2, \tau_2} \vdash \tau_2' \to \tau'_2 \)

Then, using these two statements, we can construct the following derivation where \( e = \text{fix}(f(x). \text{coerce}_{\tau_2, \tau_2} (x (c(e)))) \to \vdash \text{coerce}_{\tau_2, \tau_2} \to \tau_1' \to \tau_2' \to \tau_2 \)

Case \[ \vdash \tau \subseteq U ([\tau_1], [\tau_2]) \]
Then, we can immediately construct the derivation using the rule c-switch.

\[ \vdash \text{fix}(f(x). \text{switch} x \vdash \text{fix}(f(x). \text{switch} x) : \tau \to U ([\tau_1], [\tau_2]) \]

Case \[ \vdash \tau_1 \subseteq \tau_2 \hspace{1cm} \vdash \tau_2 \subseteq \tau_3 \]
By IH on \( \vdash \tau_1 \vdash \exists \text{coerce}_{\tau_1, \tau_2} \vdash \text{coerce}_{\tau_1, \tau_2} \vdash \tau_2 \to \tau_2 \)
By IH on \( \vdash \tau_2 \vdash \exists \text{coerce}_{\tau_2, \tau_3} \vdash \text{coerce}_{\tau_2, \tau_3} \vdash \tau_2 \to \tau_2 \)

Then, using \( \vdash \tau_1 \) and \( \vdash \tau_2 \), we can construct the derivation simply by function composition

\[ \vdash \text{fix}(f(x). \text{coerce}_{\tau_2, \tau_3} (\text{coerce}_{\tau_1, \tau_2} x) \vdash \text{fix}(f(x). \text{coerce}_{\tau_2, \tau_3} (\text{coerce}_{\tau_1, \tau_2} x) : \tau_1 \to \tau_3 \]

46
Proof of Theorem 19.1:

\[
\begin{align*}
\text{Case} & \quad \vdash U (A_1 \to A_2, A'_1 \to A'_2) \subseteq U (A_1, A'_1) \to U (A_2, A'_2) \to \text{execdiff} \\
\text{Then, we can immediately construct the following derivation where } e = \text{fix } (x). \text{fix } (y). \text{switch } (x \; y) \text{ using the } \text{c-switch} \text{ and } \text{c-app} \text{ rules.} \\
\cdot \vdash e \sim e \neq (U (A_1 \to A_2, A'_1 \to A'_2)) \to U (A_1, A'_1) \to U (A_2, A'_2)
\end{align*}
\]

\[
\begin{align*}
\text{Case} & \quad \vdash^A A_1 \subseteq A'_1 \quad \vdash^A A_2 \subseteq A'_2 \\
\text{Then, we can immediately construct the following derivation where } e = \text{fix } (x). \text{switch } x \text{ using the } \text{c-switch} \text{ and } \text{c-u-\subseteq} \text{ rules.} \\
\cdot \vdash e \sim e \neq (U (A_1, A_2)) \to U (A'_1, A'_2)
\end{align*}
\]

\[\Box\]

Theorem 18 (Types are preserved by embedding in \text{Rellnf})

1. If \(\Omega \vdash e \rightarrow e^\star : A\), then \(\Omega \vdash e^\star : c\ A\) and \(\Omega \vdash e : A\).
2. If \(\Gamma \vdash e_1 \rightarrow e_2 \rightarrow e_1^\star \rightarrow e_2^\star : \tau\), then \(\Gamma \vdash e_1^\star \rightarrow e_2^\star : \tau\) and \(\Gamma \vdash e_1 \rightarrow e_2 : \tau\).

Proof. By simultaneous induction on the given derivations.

Proof of Theorem 18.1:

\[
\begin{align*}
\text{Case} & \quad \vdash^A A_1 \rightarrow A_2 \ \text{wf} \\
\Omega & \vdash \text{fix } f(x). e \rightarrow \text{fix } f(x). e^\star : A_1 \to A_2 \\
\text{By Theorem 18.1 on the premise, we get } x : A_1, f : A_1 \to A_2, \Omega \vdash e : c A_2. \text{ Then, by unary core rule} \\
\text{c-fix, we conclude:} & \quad \vdash \text{fix } f(x). e : c A_1 \to A_2 \\
\text{c-u-fix.}
\end{align*}
\]

Proof of Theorem 18.2:

\[
\begin{align*}
\text{Case} & \quad \Gamma \vdash e_1 \rightarrow e_2 \rightarrow \text{switch } e_1^\star \rightarrow \text{switch } e_2^\star : U (A_1, A_2) \\
\text{By Theorem 18.1 on } (*) & \text{, we get } \Delta ; \Phi ; \Omega \vdash e_1^\star : c A_1 \quad (**) . \\
\text{By Theorem 18.1 on } (\diamond) & \text{, we get } \Delta ; \Phi ; \Omega \vdash e_2^\star : c A_2 \quad (\diamond\diamond) . \\
\text{Then, we conclude as follows:} & \quad \Gamma \vdash \text{switch } e_1 \sim \text{switch } e_2 : U (A_1, A_2)
\end{align*}
\]

\[\Box\]

Theorem 19 (Completeness of embedding in \text{Rellnf})

1. If \(\Omega \vdash e : A\), then there exists an \(e^\star\) such that \(\Omega \vdash e \sim e^\star : A\).
2. If \(\Gamma \vdash e_1 \sim e_2 : \tau\), then there exist \(e_1^\star, e_2^\star\) such that \(\Gamma \vdash e_1 \sim e_2 \rightarrow e_1^\star \sim e_2^\star : \tau\).

Proof. By simultaneous induction on the given typing derivations.

Proof of Theorem 19.1:
Case \( \Omega \vdash e : A \) \((\ast)\) \quad \mid \vdash^A A \sqsubseteq A' \hspace{1em} \ast \quad \square \\
\Omega \vdash e : A'

By Theorem 19.1 on \((\ast)\), we get \( \exists e^* \) such that \( \Omega \vdash e \leadsto e^* : A \) \((\ast\ast)\).

By e-u-\sqsubseteq rule using \((\ast\ast)\), \((\ast)\), we conclude as follows
\[\Omega \vdash e \leadsto e^* : A\]

\[\Omega \vdash e \leadsto e^* : A'\]

**Proof of Theorem 19.2:**

**Case**
\[\frac{|\Gamma_1| \vdash e_1 : A_1 \quad (\ast) \quad |\Gamma_2| \vdash e_2 : A_2 \quad (\ast)\quad \text{r-switch}}{\Gamma \vdash e_1 \leadsto e_2 : U(A_1,A_2)}\]

By \text{IH1} on \((\ast)\), we get \( \exists e_1^* \) such that \( |\Gamma_1| \vdash e_1 \leadsto e_1^* : A_1 \) \((\ast\ast)\).

By \text{IH1} on \((\ast)\), we get \( \exists e_2^* \) such that \( |\Gamma_2| \vdash e_2 \leadsto e_2^* : A_2 \) \((\infty)\).

By e-switch embedding rule using \((\ast\ast)\) and \((\infty)\), we can conclude as follows:
\[\frac{|\Gamma_1| \vdash e_1 \leadsto e_1^* : A_1 \quad (\ast\ast) \quad |\Gamma_2| \vdash e_2 \leadsto e_2^* : A_2 \quad (\infty)}{\Gamma \vdash e_1 \leadsto e_2 \leadsto \text{switch} e_1^* \leadsto \text{switch} e_2^* : U(A_1,A_2)}\]

\[\text{e-switch.}\]

**Theorem 20 (Soundness of algorithmic typechecking in RelInf)**

1. Assume that \( \Omega \vdash e \downarrow A \), then, \( \Omega \vdash |e| : \leq A \).
2. Assume that \( \Omega \vdash e \uparrow A \), then, \( \Omega \vdash |e| : \leq A \).
3. Assume that \( \Gamma \vdash e \ominus e' \downarrow \tau \), then, \( \Gamma \vdash |e| \ominus |e'| : \leq \tau \).
4. Assume that \( \Gamma \vdash e \ominus e' \uparrow \tau \), then, \( \Gamma \vdash |e| \ominus |e'| : \leq \tau \).

**Proof.** By simultaneous induction on the given algorithmic typing derivations.

**Proof of Theorem 20.1:**

**Case**
\[\frac{\Omega, x : A_1, f : A_1 \rightarrow A_2 \vdash e_1 \downarrow A_2 \quad \text{alg-u-fix}}{\Omega \vdash \text{fix } f(x).e_1 \downarrow A_1 \rightarrow A_2}\]

By \text{IH1} on the premise, we get \( \Omega, x : A_1, f : A_1 \rightarrow A_2 \vdash |e_1| : \leq A_2 \) \((\ast)\).

By the unary core rule c-u-fix and \((\ast)\), we conclude \( \Omega \vdash \text{fix } f(x).|e_1| : \leq A_1 \rightarrow A_2 \).

**Proof of Theorem 20.2:**

**Case**
\[\frac{\Omega \vdash e \downarrow A \quad \text{alg-u-anno}\uparrow}{\Omega \vdash (e : A) \uparrow A}\]

By \text{IH1} on the premise, we get \( \Omega \vdash |e| : \leq A \) \((\ast)\).

Because \(|(e : A)| = |e|\), We conclude \( \Omega \vdash [(e : A)] : \leq A \).

48
Proof of Theorem 20.3:

Case  
\[
\frac{|\Gamma| \vdash e_1 \downarrow A_1 \quad |\Gamma| \vdash e_2 \downarrow A_2}{\Gamma \vdash \text{switch } e_1 \sim \text{switch } e_2 \downarrow U(A_1, A_2)} \text{ alg-r-switch}_\downarrow
\]

By IH1 on the first premise, we get \(|\Gamma|_1 \vdash |e_1| : c A_1 \quad (\ast)\).

By IH1 on the second premise, we get \(|\Gamma|_2 \vdash |e_2| : c A_2 \quad (\circ)\).

By relational core rule \text{c-switch} and (\ast) and (\circ), we conclude \(|\Gamma| \vdash \text{switch } e_1 \sim \text{switch } e_1|e_2| : c U(A_1, A_2)\).

Proof of Theorem 20.4:

Case  
\[
\frac{|\Gamma| \vdash e_1 \uparrow A_1 \quad |\Gamma| \vdash e_2 \uparrow A_2}{\Gamma \vdash \text{switch } e_1 \sim \text{switch } e_2 \uparrow U(A_1, A_2)} \text{ alg-r-switch}_\uparrow
\]

By IH2 on the first premise, we get \(|\Gamma|_1 \vdash |e_1| : c A_1 \quad (\ast)\).

By IH2 on the second premise, we get \(|\Gamma|_2 \vdash |e_2| : c A_2 \quad (\circ)\).

By relational core rule \text{c-switch} and (\ast) and (\circ), we conclude \(|\Gamma| \vdash \text{switch } e_1 \sim \text{switch } e_1|e_2| : c U(A_1, A_2)\).

\[\blacksquare\]

Theorem 21 (Completeness of algorithmic typechecking in RelInf)

1. Assume that \(\Omega \vdash e : c A\). Then, there exists \(e'\) such that \(\Omega \vdash e' \downarrow A\) and \(|e'| = e\).

2. Assume that \(\Gamma \vdash e_1 \sim e_2 : c \tau\). Then, there exist \(e_1', e_2'\) such that \(\Gamma \vdash e_1' \bowtie e_2' \downarrow \tau\) and \(|e_1'| = e_1\) and \(|e_2'| = e_2\).

Proof. By simultaneous induction on the given Core typing derivations.

Proof of Theorem 21.1:

Case  
\[
\frac{\Omega(x) = A \quad \text{c-u-var}}{\Omega \vdash x : c A} \text{ c-u-var}
\]

We can conclude as follows

\[
\frac{\Omega(x) = \tau \quad \text{alg-u-var}}{\Omega \vdash x \uparrow A} \quad | A \subseteq A \quad \frac{\Omega \vdash x \downarrow A \quad \text{alg-u-} \downarrow}{\Omega \vdash x \downarrow A}
\]

Case  
\[
\frac{\Omega \vdash \text{fix } f(x).e : c A_1 \rightarrow A_2 \quad \text{c-u-fix}}{\text{alg-u-fix}}
\]

By IH1 on the premise, we get exists \(e''\) s.t \(\Omega \vdash e'' \downarrow A_2 \quad (\ast)\).

By algorithmic unary rule \text{alg-u-fix} and (\ast), we conclude where \(e' = \text{fix } f(x).e''\) and \(|e'| = e''\).

\(|\Omega \vdash e' \downarrow A_1 \rightarrow A_2\). and \(|e'| = \text{fix } f(x).|e''|\).
Proof of Theorem 21.2:

\[
\begin{align*}
\Gamma \vdash e_1 \sim e'_1 : \tau_1 \rightarrow \tau_2 \\
\Gamma \vdash e_2 \sim e'_2 : \tau_1 \\
\text{Case} \quad \Gamma \vdash e_1 \sim e'_1, e_2 \sim e'_2 : \tau_2 \\
\text{c-r-app} \\
\end{align*}
\]

By IH on the first premise, \(\exists \bar{e}_1, \bar{e}'_1\) such that

a) \(\Gamma \vdash \bar{e}_1 \odot \bar{e}'_1 \downarrow \tau_1 \rightarrow \tau_2\)

b) \(|\bar{e}_1| = e_1\) and \(|\bar{e}'_1| = e'_1\)

By IH on the second premise, \(\exists \bar{e}_2, \bar{e}'_2\) such that

c) \(\Gamma \vdash \bar{e}_2 \odot \bar{e}'_2 \downarrow \tau_1\)

d) \(|\bar{e}_2| = e_2\) and \(|\bar{e}'_2| = e'_2\)

Then, we can conclude as follows

1. \[
\begin{align*}
\Gamma \vdash \bar{e}_1 \odot \bar{e}'_1 \downarrow \tau_1 \rightarrow \tau_2 & \quad \text{alg-r-anno-\uparrow} \quad \Gamma \vdash \bar{e}_2 \odot \bar{e}'_2 \downarrow \tau_1 \rightarrow \Phi_2 \\
\Gamma \vdash (\bar{e}_1 : \tau_1 \rightarrow \tau_2) \odot (\bar{e}'_1 : \tau_1 \rightarrow \tau_2) \uparrow \tau_1 \rightarrow \tau_2 & \quad \text{\underline{c-r-app}} \\
\Gamma \vdash E_1 \odot E_2 \uparrow \tau_2 & \quad \text{alg-r-\uparrow\downarrow} \\
\Gamma \vdash E_1 \odot E_2 \downarrow \tau_2
\end{align*}
\]

where \(E_1 = (\bar{e}_1 : \tau_1 \rightarrow \tau_2) \bar{e}_2\) and \(E_2 = (\bar{e}'_1 : \tau_1 \rightarrow \tau_2) \bar{e}'_2\).

2. By using b) and e).

3. Using c) and f), \(|(\bar{e}_1 : \tau_1 \rightarrow \tau_2) \bar{e}_2| = e_1 e_2\) and \(|(\bar{e}'_1 : \tau_1 \rightarrow \tau_2) \bar{e}'_2| = e'_1 e'_2\).
4 RelRefU

4.1 Syntax of RelRefU

Unary types

A ::= bool | int | A₁ → A₂ | list[n] A | ∀i:S.A | ∃i:S.A | C & A | C ⊃ A

Relational types

τ ::= bool | int | τ₁ → τ₂ | U(A₁, A₂) | list[n]τ | ∀i:S.τ | ∃i:S.τ | C & τ | C ⊃ τ

Expressions

e ::= x | n | true | false | if e then e₁ else e₂ | fix f(x).e | e₁ e₂ | nil | cons(e₁, e₂) |
case e of nil → e₁ | h :: tl → e₂ | Λ.e | e[] |
pack e | unpack e₁ as x in e₂ | let x = e₁ in e₂ |
let e₁ as x in e₂ | celim e

Value

v ::= n | true | false | fix f(x).e | nil | cons(v₁, v₂) | Λe | pack v

Figure 38: Syntax of values and expressions in RelRefU

Unary types

A ::= bool | int | A₁ → A₂ | list[n] A | ∀i:S.A | ∃i:S.A | C & A | C ⊃ A

Relational types

τ ::= bool | int | τ₁ → τ₂ | U(A₁, A₂) | list[n]τ | ∀i:S.τ | ∃i:S.τ | C & τ | C ⊃ τ

Expressions

e ::= x | true | false | if e then e₁ else e₂ | fix f(x).e | e₁ e₂ | switch e | NC e |
split(e₁, e₂) with C | contra e | der e | Λ.e | e[I] | pack e with I |
unpack e₁ as (x, i) in e₂ | consNC(e₁, e₂) | consC(e₁, e₂) | fixNC f(x).e |
let x = e₁ in e₂ | (case e of nil → e₁ |
(h :: NC tl → e₂ | h :: C tl → e₃ ))

Value

v ::= n | fix f(x).e | fixNC f(x).e | nil |
consNC(v₁, v₂) | consC(v₁, v₂) | Λe | pack v with I

Figure 39: Syntax of values and expressions in RelRefU Core
Unary types

\[ A ::= \text{bool} | \text{int} | A_1 \rightarrow A_2 | \text{list}[n] A | \forall i::S. A | \exists i::S. A | C \land A | C \lor A \]

Relational types

\[ \tau ::= \text{bool} | \text{int} | \tau_1 \rightarrow \tau_2 | U(A_1, A_2) | \text{list}[n] \tau | \forall i::S. \tau | \exists i::S. \tau \]

Expressions

\[ e ::= x | \text{true} | \text{false} | \text{if } e \text{ then } e_1 \text{ else } e_2 | \text{switch } e | \text{NC } e | \text{split } (e_1, e_2) \text{ with } C | \text{contra } e | \text{der } e | e[I] | \text{pack } e \text{ with } I | \text{unpack } e_1 \text{ as } (x, i) \text{ in } e_2 | \text{consNC}(e_1, e_2) | \text{consC}(e_1, e_2) | \text{fixNC } f(x).e | (e : \tau) | (e : A) | \text{let } x = e_1 \text{ in } e_2 | \left( \begin{array}{c} \text{case } e \text{ of } \text{nil} \rightarrow e_1 \\
\text{h :: NC } tl \rightarrow e_2 | \text{h :: C } tl \rightarrow e_3 \end{array} \right) \]

Value

\[ v ::= n | \text{fix } f(x).e | \text{fixNC } f(x).e | \text{nil} | \text{consNC}(v_1, v_2) | \text{consC}(v_1, v_2) | \text{Ai } e | \text{pack } v \text{ with } I \]

Figure 40: Syntax of values and expressions in BiRelRefU
\[
\begin{align*}
\Gamma(x) &= \tau & \text{r-var} \\
\Delta; \Phi_a; \Gamma \vdash x \sim x : \tau & \quad \text{r-bool} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \text{bool} & \quad \text{r-if} \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 : \tau & \quad \Delta; \Phi_a; \Gamma \vdash e_2 \sim e'_2 : \tau \\
\Delta; \Phi_a; \Gamma \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 \sim \text{if } e' \text{ then } e'_1 \text{ else } e'_2 : \tau & \\
\Delta; \Phi_a; \Gamma \vdash \text{fix } f(x) \sim \text{fix } f(x) : \tau_1 \rightarrow \tau_2 & \quad \text{r-fix} \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 : \tau \rightarrow \tau_2 & \quad \Delta; \Phi_a; \Gamma \vdash e_2 \sim e'_2 : \tau_1 \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 \cdot e_2 \sim e'_2 : \tau_2 & \quad \text{r-app} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e : \tau & \quad \forall x \in \text{dom}(\Gamma). \Delta; \Phi_a \vdash \Gamma(x) \subseteq \square \Gamma(x) \\
\Delta; \Phi_a; \Gamma \vdash e \sim e : \square \tau & \quad \text{r-nochange} \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 : \tau & \quad \Delta; \Phi_a \vdash \tau \subseteq \tau' \\
\Delta; \Phi_a; \Gamma \vdash e \sim e : \tau' & \quad \Delta; \Phi_a; \Gamma \vdash \tau \subseteq \tau & \quad \text{r-} \triangleleft \\
\Delta; \Phi_a; \Gamma \vdash \exists x : \tau \rightarrow \tau_2. e \in \square \tau_2 & \quad \forall x \in \text{dom}(\Gamma). \Delta; \Phi_a \vdash \Gamma(x) \subseteq \square \Gamma(x) & \quad \text{r-fixNC} \\
\Delta; \Phi_a; \Gamma \vdash \text{fix } f(x) \sim \text{fix } f(x) : \square (\tau_1 \rightarrow \tau_2) & \\
\Delta; \Phi_a; \Gamma \vdash \text{fix } (f(x) \cdot e : \square (\tau_1 \rightarrow \tau_2)) & \quad \text{r-NC} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau & \quad \forall i \in \text{FIV}(\Phi_a; \Gamma) \\
\Delta; \Phi_a; \Gamma \vdash \text{fix } f(x) \sim \text{fix } f(x) : \square (\tau_1 \rightarrow \tau_2) & \quad \text{r-iLam} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \forall i : S. \tau & \quad \Delta; \Phi_a; \Gamma \vdash e \sim e' : \forall i : S. \tau & \quad \text{r-iApp} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau \{I/i\} & \quad \Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau \{I/i\} & \quad \text{r-pack} \\
\Delta; \Phi_a; \Gamma \vdash \text{pack } e \sim \text{pack } e' : \exists i : S. \tau & \quad \Delta; \Phi_a; \Gamma \vdash \text{pack } e \sim \text{pack } e' : \exists i : S. \tau & \quad \text{r-unpack1} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau & \quad \Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau & \quad \text{r-c-impl} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : C \supset \tau & \quad \Delta; \Phi_a; \Gamma \vdash e \sim e' : C \supset \tau & \quad \text{r-c-implE} \\
\Delta; \Phi_a \land C; \Gamma \vdash e \sim e' : \tau & \quad \Delta; \Phi_a; \Gamma \vdash e \sim e' : C \supset \tau & \quad \text{r-c-implE} \\
\Delta; \Phi_a; \Gamma \vdash e \sim e' : C \supset \tau & \quad \Delta; \Phi_a; \Gamma \vdash \text{celim } e \sim \text{celim } e' : \tau & \quad \text{r-c-implE} \\
\Delta; \Phi_a; \Gamma \vdash e_1 : A_1 & \quad \Delta; \Phi_a; \Gamma \vdash e_2 : A_2 & \quad \text{r-switch} \\
\Gamma \vdash e_1 \sim e_2 : U(A_1, A_2) & \quad \Gamma \vdash e_1 \sim e_2 : U(A_1, A_2) & \quad \text{r-switch} \\
\Delta; \Phi_a; \Gamma \vdash e_1 \sim e'_1 : \tau_1 & \quad \Delta; \Phi_a; \Gamma \vdash e_2 \sim e'_2 : \tau_2 & \quad \text{r-switch} \\
\Delta; \Phi_a; \Gamma \vdash \text{let } x = e_1 \text{ in } e_2 \sim \text{let } x = e'_1 \text{ in } e'_2 : \tau_2 & \quad \text{r-let} \\
\end{align*}
\]

Figure 41: RelRefU relational typing rules (Part 1)
\[
\Delta; \Phi \vdash \tau \text{wf}
\]
\[
\Delta; \Phi; \Gamma \vdash \text{nil} \leadsto \text{nil} : \text{list}[0]^{\alpha} \tau
\]
\[
\Delta; \Phi; \Gamma \vdash e_1 \leadsto e'_1 : \tau 
\Delta; \Phi; \Gamma \vdash e_2 \leadsto e'_2 : \text{list}[n]^{\alpha} \tau 
\frac{\Delta; \Phi; \Gamma \vdash \text{cons}(e_1, e_2) \leadsto \text{cons}(e'_1, e'_2) : \text{list}[n+1]^{\alpha+1} \tau}{\text{r-cons1}}
\]
\[
\Delta; \Phi; \Gamma \vdash e_1 \leadsto e'_1 : \Box \tau 
\Delta; \Phi; \Gamma \vdash e_2 \leadsto e'_2 : \text{list}[n]^{\alpha} \tau 
\frac{\Delta; \Phi; \Gamma \vdash \text{cons}(e_1, e_2) \leadsto \text{cons}(e'_1, e'_2) : \text{list}[n+1]^{\alpha} \tau}{\text{r-cons2}}
\]
\[
\Delta; \Phi; \Gamma \vdash 
\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e'_1 : \tau' 
\Delta; \Phi_a; \Gamma \vdash e_2 \leadsto e'_2 : \tau' 
\frac{\Delta; \Phi_a; \Gamma \vdash \text{case } e \text{ of nil } \rightarrow e_1 | h :: tl \rightarrow e_2 \leadsto e'_1 | h :: tl \rightarrow e'_2 : \tau'}{\text{r-caseL}}
\]
\[
\Delta; \Phi_a \vdash C 
\Delta; \Phi_a \vdash e \leadsto e' : \tau
\frac{\Delta; \Phi_a \vdash e \leadsto e' : C \& \tau}{\text{r-c-andI}}
\]
\[
\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e'_1 : C \& \tau_1 
\Delta; \Phi_a \vdash \text{clet } e_1 \text{ as } x \text{ in } e_2 \leadsto \text{clet } e'_1 \text{ as } x \text{ in } e'_2 : \tau_2 
\frac{\Delta; \Phi_a; \Gamma \vdash \text{clet } e \text{ as } x \text{ in } e_2 : \tau}{\text{r-c-andE}}
\]
\[
\Delta; \Phi_a \vdash e_1 \leadsto e_2 : \tau 
\Delta; \Phi_a \vdash \neg C; \Gamma \vdash e_1 \leadsto e_2 : \tau
\frac{\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e_2 : \tau}{\text{r-split}}
\]
\[
\Delta; \Phi_a \vdash \bot 
\Delta; \Phi_a \vdash \Gamma \text{wf} 
\frac{\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e_2 : \tau}{\text{r-contra}}
\]

Figure 42: RelRefU relational typing rules (Part 2)
Figure 43: RelRefU relational subtyping rules
Figure 44: RelRefU unary subtyping rules

\[ \Delta, \Phi_a \models \tau_1 \equiv \tau_2 \] checks whether \( \tau_1 \) is equivalent to \( \tau_2 \)

\[ \Delta, \Phi_a \models \text{bool}, \equiv \text{bool} \quad \text{eq-bool} \]

\[ \Delta, \Phi_a \models \text{int}, \equiv \text{int} \quad \text{eq-int} \]

\[ \Delta, \Phi_a \models \tau_1 \equiv \tau_1' \quad \Delta, \Phi_a \models \tau_2 \equiv \tau_2' \quad \text{eq-\forall} \]

\[ \Delta, \Phi_a \models \tau \equiv \tau' \quad \text{eq-\exists} \]

\[ \Delta, \Phi_a \models \tau \equiv \tau' \quad \text{eq-B-\Box} \]

\[ \Delta, \Phi_a \models A_1 \subseteq A_1' \quad \Delta, \Phi_a \models A_2 \subseteq A_2' \quad \text{eq-U} \]

\[ \Delta, \Phi_a \models C \cap \Phi_a \models C \quad \Delta, \Phi_a \models C' \cap \Phi_a \models C' \quad \text{eq-c-impl} \]

\[ \Delta, \Phi_a \models C \quad \Delta, \Phi_a \models C' \quad \Delta, \Phi_a \models C' \cap \Phi_a \models C' \quad \text{eq-c-prod} \]

Figure 45: RelRefU Core binary type equivalence rules
\[
\begin{array}{ll}
\Delta; \Phi_a; \Omega \vdash n : n & \text{c-const} \\
\Delta; \Phi_a; \Omega \vdash x : A & \text{c-var} \\
\Delta; \Phi_a ; A_1 \rightarrow A_2 & \Delta; \Phi_a ; x : A_1, f : A_1 \rightarrow A_2, \Omega \vdash e : A_2 & \text{c-fix} \\
\Delta; \Phi_a ; \Omega \vdash e_1 : A_1 & \Delta; \Phi_a ; \Omega \vdash e_2 : A_1 & \text{c-app} \\
\Delta; \Phi_a ; \Omega \vdash e : \forall i : S. A & \Delta; \Phi_a ; \Omega \vdash A[i/i] & \text{c-iApp} \\
\Delta; \Phi_a ; \Omega \vdash e : A & \Delta; \Phi_a ; \Omega \vdash e : S.A & \text{c-iLam} \\
\Delta; \Phi_a ; \Omega \vdash e : A & \Delta; \Phi_a ; \Omega \vdash e : \text{list}[n] A & \text{c-cons} \\
\Delta; \Phi_a ; \Omega \vdash e : A & \Delta; \Phi_a ; \Omega \vdash e : C & A & \text{c-candI} \\
\Delta; \Phi_a ; \Omega \vdash e : n \rightarrow A & \Delta; \Phi_a ; \Omega \vdash e : C \rightarrow A & \text{c-candE} \\
\Delta; \Phi_a ; \Omega \vdash e : C \triangleright A & \Delta; \Phi_a ; \Omega \vdash e : C \triangleright A & \text{c-cimplI} \\
\Delta; \Phi_a ; \Omega \vdash e : A & \Delta; \Phi_a ; \Omega \vdash e : A & \text{c-cimplE} \\
\Delta; \Phi_a ; \Omega \vdash e : A & \Delta; \Phi_a ; \Omega \vdash e : A & \text{c-cimplE} \\
\end{array}
\]

Figure 46: RelRefU Core unary typing rules
\[
\frac{\Gamma(x) = \tau}{\Delta; \Phi_a; \Gamma \vdash x : x : \tau} \text{ c-r-var}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash \tau_1 \to \tau_2 \ \text{wf} \quad \Delta; \Phi_a; x : \tau_1, f : \tau_1 \to \tau_2, \Gamma \vdash e_1 \sim e_2 : \tau_2}{\Delta; \Phi_a; \Gamma \vdash \text{fix}(f)(e_1) \sim \text{fix}(f)(e_2) : \tau_1 \to \tau_2} \text{ c-r-fix}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash \tau_1 \to \tau_2 \ \text{wf} \quad \Delta; \Phi_a; x : \tau_1, f : \square(\tau_1 \to \tau_2), \square \Gamma \vdash e \sim e : \tau_2}{\Delta; \Phi_a; \square \Gamma \vdash \text{fix}_{NC} f(x).e \sim \text{fix}_{NC} f(x).e : \square(\tau_1 \to \tau_2)} \text{ c-r-fixNC}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_1' : \tau \quad \Delta; \Phi_a; \Gamma \vdash e_2 \sim e_2' : \text{list}[\alpha] \quad \Delta; \Phi_a; \Gamma \vdash \text{cons}_{\text{C}}(e_1, e_2) \sim \text{cons}_{\text{C}}(e_1', e_2') : \text{list}[\alpha + 1] \tau}{\Delta; \Phi_a; \Gamma \vdash \text{cons}_{\text{NC}}(e_1, e_2) \sim \text{cons}_{\text{NC}}(e_1', e_2') : \text{list}[\alpha + 1] \tau} \text{ c-r-cons1}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_1' : \square \tau \quad \Delta; \Phi_a; \Gamma \vdash e_2 \sim e_2' : \text{list}[\alpha] \quad \Delta; \Phi_a; \Gamma \vdash \text{cons}_{\text{C}}(e_1, e_2) \sim \text{cons}_{\text{C}}(e_1', e_2') : \text{list}[\alpha + 1] \tau}{\Delta; \Phi_a; \Gamma \vdash \text{cons}_{\text{NC}}(e_1, e_2) \sim \text{cons}_{\text{NC}}(e_1', e_2') : \text{list}[\alpha + 1] \tau} \text{ c-r-cons2}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \sim e' : \text{list}[\alpha] \tau \quad \Delta; \Phi_a; \Gamma \vdash e \sim e' : \text{list}[\alpha] \tau}{\Delta; \Phi_a; \Gamma \vdash e \equiv e' : \tau'} \text{ c-r-eq}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \sim e' : \text{list}[\alpha] \tau \quad i, \beta; \Delta; \Phi_a; \Gamma \vdash n = i + 1 \land \alpha = \beta + 1 \quad i' \Gamma \vdash \text{list}[\beta] \tau, \Gamma \vdash e_1 \sim e_1', \tau'}{\Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau'} \text{ c-r-caseL}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \sim e' : \square \tau \quad \Delta; \Phi_a; \square \Gamma \vdash e \sim e : \tau}{\Delta; \Phi_a; \Gamma \vdash e \equiv e' : \tau} \text{ c-nochange}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \equiv e' : \tau \quad \Delta; \Phi_a; \Gamma \vdash e \equiv e' : \tau'}{\Delta; \Phi_a; \Gamma \vdash e \equiv e' : \tau} \text{ c-r-eq}
\]

\[
\frac{\Delta; \Phi_a \land C; \Gamma \vdash e_1 \sim e_2 : \tau \quad \Delta; \Phi_a \land C; \Gamma \vdash e_1' \sim e_2' : \tau}{\Delta; \Phi_a; \Gamma \vdash \text{split}(e_1, e_1') \text{ with } C \sim \text{split}(e_2, e_2') \text{ with } C : \tau} \text{ c-r-split}
\]

\[
\frac{\Delta; \Phi_a = \bot \quad \Delta; \Phi_a; \Gamma \vdash \text{contra} e_1 \sim \text{contra} e_2 : \tau}{\Delta; \Phi_a; \Gamma \vdash \text{contra} e_2 : \tau} \text{ c-r-contra}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \sim e' : \forall i : S. \tau \quad \Delta \vdash I : S}{\Delta; \Phi_a; \Gamma \vdash \text{iApp}(I, i, e) : \tau} \text{ c-r-iApp}
\]

\[
\frac{i : S, \Delta; \Phi_a; \Gamma \vdash e \sim e' : \tau \quad i \not\in \text{FIV}(\Phi_a; \Gamma)}{\Delta; \Phi_a; \Gamma \vdash \text{iLam}(I, e) : \forall i : S. \tau} \text{ c-r-iLam}
\]

Figure 47: RelRefU Core binary typing rules (Part 1)
\[
\frac{\Delta; \Phi \cup C; \Gamma \vdash e \sim e' : \tau}{\Delta; \Phi \vdash e \sim e' : \tau} \quad \text{c-r-cimpI}
\]

\[
\frac{\Delta; \Phi \vdash e \sim e' : \tau}{\Delta; \Phi \vdash \text{celim } e \sim \text{celim } e' : \tau} \quad \text{c-r-cimLE}
\]

\[
\frac{\Delta; \Phi \vdash e \sim e' : \tau}{\Delta; \Phi; \Gamma \vdash e \sim e' : C \cup \tau} \quad \text{c-r-candI}
\]

\[
\frac{\Delta; \Phi \vdash e \sim e' : \tau}{\Delta; \Phi; \Gamma \vdash \text{cel } e \sim \text{cel } e' : \tau} \quad \text{c-r-candE}
\]

\[
\frac{\Delta; \Phi \vdash e \sim e' : C \& \tau}{\Delta; \Phi; \Gamma \vdash e \sim e' : C \& \tau} \quad \text{c-r-capp}
\]

\[
\frac{\Delta; \Phi; \Gamma \vdash e \sim e' : C \& \tau}{\Delta; \Phi; \Gamma \vdash \text{let } x = e_1 \text{ in } e_2 \sim \text{let } x = e'_1 \text{ in } e'_2 : \tau} \quad \text{c-r-let}
\]

\[
\frac{\Delta; \Phi; \Gamma \vdash e \sim e' : \tau}{\Delta; \Phi; \Gamma \vdash \text{pack } e \text{ with } I \sim \text{pack } e' \text{ with } I : \exists i : S. \tau} \quad \text{c-r-pack}
\]

\[
\frac{\Delta; \Phi; \Gamma \vdash e \sim e' : \exists i : S. \tau}{\Delta; \Phi; \Gamma \vdash e \sim e' : \exists i : S. \tau} \quad \text{c-r-unpack1}
\]

\[
\frac{\Delta; \Phi; \Gamma \vdash e \sim e' : \exists i : S. \tau}{\Delta; \Phi; \Gamma \vdash e \sim e' : \exists i : S. \tau} \quad \text{c-switch}
\]

Figure 48: RelRefU Core binary typing rules (Part 2)
\[
\begin{align*}
\Delta; \Phi_0 \vdash^A A_1 \rightarrow A_2 \quad \Delta; \Phi_0; x : A_1, f : A_1 \rightarrow A_2, \Omega \vdash e \leadsto e^* : A_2 \\
\Delta; \Phi_0; \Omega \vdash \text{fix } f(x). e \leadsto \text{fix } f(x). e^* : A_1 \rightarrow A_2 \\
\Omega(x) = A \\
\Delta; \Phi_0; \Omega \vdash x \leadsto x : A \\
\Delta; \Phi_0; \Omega \vdash e_1 \leadsto e_1^* : A_1 \rightarrow A_2 \\
\Delta; \Phi_0; \Omega \vdash e_2 \leadsto e_2^* : A_1 \\
\Delta; \Phi_0; \Omega \vdash e_1 \leadsto e_1^* \quad \Delta; \Phi_0; \Omega \vdash e_2 \leadsto e_2^* : A_2 \\
\Delta; \Phi_0; \Omega \vdash A \quad \Delta; \Phi_0; \Omega \vdash \text{nil} \leadsto \text{nil} : \text{list}[0] A \\
\Delta; \Phi_0; \Omega \vdash \text{cons}(e_1, e_2) \leadsto \text{cons}_C(e_1, e_2) : \text{list}[n + 1] A \\
\Delta; \Phi_0 \land n = 0; \Omega \vdash e_1 \leadsto e_1^* : A' \\
i; \Delta; \Phi_0 \land n = i + 1; h : A, tl : \text{list}[i] A, \Omega \vdash e_2 \leadsto e_2^* : A' \\
\Delta; \Phi_0; \Omega \vdash e \leadsto e^* : A \\
\Delta; \Phi_0; \Omega \vdash \Lambda. e \leadsto \Lambda. i. e^* : \forall i :: S. A \\
\Delta; \Phi_0; \Omega \vdash e \leadsto e^* : \forall i :: S. A \\
\Delta; \Phi_0; \Omega \vdash e[I/i] \leadsto e^*[I] : A[I/i] \\
\Delta; \Phi_0; \Omega \vdash e \leadsto e^* : A[I/i] \\
\Delta; \Phi_0; \Omega \vdash e \leadsto e^* : A[I/i] \\
\Delta; \Phi_0; \Omega \vdash \text{pack } e \leadsto \text{pack } e^* \text{ with } I : \forall i :: S. A \\
\Delta; \Phi_0; \Omega \vdash \text{unpack } e_1 \text{ as } x \text{ in } e_2 \leadsto \text{unpack } e_1 \text{ as } (x, i) \text{ in } e_2 : \text{unpack } e_1^* \text{ as } (x, i) \text{ in } e_2^* : A_2 \\
\Delta; \Phi_0; \Omega \vdash x : A_1, \Omega \vdash e_1 \leadsto e_1^* : A_1 \\
\Delta; \Phi_0; \Omega \vdash x : A_1, \Omega \vdash e_2 \leadsto e_2^* : A_2 \\
\Delta; \Phi_0; \Omega \vdash x : A_1, \Omega \vdash e \leadsto e^* : A \\
\Delta; \Phi_0; \Omega \vdash C \land C ; \Omega \vdash e \leadsto e^* : C \land C \\
\Delta; \Phi_0; \Omega \vdash \text{clet } e_1 \text{ as } x \text{ in } e_2 \leadsto \text{clet } e_1^* \text{ as } x \text{ in } e_2^* : A_2 \\
\Delta; \Phi_0 \land \Phi_1 : C \land C ; \Omega \vdash e \leadsto e^* : C \land C \\
\Delta; \Phi_0 \land \Omega \vdash \text{celim } e \leadsto \text{celim } e^* : A \\
\Delta; \Phi_0 \vdash C \\
\Delta; \Phi_0; \Omega \vdash C \land C \land C ; \Omega \vdash e \leadsto e^* : C \land C \\
\end{align*}
\]

Figure 49: ReRefU unary embedding typing rules
\[ \Gamma(x) = \tau \]
\[ \Delta; \Phi_a; \Gamma \vdash x \sim x \sim x : \tau \quad \text{e-r-var} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{fix}(x).e_1 \sim \text{fix}(x).e_2 \sim e_1^* \sim e_2^* : \tau_2 \quad \text{e-r-fix} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{fix}(x).e_1 \sim \text{fix}(x).e_2 \sim e_1^* \sim \text{fix}(x).e_2^* : \tau_1 \rightarrow \tau_2 \quad \text{e-r-fixNC} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{fix}(x).e_1 \sim \text{fix}(x).e_2 \sim e_1^* \sim e_2^* : \tau \quad \text{e-r-impl} \]
\[ \Delta; \Phi_a; \Gamma \vdash \text{celim} e \sim \text{celim} e' \sim e^* \sim e'^* : C \supset \tau \quad \text{e-r-and} \]
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 \sim e_3 \sim e_4 \sim e_5 \sim e_6 \sim e_7 \sim e_8 \sim e_9 \sim e_{10} : \tau \quad \text{e-r-implE} \]
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 \sim e_3 \sim e_4 \sim e_5 \sim e_6 \sim e_7 \sim e_8 \sim e_9 \sim e_{10} : \tau \quad \text{e-r-andE} \]

Figure 50: RelRefU relational embedding rules (Part 1)
\[
\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e'_1 \leadsto e_1^* \leadsto e_1'^*: \tau \\
\Delta; \Phi_a; \Gamma \vdash e_2 \leadsto e'_2 \leadsto e_2^* \leadsto e_2'^*: \text{list}[n]^\alpha \tau
\]
\[
\Delta; \Phi_a; \Gamma \vdash \text{cons}(e_1, e_2) \leadsto \text{cons}(e'_1, e'_2) \leadsto \text{cons}_G(e_1', e_2'): \text{list}[n+1]^{\alpha+1} \tau
\]
\[
\Delta; \Phi_a; \Gamma \vdash e \leadsto e \leadsto e^* \leadsto e'^*: \tau \\
\forall x_i \in \text{dom}(\Gamma). \ e_i = \text{coerce}_{\Gamma(x_i) \sqcup \Gamma(x_i)} \quad \text{e-r-nochange}
\]
\[
\Delta; \Phi_a; \Gamma, \Gamma' \vdash e \leadsto e \leadsto \text{let } y_i = e_i; x_i \text{ in } \text{nc } e'[y_i/x_i] \leadsto \text{let } y_i = e_i; x_i \text{ in } \text{nc } e'^*[y_i/x_i] ; \Box \tau
\]
\[
\Delta; \Phi_a; \Gamma \vdash e \leadsto e' \leadsto e^* \leadsto e'^*: \text{list}[n]^\alpha \tau \\
i, \Delta; \Phi_a \land n = i + 1; h : \Box \tau, tl : \text{list}[i]^\alpha \tau, \Gamma \vdash e_2 \leadsto e_2' \leadsto e_2^* \leadsto e_2'^*: \tau'
\]
\[
i, \beta, \Delta; \Phi_a \land n = i + 1 \land \alpha = \beta + 1; h : \tau, tl : \text{list}[i]^\beta \tau, \Gamma \vdash e_2 \leadsto e_2' \leadsto e_2^* \leadsto e_2'^*: \tau'
\]
\[
\Delta; \Phi_a; \Gamma \vdash \text{case } e \text{ of nil } \rightarrow e_1 \quad \text{case } e \text{ of nil } \rightarrow e_1' \quad \text{case } e^* \text{ of nil } \rightarrow e_1^* \quad \text{case } e'' \text{ of nil } \rightarrow e_1''^*
\]
\[
i : S, \Delta; \Phi_a; \Gamma \vdash e \leadsto e' \leadsto e^* \leadsto e''^*: \tau \\
\ i \not\in \text{FIV}(\Phi_a; \Gamma) \quad \Delta; \Phi_a \vdash \Lambda e \leadsto \Lambda e^* \leadsto \Lambda e''^*: \forall i : S. \tau \quad \text{e-r-iLam}
\]
\[
\Delta; \Phi_a; \Gamma \vdash e \leadsto e^* \leadsto e''^*: \forall i : S. \tau \\
\ \Delta \vdash I : S \\
\ \Delta; \Phi_a; \Gamma \vdash \text{let } I = e^* \leadsto e''^*[I] \leadsto e''^*[I] : \tau\{I/I\}
\]
\[
\Delta; \Phi_a; \Gamma \vdash e \leadsto e' \leadsto e^* \leadsto e''^*: \tau \quad \Delta; \Phi_a \vdash \tau \subseteq \tau' \\
\ e' = \text{coerce}_{\tau,\tau'} \quad \text{e-r-iApp}
\]
\[
\Delta; \Phi_a; \Gamma \vdash e \leadsto e' \leadsto e^* \leadsto e''^*: \tau\{I/I\} \\
\ \Delta \vdash I : S \quad \Delta; \Phi_a; \Gamma \vdash \text{pack } e \leadsto \text{pack } e' \leadsto \text{pack } e''^* \text{ with } I \leadsto \text{pack } e''^* \text{ with } I ; \exists i : S. \tau
\]
\[
\Delta; \Phi_a; \Gamma \vdash \text{unpack } e_1 \text{ as } x \in e_2 \leadsto \text{unpack } e'_1 \text{ as } x \in e'_2 \leadsto e_1^* \leadsto e_1'^*: \tau_2
\]
\[
i \not\in \text{FIV}(\Phi_a; \Gamma, \tau_2, t_2) \\
\ e_1^* = \text{unpack } e_1' \text{ as } (x, i) \text{ in } e_1'' \\
\ e_1'^* = \text{unpack } e_1'' \text{ as } (x, i) \text{ in } e_1''
\]
\[
\Delta; \Phi_a; \Gamma \vdash \text{unpack } e_1 \text{ as } x \in e_2 \leadsto \text{unpack } e'_1 \text{ as } x \in e'_2 \leadsto e_1^* \leadsto e_1'^*: \tau_2
\]
\[
\Delta; C \land \Phi_a; \Gamma \vdash e_1 \leadsto e_2 \leadsto e_1^* \leadsto e_2^*: \tau \\
\ \Delta; \vdash C \land \Phi_a; \Gamma \vdash e_1 \leadsto e_2 \leadsto e_1^* \leadsto e_2^*: \tau
\]
\[
\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e_2 \leadsto \text{split } (e_1', e_1''^*) \text{ with } C \leadsto \text{split } (e_2', e_2''^*) \text{ with } C : \tau \\
\ \Delta \vdash C \text{ wf}
\]
\[
\Delta; \Phi_a; \Gamma \vdash e_1 \leadsto e_2 \leadsto \text{contra } e_1 \leadsto \text{contra } e_2 : \tau \quad \text{e-r-contra}
\]
\[
\Gamma_1 \vdash e_1 \leadsto e_1' : A_1 \\
\Gamma_2 \vdash e_2 \leadsto e_2' : A_2 \\
\Gamma \vdash e_1 \leadsto e_2 \leadsto \text{switch } e_1' \leadsto \text{switch } e_2' : U(A_1, A_2) \\
\text{e-switch}
\]

Figure 51: RelRefU relational embedding rules (Part 2)
\[
\begin{align*}
\Delta; \psi_a; \Phi_a &\models^A A'_1 \subseteq A_1 \Rightarrow \Phi_1 & \Delta; \psi_a; \Phi_a &\models^A A_2 \subseteq A'_2 \Rightarrow \Phi_2 & \text{alg-u-fun} \\
\Delta; \psi_a; \Phi_a &\models^A A_1 \rightarrow A_2 \subseteq A'_1 \rightarrow A'_2 \Rightarrow \Phi_1 \land \Phi_2 \\
\Delta; \psi_a; \Phi_a &\models^A A \subseteq A' \Rightarrow \Phi_1 & \Phi = n \equiv n' \land \Phi & \text{alg-u-list} \\
i; \Delta; \psi_a; \Phi_a &\models^{\forall} A \subseteq A' \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi_a &\models^{\forall} \forall i :: S.A \subseteq \forall i :: S.A' \Rightarrow \forall i :: S.\Phi_1 \land \Phi_2 & \text{alg-u-\forall} \\
i; \Delta; \psi_a; \Phi_a &\models^{\exists} A \subseteq A' \Rightarrow \Phi & i \notin FV(\Phi_a) \\
\Delta; \psi_a; \Phi_a &\models^{\exists} \exists i :: S.A \subseteq \exists i :: S.A' \Rightarrow \forall i :: S.\Phi & \text{alg-u-\exists} \\
\Delta; \psi_a; \Phi_a &\models^A A \subseteq A' \Rightarrow \Phi & \text{alg-u-impl} \\
\Delta; \psi_a; \Phi_a &\models^{C \supset A} \supset A' \subseteq C' \supset A' \Rightarrow (C' \rightarrow C) \land \Phi & \text{alg-u-prod} \\
\Delta; \psi_a; \Phi_a &\models^C C & A \subseteq A' & \Rightarrow (C \rightarrow C') \land \Phi & \text{alg-u-cprod}
\end{align*}
\]

Figure 52: RelRefU unary algorithmic subtyping rules
\[
\Delta; \psi_a; \Phi_a \models \int_r \equiv \int_r \Rightarrow \top \quad \text{alg-r-int}
\]
\[
\begin{align*}
\Delta; \psi_a; \Phi_a & \models \tau_1 \equiv \tau'_1 \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi_a & \models \tau_2 \equiv \tau'_2 \Rightarrow \Phi_1 \land \Phi_2
\end{align*}
\]
\[
\text{alg-r-fun}
\]
\[
\begin{align*}
\Delta; \psi_a; \Phi_a \models \sigma \equiv \sigma' \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a \models \text{list}[n] \equiv \text{list}[n'] \Rightarrow \Phi \land n \equiv n' \land \alpha \equiv \alpha'
\end{align*}
\]
\[
\text{alg-r-list}
\]
\[
\begin{align*}
i, \Delta; \psi_a; \Phi_a \models \sigma \equiv \sigma' \Rightarrow \Phi \\
i \not\in FV(\Phi_a) \\
\Delta; \psi_a; \Phi_a \models \exists i : S. \sigma \equiv \exists i : S. \sigma' \Rightarrow \forall i : S. \Phi
\end{align*}
\]
\[
\text{alg-r-}\exists
\]
\[
\begin{align*}
\Delta; \psi_a; \Phi_a \models \alpha \subseteq \alpha' \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi_a \models \alpha \subseteq \alpha' \Rightarrow \Phi_2
\end{align*}
\]
\[
\Delta; \psi_a; \Phi_a \models U (A_1, A_2) \equiv U (A'_1, A'_2) \Rightarrow \Phi_1 \land \Phi'_1 \land \Phi_2 \land \Phi'_2 \quad \text{U}
\]
\[
\text{c-impl}
\]
\[
\begin{align*}
\Delta; \psi_a; \Phi_a \models \sigma \equiv \sigma' \Rightarrow \Phi \\
\Delta; \psi_a; \Phi_a \models C \supset \tau \equiv C' \land \tau' \Rightarrow C \leftrightarrow C' \land \Phi
\end{align*}
\]
\[
\text{c-prod}
\]

Figure 53: RelRefU binary algorithmic equivalence rules
\[\Omega(x) = A\]
\[\Delta; \psi; \Phi_a; \Omega \vdash x \uparrow A \Rightarrow \alg-u-var\uparrow\]
\[\Delta; \psi_a; \Phi_a; f : A_1 \rightarrow A_2, x : A_1, \Omega \vdash e \downarrow A_2 \Rightarrow \Phi\]
\[\Delta; \psi_a; \Phi_a; \Omega \vdash \text{fix } f(x).e \downarrow A_1 \rightarrow A_2 \Rightarrow \Phi\]
\[\alg-u-fix\downarrow\]
\[\Delta; \psi_a; \Phi_a; \Omega \vdash e_1 \uparrow A_1 \rightarrow \Phi_1\]
\[\Delta; \psi_a; \Phi_a; \Omega \vdash e_2 \downarrow A_1 \Rightarrow \Phi_2\]
\[\alg-u-app\uparrow\]
\[\alg-u-Lam\downarrow\]
\[\alg-u-iApp\uparrow\]
\[\alg-u-pack\downarrow\]
\[\alg-u-unpack\downarrow\]
\[\alg-u-cons\downarrow\]
\[\alg-u-nil\downarrow\]
\[\alg-u-caseL\downarrow\]
\[\alg-u-let\downarrow\]
\[\alg-u-anno\uparrow\]
\[\alg-down\]

Figure 54: RelRefU unary algorithmic typing rules (Part 1)
\[
\Delta; \psi_{a}; \Phi \land C; \Omega \vdash e \downarrow A \Rightarrow \Phi \quad \text{alg-u-c-andI} \\
\Delta; \psi_{a}; \Phi; C; \Omega \vdash e \downarrow k \Rightarrow C \land (C \rightarrow \Phi) \quad \text{alg-u-c-impl} \\
\Delta; \psi_{a}; \Phi; C; \Omega \vdash e \downarrow C \supset A \Rightarrow C \rightarrow \Phi \quad \text{alg-u-c-impI} \\
\Delta; \psi_{a}; \Phi; C; \Omega \vdash \text{celim } e \uparrow A \Rightarrow C \land \Phi \quad \text{alg-u-c-implE} \\
\Delta; \psi_{a}; \Phi; C; \Omega \vdash e_{1} \uparrow C \land A \Rightarrow \forall x : A \Rightarrow e_{2} \downarrow A \Rightarrow \Phi \quad \text{alg-u-c-andE} \\
\Delta; k_{2}, t_{2}, \psi, \psi_{a}; \Phi \land C; x : A_{1}, \Omega \vdash e_{2} \downarrow A_{2} \Rightarrow \Phi_{2} \quad \Phi'_{2} = C \rightarrow \Phi_{2} \land k = (k_{1} + k_{2}) \land (t_{1} + t_{2}) \Rightarrow t \quad \text{alg-u-c-andE} \\
\Delta; \psi_{a}; \Phi; C; \Omega \vdash \text{celim } e_{1} \text{ as } x \text{ in } e_{2} \downarrow A_{2} \Rightarrow \exists (\psi). (\Phi_{1} \land \exists k_{2}, t_{2} :: R. \Phi'_{2}) \quad \text{alg-u-c-implE} \\
\Delta; \psi_{a}; \Phi; C \land \Omega \vdash e_{1} \downarrow A \Rightarrow \Phi_{1} \quad \Delta \vdash C \text{ wf} \\
\Delta; \psi_{a}; \Phi; C \land \Omega \vdash \text{split } (e_{1}, e_{2}) \text{ with } C \downarrow A \Rightarrow C \rightarrow \Phi_{1} \land \neg C \rightarrow \Phi_{2} \quad \text{alg-u-split} \\
\Delta; \psi_{a}; \Phi; \vdash \bot \quad \Delta \vdash \text{contra } e \downarrow A \Rightarrow \top \quad \text{alg-u-contra}
\]

Figure 55: RelRefU unary algorithmic typing rules (Part 2)
Figure 56: RelRefU binary algorithmic typing rules (Part 1)
\[ i :: S, \Delta; \psi_a; \Phi; \Gamma \vdash e \otimes e' \downarrow \tau \Rightarrow \Phi \]
\[ \Delta; \psi_a; \Phi; \Gamma \vdash \top_i e \otimes \Delta_i e' \downarrow \forall i :: S. \tau \Rightarrow (\forall i :: S. \Phi) \]
\[ \Delta; \psi_a; \Phi; \Gamma \vdash e \otimes e' \uparrow \forall i :: S. \tau' \Rightarrow \Phi \quad \Delta \vdash I :: S \]
\[ \Delta; \psi_a; \Phi; \Gamma \vdash e \otimes [I] \otimes e' [I] \uparrow \tau' \{i/i\} \Rightarrow \Phi \]
\[ \Delta; \psi_a; \Phi; \Gamma \vdash \text{pack} e \text{ with } I \otimes \text{pack } e' \text{ with } I \downarrow \exists i :: S. \tau \Rightarrow \Phi \]
\[ \Delta; \psi_a; \Phi; \Gamma \vdash e_1 \otimes e_1 \downarrow \tau = \Phi \quad \Delta; \psi_a; \Phi; \Gamma \vdash e_2 \otimes e_2 \downarrow \tau = \Phi \quad \Phi = (\Phi_1 \land \forall i :: S. \Phi_2) \]
\[ \Delta; \psi_a; \Phi; \Gamma \vdash \text{split } (e_1, e_2) \text{ with } C \otimes \text{split } (e_1', e_2') \text{ with } C \downarrow \tau = \Phi \land -C \Rightarrow \Phi_2 \]

Figure 57: RelRefU binary algorithmic typing rules (Part 2)
4.2 RelRefU Lemmas

Lemma 22 (Substitution of RelRefU unary Core)
Assume that

1. $\Delta; \Psi_a; \Omega \vdash e_1 :^e A$ (1).
2. $\Delta; \Psi_a; \Omega, x : A \vdash e'_1 :^e A'$ (2).

then $\Delta; \Psi_a; \Omega \vdash e'_1[e_1/x] :^e A'$.

Proof. By induction on the typing derivation of the second assumption (2).

$$\begin{align*}
\Omega(x) &= A \\
\Delta; \Phi_a; \Omega \vdash x :^e A &\text{ c-var}
\end{align*}$$

Subcase: $x = z$.
TS: $\Delta; \Psi_a; \Omega \vdash z[e_1/x] :^e A'$.
Because $x = z, z[e_1/x] = e_1$, STS: $\Delta; \Phi_a; \Gamma \vdash e_1 :^e A'$.
From ($\Omega, x : A)(z) = A'$ and $x = z$, we know $A = A'$.
By the above statements, STS: $\Delta; \Psi_a; \Omega \vdash e_1 :^e A'$, which is proved by the assumption.

Subcase: $x \neq z$.
we know: $\Omega(z) = A'$ (1).
TS: $\Delta; \Psi_a; \Omega \vdash z[e_1/x] :^e A'$.
Because $x \neq z, z[e_1/x] = z$, STS: $\Delta; \Psi_a; \Omega \vdash z :^e A'$.
By the core rule c-r-vars and (1), we construct the derivation :

$$\begin{align*}
\Omega(z) &= A' \\
\Delta; \Phi_a; \Omega \vdash z :^e A' &\text{ c-r-var}
\end{align*}$$

$$\begin{align*}
\Delta; \Phi_a \vdash A_1 \rightarrow A_2 &\text{ wf} \\
\Delta; \Phi_a; x : A_1, f : A_1 \rightarrow A_2, \Omega \vdash e :^e A_2 &\text{ c-fix}
\end{align*}$$

e'_1 = \text{fix}(f(z), e_1, A' = A_1 \rightarrow A_2).
TS: $\Delta; \Phi_a; \Omega \vdash \text{fix}(f(z), (e'_1[e_1/x]) :^e A'$.
By IH on (1) and (3), $\Delta; \Psi_a; \Omega \vdash e'_1[e_1/x] :^e A_2$ (4).
By core rule c-fix and (4), we can derive :

$\Delta; \Phi_a; \Omega \vdash \text{fix}(f(z), (e'_1[e_1/x])) :^e A'$.

Lemma 23 (Substitution of RelRefU Core)
Assume that

1. $\Delta; \Psi_a; \Gamma \vdash e_1 \sim e_2 :^e \tau$ (1).
2. $\Delta; \Psi_a; \Gamma, x : \tau \vdash e'_1 \sim e'_2 :^e \tau'$ (2).

then $\Delta; \Psi_a; \Gamma \vdash e'_1[e_1/x] \sim e'_2[e_2/x] :^e \tau'$.

Proof. By induction on the typing derivation of the second assumption (2).

$$\begin{align*}
(\Gamma, x : \tau)(z) &= \tau' \\
\Delta; \Phi_a; \Gamma, x : \tau \vdash z :^e \tau' &\text{ c-r-var}
\end{align*}$$

Subcase: $z = x$.
TS: $\Delta; \Phi_a; \Gamma \vdash z[e_1/x] :^e \tau' [e_2/x] :^e \tau'$.
Because $x = z, z[e_1/x] = e_1, z[e_2/x] = e_1$, STS: $\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 :^e \tau'$.
From ($\Gamma, x : \tau)(z) = \tau'$ and $x = z$, we know $\tau = \tau'$.
By the above statements, STS: $\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 :^e \tau$, which is proved by the assumption.
Subcase: $z \neq x$.
we know: $\Gamma(z) = \tau'$ (⋆).
TS: $\Delta; \Phi; \Gamma \vdash z[e_1/x] \leadsto z[e_2/x] : \tau'$. Because $x \neq z$, $z[e_1/x] = z$, $z[e_2/x] = z$, STS: $\Delta; \Phi; \Gamma \vdash z \leadsto z : \tau'$.
By the core rule c-r-vars and (⋆), we construct the derivation:

$\Gamma(z) = \tau'$

$\Delta; \Phi; \Gamma \vdash z \leadsto z : \tau'$  c-r-var

**Case**

$\Delta; \Phi; \Gamma; x : \tau \vdash e''_1 \leadsto e''_2 : \Box \tau'$ (3)

$\Delta; \Phi; \Gamma; x : \tau \vdash \text{der } e'_1 \leadsto \text{der } e'_2 : \tau$  c-der

$c = \text{der } e'_1$, $c = \text{der } e'_2$.

TS: $\Delta; \Phi; \Gamma \vdash \text{der } (e''_1[e_1/x]) \leadsto \text{der } (e''_2[e_2/x]) : \tau'$.

By IH on (1) and (3), $\Delta; \Psi; \Gamma \vdash e''_1[e_1/x] \leadsto e''_2[e_2/x] : \Box \tau'$ (4).

By core rule c-der and (4), we can derive:

$\Delta; \Phi; \Gamma; x : \tau \vdash e''_1[e_1/x] \leadsto e''_2[e_2/x] : \tau'$  c-der.

$\Delta; \Phi; \Gamma; x : \tau \vdash \text{der } (e''_1[e_1/x]) \leadsto (e''_2[e_2/x]) : \tau'$

**Case**

$\Delta; \Phi; \Gamma; x : \tau \vdash \text{fix } f(z) \vdash \text{fix } f(z) : \tau_1 \rightarrow \tau_2$  c-r-fix

$c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z)$.

TS: $\Delta; \Phi; \Gamma \vdash \text{fix } f(z).e''_1[e_1/x] \leadsto \text{fix } f(z).e''_2[e_2/x] : \tau'$.

By IH on (1) and (3), $\Delta; \Psi; \Gamma \vdash e''_1[e_1/x] \leadsto e''_2[e_2/x] : \tau_2$ (4).

By core rule c-r-fix and (4), we can derive:

$\Delta; \Phi; \Gamma \vdash \text{fix } f(z).e''_1[e_1/x] \leadsto \text{fix } f(z).e''_2[e_2/x] : \tau'$

**Case**

$\Delta; \Phi; \Gamma ; x : \tau \vdash \text{fix } \text{NC } f(z).e \leadsto \text{fix } \text{NC } f(z).e : \Box (\tau_1 \rightarrow \tau_2)$  c-r-fixNC

$c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z), c = \text{fix } f(z)$.

TS: $\Delta; \Phi; \Gamma \vdash \text{fix } f(z).e[e_1/x] \leadsto \text{fix } f(z).e[e_2/x] : \tau'$.

By IH on (1) and (3), $\Delta; \Psi; \Gamma \vdash e''_1[e_1/x] \leadsto e''_2[e_2/x] : \tau_2$ (4).

By core rule c-r-fixNC and (4), we can derive:

$\Delta; \Phi; \Gamma \vdash \text{fix } f(z).e[e_1/x] \leadsto \text{fix } f(z).e[e_2/x] : \tau'$

**Case**

$\Delta; \Phi; \Gamma ; x : \tau \vdash e_3 \leadsto e'_3 : \tau''$ (3)

$\Delta; \Phi; \Gamma ; x : \tau \vdash e_4 \leadsto e'_4 : \text{list}[n]^\alpha \tau''$ (4)

$c = \text{list}[n]^\alpha \tau''$.

$\Delta; \Phi; \Gamma ; x : \tau \vdash \text{cons } e_3[e_4] \vdash \text{cons } e'_3[e'_4] : \text{list}[n+1]^\alpha + 1 \tau''$  c-r-cons1

$\Delta; \Phi; \Gamma ; x : \tau \vdash e'_1 \leadsto e'_3, e'_2 \leadsto e'_4, e'_1 = \text{cons } e_3[e_4], e'_2 = \text{cons } e'_3[e'_4]$.

TS: $\Delta; \Phi; \Gamma \vdash \text{cons } e_3[e_4] \leadsto \text{cons } e'_3[e'_4] : \tau'$.

By IH on (1) and (3), $\Delta; \Phi; \Gamma \vdash e_3[e_4] \leadsto e'_3[e'_4] : \tau''$ (5).

By IH on (1) and (4), $\Delta; \Phi; \Gamma \vdash e_4[e_4] \leadsto e'_4[e'_4] : \tau''$ (6).

By core rule c-r-fixNC and (5),(6), we can derive:

$\Delta; \Phi; \Gamma \vdash \text{cons } e_3[e_4] \leadsto \text{cons } e'_3[e'_4] : \tau''$

**Case**

$\Delta; \Phi; \Gamma ; x : \tau \vdash e \leadsto e : \Box \tau''$ (3)

$c = \text{no-change}$

$\Delta; \Phi; \Gamma ; x : \tau \vdash \text{NC } e \leadsto \text{NC } e : \Box \tau''$

$c = \text{NC } e, c = \text{NC } e, c = \Box \tau''$.

TS: $\Delta; \Phi; \Gamma \vdash \text{NC } e[e_1/x] \leadsto \text{NC } e[e_2/x] : \tau'$.

By IH on (1) and (3), $\Delta; \Phi; \Gamma \vdash e[e_1/x] \leadsto e[e_2/x] : \tau''$ (4).

By core rule c-nochange and (4), we can derive:

$\Delta; \Phi; \Gamma \vdash \text{NC } e[e_1/x] \leadsto \text{NC } e[e_2/x] : \tau'$
Case $\Delta; \Phi_a; \Gamma \vdash e_1' : : A_1$ (3) $\Delta; \Phi_a; \Gamma \vdash e_2' : : A_2$ (4) $\text{c-switch}$

$e_1' = \text{switch } e_1''; e_2' = \text{switch } e_2''; \tau' = U(A_1, A_2).
\text{TS: } \Delta; \Phi_a; \Gamma \vdash e_1' \sim e_2' : : \tau'.

By Lemma 22 on (1) and (3), $\Delta; \Phi_a; \Gamma \vdash e_1'[e_1/x] : : A_1$ (5).
By Lemma 22 on (1) and (4), $\Delta; \Phi_a; \Gamma \vdash e_2'[e_2/x] : : A_2$ (6).
By $\text{c-switch}$ and (5),(6), we conclude that $\Delta; \Phi_a; \Gamma \vdash e_1' \sim e_2' : : \tau'$.

\[
\text{Lemma 24 (Reflexivity of Unary Algorithmic Subtyping in RelRefU)}
\]

$\Delta; \Phi_a \models^A A \sqsubseteq A \Rightarrow \Phi$ and $\Delta; \Phi_a \models \Phi$.

Proof. By induction on the unary type.

\[
\text{Lemma 25 (Reflexivity of Algorithmic Binary Type Equivalence in RelRefU)}
\]

$\forall \Delta, \psi_a, \Phi_a$, there exists $\Phi$ s.t $\Delta; \psi_a; \Phi_a \models \tau \equiv \tau \Rightarrow \Phi$ and $\Delta; \psi_a; \Phi_a \models \Phi$.

Proof. By induction on the binary type. Most cases are the same as RelRef's corresponding proof.

Case $U(A_1, A_2)$

By algorithmic equivalence rule $U$.

$\Delta; \psi_a; \Phi_a \models^A A_1 \sqsubseteq A_1 \Rightarrow \Phi_1$ (4) $\Delta; \psi_a; \Phi_a \models^A A_1 \sqsubseteq A_1 \Rightarrow \Phi_1$

$\Delta; \psi_a; \Phi_a \models^A A_2 \sqsubseteq A_2 \Rightarrow \Phi_2$ (4) $\Delta; \psi_a; \Phi_a \models^A A_2 \sqsubseteq A_2 \Rightarrow \Phi_2$

$\Delta; \psi; \Phi_a \models U(A_1, A_2) \equiv U(A_1, A_2) \Rightarrow \Phi_1 \wedge \Phi_1 \wedge \Phi_2 \wedge \Phi_2$

By Lemma 24 on (4) and (4), we conclude

$\Delta; \psi_a; \Phi_a \models U(A_1, A_2) \equiv U(A_1, A_2) \Rightarrow \Phi_1 \wedge \Phi_1 \wedge \Phi_2 \wedge \Phi_2$ and $\Delta; \psi_a; \Phi_a \models \Phi_1 \wedge \Phi_1 \wedge \Phi_2 \wedge \Phi_2$.

\[
\text{Lemma 26 (Transitivity of Unary Algorithmic Subtyping in RelRefU)}
\]

If $\Delta; \Phi_a \models^A A_1 \sqsubseteq A_2 \Rightarrow \Phi_1$ and $\Delta; \Phi_a \models^A A_2 \sqsubseteq A_3 \Rightarrow \Phi_2$ and $\Delta; \Phi_a \models \Phi_1 \wedge \Phi_2$, then $\Delta; \Phi_a \models^A A_1 \sqsubseteq A_3 \Rightarrow \Phi_3$.

Proof. By induction on the first two algorithmic subtyping derivation.

Case

$\Delta; \psi_a; \Phi_a \models^A A_1' \sqsubseteq A_1 \Rightarrow \Phi_1$ (1) $\Delta; \psi_a; \Phi_a \models^A A_2' \sqsubseteq A_2 \Rightarrow \Phi_2$ (2) $\text{alg-u-fun,}$ $\Delta; \psi_a; \Phi_a \models^A A_1'' \sqsubseteq A_1' \Rightarrow \Phi_1'$ (3)

$\Delta; \psi_a; \Phi_a \models^A A_1' \sqsubseteq A_1 \Rightarrow \Phi_1$

\text{u-fun}

By IH on (1) and (3), we get

$\models^A A_1'' \sqsubseteq A_1 \Rightarrow \Phi_1 \wedge \Phi_1'$ and $\Delta; \Phi_a \models \Phi_1 \wedge \Phi_1'$.

By IH on (2) and (4), we get

$\models^A A_2' \sqsubseteq A_2 \Rightarrow \Phi_2 \wedge \Phi_2'$ and $\Delta; \Phi_a \models \Phi_2 \wedge \Phi_2'$.

By rule $\text{alg-u-fun}$ and above statements, we conclude

$\Delta; \psi_a; \Phi_a \models^A A_1 \sqsubseteq A_2 \sqsubseteq A_1' \Rightarrow \Phi_1' \wedge \Phi_2' \wedge \Phi_1 \wedge \Phi_2$.

\[
\text{Theorem 27 (Soundness of the Algorithmic Unary Subtyping in RelRefU)}
\]

Assume that

1. $\Delta; \psi_a; \Phi_a \models^A A \sqsubseteq A' \Rightarrow \Phi$

2. $\text{FIV}(\Phi_a, A, A') \subseteq \Delta, \psi_a$
3. $\Delta; \Phi_a[\theta_a] \models \Phi[\theta_a]$ is provable s.t $\Delta \triangleright \theta_a : \psi_a$ is derivable.

Then $\Delta; \Phi_a[\theta_a] \vdash^A \Delta[\theta_a] \subseteq A'[\theta_a]$.

Proof. By induction on the algorithmic unary subtyping derivation.

\begin{center}
\begin{tabular}{ccc}
\text{Case} & Alg-U-Fun & \text{exec} \\
$\Delta; \psi_a; \Phi_a \models^A A' \subseteq A_1 \Rightarrow \Phi_1$ (1) & $\Delta; \psi_a; \Phi_a \models^A A_2 \subseteq A_2' \Rightarrow \Phi_2$ (2) & $\Delta; \psi_a; \Phi_a \models^A A_1 \rightarrow A_2 \subseteq A_1' \rightarrow A_2' \Rightarrow \Phi_1 \land \Phi_2$ \\
By IH on (1) and assumption 2.3, we get $\Delta; \Phi_a[\theta_a] \models^A A'_1[\theta_a] \subseteq A_1[\theta_a]$. & By IH on (2) and assumption 2.3, we get $\Delta; \Phi_a[\theta_a] \models^A A'_2[\theta_a] \subseteq A_2[\theta_a]$. & By subtyping rule $\rightarrow \text{exec}$, then we conclude $\Delta; \Phi_a[\theta_a] \models^A A_1 \rightarrow A_2[\theta_a] \subseteq A'_1 \rightarrow A'_2[\theta_a]$. \\
\end{tabular}
\end{center}

$\blacksquare$

Theorem 28 (Completeness of the Unary Algorithmic Subtyping)

Assume that $\Delta; \Phi_a \models^A A \subseteq A'$. Then $\exists \Phi$, such that $\Delta; \Phi_a \models^A A \subseteq A' \Rightarrow \Phi$ and $\Delta; \Phi_a \models \Phi$.

Proof. By induction on the unary subtyping derivation.

\begin{center}
\begin{tabular}{ccc}
\text{Case} & Alg-U-Fun & $\text{exec}$ \\
$\Delta; \Phi_a \models^A A'_1 \subseteq A_1$ (1) & $\Delta; \Phi_a \models^A A_2 \subseteq A_2'$ (2) & $\Delta; \Phi_a \models^A A_1 \rightarrow A_2 \subseteq A'_1 \rightarrow A'_2$ \\
By IH on (1), there exists $\Phi_1$ s.t $\Delta; \Phi_a \models^A A'_1 \subseteq A_1 \Rightarrow \Phi_1$ and $\Delta; \Phi_a \models \Phi_1$. & By IH on (2), there exists $\Phi_2$ s.t $\Delta; \Phi_a \models^A A_2 \subseteq A'_2 \Rightarrow \Phi_2$ and $\Delta; \Phi_a \models \Phi_2$. & By $\text{alg-u-fun}$ and the above statements, we conclude there exists $\Phi = \Phi_1 \land \Phi_2$, s.t $\Delta; \Phi_a \models^A A_1 \rightarrow A_2 \subseteq A'_1 \rightarrow A'_2 \Rightarrow \Phi$ and $\Delta; \Phi_a \models \Phi$. \\
\end{tabular}
\end{center}

$\blacksquare$

Theorem 29 (Soundness of the Algorithmic Binary Type Equality in RelRefU)

Assume that

1. $\Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi$
2. $\text{FIV}(\Phi_a, \tau, \tau') \subseteq \Delta, \psi_a$
3. $\Delta; \Phi_a[\theta_a] \models \Phi[\theta_a]$ is provable s.t $\Delta \triangleright \theta_a : \psi_a$ is derivable.

Then $\Delta; \Phi_a[\theta_a] \models \tau[\theta_a] \equiv \tau'[\theta_a]$.

Proof. By induction on the algorithmic binary type equivalence derivation. Same as RelRef.

\begin{center}
\begin{tabular}{ccc}
\text{Case} & Alg-U-Fun & U \\
$\Delta; \psi_a; \Phi_a \models^A A_1 \subseteq A'_1 \Rightarrow \Phi_1$ & $\Delta; \psi_a; \Phi_a \models^A A'_1 \subseteq A_1 \Rightarrow \Phi'_1$ & $\Delta; \psi_a; \Phi_a \models^A A_1 \models^A A'_2 \models^A A_2 \models^A A'_2 \Rightarrow \Phi_2 \models U(A_1, A_2)$ \\
From assumption 3, $\Delta; \Phi_a[\theta_a] \models (\Phi_1 \land \Phi'_1 \land \Phi_2 \land \Phi'_2)[\theta_a]$ & we know $\Delta; \Phi_a[\theta_a] \models \Phi_1[\theta_a]$ (a) & By IH on the first premise and (a), the second and (b), the third and (c) and the fourth and (d) and the rule $\text{eq-U}$, we can conclude that $\Delta; \Phi_a[\theta_a] \models U(A_1, A_2)[\theta_a] \equiv U(A'_1, A'_2)[\theta_a]$. \\
\end{tabular}
\end{center}

$\blacksquare$
Theorem 30 (Completeness of the Binary Algorithmic Type Equivalence in RelRefU)
Assume that $\Delta; \Phi_a \models \tau \equiv \tau'$. Then $\exists \Phi$, such that $\Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi$ and $\Delta; \Phi_a \models \Phi$.

Proof. By induction on the binary subtyping derivation. Same as RelRef.

Case $\Delta; \Phi_a \models^A A_1 \subseteq A_1' \quad \Delta; \Phi_a \models^A A_1' \subseteq A_1 \quad \Delta; \Phi_a \models^A A_2 \subseteq A_2' \quad \Delta; \Phi_a \models^A A_2' \subseteq A_2$ 

By IH on the first premise, we know $\Delta; \Phi_a \models^A A_1 \subseteq A_1' \Rightarrow \Phi_1$ and $\Delta; \Phi_a \models \Phi_1$.

Similarly by IH on the other premises respectively, we get

$\Delta; \Phi_a \models^A A_1' \subseteq A_1 \Rightarrow \Phi_1'$ and $\Delta; \Phi_a \models \Phi_1' \quad \Delta; \Phi_a \models^A A_2 \subseteq A_2' \Rightarrow \Phi_2$ and $\Delta; \Phi_a \models \Phi_2 \quad \Delta; \Phi_a \models^A A_2' \subseteq A_2 \Rightarrow \Phi_2'$ and $\Delta; \Phi_a \models \Phi_2'$.

By the above statements and the relational algorithmic type equivalence rule $U$, we conclude where $\Phi = \Phi_1 \land \Phi_1' \land \Phi_2 \land \Phi_2'$.

\[ \Delta; \Phi_a \models U(A_1, A_2) \equiv U(A_1', A_2') \Rightarrow \Phi \quad \text{and} \quad \Delta; \Phi_a \models \Phi \]

Lemma 31 (Existence of coercions for relational subtyping in RelRefU)
If $\Delta; \Phi_a \models \tau \subseteq \tau'$ then there exists $\text{coerce}_{\tau, \tau'} \in \text{Core}$ s.t. $\Delta; \Phi_a \vdash \text{coerce}_{\tau, \tau'} : \tau \rightarrow \tau'$.

Proof. Proof is by induction on the subtyping derivation. We denote the witness $e$ of type $\tau \rightarrow \tau'$ as $\text{coerce}_{\tau, \tau'}$ for clarity.

Case $\Delta; \Phi_a \models \Box U \left( \text{int}, \text{int} \right) \subseteq \text{int}$

Then, we can construct the derivation using the primitive function $\text{fix}: \Box U \left( \text{int}, \text{int} \right) \rightarrow \text{int}$

\[ \Delta; \Phi_a \vdash \text{fix}(f(x)).\text{box}_U x \sim \text{fix}(f(x)).\text{box}_U x : \Box U \left( \text{int}, \text{int} \right) \rightarrow \text{int} \]

Case $\Delta; \Phi_a \models U \left( \forall i : S.A, \forall i : S.A' \right) \subseteq \forall i : S.U \left( A, A' \right)$

Then, we can immediately construct the following derivation where $e = \text{fix}(f(x)).\text{switch}(x[i])$ using the $\text{c-switch}$ and $\text{c-iApp}$ rules.

$\Delta; \Phi_a \vdash e \sim e : \forall i : S.U \left( \forall i : S.A, \forall i : S.A' \right)$

Then, by unary core rule

\[ \forall i : S.U \left( A, A' \right) \]

Theorem 32 (Types are preserved by embedding in RelRefU)

1. If $\Delta; \Phi_a; \Omega \vdash e \sim e^* : A$, then $\Delta; \Phi_a; \Omega \vdash e^* \vdash A$ and $\Delta; \Phi_a; \Omega \vdash e : A$.

2. If $\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 \sim e_1' \sim e_2' : \tau$, then $\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 : \tau$ and $\Delta; \Phi_a; \Gamma \vdash e_1' \sim e_2' : \tau$.

Proof. By simultaneous induction on the given derivations.

Proof of Theorem 32.1:

Case $\Delta; \Phi_a \models^A A_1 \rightarrow A_2 \text{ u-f} \quad \Delta; \Phi_a; x : A_1, f : A_1 \rightarrow A_2, \Omega \vdash e \sim e^* : A_2$ 

By Theorem 18.1 on the premise, we get $x : A_1, f : A_1 \rightarrow A_2, \Omega \vdash e : A_2$. Then, by unary core rule $\text{c-fix}$, we conclude:

\[ \Delta; \Phi_a \models \text{fix}(f(x)).x : \forall i : S.A \rightarrow A_2 \]

73
Proof of Theorem 32.2:

\[ \Delta; \Phi_a; |\Gamma|_1 \vdash e_1 \sim c : A_1 \quad \Delta; \Phi_a; |\Gamma|_2 \vdash e_2 \sim c : A_2 \]

By Theorem 32.1 on (\footnote{\(\star\)}, we get \(\Delta; \Phi_a; \Omega \vdash c : A_1\) (\footnote{\(\star\ast\)}).

Then, we conclude as follows:

\[ \Delta; \Phi_a; |\Gamma|_1 \vdash e_1 \sim c : U(A_1, A_2) \]

By applying e-switch rule on (\footnote{\(\star\)}), we get \(\Delta; \Phi_a; \Omega \vdash c : U(A_1, A_2)\) (\footnote{\(\star\ast\)}).

Case

\[ \Delta; \Phi_a; |\Gamma|_1 \vdash e_1 \sim e_2 \sim switch e_1^* \sim switch e_2^* : U(A_1, A_2) \]

By Theorem 32.2 on (\footnote{\(\star\)}), we get \(\Delta; \Phi_a; \Omega \vdash e_1^* : A_1\) (\footnote{\(\star\ast\)}).

Then, we conclude as follows:

\[ \Delta; \Phi_a; |\Gamma|_2 \vdash e_2 \sim c : A_2 \]

By Lemma 31 using (\footnote{\(\star\)}), we know \(\Delta; \Phi_a; \Omega \vdash e_2^* : \tau \quad \\tau \equiv c\).

By applying c-r-app rule on (\footnote{\(\star\ast\)}), we get \(\Delta; \Phi_a; \Gamma \vdash e_1 \sim c \vdash \tau \quad \\tau' \equiv \tau \equiv c\).

By reflexivity of binary type equivalence, we know \(\Delta; \Phi_a \vdash \tau' \equiv \tau' \quad \\tau' \equiv \tau'\).

Then, we conclude as follows:

\[ \Delta; \Phi_a; \Gamma \vdash e' \quad e'_1 \sim c \vdash e' \quad e' \sim c \vdash \tau' \quad \\tau' \equiv \tau' \]

By Theorem 32.2 on (\footnote{\(\star\)}, the \(\Delta; \Phi_a; \Omega \vdash e' \sim e' : \tau \quad \\tau' \equiv \tau' \equiv \tau'\).

<table>
<thead>
<tr>
<th>Case</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta; \Phi_a;</td>
<td>\Gamma</td>
</tr>
<tr>
<td>c-r-(\sqsubseteq)</td>
<td>c-r-(\sqsubseteq)</td>
</tr>
</tbody>
</table>

Theorem 33 (Completeness of embedding in RelRefU)

1. If \(\Delta; \Phi_a; \Omega \vdash e : A\), then there exists an \(e^*\) such that \(\Delta; \Phi_a; \Omega \vdash e \sim e^* : A\).

2. If \(\Delta; \Phi_a; |\Gamma| \vdash e_1 \sim e_2 : \tau\), then there exist \(e_1, e_2\) such that \(\Delta; \Phi_a; \Gamma \vdash e_1 \sim e_2 \sim e_1^* \sim e_2^* : \tau\).

Proof. By simultaneous induction on the given RelCost derivations.

Proof of Theorem 33.1:

<table>
<thead>
<tr>
<th>Case</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Delta; \Phi_a; \Omega \vdash e : A) (\footnote{(\star)})</td>
<td>(\Delta; \Phi_a \vdash A \sqsubseteq A' \quad \Delta; \Phi_a \vdash A' \quad \Delta; \Phi_a \vdash A \sqsubseteq A' \quad \Delta; \Phi_a \vdash A' \quad \Delta; \Phi_a \vdash e \sim e' : A')</td>
</tr>
<tr>
<td>e-u-(\sqsubseteq)</td>
<td>e-u-(\sqsubseteq)</td>
</tr>
</tbody>
</table>

Proof of Theorem 33.2:
By Theorem 33.1 on $(\ast)$, we get $\exists e^*_1$ such that $\Delta; \Phi; \Omega \vdash_{k1}^{\ast} e_1 \sim e^*_1 : A_1$ $(\ast\ast)$.

By Theorem 33.1 on $(\circ)$, we get $\exists e^*_2$ such that $\Delta; \Phi; \Omega \vdash_{k2}^{\ast} e_2 \sim e^*_2 : A_2$ $(\circ\circ)$.

By $e$-switch embedding rule using $(\ast\ast)$ and $(\circ\circ)$, we can conclude as follows:

$$\Delta; \Phi; \Gamma \vdash e_1 \sim e_2 \sim \text{switch } e^*_1 \sim \text{switch } e^*_2 : U(A_1, A_2)$$

By IH2 on $(\circ)$, we get $\exists e^*$ such that $\Delta; \Phi; \Gamma \vdash e \sim e^* \sim e^* : \tau$ $(\dagger)$. By Lemma 11 on $(\circ)$, we get $\exists x_i = \text{coerce}_{(\Gamma(x_i))} \square (\Gamma(x_i))$ for all $x_i \in \text{dom}(\Gamma)$ $(\vdagger)$. By $e$-nochange embedding rule using $(\dagger)$ and $(\vdagger)$, we can conclude as follows:

$$\forall x_i \in \text{dom}(\Gamma), \ e_i = \text{coerce}_{(\Gamma(x_i))} \square (\Gamma(x_i)) \ (\dagger) \ \forall x_i \in \text{dom}(\Gamma), \ \Delta; \Phi; \Gamma \vdash e_i \sim \tau \ (\dagger) \ \forall x_i \in \text{dom}(\Gamma), \ e_i \sim \text{coerce}_{(\Gamma(x_i))} \square (\Gamma(x_i)) \ (\dagger)$$

By IH2 on $(\circ)$, we get $\exists e^*$ and $\exists e'^* s.t. \Delta; \Phi; \Gamma \vdash e \sim e^* \sim e'^* : \text{list}[n][\tau]$ $(\ast\ast)$. By IH2 on $(\dagger)$, we get $\exists e'_2$ and $\exists e^*_2 s.t.

$$i : S, \Delta; n \vdash i + 1 \land \Delta; h : \square \tau, t : \text{list}[i][\tau], \Gamma \vdash e_2 \sim e^*_2 \sim e'^*_2 : \tau' \ (\dagger\dagger)$

By IH2 on $(\circ\circ)$, we get $\exists e^*_2$ and $\exists e'^*_2 s.t.

$$i : S, \beta :: S, \Delta; n \vdash i + 1 \land \Delta; h : \square \tau, t : \text{list}[i][\tau], \Gamma \vdash e_2 \sim e^*_2 \sim e'^*_2 : \tau' \ (\dagger\dagger\dagger)$$

By e-caseL embedding rule using $(\ast\ast)$, $(\circ\circ)$, and $(\dagger\dagger\dagger)$, we can conclude as follows:

$$\Delta; \Phi; \Gamma \vdash \text{case } e \sim \text{of nil } \rightarrow e_1 \ | \ h :: t \rightarrow e_2 \sim \text{case } e' \sim \text{of nil } \rightarrow e'_1 \ | \ h :: t \rightarrow e'_2 : \tau'$$

$$i, \Delta; \Phi; \Gamma \vdash e \sim e^* \sim e'^* : \text{list}[n][\tau] \ (\ast\ast) \ i, \Delta; \Phi; \Gamma \vdash e_2 \sim e'^*_2 \sim e'^* : \tau' \ (\dagger\dagger\dagger)$$

$$\Delta; \Phi; \Gamma \vdash f : \square (\tau_1 \rightarrow \tau_2), \ \forall x \in \text{dom}(\Gamma), \Delta; \Phi; \Gamma \vdash \Gamma(x) \subseteq \square (\Gamma(x)) \ (\circ)$$

$$\Delta; \Phi; \Gamma \vdash \text{fix } f(x), e \sim \text{fix } f(x), e : \square (\tau_1 \rightarrow \tau_2)$$
By IH2 on (⋆), we get \( \exists e^* \) such that \( \Delta; \Phi_a; x : \tau_1, f : \Box (\tau_1 \to \tau_2), \Gamma \vdash e \circ e \leadsto e^* \circ e^* : \tau_2 \)  
By Lemma 31 on (⋆), we get \( \exists e_i = \text{coerce}(x_i), \Box (x_i) \) for all \( x_i \in \text{dom}(\Gamma) \) (∞).

By e-fixNC embedding rule using (⋆⋆) and (∞), we can conclude as follows:

\[
\Delta; \Phi_a; x : \tau_1, f : \Box (\tau_1 \to \tau_2), \Gamma \vdash e \circ e \leadsto e^* \circ e^* : \tau_2 \quad \forall x_i \in \text{dom}(\Gamma), \quad \Delta; \Phi_a \vdash \Gamma(x) \subseteq \Box \Gamma(x) \quad e^{**} = \text{let } y_i = x_i \text{ in } \text{fix}_{NC} f(x).e^*[^{\text{der}} y_i/x_i] \quad \text{e-fixNC.}
\]

\[
i : S, \Delta; \Phi_a; \Gamma \vdash e \leadsto e' : \tau \quad (⋆)
\]

\[
i \not\in \text{FIV}(\Phi_a; \Gamma) \quad \text{rr-iLam.}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e \mapsto e' : \forall i : S. \tau \quad (⋆)
\]

\[
\Delta \vdash I : S \quad (⋆)
\]

\[
\Delta; \Phi_a; \Gamma \vdash e \mapsto e' : [I/i] \quad \text{rr-iApp.}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e \mapsto e' : [I/i] \leadsto e^*[I/i] : \tau[I/i] \quad \text{e-iApp.}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e \mapsto e' : [I/i] \leadsto e^*[I/i] : \tau[I/i] \quad \text{rr-pack.}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e \mapsto e' : [I/i] \leadsto e^*[I/i] : \tau[I/i] \quad \text{e-pack.}
\]

\[
i : S, \Delta; \Phi_a; x : \tau_1, \Gamma \vdash e_2 \leadsto e'_2 ; \tau_2 \quad (⋆)
\]

\[
i \not\in \text{FIV}(\Phi_a; \Gamma, \tau_2, I_2) \quad \text{rr-unpack1.}
\]
\[ \Delta; \Phi_a; \Gamma \models e_1 \sim e'_1 \sim e_1^* \sim e_1^{**} : \exists i : S, \tau_i \quad i : S, \Delta; \Phi_a; x : \tau_i, \Gamma \vdash e_2 \sim e'_2 \sim e_2^* \sim e_2^{**} : \tau_2 \]

\[ i \not\in \text{FV}(\Phi_a; \Gamma, \tau_2, \ell_2) \quad e_1^{**} = \text{unpack } e_1^* \text{ as } (x, i) \text{ in } e_2^* \quad e_2^{**} = \text{unpack } e_2^* \text{ as } (x, i) \text{ in } e_2^* \]

\[ \Delta; \Phi_a; \Gamma \models \text{e-r-unpack.} \]

**Theorem 34 (Soundness of algorithmic typechecking in RelRefU)**

1. Assume that \( \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A \rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \) and \( \theta_a \) is a valid substitution for \( \psi_a \) s.t. \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) holds. Then, \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash [e] : \mathcal{C}[\theta_a] \).
2. Assume that \( \Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow A \Rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \) and \( \theta_a \) is the valid substitutions for \( \psi_a \) such that \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) holds. Then, \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash [e] : \mathcal{C}[\theta_a] \).
3. Assume that \( \Delta; \psi_a; \Phi_a; \Gamma \vdash e \uparrow e' \downarrow \tau \Rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_a) \) and \( \theta_a \) is the valid substitution for \( \psi_a \) such that \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) holds. Then, \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e] \sim [e'] : \mathcal{C}[\theta_a] \).
4. Assume that \( \Delta; \psi_a; \Phi_a; \Gamma \vdash e \uparrow e' \uparrow \tau \Rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_a) \) and \( \theta_a \) is the valid substitution for \( \psi_a \) such that \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) holds. Then, \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e] \sim [e'] : \mathcal{C}[\theta_a] \).

**Proof.** By simultaneous induction on the given algorithmic typing derivations.

**Proof of Theorem 34.1:**

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow A' \Rightarrow \Phi_1 \quad \Delta; \psi_a; \Phi_a \vdash^A A' \subseteq A \Rightarrow \Phi_2 \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A \Rightarrow \Phi_1 \land \Phi_2 \] 

**TS:** \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \models [e] : \mathcal{C}[\theta_a] \).

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \) (\( * \)) and \( \Delta; \Phi_a[\theta_a] \models \Phi_1 \land \Phi_2[\theta_a] \) (\( ** \))

Using (\( * \)) and (\( \circ \)), (\( ** \))’s derivation must be in a form such that we have

a) \( \Delta \triangleright \theta_a : \psi_a \)

b) \( \Delta; \Phi_a[\theta_a] \models \Phi_1[\theta_a] \)

c) \( \Delta; \Phi_a[\theta_a] \models \Phi_2[\theta_a] \)

By IH2 on the first premise using (\( * \), a) and b), we can show that

\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash [e] : \mathcal{A}'[\theta_a] \] (4.1)
By Theorem 27 using the second premise and c), we obtain

$$\Delta; \Phi_a[\theta_a] \vdash A'[\theta_a] \subseteq A[\theta_a]$$  \(4.2\)

Note that due to \((\star)\), we can conclude by the c-\(\subseteq\) exec rule using eqs. (5.6) and (5.7) that \(\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash [e] :^c A[\theta_a] \).

**Case**

\[
\frac{\Delta; \psi_a; \Phi_a; f : A_1 \rightarrow A_2, x : A_1, \Omega \vdash e \downarrow A_2 \Rightarrow \Phi}{\Delta; \psi_a; \Phi_a; \Omega \vdash \text{fix} f(x).e \downarrow A_1 \Rightarrow A_2 \Rightarrow \Phi} \quad \text{alg-u-fix-}\downarrow
\]

TS: \(\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{fix} f(x).[e] :^c A_1[\theta_a] \Rightarrow A_2[\theta_a] \).

By the main assumptions, we have \(\text{FIV}(\Phi_a, \Omega, A_1 \rightarrow A_2) \subseteq \text{dom}(\Delta, \psi_a) \)(\(\star\)) and \(\Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) (\(\star\)\)

Using \((\star)\), we can show that

\(a)\) \(\text{FIV}(\Phi_a, \Omega, A_1 \rightarrow A_2) \subseteq \text{dom}(\Delta, \psi_a)\).

We also can show that \((\star \star)\)'s derivation must be in a form such that we have

\(b)\) \(\Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \)

By IH1 on the first premise using a) and b), we can show that

$$\Delta; \Phi_a[\theta_a]; x : A_1[\theta_a], f : A_1[\theta_a] \rightarrow A_2[\theta_a], \Omega[\theta_a] \vdash [e] :^c A_2[\theta_a]$$  \(4.3\)

By the c-fix rule using eq. (5.8), we obtain

\(\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{fix} f(x).[e] :^c A_1[\theta_a] \rightarrow A_2[\theta_a] \).

**Case**

\[
\frac{i :: S, \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A \Rightarrow \Phi}{i :: S, \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{fix} f(x).[e] :^c A_1[\theta_a] \rightarrow A_2[\theta_a] \Rightarrow \Phi} \quad \text{alg-u-iLam-}\downarrow
\]

TS: \(\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{fix} f(x).[e] :^c A_1[\theta_a] \Rightarrow A_2[\theta_a] \).

By the main assumptions, we have \(\text{FIV}(\Phi_a, \Omega, \forall i :: S. A) \subseteq \text{dom}(\Delta, \psi_a) \)(\(\star\)) and \(\Delta; \Phi_a[\theta_a] \models \forall i :: S. \Phi[\theta_a] \) (\(\star \star\))

Using \((\star)\), we can show that

\(a)\) \(\text{FIV}(\Phi_a, \Omega, A) \subseteq i, \text{dom}(\Delta, \psi_a)\).

We can also show that \((\star \star)\)'s derivation must be in a form such that we have

\(b)\) \(i :: S, \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \)

By IH1 on the premise using a) and b), we can show that

\(i :: S, \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash [e] :^c A[\theta_a] \)  \(4.4\)
By the **c-iLam** rule using eq. (5.9), we obtain $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \forall \iota:A[\theta_a]$.

**Case** $\frac{\Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A[I/i]}{\Delta; \psi_a; \Phi_a; \Omega \vdash \text{pack } e \text{ with } I \downarrow \exists \iota:A \Rightarrow \Phi}$  

**alg-u-pack-↓**

TS: $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{pack } |e| \text{ with } I \downarrow \exists \iota:A[\theta_a]$.  

By the main assumptions, we have $\text{FIV}(\Phi_a, \Omega, \exists \iota:A) \subseteq \text{dom}(\Delta, \psi_a)$ (**) and $\Delta; \Phi_a[\theta_a] \models \Phi[\theta_a]$ (***)

Using (**) and the second premise, we can show that

a) $\text{FIV}(\Phi_a, \Omega, A[I/i]) \subseteq \text{dom}(\Delta, \psi_a)$.

By IH1 on the premise using a) and (**), we can show that

$$\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| : \subseteq A[\theta_a][I/i]$$  \hspace{1cm} (4.5)

By the **c-pack** rule using eq. (5.10) and the second premise, we obtain $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{pack } |e| \text{ with } I : \subseteq \exists \iota:A[\theta_a]$.  

**Case** $\frac{i :: S; \Delta; \psi_a; \Phi_a; x : A_1, \Omega \vdash e_2 \downarrow A_2 \Rightarrow \Phi_2 \quad i \not\in \text{FV}(\Phi_a; \Omega; A_2) \quad \Phi = \Phi_1 \land \forall \iota:S. \Phi_2}{\Delta; \psi_a; \Phi_a; \Omega \vdash \text{unpack } e_1 \text{ as } (x, i) \text{ in } e_2 \downarrow A_2 \Rightarrow \Phi}$  

**alg-u-pack-↓**

TS: $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{unpack } |e_1| \text{ with } (x, i) \text{ in } |e_2| : \subseteq A_2[\theta_a]$.  

By the main assumptions, we have $\text{FIV}(\Phi_a, \Omega, A_2) \subseteq \text{dom}(\Delta, \psi_a)$ (**) and $\Delta; \Phi_a[\theta_a] \models \Phi_1 \land \forall \iota:S. \Phi_2[\theta_a]$ (***)

Using (**) and (**) and the 3rd premise, (***)’s derivation must be in a form such that we have

a) $\Delta; \Phi_a[\theta_a] \models \Phi_1[\theta_a]$  

b) $i :: S; \Delta; \Phi_a[\theta_a] \models \Phi_2[\theta_a]$  

By IH2 on the first premise using (**), a), we can show that

$$\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_1| : \subseteq \exists \iota:A_1[\theta_a]$$  \hspace{1cm} (4.6)

From (**), we can show that

c) $\text{FIV}(\Phi_a, A_1, \Omega, A_2) \subseteq i, \text{dom}(\Delta, \psi, \psi_a)$

By IH1 on the second premise using b), c), (**), we obtain

$$i :: S; \Delta; \Phi_a[\theta_a]; x : A_1[\theta \theta_a], \Omega[\theta_a] \vdash |e_2| : \subseteq A_2[\theta_a]$$  \hspace{1cm} (4.7)

Then by the **c-unpack** rule using eqs. (5.11) and (5.12), we can show that $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{unpack } |e_1| \text{ with } (x, i) \text{ in } |e_2|$
Case

Case \( \Delta; \psi; C \land \Phi; \Omega \vdash e_1 \downarrow A \Rightarrow \Phi \)

\( \Delta; \psi; C \land \Phi; \Omega \vdash e_2 \downarrow A \Rightarrow \Phi_2 \)

\( \Delta \vdash C \text{ wf} \)

alg-u-split \downarrow

\[ \Delta; \psi; C \land \Phi; \Omega \vdash \text{ split} (e_1, e_2) \text{ with } C \downarrow A \Rightarrow C \rightarrow \Phi_1 \land \neg C \rightarrow \Phi_2 \]

TS: \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{ split } (|e_1|, |e_2|) \text{ with } C : \subseteq A[\theta_a] \).

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \) (**)

Using (**) and the third premise, we can show that

a) \( \text{FIV}(C \land \Phi, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \).

b) \( \text{FIV}(\neg C \land \Phi, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \).

Using (**) and the third premise, we can show that

c) \( \Delta; \neg C \land \Phi_a[\theta_a] \vdash \Phi_1[\theta_a] \)

d) \( \Delta; C \land \Phi_a[\theta_a] \vdash \Phi_2[\theta_a] \)

By IH1 on the first premise using (**) and c), we can show that

\[ \Delta; C \land \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_1| : \subseteq A[\theta_a]\{I[\theta_a]/i\} \tag{4.8} \]

By IH1 on the second premise using (**) and d), we can show that

\[ \Delta; \neg C \land \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_2| : \subseteq A[\theta_a]\{I[\theta_a]/i\} \tag{4.9} \]

By the \textit{c-split} rule using eqs. (4.13) and (5.14) and the third premise, we obtain

\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \text{ split } (|e_1|, |e_2|) \text{ with } C : \subseteq A[\theta_a] \).

Case

Case \( \Delta; \Phi \land C; \Omega \vdash e \downarrow A \Rightarrow \Phi \)

alg-u-c-impI \downarrow

\( \Delta; \psi; \Phi_a; \Omega \vdash e \downarrow C \supset A \Rightarrow C \rightarrow \Phi \)

TS: \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| : \subseteq C[\theta_a] \supset A[\theta_a] \).

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Omega, C \supset A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \) (**)

Using (**), we can show that

a) \( \text{FIV}(C \land \Phi, \Omega, A) \subseteq \text{dom}(\Delta, \psi_a) \).

By IH1 on the premise using (** and a), we can show that

\[ \Delta; C[\theta_a] \land \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| : \subseteq A[\theta_a] \tag{4.10} \]

By the \textit{c-impI} rule using eq. (5.15), we obtain \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| : \subseteq C[\theta_a] \supset A[\theta_a] \).
Proof of Theorem 34.2:

\[
\begin{align*}
\text{Case } & \quad \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A \Rightarrow \Phi & \quad \Delta; \Phi_a \vdash A \text{ wf} & \quad \text{FIV}(A, k, t) \in \Delta \\
& \quad \text{alg-u-anno} \uparrow \\
\Delta; \psi_a; \Phi_a; \Omega \vdash (e : A, k, t) \uparrow A \Rightarrow \Phi \\
\text{TS: } & \quad \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |(e : A, k, t)| : \subseteq A[\theta_a]. \\
\text{Since by definition, } & \quad \forall e. |(e : \_)| = |e|, \quad \text{STS: } \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| : \subseteq A[\theta_a].
\end{align*}
\]

By the main assumptions, we have FIV(\(\Phi_a, \Omega\)) \(\subseteq dom(\Delta, \psi_a)\) (\(\ast\)) and \(\Delta; \Phi_a[\theta_a] \models \Phi[\theta_a]\) (\(\ast\ast\)) such that \(\Delta \triangleright \theta_a : \psi_a\) are derivable.

Using the third premise, we can show that

\[
\begin{align*}
a) & \quad \text{FIV}(\Phi_a, \Omega, A) \subseteq dom(\Delta, \psi_a).
\end{align*}
\]

By IH on the first premise using (\(\ast\ast\)) and \(a\), we can conclude that

\[
\begin{align*}
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| : \subseteq A[\theta_a].
\end{align*}
\]

\[
\begin{align*}
\text{Case } & \quad \Delta; \psi_a; \Phi_a; \Omega \vdash e_1 \uparrow A_1 \rightarrow A_2 \Rightarrow \Phi_1 & \quad \Delta; \psi_a; \Phi_a; \Omega \vdash e_2 \downarrow A_1 \Rightarrow \Phi_2 \\
& \quad \text{alg-u-app} \uparrow \\
\Delta; \psi_a; \Phi_a; \Omega \vdash e_1 \downarrow A_2 \Rightarrow \Phi_1 \land \Phi_2 \\
\text{TS: } & \quad \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_1| \vdash A_2[\theta_a]. \\
\end{align*}
\]

By the main assumptions, we have FIV(\(\Phi_a, \Omega\)) \(\subseteq dom(\Delta, \psi_a)\) (\(\ast\)) and \(\Delta; \Phi_a[\theta_a] \models (\Phi_1 \land \Phi_2)[\theta_a]\) (\(\ast\ast\)) such that \(\Delta \triangleright \theta_a : \psi_a\) are derivable.

From (\(\ast\ast\)), we know

\[
\begin{align*}
a) & \quad \Delta; \Phi_a[\theta_a] \models \Phi_1[\theta_a] \\
b) & \quad \Delta; \Phi_a[\theta_a] \models \Phi_2[\theta_a]
\end{align*}
\]

By IH on the first premise using (\(\ast\)) and \(a\), we obtain

\[
\begin{align*}
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_1| : \subseteq A_1[\theta_a] \rightarrow A_2[\theta_a] & \quad (4.11)
\end{align*}
\]

By IH2 on the third premise using (\(\ast\)) and \(b\), we obtain

\[
\begin{align*}
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_2| : \subseteq A_1[\theta_a] & \quad (4.12)
\end{align*}
\]

Then, by using \(c\)-\textsc{app} rule using eqs. (5.16) and (5.17), we can show that

\[
\begin{align*}
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_1| \vdash |e_2| : \subseteq A_2[\theta_a].
\end{align*}
\]

\[
\begin{align*}
\text{Case } & \quad \Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow \forall i : S. A' \Rightarrow \Phi & \quad \Delta \vdash I :: S \\
& \quad \text{alg-u-iApp} \uparrow \\
\Delta; \psi_a; \Phi_a; \Omega \vdash e \{I/i\} \uparrow A'[I/i] \Rightarrow \Phi \\
\text{TS: } & \quad \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e| \{I\} : \subseteq (A'[I/i])[\theta_a].
\end{align*}
\]

By the main assumptions, we have FIV(\(\Phi_a, \Omega\)) \(\subseteq dom(\Delta, \psi_a)\) (\(\ast\)) and \(\Delta; \Phi_a[\theta_a] \models \Phi_2[\theta_a]\) (\(\ast\ast\)) such that \(\Delta \triangleright \theta_a : \psi_a\) are derivable.
By IH2 on the first premise using (⋆), we obtain
\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{e\} : \forall i : S. A'[\theta_a] \] (4.13)

Then, by **c-iApp** rule using eq. (5.18) and the second premise, we can conclude that
\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{e\} \{I/I\} : A'[\theta_a] \] (4.14)

**Proof of Theorem 34.3:**

**Case**
\[ \Delta; \psi_a; \Phi_a; \Box \Gamma \vdash e \odot e \downarrow \tau \Rightarrow \Phi \]  
**alg-r-nochange**
\[ \Delta; \psi_a; \Phi_a; \Box' \Gamma' \vdash e \odot e \downarrow \Box \tau \Rightarrow \Phi \]  
**TS:**  
\[ \Delta; \Phi_a[\theta_a]; \Box' \Gamma'[\theta_a]; \Box \Gamma[\theta_a] \vdash NC \{e \odot e \downarrow \Box \tau[\theta_a]\} \]

By the main assumptions, we have FIV(\(\Phi_a, \Box', \Box \Gamma, \tau\)) \(\subseteq dom(\Delta, \psi_a)\) (⋆) and
\[ \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \] (⋆⋆)

Using (⋆), we can show that

a) FIV(\(\Phi_a, \Box, \tau\)) \(\subseteq dom(\Delta, \psi_a)\).

By IH3 on the premise using a) and (⋆⋆), we can show that
\[ \Delta; \Phi_a[\theta_a]; \Box \Gamma[\theta_a] \vdash \{e\} \odot \{e\} : \tau[\theta_a] \] (4.14)

By the **c-nochange** rule using eq. (4.14), we obtain
\[ \Delta; \Phi_a[\theta_a]; \Box' \Gamma'[\theta_a]; \Box \Gamma[\theta_a] \vdash NC \{e \odot e \downarrow \Box \tau[\theta_a]\} \]

**Case**

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e \odot e' \uparrow \text{list}\[n\] \odot \tau \Rightarrow \Phi_e \]  
**alg-r**
\[ \Delta; \psi_a; \Phi_a; \Gamma' \vdash e_1 \odot e'_1 \downarrow \tau' \Rightarrow \Phi_1 \]  
\[ i : N, \Delta; \psi_a; n \equiv i + 1 \wedge \Phi_a; h : \Box \tau; \text{tl} : \text{list}\[i\] \odot \tau, \Gamma \vdash e_2 \odot e'_2 \downarrow \tau' \Rightarrow \Phi_2 \]  
\[ i : N, \beta : N, \Delta; \psi_a; n \equiv i + 1 \wedge \alpha \equiv \beta + 1 \wedge \Phi_a; h : \tau; \text{tl} : \text{list}\[i\] \odot \tau, \Gamma \vdash e_3 \odot e'_3 \downarrow \tau' \Rightarrow \Phi_3 \]  
\[ \Phi_{body} = (n \equiv 0 \Rightarrow \Phi_1) \land (\forall i : N. (n \equiv i + 1) \Rightarrow (\Phi_2 \land \forall \beta : N. (\alpha \equiv \beta + 1) \Rightarrow \Phi_3)) \]

**Case**

\[ \text{case } e \text{ of nil } \rightarrow e_1 \]  
**alg-r**
\[ \text{case } e' \text{ of nil } \rightarrow e'_1 \]

**TS:**
\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \ {h} : \text{nc} \ \rightarrow e_2 \uplus \ {h} : \text{nc} \ \rightarrow e'_2 \downarrow \tau' \Rightarrow (\Phi_e \land \Phi_{body}) \]  
\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \ {h} : \text{c} \ \rightarrow e_3 \downarrow \tau' \Rightarrow \Phi_3 \]

By the main assumptions, we have FIV(\(\Phi_a, \Gamma, \tau'\)) \(\subseteq dom(\Delta, \psi_a)\) (⋆) and
\[ \Delta; \Phi_a[\theta_a] \models \Phi_e[\theta_a] \] (⋆⋆)

Using (⋆), (⋆⋆)'s derivation must be in a form such that we have

a) \(\Delta; \Phi_a[\theta_a] = \Phi_e[\theta_a]\)
b) $\Delta; n[\theta_a] \Rightarrow 0 \land \Phi_a[\theta_a] \models \Phi_1[\theta_a]$

c) $i :: S, \Delta; n[\theta_a] \Rightarrow i + 1 \land \Phi_a[\theta_a] \models \Phi_2[\theta_a]$

d) $i :: S, \beta :: S, \Delta; n[\theta_a] \Rightarrow i + 1 \land \alpha[\theta_a] \Rightarrow \beta + 1 \land \Phi_a[\theta_a] \models \Phi_3[\theta_a]$

By IH4 on the first premise using a) and (*), we can show that

$$\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e] \odot [e'] : \text{list}[n[\theta_a]]^\alpha[\theta_a] \tau[\theta_a]$$  \hspace{1cm} (4.15)

By IH3 on the second premise using b) and (*), we can show that

$$\Delta; n[\theta_a] \Rightarrow 0 \land \Phi_a[\theta_a]; \Gamma'[\theta_a] \vdash [e_1] \odot [e'_1] : \tau'[\theta_a]$$  \hspace{1cm} (4.16)

By IH3 on the third premise using c) and (*), we can show that

$$i :: S, \Delta; n[\theta_a] \Rightarrow i + 1 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e_2] \odot [e'_2] : \tau'[\theta_a]$$  \hspace{1cm} (4.17)

By IH3 on the fourth premise using d) and (*), we can show that

$$i :: S, \beta :: S, \Delta; n[\theta_a] \Rightarrow i + 1 \land \alpha[\theta_a] \Rightarrow \beta + 1 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e_3] \odot [e'_3] : \tau'[\theta_a]$$  \hspace{1cm} (4.18)

Then by c-r-caseL rule using eqs. (4.15) to (4.18), we can show that

$$\Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \odot e'_1 \downarrow \Box \tau \Rightarrow \Phi_1$$

$$\Delta; i, \psi_a; \Phi_a; \Gamma \vdash e_2 \odot e'_2 \downarrow \text{list}[n]^\alpha \tau \Rightarrow \Phi_2$$

$$\Phi'_2 = \Phi_2 \land n \Rightarrow (i + 1)$$

$$\Delta; \psi_a; \Phi_a; \Gamma \vdash \text{cons}_{NC}(e_1, e_2) \odot \text{cons}_{NC}(e'_1, e'_2) \downarrow \text{list}[n]^\alpha \tau \Rightarrow \Phi_1 \land \exists i :: N. \Phi'_2$$

TS: $\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{cons}_{NC}(e_1, e_2) \odot \text{cons}_{NC}(e'_1, e'_2) : \text{list}[n[\theta_a]]^\alpha[\theta_a] \tau[\theta_a]$.  

By the main assumptions, we have FIV($\Phi_a, \Gamma, \text{list}[n]^\alpha \tau$) \subseteq dom($\Delta, \psi_a$)  \hspace{1cm} (*)

Using (a), (**)'s derivation must be in a form such that we have

a) $\Delta; \Phi_a[\theta_a] \models \Phi_1[\theta_a]$

b) $\Delta \vdash I :: N$

c) $\Delta; \Phi_a[\theta_a] \models \Phi_2[\theta_a, i \mapsto I]$

d) $\Delta; \Phi_a[\theta_a] \models (I + 1) \models n[\theta_a]$
By IH3 on the second premise using (*) and a), we can show that

$$\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_1| \odot |e'_1| : \square \tau[\theta_a]$$  \hfill (4.19)

By IH3 on the third premise using (*) and b), we can show that

$$\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_2| \odot |e'_2| : \text{list}[I]^{\alpha[\theta_a]} \tau[\theta_a]$$  \hfill (4.20)

By c-r-cons2 typing rule using eqs. (4.19) and (4.20), we obtain

$$\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{cons}_{NC}(|e_1|, |e_2|) \odot \text{cons}_{NC}(|e'_1|, |e'_2|) : \text{list}[I+1]^{\alpha[\theta_a]} \tau[\theta_a].$$

We conclude by applying c-r-\subseteq\subseteq rule to this using d).

**Proof of Theorem 34.4:**

Case \( \Delta; \psi_a; \Phi_a; \Gamma \vdash e \odot e' \downarrow \tau \Rightarrow \Phi \) \quad \Delta; \Phi_a \vdash \tau \text{ wf} \quad \text{FIV}(\tau) \in \Delta \quad \text{alg-r-anno}\uparrow

\[
\Delta; \psi_a; \Phi_a; \Gamma \vdash (e : \tau) \odot (e' : \tau) \uparrow \tau \Rightarrow \Phi
\]

TS: \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [(e : \tau, t)] \odot [(e' : \tau, t)] : \tau[\theta_a]. \)

Since by definition, \( \forall e. [(e : \tau, t)] = |e| \), STS: \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \tau[\theta_a]. \)

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Gamma) \subseteq \text{dom}(\Delta, \psi_a) \) (**) and \( \Delta; \Phi_a[\theta_a] \vdash \Phi[\theta_a] \) (***)

Using the third premise, we can show that

a) \( \text{FIV}(\Phi_a, \Gamma, \tau) \subseteq \text{dom}(\Delta, \psi_a) \).

By IH4 on the first premise using (**) and a), we can conclude that

\( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \tau[\theta_a] \).

Case \( \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \odot e_2 \uparrow \Box \tau \Rightarrow \Phi \quad \text{alg-r-der}\uparrow

\[
\Delta; \psi_a; \Phi_a; \Gamma \vdash \text{der} e_1 \odot \text{der} e_2 \uparrow \tau \Rightarrow \Phi
\]

TS: \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{der} |e| \odot \text{der} |e'| : \tau[\theta_a]. \)

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Gamma) \subseteq \text{dom}(\Delta, \psi_a) \) (*) and \( \Delta; \Phi_a[\theta_a] \vdash \Phi[\theta_a] \) (**) such that \( \Delta \triangleright \theta_a : \psi_a \) are derivable.

By IH4 on the first premise using (*) and ***, we obtain

$$\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| : \Box \tau[\theta_a]$$  \hfill (4.21)

Then, by c-der rule using eq. (4.21), we can conclude that

$$\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{der} |e| \odot \text{der} |e'| : \tau[\theta_a].$$

\[\square\]

**Theorem 35** (Completeness of algorithmic typechecking in RelRefU)

1. Assume that \( \Delta; \Phi_a; \Omega \vdash e : \text{c} A \). Then, there exists \( e' \) such that \( \Delta; ; \Phi_a; \Omega \vdash e' \downarrow A \) and \( \Delta; \Phi_a \vdash \Phi \) and \( |e'| = e \).
2. Assume that $\Delta; \Phi; \Gamma \vdash e_1 \sim e_2: \tau$. Then, there exist $e_1', e_2'$ such that $\Delta; \vdash e_1' \tau e_2' \downarrow \tau \Rightarrow \Phi$ and $\Delta; \Phi; \vdash \Phi$ and $|e_1'| = e_1$ and $|e_2'| = e_2$.

Proof. By simultaneous induction on the given Core typing derivations.

Proof of Theorem 35.1:

Case $\Omega(x) = A$

\begin{align*}
\Delta; \Phi; \Omega \vdash x : A & \quad \text{c-var} \\
\end{align*}

We can conclude as follows

\begin{align*}
\Omega(x) = A & \quad \text{alg-u-var-} \uparrow \quad \Delta; \Phi; \vdash A \subseteq A \Rightarrow \Phi \\
\Delta; \Phi; \vdash \uparrow A & \quad \text{alg-r-} \uparrow \downarrow \\
\end{align*}

Case $\Delta; \Phi; \Omega \vdash e_1 : A \quad \Delta; \Phi; \Omega \vdash e_2 : \text{list}[n] A$

\begin{align*}
\Delta; \Phi; \Omega \vdash \text{cons}(e_1, e_2) : \text{list}[n+1] A & \quad \text{c-cons} \\
\end{align*}

By IH1 on the first premise, $\exists e_1'$ such that

a) $\Delta; \vdash e_1' \downarrow A \Rightarrow \Phi_1$

b) $\Delta; \Phi; \vdash \Phi_1$

c) $|e_1'| = e_1$

By IH1 on the second premise, $\exists e_2'$ such that

d) $\Delta; \vdash e_2' \downarrow \text{list}[n] A \Rightarrow \Phi_2$

e) $\Delta; \Phi; \vdash \Phi_2$

f) $|e_2'| = e_2$

Then, we can conclude as follows

1. $i \in \text{fresh}(\mathbb{N})$

\begin{align*}
\Delta; i, k_1, k_2, \psi; \Phi; \Omega \vdash e_1' \downarrow A \Rightarrow \Phi_1' & \quad \text{alg-u-cons-} \downarrow \\
\Delta; i, k_2, e_1', \psi; \Phi; \Omega \vdash \text{list}[i] A \Rightarrow \Phi_2' & \quad \Phi_2'' = (\Phi_2 \land n + 1 \equiv (i + 1) \land) \\
\Delta; \psi; \Phi; \Omega \vdash \text{cons}(e_1', e_2') \downarrow \text{list}[n + 1] A \Rightarrow \Phi_1' \land \exists i : \mathbb{N}. \Phi_2'' & \\
\end{align*}

2. Using c) and f), $|\text{cons}(e_1', e_2')| = \text{cons}(e_1, e_2)$.

Proof of Theorem 35.2:

Case $\Delta; \Phi; \Gamma \vdash e_1 \sim e_2 : \square \tau$

\begin{align*}
\Delta; \Phi; \Gamma \vdash \text{der} e_1 \sim \text{der} e_2 : \tau & \quad \text{c-der} \\
\end{align*}

By IH2 on the premise, $\exists e_1', e_2'$ such that
a) $\Delta; \Phi; \Gamma \vdash e'_1 \land e'_2 \downarrow \tau, t \Rightarrow \Phi$

b) $\Delta; \Phi \models \Phi$

c) $|e'_1| = e_1$ and $|e'_2| = e_2$

Then, we can conclude by using a), b) and c) as follows:

\[
\Delta; \Phi; \Gamma \vdash e'_1 \land e'_2 \downarrow \tau \Rightarrow \Phi
\]

1. $\begin{align*}
\Delta; \Phi; \Gamma \vdash e'_1 \land e'_2 \downarrow \tau \Rightarrow \Phi & \quad \text{alg-r-der-↓} \\
\Delta; \Phi; \Gamma \vdash \text{der } e'_1 \land \text{der } e'_2 \downarrow \tau \Rightarrow \Phi & \quad \text{alg-r-anno-↑} \\
\Delta; \Phi \models \tau \equiv \tau' \Rightarrow \Phi \quad & \text{by Lemma 8} \\
\Delta; \Phi; \Gamma \vdash (\text{der } e'_1 : \tau) \land (\text{der } e'_2 : \tau) \downarrow \tau \Rightarrow \Phi \land \Phi' & \quad \text{alg-r-↑↓}
\end{align*}
\]

2. By c), $|\text{der } e'_1 : \tau| = \text{der } |e'_1|$.

3. By b) and Lemma 25.

Case $\Delta; \Phi; \Gamma \vdash e \sim e' : \tau$

\[
\Delta; \Phi ; \Gamma \models \tau \equiv \tau' \quad \text{c-r-≡}
\]

By IH2 on the first premise, $\exists e'_1, e'_2$ such that

a) $\Delta; \Phi; \Gamma \vdash e'_1 \land e'_2 \downarrow \tau \Rightarrow \Phi_1$

b) $\Delta; \Phi \models \Phi_1$

c) $|e'_1| = e_1$ and $|e'_2| = e'$

By IH2 on the second premise,

d) $\Delta; \Phi \models \tau \equiv \tau' \Rightarrow \Phi_2$

e) $\Delta; \Phi \models \Phi_2$.

Then, we can conclude as follows

1. By using a) and d)

\[
\begin{align*}
\Delta; \Phi; \Gamma \vdash e'_1 \land e'_2 \downarrow \tau \Rightarrow \Phi_1 & \quad \text{alg-r-anno-↑} \\
\Delta; \Phi; \Gamma \vdash (e'_1 : \tau) \land (e'_2 : \tau) \uparrow \tau \Rightarrow \Phi_1 & \quad \text{alg-r-↑↓} \\
\Delta; \Phi \models \tau \equiv \tau' \Rightarrow \Phi \land \Phi' & \quad \text{alg-r-↑↓}
\end{align*}
\]

2. By using b), e), we can show that $\Delta; \Phi \models \Phi_1 \land \Phi_2$

3. By c), $|e'_1 : \tau| = e'_1$ and $(e'_2 : \tau) = e'_2$
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e'_1 : \tau_1 \rightarrow \tau_2 \]

Case  \(\Delta; \Phi_a; \Gamma \vdash e_2 \rightsquigarrow e'_2 : \tau_1\)  c-r-app

\[ \Delta; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e'_1, e'_2 : \tau_2 \]

By IH2 on the first premise, \(\exists \bar{r}_1, \bar{r}'_1\) such that

1. \[ \Delta; ; \Phi_a; \Gamma \vdash \bar{r}_1 \cap \bar{r}'_1 \downarrow \tau_1 \rightarrow \tau_2 \Rightarrow \Phi_1 \] alg-r-anno→

\[ \Delta; ; \Phi_a; \Gamma \vdash (\bar{r}_1 : \tau_1 \rightarrow \tau_2) \cap \bar{r}'_1 : \tau_1 \rightarrow \tau_2 \uparrow \tau_1 \rightarrow \tau_2 \Rightarrow \Phi_1 \] c-r-app

\[ \Delta; ; \Phi_a; \Gamma \vdash E_1 \cap E_2 \uparrow \tau_2 \Rightarrow \Phi_1 \land \Phi_2 \] alg-r-↓

\[ \Delta; ; \Phi_a; \Gamma \vdash E_1 \cap E_2 \downarrow \tau_2 \Rightarrow \Phi_1 \land \Phi_2 \]

where \(E_1 = (\bar{r}_1 : \tau_1 \rightarrow \tau_2) \bar{r}_2\) and \(E_2 = (\bar{r}'_1 : \tau_1 \rightarrow \tau_2) \bar{r}'_2\).

2. By using b) and c).

3. Using c) and f), \(|\bar{r}_1| = e_1, e_2\) and \(|\bar{r}'_1| = e'_1, e'_2| = e'_1, e'_2|.

\[ \Delta; ; \Phi_a; \Gamma \vdash e_1 \rightsquigarrow e'_1 : \exists i : S, \tau_1 \]

Case \[ \Delta; ; \Phi_a; \Gamma \vdash \text{unpack} \ e_1 \ \text{as} \ (x, i) \ \text{in} \ e_2 \ \text{unpack} \ e'_1 \ \text{as} \ (x, i) \ \text{in} \ e'_2 : \tau_2 \]

By IH2 on the first premise, \(\exists \bar{r}_1, \bar{r}'_1\) such that

a) \[ \Delta; ; \Phi_a; \Gamma \vdash \bar{r}_1 \cap \bar{r}'_1 \\downarrow \exists i : S, \tau_1 \Rightarrow \Phi_1 \]

b) \[ \Delta; ; \Phi_a \vdash \Phi_1 \]

c) \[ |\bar{r}_1| = e_1 \text{ and } |\bar{r}'_1| = e'_1 \]

By IH2 on the second premise, \(\exists \bar{r}_2, \bar{r}'_2\) such that

d) \[ i : S, \Delta; ; \Phi_a; x : \tau_1, \Gamma \vdash \bar{r}_2 \cap \bar{r}'_2 \downarrow \tau_2 \Rightarrow \Phi_2 \]

e) \[ i : S, \Delta; ; \Phi_a \vdash \Phi_2 \]

f) \[ |\bar{r}_2| = e_2 \text{ and } |\bar{r}'_2| = e'_2 \]
Then, we can conclude as follows

1. 

\[
\Delta; \Phi \vdash \overline{\tau}_1 \hat{\odot} \overline{\tau}_1' \downarrow \exists i : S. \tau_1 \Rightarrow \Phi_1 \quad (a)
\]

\[
\Delta; \Phi \vdash E_1 \hat{\odot} E_1' \uparrow \exists i : S. \tau_1 \Rightarrow \Phi_1
\]

\[
i : S, \Delta; \Phi \vdash x : \tau_1, \Gamma \vdash \overline{\tau}_2 \hat{\odot} \overline{\tau}_2' \downarrow \tau_2 \Rightarrow \Phi_2
\]

\[
\Phi' = \Phi_1 \land \Phi_2'
\]

\[
\Delta; \Phi \vdash \text{unpack } E_1 \text{ as } (x, i) \text{ in } e_2 \quad \text{alg-r-unpack-↓}
\]

\[
\text{where } E_1 = (\overline{\tau}_1 : \exists i : S. \tau_1) \text{ and } E_2 = (\overline{\tau}_1' : \exists i : S. \tau_1)
\]

2. By using b) and e).

3. Using c) and f), \(|\text{unpack } (\overline{\tau}_1 : \exists i : S. \tau_1)\text{ as } (x, i)\text{ in } e_2| = \text{unpack } e_1\text{ as } (x, i)\text{ in } e_2\) and \(|\text{unpack } (\overline{\tau}_1' : \exists i : S. \tau_1)\text{ as } (x, i)\text{ in } e_2'| = \text{unpack } e_1'\text{ as } (x, i)\text{ in } e_2'.\)
5 RelCost

This appendix considers the following additions to the main paper.

- The constrained type $C \supset \tau$, which is eliminated with the “celim e” construct.

- The type $U A$ is generalized to $U (A_1, A_2)$ (like in RelCost’s appendix), allowing us to relate two expressions of two different unary types $A_1$ and $A_2$, respectively. As a result of this change, switch rule, $\rightarrow$ exec subtyping rule, and some of the asynchronous rules are also generalized.

We first present RelCost’s syntax, typing and subtyping rules. Then, we introduce RelCostCore and the embedding of RelCost into RelCostCore. Finally, the bidirectional system BiRelCost is introduced.

In Section 6, we present our benchmark programs along with the results of the experimental evaluation.

We use some abbreviations throughout. STS stands for “suffices to show”, TS stands for “to show”, and RTS stands for “remains to show”.

89
Relational types

\[ \tau ::= \text{unit} \mid \text{int} \mid \tau_1 \times \tau_2 \mid \tau_1 + \tau_2 \mid \tau_1 \text{diff}(t) \rightarrow \tau_2 \mid \text{list}[n]^{\alpha}\tau \]

\[ \forall i \text{diff}(t) ::= S.\tau \mid \exists i :: S.\tau \mid U(A_1, A_2) \mid \Box \tau \mid C \& \tau \mid C \triangleright \tau \]

Unary types

\[ A ::= \text{unit}_r \mid \text{int}_r \mid A_1 \times A_2 \mid A_1 + A_2 \mid A_2 \text{exec}(k, t) \rightarrow A_2 \mid \text{list}[n] A \]

\[ \forall i \text{exec}(k, t) ::= S.A \mid \exists i :: S.A \mid C \& A \mid C \triangleright A \]

Sorts

\[ S ::= \mathbb{N} \mid \mathbb{R} \]

Index terms

\[ I, k, t, \alpha ::= i \mid 0 \mid \infty \mid I + 1 \mid I_1 + I_2 \mid I_1 - I_2 \mid \frac{I_1}{I_2} \mid I_1 \cdot I_2 \mid [I] \mid [I] \]

\[ \log_2(I) \mid I_1^I \mid \min(I_1, I_2) \mid \max(I_1, I_2) \mid \sum_{i=I_1}^{I_2} I \]

Constraints

\[ C ::= I_1 \equiv I_2 \mid I_1 < I_2 \mid \neg C \]

Constraint env.

\[ \Phi ::= \top \mid C \land \Phi \]

Sort env.

\[ \Delta ::= \emptyset \mid \Delta, i :: S \]

Unary type env.

\[ \Omega ::= \emptyset \mid \Omega, x :: A \]

Relational type env.

\[ \Gamma ::= \emptyset \mid \Gamma, x :: \tau \]

Primitive env.

\[ \Upsilon ::= \emptyset \mid \Upsilon, \zeta :: \tau \text{diff}(t) \rightarrow \tau_2 \mid \Upsilon, \zeta :: A \text{exec}(k, t) \rightarrow A_2 \]

RelCost typing judg.

\[ \Omega \vdash_k e :: A \]

\[ \Gamma \vdash e_1 \odot e_2 \preceq t :: \tau \]

RelCostCore typing judg.

\[ \Omega \vdash_k e :: c A \]

\[ \Gamma \vdash e_1 \odot e_2 \preceq t :: c \tau \]

Figure 58: Syntax of types and contexts

Terms

\[ e ::= x \mid n \mid \text{fix}\ f(x).e \mid e_1 e_2 \mid \zeta e \mid \langle e_1, e_2 \rangle \mid \pi_1(e) \mid \pi_2(e) \mid \text{inl}(e) \mid \text{inr}(e) \mid \text{case}\ (e, x.e_1, y.e_2) \mid \text{nil} \mid \text{cons}(e_1, e_2) \mid \text{case}\ e\ of\ \text{nil} \rightarrow e_1 \mid h :: t \rightarrow e_2 \mid \Lambda e \mid e[\ ] \mid \text{pack}(e) \mid \text{unpack}(e) \mid \text{let}\ e_1 = e_2 \mid \text{let}\ x = e_1\ in\ e_2 \mid () \mid \text{celt}\ e_1\ as\ x\ in\ e_2 \mid \Lambda e \mid \text{pack}\ e \mid () \]

Values

\[ v ::= n \mid \text{fix}(f).v \mid \langle v_1, v_2 \rangle \mid \text{inl}(v) \mid \text{inr}(v) \mid \text{nil} \mid \text{cons}(v_1, v_2) \mid \Lambda e \mid \text{pack}\ v \mid () \]

Figure 59: Syntax of values and expressions in RelCost
Terms \( e ::= x \mid n \mid \text{fix}(x).e \mid \text{fix}_{NC}(x).e \mid e_1 \cdot e_2 \mid \zeta e \mid \langle e_1, e_2 \rangle \mid \pi_1(e) \mid \pi_2(e) \mid \text{inl} e \mid \text{inr} e \mid \text{case} (e, x, e_1, y, e_2) \mid \text{nil} \mid \text{cons}_{NC}(e_1, e_2) \mid \text{cons}(e_1, e_2) \mid \text{case} e \mid \text{pack} e \mid \text{fix} e \mid \text{unpack} e_1 \text{as} (x, i) \text{in} e_2 \mid \text{let} x = e_1 \text{in} e_2 \mid () \mid \text{clet} e_1 \text{as} x \text{in} e_2 \mid \text{celim} e \mid \text{der} e \mid \text{switch} e \mid \text{NC} e \mid \text{split} (e_1, e_2) \text{with} C \mid \text{contra} e \)

Values \( v ::= n \mid \text{fix}(x).e \mid \text{fix}_{NC}(x).e \mid \langle v_1, v_2 \rangle \mid \text{inl} v \mid \text{inr} v \mid \text{nil} \mid \text{cons}_{NC}(v_1, v_2) \mid \text{cons}(v_1, v_2) \mid \lambda i.e \mid \text{pack} v \text{with} I \mid () \)

Figure 60: Syntax of values and expressions in RelCostCore

Terms \( e ::= x \mid n \mid \text{fix}(x).e \mid \text{fix}_{NC}(x).e \mid e_1 \cdot e_2 \mid \zeta e \mid \langle e_1, e_2 \rangle \mid \pi_1(e) \mid \pi_2(e) \mid \text{inl} e \mid \text{inr} e \mid \text{case} (e, x, e_1, y, e_2) \mid \text{nil} \mid \text{cons}_{NC}(e_1, e_2) \mid \text{cons}(e_1, e_2) \mid \text{case} e \mid \text{pack} e \mid \text{fix} e \mid \text{unpack} e_1 \text{as} (x, i) \text{in} e_2 \mid \text{let} x = e_1 \text{in} e_2 \mid () \mid \text{clet} e_1 \text{as} x \text{in} e_2 \mid \text{celim} e \mid \text{der} e \mid \text{switch} e \mid \text{NC} e \mid \text{split} (e_1, e_2) \text{with} C \mid \text{contra} e \mid (e : \tau, t) \mid (e : A, k, t) \)

Values \( v ::= n \mid \text{fix}(x).e \mid \text{fix}_{NC}(x).e \mid \langle v_1, v_2 \rangle \mid \text{inl} v \mid \text{inr} v \mid \text{nil} \mid \text{cons}_{NC}(v_1, v_2) \mid \text{cons}(v_1, v_2) \mid \lambda i.e \mid \text{pack} v \text{with} I \mid () \)

Figure 61: Syntax of values and expressions in BiRelCost
$\Delta; \Phi \vdash A \text{wf}$ checks well-formedness of the unary type $A$

$\Delta; \Phi \vdash \tau \text{wf}$ checks well-formedness of the binary type $\tau$

Figure 62: Well-formedness of binary types
\( \Delta \vdash A \text{ wf} \) checks well-formedness of the unary type \( A \Phi \)

\[
\begin{array}{c}
\Delta; \Phi_a \vdash \text{unit} \text{ wf} \\
\Delta; \Phi_a \vdash \text{int} \text{ wf} \\
\Delta; \Phi_a \vdash A_1 \times A_2 \text{ wf} \\
\Delta; \Phi_a \vdash A_1 + A_2 \text{ wf} \\
\Delta; \Phi_a \vdash A_1 \text{ exec}(k,t) \text{ wf} \\
\Delta; \Phi_a \vdash \text{list}[n] \text{ wf} \\
\Delta; \Phi_a \vdash \exists i : S.A \text{ wf} \\
\Delta; \Phi_a \vdash C \supset A \text{ wf} \\
\end{array}
\]

Figure 63: Well-formedness of unary types

\[
\begin{array}{c}
\emptyset \\
\Delta \vdash I : S \\
\Delta \vdash \Theta : \psi \\
\end{array}
\]

Figure 64: Sorting of Substitutions
Figure 65: RelCost subtyping rules (part 1)
\[ \Delta; \Phi \vdash \tau_1 \subseteq \tau_2 \] Binary type \( \tau_1 \) is a subtype of type \( \tau_2 \)

\[
\begin{array}{c}
  \frac{i :: S, \Delta; \Phi \vdash \tau \subseteq \tau'}{\Delta; \Phi \vdash \exists i :: S. \tau \subseteq \exists i :: S. \tau'} \\
  \Delta; \Phi \vdash C \land C' \supseteq \Delta; \Phi \vdash \tau \subseteq \tau' \\
  \Delta; \Phi \vdash C \supseteq \tau \subseteq C' \supseteq \tau' \\
  \Delta; \Phi \vdash \tau \subseteq \tau_1 \subseteq \tau_2 \\
  \Delta; \Phi \vdash \tau \subseteq \tau_2 \subseteq \tau_3 \\
  \Delta; \Phi \vdash \tau \subseteq \tau_1 \subseteq \tau_3 \\
  \end{array}
\]

Figure 66: RelCost subtyping rules (Part 2)
Unary type $A_1$ is a subtype of type $A_2$

$$\Delta; \Phi \vdash A_1 \subseteq A_2$$

$$\Delta; \Phi \vdash A_1' \subseteq A_1 \quad \Delta; \Phi \vdash A_2 \subseteq A_2' \quad \Delta; \Phi \vdash k' \leq k \quad \Delta; \Phi \vdash t \leq t'$$

$$\Delta; \Phi \vdash \text{exec}(k, t) \rightarrow \text{exec}$$

$$\Delta; \Phi \vdash A_1 \subseteq A_2 \quad i :: S; \Delta; \Phi \vdash k \leq k \quad i :: S; \Delta; \Phi \vdash t \leq t' \quad i \not\in \text{FV}(\Phi)$$

$$\Delta; \Phi \vdash \forall i :: S. A \subseteq \forall i :: S. A$$

$$\Delta; \Phi \vdash A_1 \times A_2 \subseteq A_1' \times A_2'$$

$$\Delta; \Phi \vdash n \equiv n' \quad \Delta; \Phi \vdash A \subseteq A'$$

$$\Delta; \Phi \vdash \text{list}[n] A \subseteq \text{list}[n'] A'$$

$$\Delta; \Phi \vdash C \land A \subseteq C' \land A'$$

$$\Delta; \Phi \vdash u-\text{and}$$

$$\Delta; \Phi \vdash A \subseteq A$$

$$\Delta; \Phi \vdash u-\text{refl}$$

$$\Delta; \Phi \vdash A_1 \subseteq A_2$$

$$\Delta; \Phi \vdash A_1 \subseteq A_3$$

$$\Delta; \Phi \vdash u-\text{tran}$$

Figure 67: RelCost unary subtyping rules
General rules

\[
\begin{align*}
\Delta; \Phi_a; \Gamma &\vdash_{\Delta}^1 e_1 : A \\
\Delta; \Phi_a; \Gamma &\vdash_{\Delta}^2 e_2 : A
\end{align*}
\]
\[
\Delta; \Phi_a; \Gamma \vdash e_1 \oplus e_2 \lessdot t_1 - k_2 : U/A
\]
\[
\text{switch} \quad \Delta; \Phi_a; \Gamma \vdash e \circ e \lessdot t : \tau
\]
\[
\forall x \in \text{dom}(\Gamma). \quad \Delta; \Phi_a \vdash \Gamma(x) \subseteq \Box \Gamma(x)
\]
\[
\Delta; \Phi_a; \Gamma^{\tau'}; \Omega \vdash e \circ e \lessdot 0 : \Box \tau
\]
\[
\text{nochange}
\]

Constant integers and unit

\[
\Delta; \Phi_a; \Omega \vdash_0 n : \text{int} \quad \text{const}
\]
\[
\Delta; \Phi_a; \Gamma \vdash n \odot n \lessdot 0 : \text{int}_r \quad \text{r-const}
\]
\[
\Delta; \Phi_a; \Omega \vdash_0 () : \text{unit} \quad \text{unit}
\]
\[
\Delta; \Phi_a; \Gamma \vdash () \odot () \lessdot 0 : \text{unit}_r \quad \text{r-unit}
\]

Variables \(x\)

\[\Omega(x) = A\]
\[\Delta; \Phi_a; \Omega \vdash_0 x : A \quad \text{var}\]
\[\Gamma(x) = \tau\]
\[\Delta; \Phi_a; \Gamma \vdash x \odot x \lessdot 0 : \tau \quad \text{r-var}\]

\[\text{inl } e\]
\[
\Delta; \Phi_a; \Omega \vdash_k e : A_1 \quad \Delta; \Phi_a \vdash^A A_2 \quad \text{wf}
\]
\[\Delta; \Phi_a; \Omega \vdash_k \text{inl } e : A_1 + A_2 \quad \text{inl}\]
\[
\Delta; \Phi_a; \Omega \vdash_k e : A_2 \quad \Delta; \Phi_a \vdash^A A_1 \quad \text{wf}
\]
\[\Delta; \Phi_a; \Omega \vdash_k \text{inr } e : A_1 + A_2 \quad \text{inr}\]

\[\text{case } (e, x_1, y, e_2)\]
\[
\Delta; \Phi_a; \Omega \vdash_k e : A_1 + A_2 \\
\Delta; \Phi_a; x : A_1, \Omega \vdash_{k'} e_1 : A \\
\Delta; \Phi_a; y : A_2, \Omega \vdash_{k'} e_2 : A
\]
\[
\Delta; \Phi_a; \Omega \vdash_{k+k'+c_{\text{case}}} \text{case } (e, x_1, y, e_2) : A
\]
\[
\Delta; \Phi_a; \Gamma \vdash e \odot e' \lessdot t : \tau_1 + \tau_2
\]
\[
\Delta; \Phi_a; x : \tau_1, \Gamma \vdash e_1 \odot e'_1 \lessdot t' : \tau
\]
\[
\Delta; \Phi_a; y : \tau_2, \Gamma \vdash e_2 \odot e'_2 \lessdot t' : \tau
\]
\[
\Delta; \Phi_a; \Gamma \vdash \text{case } (e, x_1, y, e_2) \odot \text{case } (e', x_1', y, e_2') \lessdot t + t' : \tau \quad \text{r-case}
\]

Figure 68: RelCost typing rules (Part 1)
\[
\begin{align*}
\Delta; \Phi_a \vdash A_1 & \\
\text{exec}(k, t) & \quad \Delta; \Phi_a; x : A_1, f : A_1 \quad \text{exec}(k, t) \\
\hline
\Delta; \Phi_a; \Omega \vdash \tau f x . e : A_2 & \\
\text{fix} & \\
\Delta; \Phi_a; \Omega \vdash \tau f x . e : A_1 & \\
\end{align*}
\]

\[
\Delta; \Phi_a \vdash \tau_1 \quad \text{diff}(t) \quad \Delta; \Phi_a; x : \tau_1, f : \tau_1 \quad \text{diff}(t) \\
\hline
\Delta; \Phi_a; \Gamma \vdash f (x). e \circ f (x). e_2 \leq 0 : \tau_1 \quad \text{r-fix} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Gamma \vdash f (x). e \circ f (x). e \leq 0 : \Box (\tau_1 \quad \text{r-fixNC}) & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Omega \vdash \tau_{k_1} e_1 : A_1 & \\
\text{exec}(k, t) \quad \Delta; \Phi_a; \Omega \vdash \tau_{k_2} e_2 : A_1 & \\
\hline
\Delta; \Phi_a; \Omega \vdash \tau_{k_1 + k_2 + k e p p} e_1 e_2 : A_2 & \\
\text{app} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e_1 \circ e_1' \leq t_1 : \tau_1 & \\
\text{diff}(t) & \\
\Delta; \Phi_a; \Gamma \vdash e_2 \circ e_2' \leq t_2 : \tau_1 & \\
\text{r-app} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Omega \vdash \tau_{k_1} e_1 : A_1 & \\
\hline
\Delta; \Phi_a; \Omega \vdash \tau_{k_2} e_2 : A_2 & \\
\text{prod} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e_1 \circ e_1' \leq t_1 : \tau_1 & \\
\Delta; \Phi_a; \Gamma \vdash e_2 \circ e_2' \leq t_2 : \tau_1 & \\
\text{r-prod} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Omega \vdash \tau_{k_1} e : A_1 \times A_2 & \\
\hline
\Delta; \Phi_a; \Omega \vdash \tau_{k_1 + \text{proj}_{k_1 \text{proj}}} \pi_1 (e) : A_1 & \\
\text{proj1} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Gamma \vdash e \circ e' \leq t : \tau_1 \times \tau_2 & \\
\text{r-proj1} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Omega \vdash \tau_{k_1} e : A_1 & \\
\hline
\Delta; \Phi_a; \Omega \vdash \tau_{k_2} e : A_2 & \\
\text{proj1} & \\
\end{align*}
\]

\[
\Delta; \Phi_a; \Gamma \vdash \pi_1 (e) \circ \pi_1 (e') \leq t : \tau_1 & \\
\text{r-proj1} & \\
\end{align*}
\]

Symmetric rules.

Figure 69: RelCost typing rules (Part 2)
$$\Delta; \Phi_0 \vdash^A A \text{ w.f.} \quad \Delta; \Phi_0; \Omega \vdash^\emptyset 0 \text{ nil : list}[0]A \quad \Delta; \Phi_0 \vdash \tau \text{ w.f.} \quad \Delta; \Phi_0; \Gamma \vdash \text{ nil } \odot \text{ nil } \preceq 0 : \text{ list}[0]^\alpha \tau$$

$$\text{cons}(e_1, e_2)$$

$$\Delta; \Phi_a; \Omega \vdash^{t_1}_{k_1} e_1 : A \quad \Delta; \Phi_a; \Omega \vdash^{t_2}_{k_2} e_2 : \text{ list}[n]A \quad \Delta; \Phi_a; \Omega \vdash_{k_1 + k_2} \text{ cons}(e_1, e_2) : \text{ list}[n + 1]A$$

$$\Delta; \Phi_a; \Gamma \vdash e_1 \odot e_1' \preceq t_1 : \tau \quad \Delta; \Phi_a; \Gamma \vdash e_2 \odot e_2' \preceq t_2 : \text{ list}[n]^\alpha \tau \quad \Delta; \Phi_a; \Gamma \vdash e_2 \odot e_2' \preceq t_2 : \text{ list}[n + 1]^\alpha \tau$$

$$\text{case } e \text{ of nil } \rightarrow e_1 | h :: tl \rightarrow e_2$$

$$\Delta; \Phi_a; \Omega \vdash^{t'}_{k'} e \vdash \text{ list}[n]A \quad \Delta; \Phi_a \land n = 0; \Omega \vdash^{t'}_{k'} e_1 : A' \quad i, \Delta; \Phi_a \land n = i + 1; h : A, tl : \text{ list}[i]A, \Omega \vdash^{t'}_{k'} e_2 : A'$$

$$\Delta; \Phi_a; \Omega \vdash_{k_1 + k_2 + e_{caseL}} \text{ case } e \text{ of nil } \rightarrow e_1 | h :: tl \rightarrow e_2 : A'$$

$$\Delta; \Phi_a; \Gamma \vdash e \odot e' \preceq t : \text{ list}[n]^\alpha \tau \quad \Delta; \Phi_a \land n = 0; \Gamma \vdash e_1 \odot e_1' \preceq t' : \tau' \quad i, \Delta; \Phi_a \land n = i + 1; h : \square \tau, tl : \text{ list}[i]^{\beta} \tau, \Gamma \vdash e_2 \odot e_2' \preceq t' : \tau'$$

$$\Delta; \Phi_a; \Gamma \vdash \text{ case } e \text{ of nil } \rightarrow e_1 | h :: tl \rightarrow e_2 \quad \text{ case } e' \text{ of nil } \rightarrow e_1' | h :: tl \rightarrow e_2' \preceq t + t' : \tau'$$

$$\text{leaf}$$

$$\Delta; \Phi_a; \Omega \vdash^A A \text{ w.f.} \quad \Delta; \Phi_a; \Omega \vdash \tau \text{ w.f.} \quad \Delta; \Phi_a; \Gamma \vdash \text{ leaf } \odot \text{ leaf } \preceq 0 : \text{ tree}[0]^\alpha \tau$$

$$\text{node}(e_l, e_r, e)$$

$$\Delta; \Phi_a; \Omega \vdash^{t_1}_{k_1} e_l : \text{ tree}[i]A \quad \Delta; \Phi_a; \Omega \vdash^{t_2}_{k_2} e_r : \text{ tree}[j]A \quad \Delta; \Phi_a; \Omega \vdash_{k_1 + k_2 + e_{node}} \text{ node}(e_l, e_r, e) : \text{ tree}[i + j + 1]A$$

$$\Delta; \Phi_a; \Gamma \vdash e \odot e' \preceq t : \tau \quad \Delta; \Phi_a; \Gamma \vdash e_l \odot e_l' \preceq t_1 : \text{ tree}[i]^{\beta} \tau \quad \Delta; \Phi_a; \Gamma \vdash e_r \odot e_r' \preceq t_2 : \text{ tree}[j]^{\beta} \tau$$

$$\Delta; \Phi_a; \Gamma \vdash \text{ node}(e_l, e_r, e) \odot \text{ node}(e_l', e_r', e_r) \preceq t + t_1 + t_2 : \text{ tree}[i + j + 1]^{\alpha + \beta + 1} \tau$$

Figure 70: RelCost typing rules (Part 3)
\[
\text{\textbf{case e of leaf} \rightarrow e_1 \mid \text{node}(l, x, r) \rightarrow e_2}
\]

\[
\begin{array}{c}
\Delta; \Phi_a; \Omega \vdash \text{k}_e : \text{tree}[n] A \\
i, j, \Delta; \Phi_a \land n = i + j + 1; x : A, l : \text{tree}[i] A, r : \text{tree}[j] A, \Omega \vdash \text{t}', e_2 : A' \\
\Delta; \Phi_a; \Omega \vdash \text{k}_e + \text{t}' + \text{caseT} \\
\text{case e of leaf} \rightarrow e_1 \mid \text{node}(l, x, r) \rightarrow e_2 : A'
\end{array}
\]

caseT

\[
\begin{array}{c}
\Delta; \Phi_a; \Gamma \vdash e \circ e' \subseteq t : \text{tree}[n] \tau \\
i, j, \beta, \Delta; \Phi_a \land n = i + j + 1; x : \beta + \theta; l : \theta, l : \text{tree}[i] \beta, r : \text{tree}[j] \theta, \Gamma \vdash e_2 \circ e_2' \subseteq t' : \tau' \\
\Delta; \Phi_a; \Gamma \vdash e \circ e' \subseteq t : \tau \\
i, j, \beta, \theta, \Delta; \Phi_a \land n = i + j + 1; k = \beta + \theta; l = t : \text{tree}[i] \beta, r : \text{tree}[j] \theta, \Gamma \vdash e_2 \circ e_2' \subseteq t' : \tau' \\
\Delta; \Phi_a; \Gamma \vdash e \circ e' \subseteq t : \tau
\end{array}
\]

r-caseT

\[
\begin{array}{c}
\Lambda e \\
i :: S, \Delta; \Phi_a; \Omega \vdash \text{k}_e : A \\
i \notin \text{FIV}(\Phi_a; \Omega) \\
\Delta; \Phi_a; \Omega \vdash \text{k}_e \Lambda e : \forall \text{exec}(k, t) :: S. A \\
i :: S, \Delta; \Phi_a; \Gamma \vdash e \circ e' \subseteq t : \tau \\
i \notin \text{FIV}(\Phi_a; \Gamma) \\
\Delta; \Phi_a; \Gamma \vdash \Lambda e \circ \Lambda e' \subseteq 0 : \forall i :: S. \tau
\end{array}
\]

r-Lam

\[
\begin{array}{c}
i [] \\
\Delta; \Phi_a; \Omega \vdash \text{k}_e : \forall i \text{exec}(k', t') :: S. A \\
\Delta \vdash I : S \\
\Delta; \Phi_a; \Omega \vdash \text{k}_e [] : A[I/i] \\
\Delta; \Phi_a; \Gamma \vdash e \circ e' \subseteq t : \forall i :: S. \tau \\
\Delta \vdash I : S \\
\Delta; \Phi_a; \Gamma \vdash e [[] \circ e' []] \subseteq t + t'[I/i] : \tau[I/i]
\end{array}
\]

r-App

\[
\begin{array}{c}
\text{pack e} \\
\Delta; \Phi_a; \Omega \vdash \text{k}_e A[I/i] : A \rightarrow S. A \\
\Delta \vdash I : S \\
\Delta; \Phi_a; \Omega \vdash \text{k}_p \text{pack e} : \forall i :: S. A \\
\Delta; \Phi_a; \Gamma \vdash \text{pack e} \circ \text{pack e'} \subseteq t : \forall i :: S. \tau
\end{array}
\]

r-pack

\[
\begin{array}{c}
\text{unpack e as x in e'} \\
\Delta; \Phi_a; \Omega \vdash \text{k}_1 e_1 : \exists i :: S. A_1 \\
i :: S, \Delta; \Phi_a; x : A_1, \Omega \vdash \text{k}_2 e_2 : A_2 \\
i \notin \text{FV}(\Phi_a; \Gamma, A_2, k_2, t_2) \\
\Delta; \Phi_a; \Omega \vdash \text{k}_1 + \text{k}_2 \text{unpack e}_1 \text{as x in e}_2 \rightarrow A_2 \\
\Delta; \Phi_a; \Gamma \vdash e_1 \circ e_2' \subseteq t_1 : \exists i :: S. \tau_1 \\
i :: S, \Delta; \Phi_a; x : \tau_1, \Gamma \vdash e_2 \circ e_2' \subseteq t_2 : \tau_2 \\
i \notin \text{FV}(\Phi_a; \Gamma, \tau_2, t_2) \\
\Delta; \Phi_a; \Gamma \vdash \text{unpack e}_1 \text{as x in e}_2 \circ \text{unpack e}_2' \text{as x in e}_2' \rightarrow \exists i :: S. \tau_1 + t_2 : \tau_2
\end{array}
\]

r-unpack1

Figure 71: Typing rules (Part 4)
Primitive application

\[
\begin{align*}
\text{\text{Primapp}} & \quad \Rightarrow \\
\Delta; \Phi_a; \Omega \vdash \delta^e_k e : A_1 & \quad \Rightarrow \\
\Delta; \Phi_a; \Omega \vdash \delta^{\tau} e : A_2
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e \ast e' \leq t' : \tau_1 & \quad \Rightarrow \\
\Delta; \Phi_a; \Gamma \vdash \zeta e \ast \zeta' \leq t + t' : \tau_2
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e_1 \circ e_1' \leq l_1 : C & \quad \Rightarrow \\
\Delta; \Phi_a; \Gamma \vdash e \circ e' \leq t : \tau & \quad \Rightarrow \\
\Delta; \Phi_a; \Gamma \vdash e \ast e' \leq t : C \ast \tau
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash C & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast C; \Omega \vdash \delta^e_k e : A & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^e_k e : C \ast A
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash C \ast \delta^e_k e : C \ast A & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^{\tau} e : \tau & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^{\tau} e : \tau
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash C & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^e_k e : C \ast A & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^{\tau} e : \tau & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^{\tau} e : \tau
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash C & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^e_k e : C \ast A & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^{\tau} e : \tau & \quad \Rightarrow \\
\Delta; \Phi_a \vdash C \ast \delta^{\tau} e : \tau
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash e_1 \ast e_1' \leq l_1 : C & \quad \Rightarrow \\
\Delta; \Phi_a \vdash e \circ e' \leq t : \tau & \quad \Rightarrow \\
\Delta; \Phi_a \vdash e \ast e' \leq t : C \ast \tau
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash e_1 \ast e_1' \leq l_1 : C & \quad \Rightarrow \\
\Delta; \Phi_a \vdash e \circ e' \leq t : \tau & \quad \Rightarrow \\
\Delta; \Phi_a \vdash e \ast e' \leq t : C \ast \tau
\end{align*}
\]

\[
\begin{align*}
\Delta; \Phi_a \vdash e_1 \ast e_1' \leq l_1 : C & \quad \Rightarrow \\
\Delta; \Phi_a \vdash e \circ e' \leq t : \tau & \quad \Rightarrow \\
\Delta; \Phi_a \vdash e \ast e' \leq t : C \ast \tau
\end{align*}
\]
Subtyping

\[
\frac{\Delta; \Phi_a; \Omega \vdash_s e : A \quad \Delta; \Phi_a \models A \sqsubseteq A' \quad \Delta; \Phi_a \models k' \leq k \quad \Delta; \Phi_a \models t \leq t'}{\Delta; \Phi_a; \Omega \vdash_{exec} e : A'}
\]

Constraint dependent typing

\[
\frac{\Delta; \Phi_a \land C; \Gamma \vdash_k e : A \quad \Delta; \Phi_a \land \neg C; \Gamma \vdash_k e : A \quad \Delta \vdash C \text{ wf}}{\Delta; \Phi_a; \Gamma \vdash_k e : A \text{ split}}
\]

\[
\frac{\Delta; \Phi_a \land C; \Gamma \vdash e_1 \odot e_2 \leq t : \tau \quad \Delta; \Phi_a \land \neg C; \Gamma \vdash e_1 \odot e_2 \leq t : \tau \quad \Delta \vdash C \text{ wf}}{\Delta; \Phi_a; \Gamma \vdash e_1 \odot e_2 \leq t : \tau \text{ r-split}}
\]

Asynchronous typing

\[
\frac{\Delta; \Phi_a; \Gamma \mid \vdash_{k_1} e_1 : A_1}{\Delta; \Phi_a; \Gamma \mid \vdash_{exec(k_1,t)} e_1 : A_1 \quad \Delta \vdash \text{case}(e, x, e_1, y, e_2) \odot e' \leq t' + t + c_{\text{case}} : \tau \text{ r-case-e}}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2 \quad \Delta; \Phi_a; \Gamma \mid \vdash_{exec(k_1,t)} e' : A_1 + A_2 \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e-app} e \odot e' \leq t' + k - k - c_{\text{app}} : U(A_2, A'_2) \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2}{\Delta; \Phi_a; \Gamma \mid \vdash_{r-e-let} e \odot e' \leq t' : \tau \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e-let} e \odot e' \leq t' : \tau \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e-let} e \odot e' \leq t' : \tau \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e-let} e \odot e' \leq t' : \tau}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2 \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2 \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2}{\Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2 \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2 \quad \Delta; \Phi_a; \Gamma \mid \vdash_{r-e} e : A_1 + A_2}
\]

Figure 72: RelCost typing rules (Part 6)
Expressions $e_1 \oplus e_2$ are embedded into $e_1^* \oplus e_2^*$ with the relational type $\tau$ and the relational cost $t$.

Expression $e$ is embedded into $e^*$ with the unary type $A$ and the minimum and maximum execution costs $k$ and $t$, respectively.

**General rules**

\[
\Delta; \Phi; \Gamma \vdash \frac{\Delta; \Phi; \Gamma \vdash e \ominus e^* \leq t : \tau}{\forall x_i \in \text{dom}(\Gamma), \quad e_i = \text{coerce}_{\Gamma(x_i), \Gamma(x_i)}} \text{ e-nochange}
\]

**Constant integers and unit**

\[
\Delta; \Phi; \Omega \vdash_0 n \leadsto n : \text{int} \quad \Delta; \Phi; \Gamma \vdash n \ominus n \leq 0 : \text{int}_\tau \quad \Delta; \Phi; \Gamma \vdash n \ominus n \leq 0 : \text{int}_\tau
\]

**Variables $x$**

\[
\Omega(x) = A \quad \Gamma(x) = \tau
\]

**Inl**

\[
\Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A \quad \Delta; \Phi; \Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A \quad \Delta; \Phi; \Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A \quad \Delta; \Phi; \Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A
\]

**Inr**

\[
\Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A \quad \Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A \quad \Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A
\]

**Case**

\[
\Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A \quad \Delta; \Phi; \Omega \vdash_0 e \leadsto e^* : A
\]

**Figure 73: Embedding Rules (Part 1)**
\[\text{Figure 74: Embedding Rules (Part 2)}\]
Figure 75: Embedding Rules (Part 3)
pack e

$$\Delta; \Phi; \Omega, e \vdash e^* : A \{i/i\} \quad \Delta \vdash I :: S \quad \text{e-u-pack}$$

$$\Delta; \Phi; \Omega, e \vdash e^* \quad \text{pack e} \quad \text{with } I \vdash e^* : \exists i : S. A \quad \Delta; \Phi; \Gamma \vdash \text{e-r-pack}$$

$$\Delta; \Phi; \Gamma \vdash e \circ e' \vdash e^* \circ e'^* \leq t : \tau \{i/i\} \quad \Delta \vdash I :: S$$

unpack e as x in e'

$$\Delta; \Phi; \Omega, e_1 \vdash e_1^* : \exists i : S. A_1 \quad i :: S, \Delta; \Phi; \Omega, x : A_1, \Omega, e_2 \vdash e_2^* : A_2 \quad i \notin \text{FV}(\Phi, \Omega, A_2, k_2, t_2)$$

$$\Delta; \Phi; \Omega, e_1 \vdash e_1^* \circ e_1'^* \leq t_1 : \exists i : S. \tau_1 \quad i :: S, \Delta; \Phi; \Omega, x : \tau_1, \Gamma \vdash e_2 \circ e_2' \vdash e_2^* \circ e_2'^* \leq t_2 : \tau_2$$

$$\Delta; \Phi; \Omega, e_1 \vdash e_1^* \circ e_1'^* \leq t_1 + t_2 : \tau_2$$

let x = e_1 in e_2

$$\Delta; \Phi; \Omega, e_1 \vdash e_1^* : A_1 \quad \Delta; \Phi; \Omega, x : A_1, \Omega, e_2 \vdash e_2^* : A_2$$

$$\Delta; \Phi; \Omega, e_1 \vdash e_1^* \circ e_1'^* \leq t_1 : \tau_1 \quad \Delta; \Phi; \Omega, x : \tau_1, \Gamma \vdash e_2 \circ e_2' \vdash e_2^* \circ e_2'^* \leq t_2 : \tau_2$$

$$\Delta; \Phi; \Omega, e_1 \vdash e_1^* \circ e_1'^* \leq t_1 + t_2 : \tau_2$$

C & $\tau$ intro. rules

$$\Delta; \Phi; \vdash C \quad \Delta; \Phi; \Gamma \vdash e \vdash e^* : C \land A \quad \Delta; \Phi; \Omega, e \vdash e^* : A \quad \text{e-u-andI}$$

$$\Delta; \Phi; \vdash C \quad \Delta; \Phi; \Lambda; C; \Gamma \vdash e \circ e' \vdash e^* \circ e'^* \leq t : \tau \quad \Delta; \Phi; \Gamma \vdash e \circ e' \vdash e^* \circ e'^* \leq t : C \land \tau \quad \text{e-r-andI}$$

C & $\tau$ elim. rules

$$\Delta; \Phi; \Omega, e \vdash e^* : C \land A \quad \Delta; \Phi; \Omega, x : A_1, \Omega, e \vdash e_2^* : A_2$$

$$\Delta; \Phi; \Omega, e \vdash e^* \circ e' \circ e^* \leq t : \tau \quad \Delta; \Phi; \Omega, e \vdash e^* \circ e' \circ e^* \leq t : C \land \tau \quad \text{e-u-c-andE}$$

$$\Delta; \Phi; \Omega, e \vdash e^* \circ e' \circ e^* \leq t : \tau \quad \Delta; \Phi; \Omega, e \vdash e^* \circ e' \circ e^* \leq t : C \land \tau \quad \text{e-r-c-andE}$$

C $\Rightarrow$ $\tau$ intro. rules

$$\Delta; \Phi; \Lambda; C; \vdash e \vdash e^* : A \quad \Delta; \Phi; \Omega, e \vdash e^* : C \land A \quad \Delta; \Phi; \Gamma \vdash e \vdash e^* : C \land \tau \quad \text{e-u-c-impl}$$

$$\Delta; \Phi; \Lambda; C; \vdash e \vdash e^* : C \land \tau \quad \Delta; \Phi; \Omega, e \vdash e^* : C \Rightarrow \tau \quad \text{e-r-c-impl}$$

Figure 76: Embedding Rules (Part 4)
\[ \begin{align*}
\Delta; \Phi_a; \Omega \vdash _k^e e & \rightsquigarrow e^* : C \supset A & \Delta; \Phi_a \models C & \text{e-u-c-implE} \\
\Delta; \Phi_a; \Omega \vdash _k^e \text{elim} e & \rightsquigarrow \text{elim} e^* : A
\end{align*} \]

**Subsumption**

\[ \begin{align*}
\Delta; \Phi_a; \Omega \vdash _k^e e & \rightsquigarrow e^* : A & \Delta; \Phi_a \models A \subseteq A' & \Delta; \Phi_a \models k' \leq k & \Delta; \Phi_a \models t \leq t' & \text{e-u-\subseteq}
\end{align*} \]

\[ \begin{align*}
\Delta; \Phi_a; \Gamma & \vdash e_1 \odot e_2 \rightsquigarrow e_1^* \odot e_2^* \leq t : \tau & \Delta; \Phi_a \models \tau \subseteq \tau' & e' = \text{coerce}_{\tau,\tau'} & \Delta; \Phi_a \models t \leq t' & \text{e-r-\subseteq}
\end{align*} \]

**Constraint-based rules**

\[ \begin{align*}
\Delta; C \land \Phi_a; \Omega \vdash _k^e e & \rightsquigarrow e^* : A & \Delta; \neg C \land \Phi_a; \Omega \vdash _k^e e & \rightsquigarrow e^* : A & \Delta; \models C \text{ wf} & \text{e-u-split}
\end{align*} \]

\[ \begin{align*}
\Delta; C \land \Phi_a; \Gamma & \vdash e_1 \odot e_2 \rightsquigarrow e_1^* \odot e_2^* \leq t : \tau & \Delta; \neg C \land \Phi_a; \Gamma \vdash e_1 \odot e_2 \rightsquigarrow e_1^* \odot e_2^* \leq t : \tau & \Delta; \models C \text{ wf}
\end{align*} \]

\[ \begin{align*}
\Delta; C \land \Phi_a; \Gamma & \models \text{split} (e_1^*, e_1^{**}) \text{ with } C \land \text{split} (e_2^*, e_2^{**}) \text{ with } C \leq t : \tau & \text{e-r-split}
\end{align*} \]

Figure 77: Embedding Rules (Part 5)
Asynchronous typing rules

\[
\begin{align*}
\Delta; \Phi: |\Gamma|_1 \vdash_1 e_1 & \rightarrow e_1^* : A_1 & \Delta; \Phi: |\Gamma|_1 \vdash_1 x : U(A_1, A_1), \Gamma \vdash e_2 \odot e \rightarrow e_2^* \odot e^* \leq t_2 \quad \text{e-let-e} \\
\Delta; \Phi: \Gamma \vdash \text{let } x = e_1 \text{ in } e_2 \odot e & \rightarrow \text{let } x = e_1^* \text{ in } e_2^* \odot e^* \leq t_1 + t_2 + c_{\text{let}} : \tau_2 \\
\Delta; \Phi: |\Gamma|_2 \vdash_1 e_1 & \rightarrow e_1^* : A_1 & \Delta; \Phi: |\Gamma|_2 \vdash_1 x : U(A_1, A_1), \Gamma \vdash e \odot e_2 \rightarrow e^* \odot e_2^* \leq t_2 \quad \text{e-let-e} \\
\Delta; \Phi: \Gamma \vdash e \odot \text{let } x = e_1 \text{ in } e_2 \odot e^* & \rightarrow \text{let } x = e_1^* \text{ in } e_2^* \odot e^* \leq t_2 - k_1 - c_{\text{let}} : \tau_2 \\
\Delta; \Phi: |\Gamma|_1 \vdash_1 e_1 & \rightarrow e_1^* : A_1 & \Delta; \Phi: |\Gamma|_1 \vdash_1 \text{exec}((k, t)) & \rightarrow A_2 \quad \Delta; \Phi: \Gamma \vdash e_2 \odot e_2' \rightarrow e_2^* \odot e_2'^* \leq t_2 : U(A_1, A_2') \\
\Delta; \Phi: \Gamma \vdash e_2 \odot e_2' & \rightarrow e_2^* \odot e_2'^* \leq t_2 + t + c_{\text{app}} : U(A_2, A_2') \\
\Delta; \Phi: |\Gamma|_2 \vdash_1 e_1 & \rightarrow e_1^* : A_1 & \Delta; \Phi: |\Gamma|_2 \vdash_1 \text{exec}((k, t)) & \rightarrow A_2 \quad \Delta; \Phi: \Gamma \vdash e_2 \odot e_2' \rightarrow e_2^* \odot e_2'^* \leq t_2 : U(A_2, A_2') \\
\Delta; \Phi: \Gamma \vdash e_2 \odot e_2' & \rightarrow e_2^* \odot e_2'^* \leq t_2 - k_1 - k - c_{\text{app}} : U(A_2, A_2') \\
\Delta; \Phi: |\Gamma|_1 \vdash_1 e & \rightarrow e^* : A_1 + A_2 & \Delta; \Phi: |\Gamma|_1 \vdash_1 x : U(A_1, A_1), \Gamma \vdash e_1 \odot e' \rightarrow e_1^* \odot e'^* \leq t' : \tau \\
\Delta; \Phi: \Gamma \vdash y : U(A_2, A_2), \Gamma \vdash e_2 \odot e' & \rightarrow e_2^* \odot e'^* \leq t_2 - k_1 - k - c_{\text{app}} : U(A_2, A_2') \\
\Delta; \Phi: \Gamma \vdash \text{case } (e, \tau, e_1, y, e_2) \odot e' & \rightarrow \text{case } (e^*, \tau, e_1', y, e_2') \odot e'^* \leq t' + t + c_{\text{case}} : \tau \\
\Delta; \Phi: \Gamma \vdash (e, \tau, e_1, y, e_2) \odot e' & \rightarrow \text{case } (e^*, \tau, e_1', y, e_2') \odot e'^* \leq t - k' - c_{\text{case}} : \tau \\
\end{align*}
\]

Figure 78: Embedding Rules (Part 6)
\[ \Delta; \Phi_a \models \tau_1 \equiv \tau_2 \] checks whether \( \tau_1 \) is equivalent to \( \tau_2 \)

\[
\begin{align*}
\Delta; \Phi_a \models \text{int}_r \equiv \text{int}_r & \quad \text{eq-int} \\
\Delta; \Phi_a \models \tau_1 \equiv \tau_1' & \quad \Delta; \Phi_a \models \tau_2 \equiv \tau_2' & \quad \text{eq-fun} \\
\Delta; \Phi_a \models t \equiv t' & \quad \text{eq-sum} \\
\Delta; \Phi_a \models \forall \Delta; \Phi_a \models \tau \equiv \tau' & \quad \text{eq-∀} \\
\Delta; \Phi_a \models C \equiv C' & \quad \text{eq-B-□} \\
\end{align*}
\]

\[ \begin{align*}
\Delta; \Phi_a \models A_1 \sqsubseteq A_1' & \quad \Delta; \Phi_a \models A_1' \sqsubseteq A_1 & \quad \Delta; \Phi_a \models A_2 \sqsubseteq A_2' & \quad \Delta; \Phi_a \models A_2' \sqsubseteq A_2 & \quad \text{eq-U} \\
\Delta; \Phi_a \models U (A_1, A_2) \equiv U (A_1', A_2') & \quad \text{eq-c-impl} \\
\Delta; \Phi_a \models C \sqcap \Phi_a \models C' & \quad \Delta; \Phi_a \models \tau \equiv \tau' & \quad \text{eq-c-prod} \\
\end{align*}
\]
General rules

\[ \Delta; \Phi_a; \Gamma \vdash e \in \Omega, \Gamma \vdash e \in \tau \]
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \otimes e_2 \leq t_1 - k_2 : \tau \quad \text{c-switch} \]
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \otimes e_2 \leq t : \tau \quad \text{c-nochange} \]
\[ \Delta; \Phi_a; \Gamma \vdash e_1 \otimes e_2 \leq t : \tau \quad \text{c-der} \]

Constant integers and unit

\[ \Delta; \Phi_a; \Omega \vdash \mathbb{N} : \text{int} \quad \text{c-const} \]
\[ \Delta; \Phi_a; \Omega \vdash \mathbb{N} : \text{unit} \quad \text{c-unit} \]

Variables \( x \)

\[ \Omega(x) = A \quad \text{c-var} \]
\[ \Gamma(x) = \tau \quad \text{c-r-var} \]

inl \( e \)

\[ \Delta; \Phi_a; \Omega \vdash e : \text{inl} \quad \Delta; \Phi_a; \Omega \vdash A_1 \quad \Delta; \Phi_a; \Omega \vdash A_2 \quad \text{c-inl} \]
\[ \Delta; \Phi_a; \Gamma \vdash e \in \text{inl} \quad \Delta; \Phi_a; \Gamma \vdash t_1 : \tau_1 \quad \Delta; \Phi_a; \Gamma \vdash t_2 : \tau_2 \quad \text{c-r-inl} \]

inr \( e \)

\[ \Delta; \Phi_a; \Omega \vdash e : \text{inr} \quad \Delta; \Phi_a; \Omega \vdash A_1 \quad \Delta; \Phi_a; \Omega \vdash A_2 \quad \text{c-inr} \]
\[ \Delta; \Phi_a; \Gamma \vdash e \in \text{inr} \quad \Delta; \Phi_a; \Gamma \vdash t_1 : \tau_1 \quad \Delta; \Phi_a; \Gamma \vdash t_2 : \tau_2 \quad \text{c-r-inr} \]

\[ \text{case } (e, e_1, y, e_2) \]

\[ \Delta; \Phi_a; \Omega \vdash e : \text{case } (e, e_1, y, e_2) \quad \Delta; \Phi_a; \Omega \vdash e_1 : \text{case } (e, e_1, y, e_2) \quad \Delta; \Phi_a; \Omega \vdash e_2 : \text{case } (e, e_1, y, e_2) \quad \text{c-case} \]

\[ \Delta; \Phi_a; \Gamma \vdash e \in \text{case } (e, e_1, y, e_2) \quad \Delta; \Phi_a; \Gamma \vdash e_1 \in \text{case } (e, e_1, y, e_2) \quad \Delta; \Phi_a; \Gamma \vdash e_2 \in \text{case } (e, e_1, y, e_2) \quad \text{c-r-case} \]

Figure 80: RelCostCore typing rules (Part 1)
Symmetric rules.

Figure 81: RelCostCore typing rules (Part 2)
\[\begin{align*}
\Delta; \Phi_a \vdash \text{c-nil} & \quad \Delta; \Phi_a \vdash \tau \text{wf} \\
\Delta; \Phi_a; \Omega \vdash^0 \text{nil}; : \text{list}[0] A & \quad \Delta; \Phi_a; \Gamma \vdash \text{nil} \circ \text{nil} \leq 0 : \text{list}[0] \tau
\end{align*}\]

Figure 82: RelCostCore typing rules (Part 3)
\[ i :: S, \Delta; \Phi_a; \Omega \vdash_\text{exec(k,t)} e :: A \quad i \notin \text{FIV}(\Phi_a; \Omega) \quad \text{c-lam} \]
\[ \Delta; \Phi_a; \Omega \vdash_0 \forall i. e :: S, A \]
\[ i :: S, \Delta; \Phi_a; \Gamma \vdash e \odot e' \subseteq t :: \tau \quad i \notin \text{FIV}(\Phi_a; \Gamma) \quad \text{c-r-lam} \]
\[ \Delta; \Phi_a; \Gamma \vdash \forall i. e \odot e' \subseteq 0 :: \text{diff}(t) :: S, \tau \]

\[ \Delta; \Phi_a; \Omega \vdash_\text{exec(k',t')} e :: \forall i. A \quad \Delta \vdash I :: S \quad \text{c-app} \]
\[ \Delta; \Phi_a; \Omega \vdash e[I] :: \forall i. A[I/i] \quad \Delta \vdash I :: S \quad \text{c-r-app} \]

\[ \Delta; \Phi_a; \Omega \vdash e :: \forall i. A[I/i] \quad \Delta \vdash I :: S \quad \text{c-pack} \]
\[ \Delta; \Phi_a; \Omega \vdash \text{pack} e \quad \text{with I} :: \exists i :: S.A \quad \text{c-r-pack} \]

\[ \Delta; \Phi_a; \Omega \vdash e :: \forall i. A[I/i] \quad \Delta \vdash I :: S \quad \text{c-pack} \]
\[ \Delta; \Phi_a; \Omega \vdash \text{pack} e \quad \text{with I} :: \forall i. A[I/i] \quad \text{c-r-pack} \]

\[ \Delta; \Phi_a; \Omega \vdash_\text{exec(k,t)} e :: A \quad \Delta \vdash I :: S \quad \text{c-unpack} \]
\[ \Delta; \Phi_a; \Omega \vdash \text{unpack} e_1 \quad \text{as (x,i)} \quad \text{in e_2 :: A_2} \]
\[ \Delta; \Phi_a; \Omega \vdash \text{unpack} e_1 \quad \text{as (x,i)} \quad \text{in e_2 :: A_2} \quad \text{c-r-unpack1} \]

Figure 83: RelCostCore typing rules (Part 4)
Primitive application

\[
\Gamma(\zeta) = \frac{\text{exec}(k,t) \rightarrow A_2}{\Delta; \Phi_a; \Omega \vdash e : : A_1} \quad \text{c-primapp}
\]

\[
\Gamma(\zeta) = \frac{\text{diff}(t) \rightarrow \tau_2}{\Delta; \Phi_a; \Gamma \vdash e \otimes e' \leq t' : : \tau_1} \quad \text{c-r-primapp}
\]

\(\text{C} \& \tau\) intro. rules

\[
\frac{\Delta; \Phi_a \vdash C}{\Delta; \Phi_a \vdash C \& \tau; \Omega \vdash e : : A} \quad \text{c-c-andI}
\]

\[
\frac{\Delta; \Phi_a \vdash C \& \Gamma; \Gamma \vdash e \otimes e' \leq t : : \tau}{\Delta; \Phi_a; \Gamma \vdash e \otimes e' \leq t : : \tau} \quad \text{c-c-andI}
\]

\(\text{C} \& \tau\) elim. rules

\[
\frac{\Delta; \Phi_a; \Omega \vdash e_1 : : C \& A_1 \quad \Delta; \Phi_a \vdash C \& C; x : : A_1, \Omega \vdash e_2 : : A_2}{\Delta; \Phi_a; \Omega \vdash e_1 \otimes e_2 \text{ clet } e_1 \text{ as } x \in e_2 : : A_2} \quad \text{c-c-andE}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e_1 \otimes e_1' \leq t_1 : : C \& \tau_1 \quad \Delta; \Phi_a \vdash C \& \Gamma; x : : \tau_1, \Gamma \vdash e_2 \otimes e_2' \leq t_2 : : \tau_2}{\Delta; \Phi_a; \Gamma \vdash \text{ clet } e_1 \text{ as } x \in e_2 \otimes \text{ clet } e_1' \text{ as } x \in e_1' \leq t_1 + t_2 : : \tau_2} \quad \text{c-r-c-andE}
\]

\(\text{C} \supset \tau\) intro. rules

\[
\frac{\Delta; \Phi_a \vdash C \& \tau; \Omega \vdash e : : A}{\Delta; \Phi_a \vdash C \supset A} \quad \text{c-c-impI}
\]

\[
\frac{\Delta; \Phi_a \vdash C \& \Gamma; \Gamma \vdash e \otimes e' \leq t : : \tau}{\Delta; \Phi_a \vdash C \supset \tau} \quad \text{c-r-c-impI}
\]

\(\text{C} \supset \tau\) elim. rules

\[
\frac{\Delta; \Phi_a; \Omega \vdash e : : C \supset A}{\Delta; \Phi_a \vdash C \supset A} \quad \text{c-c-implE}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \otimes e' \leq t : : C \supset \tau \quad \Delta; \Phi_a \vdash C}{\Delta; \Phi_a; \Gamma \vdash \text{ celim } e \otimes \text{ celim } e' \leq t : : \tau} \quad \text{c-r-c-implE}
\]

let \(x = e_1\) in \(e_2\)

\[
\frac{\Delta; \Phi_a; \Omega \vdash e_1 : : A_1 \quad \Delta; \Phi_a; x : : A_1, \Omega \vdash e_2 : : A_2}{\Delta; \Phi_a; \Omega \vdash e_1 \text{ clet } e_1 \text{ let } x = e_1 \text{ in } e_2 : : A_2} \quad \text{c-let}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \otimes e_1' \leq t_1 : : \tau_1 \quad \Delta; \Phi_a; x : : \tau_1, \Gamma \vdash e_2 \otimes e_2' \leq t_2 : : \tau_2}{\Delta; \Phi_a; \Gamma \vdash \text{ let } x = e_1 \text{ in } e_2 \otimes \text{ let } x = e_1' \text{ in } e_1' \leq t_1 + t_2 : : \tau_2} \quad \text{c-r-let1}
\]

Figure 84: RelCostCore typing rules (Part 5)
Unary Subtyping

\[
\frac{\Delta; \Phi_a; \Omega \vdash^t_k e : \tau}{\Delta; \Phi_a \vdash k \leq k'}\quad \frac{\Delta; \Phi_a \vdash A \sqsupset A'}{\Delta; \Phi_a; \Omega \vdash^t_{k'} e : \tau'} \quad \text{c- } \sqsupset
\]

Binary Subeffecting

\[
\frac{\Delta; \Phi_a; \Gamma \vdash e \circ e' \leq t : \tau \quad \Delta; \Phi_a \vdash \tau \equiv \tau'}{\Delta; \Phi_a \vdash t \leq t'} \quad \text{c-r- } \equiv
\]

Constraint dependent typing

\[
\frac{\Delta; \Phi_a \land C; \Omega \vdash^t_k e_1 : \tau}{\Delta; \Phi_a \land \neg C; \Omega \vdash^t_k e_2 : \tau} \quad \text{c-split}
\]

\[
\frac{\Delta; \Phi_a \land C; \Gamma \vdash e_1 \circ e_2 \leq t : \tau \quad \Delta; \Phi_a \land \neg C; \Gamma \vdash e'_1 \circ e'_2 \leq t : \tau}{\Delta; \Phi_a; \Gamma \vdash \text{split} (e_1, e_2) \text{ with } C \circ \text{split} (e'_1, e'_2) \text{ with } C \land t \leq \tau} \quad \text{c-r-split}
\]

\[
\frac{\Delta; \Phi_a \vdash \bot}{\Delta; \Phi_a; \Omega \vdash \text{contra } e : \tau} \quad \text{c-contra}
\]

Asynchronous typing

\[
\frac{\Delta; \Phi_a; \Gamma \vdash^t_k e_1 : A_1}{\Delta; \Phi_a; x : U (A_1, A_1), \Gamma \vdash e \circ e \leq t_2 : \tau_2} \quad \frac{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 \circ e \leq t_2 + c_{\text{let}} : \tau_2}{\Delta; \Phi_a; \Gamma \vdash e \circ \text{let } x = e_1 \text{ in } e_2 \leq t_2 - k_1 - c_{\text{let}} : \tau_2} \quad \text{c-r-e-let}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash^t_k e_1 : A_1}{\Delta; \Phi_a; x : U (A_1, A_1), \Gamma \vdash e \circ e_2 \leq t_2 : \tau_2} \quad \frac{\exec(k, t)}{\Delta; \Phi_a; \Gamma \vdash \text{exec}(k, t)} \quad \frac{\Delta; \Phi_a; \Gamma \vdash e \circ e_2 \leq t_2 : \tau_2}{\Delta; \Phi_a; \Gamma \vdash e \circ \text{exec}(k, t) : \tau_2} \quad \text{c-r-app-e}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash^t_k e'_1 : A'_1}{\Delta; \Phi_a; \Gamma \vdash e'_2 \circ e'_2 \leq t_2 : \tau_2} \quad \frac{\exec(k, t)}{\Delta; \Phi_a; \Gamma \vdash \text{exec}(k, t)} \quad \frac{\Delta; \Phi_a; \Gamma \vdash e'_2 \circ e'_2 \leq t_2 : \tau_2}{\Delta; \Phi_a; \Gamma \vdash e'_2 \circ \text{exec}(k, t) : \tau_2} \quad \text{c-r-app-e}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash^t_k e : \tau}{\Delta; \Phi_a; x : U (A_1, A_1), \Gamma \vdash e_1 \circ e' \leq t : \tau \quad \Delta; \Phi_a; y : U (A_2, A_2), \Gamma \vdash e_2 \circ e' \leq t' : \tau} \quad \frac{\Gamma \vdash \text{case } (e, x, e_1, y, e_2) \circ e' \leq t' + c_{\text{case}} : \tau}{\Delta; \Phi_a; \Gamma \vdash \text{case } (e, x, e_1, y, e_2) \leq t' - k' - c_{\text{case}} : \tau} \quad \text{c-r-case-e}
\]

\[
\frac{\Delta; \Phi_a; \Gamma \vdash^t_k e : \tau}{\Delta; \Phi_a; x : U (A_1, A_1), \Gamma \vdash e \circ e' \leq t : \tau \quad \Delta; \Phi_a; y : U (A_2, A_2), \Gamma \vdash e \circ e' \leq t : \tau} \quad \frac{\Gamma \vdash \text{case } (e, x, e_1, y, e_2) \leq t' - c_{\text{case}} : \tau}{\Delta; \Phi_a; \Gamma \vdash e \circ \text{case } (e, x, e_1, y, e_2) \leq t - k' - c_{\text{case}} : \tau} \quad \text{c-r-case-e}
\]

Figure 85: RelCostCore typing rules (Part 6)
\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_2 \downarrow \tau, t \Rightarrow \Phi \] Under the existential variable context \( \psi_a \) and the assumption \( \Phi_a \), \( e_1 \circ e_2 \) checks against the input type \( \tau \) and the difference cost \( t \). Finally, it generates the constraint \( \Phi \).

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_2 \uparrow \tau \Rightarrow [\psi], t, \Phi \] Under the existential variable context \( \psi_a \) and the assumption \( \Phi_a \), \( e_1 \circ e_2 \) synthesizes the output type \( \tau \) and the relative cost \( t \) where all the newly generated existential variables are defined in \( \psi \). Finally, it generates the constraint \( \Phi \).

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k, t \Rightarrow \Phi \] Under the existential variable context \( \psi_a \) and the assumption \( \Phi_a \), \( e \) checks against the unary input type \( A \) and the minimum execution cost \( k \) and maximum execution cost \( t \). Finally, it generates the constraint \( \Phi \).

\[ \text{switch } e \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \uparrow A \Rightarrow [\psi], \tau_1, \Phi_1 \quad \Delta; \psi_a; \Phi_a; \Gamma \vdash e_2 \uparrow A \Rightarrow [\psi], k_2, \Phi_2 \quad \text{alg-r-switch}^\uparrow \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash \text{switch } e_1 \odot \text{switch } e_2 \uparrow U A \Rightarrow [\psi], \psi_2, t_1 - k_2, \Phi_1 \land \Phi_2 \quad \text{alg-r-switch}^\downarrow \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \circ e_2 \downarrow t, U A \Rightarrow k_1, k_1 \in \text{fresh}(R) \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \uparrow A, k_1, t_1 \Rightarrow \Phi_1 \quad \Delta; \psi_a; \Phi_a; \Gamma \vdash e_2 \downarrow A, k_2, t_2 \Rightarrow \Phi_2 \quad \text{alg-r-switch}^\downarrow \]

\[ \text{NC } e \]

\[ t' \in \text{fresh}(R) \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e \odot e \downarrow \tau, t' \Rightarrow \Phi \quad \text{alg-r-nochange}^\downarrow \]

\[ \text{der } e \]

\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \odot \text{der } e_2 \uparrow \tau \Rightarrow [\psi], t, \Phi \quad \text{alg-r-der}^\uparrow \]

Constant Integers \( n \) and unit

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash n \uparrow \text{int} \Rightarrow [\text{int}], 0, 1 \quad \text{alg-u-n}^\uparrow \quad \Delta; \psi_a; \Phi_a; \Gamma \vdash n \circ n \uparrow \text{int}_r \Rightarrow [\text{int}_r], 0, 1 \quad \text{alg-r-n}^\uparrow \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash () \uparrow \text{unit} \Rightarrow [\text{unit}], 0, 1 \quad \text{alg-u-unit}^\uparrow \quad \Delta; \psi_a; \Phi_a; \Gamma \vdash () \odot n \uparrow \text{unit}_r \Rightarrow [\text{unit}_r], 0, 1 \quad \text{alg-r-unit}^\uparrow \]

Variables \( x \)

\[ \Omega(x) = A \quad \Delta; \psi_a; \Phi_a; \Omega \vdash x \uparrow A \Rightarrow [\text{unit}], 0, 1 \quad \text{alg-u-var}^\uparrow \]

\[ \Gamma(x) = \tau \quad \Delta; \psi_a; \Phi_a; \Gamma \vdash x \odot x \uparrow \tau \Rightarrow [\text{unit}], 0, 1 \quad \text{alg-r-var}^\uparrow \]

Figure 86: Algorithmic typing rules (part 1) 116
\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A_1, k, t \Rightarrow \Phi \quad \Delta; \psi_a; \Phi_a \vdash ^A A_2 \text{ wf} \]
\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{inl } e \downarrow A_1 + A_2, k, t \Rightarrow \Phi \quad \text{alg-u-inl} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \ominus e_2 \downarrow \tau_1, t \Rightarrow \Phi \quad \Delta; \psi_a; \Phi_a \vdash \tau_2 \text{ wf} \quad \text{alg-r-inl} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash \text{inr } e_1 \ominus \text{inr } e_2 \downarrow \tau_1 + \tau_2, t \Rightarrow \Phi \quad \text{alg-r-inr} \downarrow \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A_2, k, t \Rightarrow \Phi \quad \Delta; \psi_a; \Phi_a \vdash ^A A_1 \text{ wf} \]
\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{inr } e \downarrow A_1 + A_2, k, t \Rightarrow \Phi \quad \text{alg-u-inr} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash e_1 \ominus e_2 \downarrow \tau_2, t \Rightarrow \Phi \quad \Delta; \psi_a; \Phi_a \vdash \tau_1 \text{ wf} \quad \text{alg-r-inr} \downarrow \]

\[ \text{case } (e, x, e_1, y, e_2) \]
\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow A_1 + A_2 \Rightarrow [\psi], k_e, t_e, \Phi_1 \]
\[ k', t' \in \text{fresh}(\mathbb{R}) \quad \Delta; k', t', \psi, \psi_a; \Phi_a; \Omega, x : A_1 \vdash e_1 \downarrow A, k', t' \Rightarrow \Phi_2 \]
\[ \Delta; k', t', \psi, \psi_a; \Phi_a; \Omega, y : A_2 \vdash e_2 \downarrow A, k', t' \Rightarrow \Phi_3 \]
\[ \Phi = \exists (\psi). \Phi_1 \land (\exists k', t' :: \mathbb{R}. \Phi_2 \land \Phi_3 \land k \equiv k' + k_e + c_{\text{case}} \land t' + t_e + c_{\text{case}} = t) \]
\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{case } (e, x, e_1, y, e_2) \downarrow A, k, t \Rightarrow \Phi \quad \text{alg-u-case} \downarrow \]
\[ t' \in \text{fresh}(\mathbb{R}) \quad \Delta; t', \psi, \psi_a; \Phi_a; \Gamma, x : \tau_1 \vdash e_1 \ominus e'_1 \downarrow \tau, t' \Rightarrow \Phi_2 \]
\[ \Delta; t', \psi, \psi_a; \Phi_a; \Gamma, y : \tau_2 \vdash e_2 \ominus e'_2 \downarrow \tau, t' \Rightarrow \Phi_3 \quad \Phi = \exists (\psi). \Phi_1 \land (\exists t' :: \mathbb{R}. \Phi_2 \land \Phi_3 \land (t' + t_e = t)) \]
\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash \text{case } (e, x, e_1, y, e_2) \ominus \text{case } (e', x, e_1', y, e_2') \downarrow \tau, t \Rightarrow \Phi \quad \text{alg-r-case} \downarrow \]

\[ \text{fix } f(x).e \]
\[ \Delta; \psi_a; \Phi_a; f : A_1 \xrightarrow{\text{exec}} A_2, x : A_1, \Omega \vdash e \downarrow A_2, k', t' \Rightarrow \Phi \quad \text{alg-u-fix} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{fix } f(x).e \downarrow A_1 \xrightarrow{\text{exec}} A_2, k, t \Rightarrow \Phi \land k \equiv 0 \land 0 \equiv t \quad \text{alg-r-fix} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; f : \tau_1 \xrightarrow{\text{diff}} \tau_2, x : \tau_1, \Gamma \vdash e \ominus e' \downarrow \tau_2, t' \Rightarrow \Phi \quad \text{alg-r-fix} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; \Gamma \vdash \text{fix } f(x).e \ominus \text{fix } f(x).e' \downarrow \tau_1 \xrightarrow{\text{diff}} \tau_2, t \Rightarrow \Phi \land 0 \equiv t \quad \text{alg-r-fix} \downarrow \]
\[ \Delta; \psi_a; \Phi_a; f : \square (\tau_1 \xrightarrow{\text{diff}} \tau_2), x : \tau_1, \square \Gamma \vdash e \ominus e' \downarrow \tau_2, t' \Rightarrow \Phi \quad \square \text{alg-r-fix} \downarrow \]

Figure 87: Algorithmic typing rules (part 2)
Figure 88: Algorithmic typing rules (part 3)
case $e$ of nil $\rightarrow e_1$ | $h :: tl \rightarrow e_2$

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow \text{list}[n] A \Rightarrow \lfloor \psi \rfloor, k_1, t_1, \Phi \]

\[ k_2, t_2 \in \text{fresh}(\mathbb{R}) \quad \Delta; k_2, t_2, \psi, \psi_a; n = 0 \land \Phi_a; \Omega \vdash e_1 \downarrow A', k_2, t_2 \Rightarrow \Phi_2 \]

\[ i \in \text{fresh}(\mathbb{N}) \quad i :: \text{NC}; k_2, t_2, \psi, \psi_a; n = i + 1 \land \Phi_a; h :: A, tl \vdash \text{list}[i] A, \Omega \vdash e_2 \downarrow A', k_2, t_2 \Rightarrow \Phi_3 \]

\[ \Phi_{\text{body}} = (n \equiv 0 \rightarrow \Phi_2) \land \forall i :: \mathbb{N} \ldotp (n \equiv i + 1) \rightarrow (\Phi_3) \land k \equiv (k_1 + k_2 + c_{\text{caseL}}) \land (t_1 + t_2 + c_{\text{caseL}}) \equiv t \]

\[ \text{alg-u-caseL} \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \begin{cases} 1 & \text{case } e \text{ of nil } \rightarrow e_1 \end{cases} \]

\[ \begin{cases} 1 & \text{case } e' \text{ of nil } \rightarrow e'_1 \end{cases} \]

\[ \text{alg-r-caseL} \]

\[ \tau \vdash \text{list}[n] A \Rightarrow \lfloor \psi \rfloor, t_1, \Phi \]

\[ t_2 \in \text{fresh}(\mathbb{R}) \quad \Delta; t_2, \psi, \psi_a; n = i + 1 \land \Phi_a; h :: \Box \tau, tl \vdash \text{list}[i] \tau, \Gamma \vdash e_2 \downarrow e'_2 \downarrow \tau', t_2 \Rightarrow \Phi_1 \]

\[ i :: \mathbb{N}, \Delta; t_2, \psi, \psi_a; n = i + 1 \land \alpha \equiv \beta + 1 \land \Phi_a; h :: \tau, tl :: \text{list}[i] \tau, \Gamma \vdash e_3 \downarrow e'_3 \downarrow \tau', t_2 \Rightarrow \Phi_3 \]

\[ \Phi_{\text{body}} = (n \equiv 0 \rightarrow \Phi_1) \land \forall i :: \mathbb{N} \ldotp (n \equiv i + 1) \rightarrow (\Phi_2 \land \forall \beta :: \mathbb{N} \ldotp (\alpha \equiv \beta + 1) \rightarrow \Phi_3)) \land t_1 + t_2 + t \]

\[ \text{alg-lam} \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k_e, t_e \Rightarrow \Phi \]

\[ i :: \mathbb{S}, \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k_e, t_e \Rightarrow \Phi \]

\[ \text{alg-u-iLam} \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \Lambda i.e \downarrow \forall i \text{exec}(k_e, t_e) \vdash S, A, k, t \Rightarrow (\forall i :: S, \Phi) \land k \equiv 0 \land t \equiv \top \]

\[ \text{alg-r-iLam} \]

\[ \begin{cases} 1 & \text{case } e \text{ of nil } \rightarrow e_1 \end{cases} \]

\[ \begin{cases} 1 & \text{case } e' \text{ of nil } \rightarrow e'_1 \end{cases} \]

\[ \text{alg-lam} \]

\[ \tau \vdash \text{list}[n] A \Rightarrow \lfloor \psi \rfloor, k, t, \Phi \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A', (I/i), \tau \Rightarrow \forall i :: S, \Phi \]

\[ \text{alg-u-iApp} \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A', (I/i), \tau \Rightarrow \forall i :: S, \Phi \]

\[ \text{alg-r-iApp} \]

\[ \text{pack with } I \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A(I/i), k, t \Rightarrow \Phi \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{pack } e \text{ with } I \downarrow \exists i :: S, A, k, t \Rightarrow \Phi \]

\[ \text{alg-u-pack} \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow \tau(I/i), t \Rightarrow \Phi \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{pack } e \text{ with } I \downarrow \exists i :: S, \tau, t \Rightarrow \Phi \]

\[ \text{alg-r-pack} \]

Figure 89: Algorithmic typing rules (part 4)
unpack e as \((x, i)\) in \(e'\)

\[
\begin{align*}
\Delta; \psi; \Phi; \Omega \vdash e_1 \uparrow \exists! S. A_1 \Rightarrow [\psi], k_1, t_1, \Phi_1 & \quad k_2, t_2 \in \text{fresh}(R) \\
i \vdash S, \Delta; k_2, t_2, \psi, \psi; \Phi; x : A_1, \Omega \vdash e_2 \downarrow A_2, k_2, t_2 \Rightarrow \Phi_2 & \quad \i \notin FV(\Phi_0, \Omega, A_2, k_2, t_2) \\
\Phi = \exists(\psi). (\Phi_1 \land \exists k_2, t_2 :: R, \forall i :: S. \Phi_2 \land k \Rightarrow k_1 + k_2 + \text{unpack} \land t_1 + t_2 + \text{unpack} \Rightarrow t) \\
\Delta; \psi; \Phi; \Omega \vdash \text{unpack} e_1 \text{ as } (x, i) \text{ in } e_2 \downarrow A_2, k, t \Rightarrow \Phi & \quad \text{alg-u-unpack-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi; \Gamma \vdash e_1 \text{ as } (x, i) \text{ in } e_2 \downarrow A_2, k, t \Rightarrow \Phi & \quad \text{alg-r-unpack-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi_0; \Gamma \vdash \text{unpack} e_1 \text{ as } (x, i) \text{ in } e_2 \downarrow A_2, k, t \Rightarrow \Phi & \quad \text{alg-u-unpack-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi_0; \Gamma \vdash \text{unpack} e_1 \text{ as } (x, i) \text{ in } e_2 \downarrow A_2, k, t \Rightarrow \Phi & \quad \text{alg-r-unpack-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi \land C \vdash e \downarrow A, k, t \Rightarrow \Phi & \quad \text{alg-u-andI-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi_0; \Gamma \vdash e \downarrow k, t, C \land A \Rightarrow C \land (C \Rightarrow \Phi) & \quad \text{alg-r-andI-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi_0; \Gamma \vdash e_1 \downarrow \tau, t \Rightarrow \Phi & \quad \text{alg-u-andE-\downarrow}
\end{align*}
\]

\[
\begin{align*}
\Delta; \psi; \Phi_0; \Gamma \vdash \text{clet} e_1 \text{ as } x \text{ in } e_2 & \quad \text{alg-r-andE-\downarrow}
\end{align*}
\]

Figure 90: Algorithmic typing rules (part 5)
\[ \Delta; \Phi \land C; \Omega \vdash e \downarrow A, k, t \Rightarrow \Phi \]
\[ \Delta; \psi_0; \Phi_a; \Omega \vdash e \downarrow C \supset A, k, t \Rightarrow C \rightarrow \Phi \] alg-u-c-impl ↓
\[ \Delta; \Phi \land C; \Gamma \vdash e \land e' \downarrow \tau, t \Rightarrow \Phi \]
\[ \Delta; \psi_0; \Phi_a; \Gamma \vdash e \land e' \downarrow C \supset \tau, t \Rightarrow C \rightarrow \Phi \] alg-r-c-impl ↓

**celim e**

\[ \Delta; \psi_0; \Phi_a; \Omega \vdash e \uparrow C \supset A \Rightarrow [\psi], k, t, \Phi \]
\[ \Delta; \psi_0; \Phi_a; \Omega \vdash \text{celim } e \uparrow A \Rightarrow [\psi], k, t, C \land \Phi \] alg-u-c-implE ↑
\[ \Delta; \psi_0; \Phi_a; \Gamma \vdash e \land e' \uparrow C \supset \tau \Rightarrow [\psi], t, \Phi \]
\[ \Delta; \psi_0; \Phi_a; \Gamma \vdash \text{celim } e \land e' \uparrow \tau \Rightarrow [\psi], t, C \land \Phi \] alg-r-c-implE ↑

**let x = e₁ in e₂**

\[ \Delta; \psi_0; \Phi_a; \Omega \vdash e_1 \uparrow A_1 \Rightarrow [\psi], k_1, t_1, \Phi_1 \]
\[ k_2, t_2 \in \text{fresh}(\mathbb{R}) \]
\[ \Delta; k_2, t_2, \psi_0; \psi_a; x : A_1, \Omega \vdash e_2 \downarrow A_2, k_2, t_2 \Rightarrow \Phi_2 \]
\[ \Phi_2' = \Phi_2 \land k \equiv (k_1 + k_2 + c_{let}) \land (t_1 + t_2 + c_{let}) \equiv t \]
\[ \Delta; \psi_0; \Phi_a; \Omega \vdash \text{let } x = e_1 \text{ in } e_2 \downarrow A_2, k, t \Rightarrow \exists(\psi).\Phi_1 \land \exists k_2, t_2 :: \mathbb{R}.\Phi_2 \]
alg-u-let-↓
\[ \Delta; \psi_0; \Phi_a; \Gamma \vdash e_1 \land e_1' \uparrow \tau_1 \Rightarrow [\psi], t_1, \Phi_1 \]
\[ t_2 \in \text{fresh}(\mathbb{R}) \]
\[ \Delta; t_2, \psi_0; \psi_a; x : \tau_1, \Gamma \vdash e_2 \land e_2' \downarrow \tau_2, t_2 \Rightarrow \Phi_2 \]
\[ \Delta; \psi_0; \Phi_a; \Gamma \vdash \text{let } x = e_1 \text{ in } e_2 \land e_2' \downarrow \tau_2, t \Rightarrow \exists(\psi).\Phi_1 \land \exists \tau_2 :: \mathbb{R}.\Phi_2 \land t_1 + t_2 = t \] alg-r-let-↓

**split e**

\[ \Delta; \psi_0; C \land \Phi_a; \Omega \vdash e_1 \downarrow A, k, t \Rightarrow \Phi_1 \]
\[ \Delta; \psi_0; \neg C \land \Phi_a; \Omega \vdash e_2 \downarrow A, k, t \Rightarrow \Phi_2 \]
\[ \Delta; \neg C \land \Phi_a; \Omega \vdash \text{split } (e_1, e_2) \text{ with } C \downarrow A, k, t \Rightarrow C \rightarrow \Phi_1 \land \neg C \rightarrow \Phi_2 \]
alg-u-split ↓
\[ \Delta; \psi_0; C \land \Phi_a; \Gamma \vdash e_1 \land e_1' \downarrow \tau, t \Rightarrow \Phi_1 \]
\[ \Delta; \psi_0; \neg C \land \Phi_a; \Gamma \vdash e_2 \land e_2' \downarrow \tau, t \Rightarrow \Phi_2 \]
\[ \Delta; \neg C \land \Phi_a; \Gamma \vdash \text{split } (e_1, e_2) \text{ with } C \supset \text{split } (e_1', e_2') \text{ with } C \supset \tau, t \Rightarrow C \rightarrow \Phi_1 \land \neg C \rightarrow \Phi_2 \]
alg-r-split ↓

**contra e**

\[ \Delta; \psi_0; \Phi_a \models \bot \]
\[ \Delta; \psi_0; \Phi_a; \Omega \vdash \text{contra } e \downarrow A, k, t \Rightarrow \top \] alg-u-contra ↓
\[ \Delta; \psi_0; \Phi_a \models \bot \]
\[ \Delta; \psi_0; \Phi_a; \Gamma \vdash \text{contra } e \land \text{contra } e' \downarrow \tau, t \Rightarrow \top \] alg-r-contra ↓

Figure 91: Algorithmic typing rules (part 6)
\[\Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow A' \Rightarrow [\psi], k', t', \Phi_1 \quad \Delta; \psi, \psi_a; \Phi_a \vdash A' \subseteq A \Rightarrow \Phi_2 \]
\[\Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k, t \Rightarrow \exists (\psi). (\Phi_1 \land \Phi_2 \land t' \leq t \land k \leq k') \]
\[\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \uparrow \tau' \Rightarrow [\psi], t', \Phi_1 \quad \Delta; \psi, \psi_a; \Phi_a \vdash \tau' \equiv \tau \Rightarrow \Phi_2 \]
\[\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \tau, t \Rightarrow \exists (\psi). (\Phi_1 \land \Phi_2 \land t' \leq t) \]

\((e : A, k, t)\)
\[\Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k, t \Rightarrow \Phi \quad \Delta; \psi_a \vdash A \subseteq A \Rightarrow \Phi \quad \text{FIV}(A, k, t) \in \Delta \]
\[\Delta; \psi_a; \Phi_a; \Omega \vdash (e : A, k, t) \uparrow A \Rightarrow [\psi], k, t, \Phi \]

\((e : \tau, t)\)
\[\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \quad \Delta; \psi_a \vdash \tau \subseteq A \Rightarrow \Phi \quad \text{FIV}(\tau, t) \in \Delta \]
\[\Delta; \psi_a; \Phi_a; \Gamma \vdash (e : \tau, t) \circ (e' : \tau, t) \uparrow \tau \Rightarrow [\psi], t, \Phi \]

Asynchronous rules
\[t_2 \in \text{fresh}(\mathbb{R}) \quad \Delta; t_2, \psi, \psi_a \vdash x : U (A_1, A_2) \Rightarrow \exists (\psi). (\Phi_1 \land \exists t_2 : \mathbb{R}. \Phi_2 \land t_1 + t_2 + c_{let} \equiv t) \]
\[\text{alg-u-anno-}\uparrow \]

\[t_2 \in \text{fresh}(\mathbb{R}) \quad \Delta; t_2, \psi, \psi_a \vdash x : U (A_1, A_2) \Rightarrow \exists (\psi). (\Phi_1 \land \exists t_2 : \mathbb{R}. \Phi_2 \land t_1 + t_2 + c_{app} \equiv t) \]
\[\text{alg-r-anno-}\uparrow \]

\[\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \quad \Delta; \psi_a \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \]
\[\text{alg-r-let-e-}\downarrow \]

\[\Delta; t_2, \psi, \psi_a \vdash x : U (A_1, A_2) \Rightarrow \exists (\psi). (\Phi_1 \land \exists t_2 : \mathbb{R}. \Phi_2 \land t_1 + t_2 + c_{let} \equiv t) \]
\[\text{alg-r-let-}\downarrow \]

\[\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \quad \Delta; \psi_a \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \]
\[\text{alg-r-app-e-}\downarrow \]

\[\text{alg-r-app-}\downarrow \]

\[\Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \quad \Delta; \psi_a \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \]
\[\text{alg-r-case-e-}\downarrow \]

\[\Delta; \psi_a; \Phi_a; \Gamma \vdash (e, x, e_1, y, e_2) \circ e' \downarrow \tau, t \Rightarrow \exists (\tau). (\Phi_1 \land (\exists t_2 : \mathbb{R}. \Phi_2 \land t_1 + t_2 + t + c_{case} \equiv t)) \]
\[\text{alg-r-e-case-}\downarrow \]

Figure 92: Algorithmic typing rules (part 7)
\[ \Delta; \psi_a; \Phi_a \models A_1 \subseteq A_2 \Rightarrow \Phi \] checks whether \( A_1 \) is subtype of \( A_2 \) and generates constraints \( \Phi \)

\[ \Delta; \psi_a; \Phi_a \models \tau_1 \equiv \tau_2 \Rightarrow \Phi \] checks whether \( \tau_1 \) is equivalent to \( \tau_2 \) and generates constraints \( \Phi \)

\begin{align*}
\Delta; \psi_a; \Phi_a \models \text{int}_r \equiv \text{int}_r \Rightarrow \top & \quad \text{alg-r-int} \\
\Delta; \psi_a; \Phi_a \models \text{unit}_r \equiv \text{unit}_r \Rightarrow \top & \quad \text{alg-r-unit} \\
\Delta; \psi_a; \Phi_a \models \tau_1 \equiv \tau'_1 \Rightarrow \Phi_1 & \quad \text{alg-r-fun} \\
\Delta; \psi_a; \Phi_a \models \text{diff}(\tau_1) \rightarrow \tau_2 \equiv \tau'_1 \rightarrow \tau'_2 \Rightarrow \Phi_1 \land \Phi_2 \land t \equiv t' \\
\Delta; \psi_a; \Phi_a \models \tau_2 \equiv \tau'_2 \Rightarrow \Phi_2 & \quad \text{alg-r-prod} \\
\Delta; \psi_a; \Phi_a \models \tau_1 + \tau_2 \equiv \tau'_1 + \tau'_2 \Rightarrow \Phi_1 \land \Phi_2 & \quad \text{alg-r-sum} \\
\Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi & \quad \text{alg-r-list} \\
\Delta; \psi_a; \Phi_a \models \text{list}[n]^\alpha \tau \equiv \text{list}[n'^\alpha] \tau' \Rightarrow \Phi \land n \equiv n' \land \alpha \equiv \alpha' & \quad \text{alg-r-nil} \\
\end{align*}

\[ \Delta; \psi_a; \Phi_a \models \forall i : \tau \equiv \tau' \Rightarrow \Phi & \quad \text{alg-r-nil} \]

\[ \Delta; \psi_a; \Phi_a \models \exists i : \tau \equiv \exists i : S \cdot \tau \Rightarrow \forall i : S \cdot \Phi & \quad \text{alg-r-nil} \]

\[ \Delta; \psi_a; \Phi_a \models \exists A_1 \subseteq A' \Rightarrow \Phi_1 \\
\Delta; \psi_a; \Phi_a \models \exists A_2 \subseteq A'' \Rightarrow \Phi_2 \\
\Delta; \psi_a; \Phi_a \models \exists A'_1 \subseteq A_1 \Rightarrow \Phi'_1 \\
\Delta; \psi_a; \Phi_a \models \exists A'_2 \subseteq A_2 \Rightarrow \Phi'_2 \\
\Delta; \psi_a; \Phi_a \models U(A_1, A_2) \equiv U(A'_1, A'_2) \Rightarrow \Phi_1 \land \Phi'_1 \land \Phi_2 \land \Phi'_2 & \quad \text{B-\Box} \]

\[ \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi & \quad \text{c-impl} \]

\[ \Delta; \psi_a; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi & \quad \text{c-prod} \]

Figure 93: Algorithmic type equivalence rules
\[ |.| : \text{Expression} \rightarrow \text{Expression} \]

\[
\begin{align*}
|n| & = n \\
|(\|)| & = () \\
|x| & = x \\
|\text{fix } f(x).e| & = \text{fix } f(x).|e| \\
|\text{fix}_{NC} f(x).e| & = \text{fix}_{NC} f(x).|e| \\
|e_1 e_2| & = |e_1| |e_2| \\
\vdots \\
|(e : A, k, t)| & = |e| \\
|(e : \tau, t)| & = |e| \\
\end{align*}
\]

Figure 94: Annotation erasure
5.1 Metatheory

Lemma 36 (Embedding of Binary Subtyping in RelCost)
If $\Delta; \Phi \models \tau \sqsubseteq \tau'$ then $\exists e \in \text{RelCostCore}$ such that $\Delta; \Phi; \cdot \vdash e \sqsubseteq 0 : \text{coerce}_{\tau, \tau'}$.

Proof. Proof is by induction on the subtyping derivation. We denote the witness $e$ of type $\tau \sqsubseteq \tau'$ as $\text{coerce}_{\tau, \tau'}$ for clarity.

Case $\Delta; \Phi \vdash \Delta: \Phi \vdash \tau_1 \sqsubseteq \tau_1$ (⋆) $\Delta; \Phi \vdash \tau_2 \sqsubseteq \tau_2$ (⋆)

Then, using these two statements and $\Delta; \Phi \vdash t \sqsubseteq t'$ with binary subeffecting rule (rule $c-r-$ in Figure 84), we can construct the following derivation where $e = \lambda x.\lambda y.\text{coerce}_{\tau_2, \tau_2'} (x (\text{coerce}_{\tau_1, \tau_1} y))$

Case $\Delta; \Phi \vdash \Delta: \Phi \vdash \text{unit}_r \sqsubseteq \square \text{unit}_r$

Then, we can immediately construct the derivation using the rule $c-nochange$ in Figure 80.

Case $\Delta; \Phi \vdash \Delta: \Phi \vdash \text{int}_r \sqsubseteq \square \text{int}_r$

Then, we can construct the derivation using the primitive function $\text{box}_{\int : \text{int}_r} : \text{int}_r \to \square \text{int}_r$

Case $\Delta; \Phi \vdash \Delta: \Phi \vdash \square (\text{int}, \text{int}) \sqsubseteq \text{int}_r$

Then, we can construct the derivation using the primitive function $\text{box}_{U : \square (\text{int}, \text{int})} : \square (\text{int}, \text{int}) \to \text{int}_r$

Case $\Delta; \Phi \vdash \Delta: \Phi \vdash \square \tau \sqsubseteq \tau$

Then, we can immediately construct the derivation using the rule $c-der$ in Figure 80.

Case $\Delta; \Phi \vdash \Delta: \Phi \vdash \square \square \tau \sqsubseteq \tau$

Then, we can immediately construct the derivation using the rule $c-nochange$ in Figure 80.
Case \(\Delta; \Phi_i \vdash \tau_1 \subseteq \tau_2(\ast)\) B-\(\Box\)

By IH on (\(\ast\)), \(\exists \coerce_{\tau_1, \tau_2}. \Delta; \Phi; \vdash \coerce_{\tau_1, \tau_2} \circ \coerce_{\tau_1, \tau_2} \not\subseteq 0 : c \tau_1 \xrightarrow{\text{diff}(0)} \tau_2\)

Then, using (\(\ast\)) and the rules c-der and c-nochange in Figure 80, we can construct the derivation

\[
\Delta; \Phi; \vdash \lambda x.\text{NC} (\coerce_{\tau_1, \tau_2} (\text{der} \ x)) \circ \lambda x.\text{NC} (\coerce_{\tau_1, \tau_2} (\text{der} \ x)) \not\subseteq 0 : c \Box \tau_1 \xrightarrow{\text{diff}(0)} \Box \tau_2
\]

Case \(\Delta; \Phi_a \vdash \tau \subseteq U |\tau|\) \(\text{W}\)

Then, we can immediately construct the derivation using the rule c-switch in Figure 80.

\[
\Delta; \Phi; \vdash \lambda x.\text{switch} \ x \circ \lambda x.\text{switch} \ x \not\subseteq 0 : c \tau \xrightarrow{\text{diff}(0)} U (|\tau_1|, |\tau_2|)
\]

Case \(\Delta; \Phi_a \vdash \tau \subseteq \tau_1 \ast \Delta; \Phi_a \vdash \tau_2 \subseteq \tau_3 (\ast)\) trans

By IH on (\(\ast\)), \(\exists \coerce_{\tau_1, \tau_2}. i :: S, \Delta; \Phi; \vdash \coerce_{\tau_1, \tau_2} \circ \coerce_{\tau_1, \tau_2} \not\subseteq 0 : c \tau_1 \xrightarrow{\text{diff}(0)} \tau_2\)

By IH on (\(\ast\)), \(\exists \coerce_{\tau_2, \tau_3}. i :: S, \Delta; \Phi; \vdash \coerce_{\tau_2, \tau_3} \circ \coerce_{\tau_2, \tau_3} \not\subseteq 0 : c \tau_2 \xrightarrow{\text{diff}(0)} \tau_3\)

Then, using (\(\ast\)) and (\(\ast\)), we can construct the derivation simply by function composition

\[
\Delta; \Phi; \vdash \lambda x.\coerce_{\tau_2, \tau_3} (\coerce_{\tau_1, \tau_2} \ x) \circ \lambda x.\coerce_{\tau_2, \tau_3} (\coerce_{\tau_1, \tau_2} \ x) \not\subseteq 0 : c \tau_1 \xrightarrow{\text{diff}(0)} \tau_3
\]

Case \(\Delta; \Phi_a \vdash \Box \tau_1 \xrightarrow{\text{diff}(k)} \tau_2 \not\subseteq \Box \tau_1 \xrightarrow{\text{diff}(0)} \Box \tau_2\) \(\text{exec}\)

Then, we can immediately construct the derivation where \(e = \lambda x.\lambda y.\text{NC} (\text{der} \ x) (\text{der} \ y)\)

\[
\Delta; \Phi; \vdash e \circ e \not\subseteq 0 : c \Box \tau_1 \xrightarrow{\text{diff}(k)} \tau_2 \xrightarrow{\text{diff}(0)} \Box \tau_1 \xrightarrow{\text{diff}(0)} \Box \tau_2
\]

Case \(\Delta; \Phi_a \vdash U (A_1 \xrightarrow{\text{exec}(k,t)} A_2) \not\subseteq U A_1 \xrightarrow{\text{diff}(t-k)} U A_2\) \(\text{exec}\)

Then, we can immediately construct the following derivation where \(e = \lambda x.\lambda y.\text{switch} \ (x \ y)\) using the c-switch and c-app rules.

\[
\Delta; \Phi; \vdash e \circ e \not\subseteq 0 : c (U (A_1 \xrightarrow{\text{exec}(k,t)} A_2, A_1' \xrightarrow{\text{exec}(k',t')} A_2')) \xrightarrow{\text{diff}(0)} U (A_1, A_1') \xrightarrow{\text{diff}(t-k')} U (A_2, A_2')
\]

Case \(i :: S, \Delta; \Phi_a \vdash \tau \subseteq \tau' (\ast)\) \(\Delta; \Phi_a \vdash i \leq t'\) \(i \not\in \text{FV}(\Phi_a)\) \(\forall\text{diff}\)

By IH on (\(\ast\)), \(\exists \coerce_{\tau, \tau'}. i :: S, \Delta; \Phi; \vdash \coerce_{\tau, \tau'} \circ \coerce_{\tau, \tau'} \not\subseteq 0 : c \tau \xrightarrow{\text{diff}(0)} \tau'
\]

126
Then, using this, the second premise and the c-r-iLam and c-r-iApp rules in RelCostCore, we can construct the following derivation:

\[
\Delta; \Phi \vdash \lambda x. \text{coerce}_{\tau, \tau'} (x \,[i]) \odot \lambda x. \text{coerce}_{\tau, \tau'} (x \,[i]) \leq 0 : c \otimes (\forall i) \vdash_i \tau \rightarrow \tau' \]

Case
\[
\Delta; \Phi \vdash \square \forall (\forall i) \vdash_i \tau \rightarrow \tau' \]

Then, we can immediately construct the following derivation using the c-der, c-nochange, c-r-iLam and c-r-iApp rules in Figures 80 and 83.

\[
\Delta; \Phi \vdash \lambda x. \text{NC} \ ((\text{der} \ x) \,[i]) \odot \lambda x. \text{NC} \ ((\text{der} \ x) \,[i]) \leq 0 : c \otimes (\forall i) \vdash_i \tau \rightarrow \tau' \]

Case
\[
\Delta; \Phi \vdash U(\forall i) \vdash_i \tau \rightarrow \tau' \]

Then, using these two statements and the rules c-prod and c-proj in Figure 81, we can show the following derivation where \(e = \lambda x. (\text{coerce}_{\tau_1, \tau_1'} (\pi_1 x), \text{coerce}_{\tau_2, \tau_2'} (\pi_2 x))\)

\[
\Delta; \Phi \vdash e \otimes e \leq 0 : c \forall (\forall i) \vdash_i \tau_1 \times \tau_2 \rightarrow \tau_1' \times \tau_2' \]

Case
\[
\Delta; \Phi \vdash \square \forall (\forall i) \vdash_i \tau_1 \times \tau_2 \rightarrow \tau_1' \times \tau_2' \]

We show the direction from right-to-left using the rules c-der, c-nochange, c-r-proj, c-r-let and c-r-prod in Figures 80, 81 and 83 where the expression \(e = \lambda x. \text{let} a = \pi_1 x \text{ in } b = \pi_2 x \text{ in } \text{NC} \ ((\text{der} \ a, \text{der} \ b))\).

\[
\Delta; \Phi \vdash e \otimes e \leq 0 : c \forall (\forall i) \vdash_i \tau_1 \times \tau_2 \rightarrow \tau_1 \times \tau_2 \]

Case
\[
\Delta; \Phi \vdash U (A_1 \times A_2, A_1' \times A_2' ) \subseteq U (A_1, A_1') \times U (A_2, A_2') \]

Then, we can immediately construct the following derivation where \(e = \lambda x. (\text{switch} \ \pi_1 x, \text{switch} \ \pi_2 x)\) using the c-switch, c-r-prod and c-r-proj rules in Figures 80 and 81.

\[
\Delta; \Phi \vdash e \otimes e \leq 0 : c \forall (\forall i) \vdash_i \tau_1 \times \tau_2 \rightarrow \tau_1 \times \tau_2 \]

Case
\[
\Delta; \Phi \vdash \square \forall (\forall i) \vdash_i \tau_1 \times \tau_2 \rightarrow \tau_1 \times \tau_2 \]

We can construct the following derivation by using the rules c-der, c-nochange, c-r-case, c-r-inl and c-r-inr in Figure 80 where the expression \(e = \lambda x. \text{case} \ (x, \text{a,NC} \ (\text{inl} \ a), \text{b,NC} \ (\text{inr} \ b))\).

\[
\Delta; \Phi \vdash e \otimes e \leq 0 : c \forall (\forall i) \vdash_i \tau_1 \times \tau_2 \rightarrow \tau_1 \times \tau_2 \]
Then, we can instantiate `fList` with a concrete

We first construct the more generic term for type:

\[
\Delta; \Phi \vdash \text{eqref}(5.1) \quad (\text{5.1})
\]

and then instantiate the term for eq. (5.1) later.

It can be shown that such a derivation can be constructed for expression

\[
e' = \text{fix } \lambda x. \text{fList (.).} \Delta_n.\Lambda \alpha. \Lambda \alpha'. \Lambda x, \text{clet } x \text{ as } e \text{ in case } e \text{ of}
\]

\[
\text{case } e \text{ of}
\]

\[
| h :: N \text{ tl } \rightarrow \text{let } r = \text{fList (.)} [n - 1] [n' - 1] [\alpha] [\alpha'] tl \text{ in cons}_{N_C}(NC(\text{coerce}_{\tau', \tau} \text{ der } h), r)
\]

\[
| h :: C \text{ tl } \rightarrow \text{let } r = \text{fList (.)} [n - 1] [n' - 1] [\alpha - 1] [\alpha' - 1] tl \text{ in cons}_{C}(\text{coerce}_{\tau', \tau} \text{ h, r))}
\]

Then, we can instantiate fList using (⋆) and (⋆) as follows where

\[
e'' = \lambda x. \text{fList (.)} [n][n'][\alpha][\alpha'] x
\]

\[
\Delta; \Phi \vdash e'' \rightarrow e'' \quad 0 : e \text{ list}[n][n'][\alpha][\alpha'] \rightarrow \text{list}[n'][n'][\alpha'][\alpha']
\]

We first construct the more generic term for type

\[
\Delta; \Phi \vdash \text{list}[n][n][\alpha][\alpha] \quad \text{(5.2)}
\]

and then instantiate the term for eq. (5.2) later. It can be shown that such a derivation can be constructed for expression

\[
e' = \text{fix } \lambda x. \text{fList (.).} \Delta_n.\Lambda \alpha \text{.} \Lambda x, \text{clet } x \text{ as } e \text{ in case } e \text{ of}
\]

\[
| \text{nil } \rightarrow \text{nil}
\]

\[
| h :: N \text{ tl } \rightarrow \text{let } r = \text{fList (.)} [n - 1] [\alpha] tl \text{ in } \text{NC}(\text{cons}_{N_C}(\text{der } h, \text{der } r))
\]

\[
| h :: C \text{ tl } \rightarrow \text{let } r = \text{fList (.)} [n - 1] [\alpha - 1] tl \text{ in } \text{NC}(\text{cons}_{C}(\text{der } h, \text{der } r))
\]

Then, we can instantiate fList with a concrete `n` and `α` as follows where

\[
e'' = \lambda x. \text{fList (.)} [n][\alpha] x
\]

\[
\Delta; \Phi \vdash e'' \rightarrow e'' \quad 0 : e \text{ list}[n][\alpha] \rightarrow \text{list}[n][\alpha]
\]

We first construct the more generic term for type

\[
\Delta; \Phi \vdash \alpha = 0 \quad \text{(5.3)}
\]

and then instantiate the term for eq. (5.3) later. It can be shown that such a derivation can be constructed for expression

\[
e' = \text{fix } \lambda x. \text{fList (.).} \Delta_n.\Lambda \alpha \text{.} \Lambda x \text{ clet } x \text{ as } e \text{ in case } e \text{ of}
\]

\[
| \text{nil } \rightarrow \text{nil}
\]

\[
| h :: N \text{ tl } \rightarrow \text{let } r = \text{fList (.)} [n - 1] [\alpha] tl \text{ in } \text{cons}_{N_C}(\text{NC} h, r)
\]

\[
| h :: C \text{ tl } \rightarrow \text{contra}
\]

11
Then, we can instantiate fList with a concrete \( n \) and \( \alpha \) (note the premise \( \alpha = 0 \)) as follows where \( e'' = \lambda x.\text{fList}()\ [n][\alpha] \ x \)

\[
\Delta; \Phi; \vdash e'' \circ e'' \triangleleft 0 : \text{c-der}[n]0 \rel \text{list}[n]0 \ \tau \ \text{diff}(0) \rightarrow \text{list}[n]0 \ \square \tau
\]

**Case**

\[
i :: S, \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\star) \quad i \notin \text{FV}(\Phi_a)
\]

By IH on (\( \star \)), \( \exists \text{coerce}_{\tau,\tau'} \cdot i :: S, \Delta; \Phi; \vdash \text{coerce}_{\tau,\tau'} \otimes \text{coerce}_{\tau,\tau'} \triangleleft 0 : \text{c-der}[n]0 \rel \tau' \quad (\diamond)
\]

Then, using this and the \textbf{c-r-pack} and \textbf{c-r-unpack} rules in \text{RelCostCore} in Figure 83, we can construct the following derivation where \( e = \lambda x.\text{unmap} \ x \) as \( (y, i) \) in pack \( \text{coerce}_{\tau,\tau'} \ y \) with \( i \)

\[
\Delta; \Phi; \vdash e \circ e \triangleleft 0 : \text{c-der}[n]0 \rel \exists i :: S. \tau' \quad (\nabla)
\]

**Case**

\[
\Delta; \Phi_a \models \exists i :: S. \tau \subseteq \exists i :: S. \tau'
\]

Then, we can immediately construct the following derivation using the \textbf{c-der}, \textbf{c-nochange}, \textbf{c-r-pack} and \textbf{c-r-unpack} rules in in Figures 80 and 83 where \( e = \lambda x.\text{unmap} \ x \) as \( (y, i) \) in \text{NC} (pack \text{der} \ y \ with \ i).

\[
\Delta; \Phi; \vdash e \circ e \triangleleft 0 : \text{c-der}[n]0 \rel \exists i :: S. \tau \quad (\nabla)
\]

**Case**

\[
\Delta; \Phi_a \models C \implies C' \quad (\star) \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\diamond)
\]

By IH on (\( \star \)), \( \exists \text{coerce}_{\tau,\tau'} \cdot \Delta; \Phi; \vdash \text{coerce}_{\tau,\tau'} \otimes \text{coerce}_{\tau,\tau'} \triangleleft 0 : \text{c-der}[n]0 \rel \tau' \quad (\star)
\]

Then, using this and the premise (\( \star \)) along with the \textbf{c-r-c-implII} and \textbf{c-r-c-implE} rules in Figure 84, we can construct the following derivation where \( e = \lambda x.\text{coerce}_{\tau,\tau'} \) (celim \( x \))

\[
\Delta; \Phi; \vdash e \circ e \triangleleft 0 : \text{c-der}[n]0 \rel C' \supset \tau' \quad (\star)
\]

**Case**

\[
\Delta; \Phi_a \models \square(C \supset \tau) \subseteq C \supset \square \tau \quad \textbf{c-impl-\square}
\]

Then, we can immediately construct the following derivation using the \textbf{c-der}, \textbf{c-nochange} and \textbf{c-r-c-implE} rules in \text{RelCostCore} where \( e = \lambda x.\text{NC} \) (celim \( \text{der} \ x \)).

\[
\Delta; \Phi; \vdash e \circ e \triangleleft 0 : \text{c-der}[n]0 \rel \square(C \supset \tau) \quad (\star)
\]

**Case**

\[
\Delta; \Phi_a \models C \implies C' \quad (\star) \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\diamond)
\]

By IH on (\( \star \)), \( \exists \text{coerce}_{\tau,\tau'} \cdot \Delta; \Phi; \vdash \text{coerce}_{\tau,\tau'} \otimes \text{coerce}_{\tau,\tau'} \triangleleft 0 : \text{c-der}[n]0 \rel \tau' \quad (\star)
\]

Then, using this and the premise (\( \star \)) along with the \textbf{c-r-c-prodII} and \textbf{c-r-c-prodE} rules in Figure 81, we can construct the following derivation where \( e = \lambda x.\text{celim} \ x \) as \( y \) in \text{coerce}_{\tau,\tau'} \ y
\[ \Delta; \Phi_1 \vdash e \odot e \triangleleft 0 : (C \& \tau) \xrightarrow{\text{diff}(0)} C' \& \tau' \]

**Case** \[ \Delta; \Phi \models C \& \Box \tau \subseteq \Box (C \& \tau) \]

Then, we can immediately construct the following derivation using the \textit{c-der}, \textit{c-nochange}, \textit{c-r-c-prodI} and \textit{c-r-c-prodE} rules in Figures 80, 81 and 83 where \( e = \lambda x. \text{clet} x \) as \( y \) in \( \text{NC (der y)} \).

\[ \Delta; \Phi_1 \vdash e \odot e \triangleleft 0 : (C \& \Box \tau) \xrightarrow{\text{diff}(0)} \Box (C \& \tau) \]

**Lemma 37 (Reflexivity of Algorithmic Binary Type Equivalence in RelCost)**

\[ \Delta; \psi_1; \Phi_a \models \tau \equiv \tau \Rightarrow \Phi \text{ and } \Delta; \psi_1; \Phi_a \models \Phi. \]

**Proof.** By induction on the binary type.

**Lemma 38 (Reflexivity of Unary Algorithmic Subtyping in RelCost)**

\[ \Delta; \Phi_a \models^A A \subseteq A \Rightarrow \Phi \text{ and } \Delta; \Phi_a \models \Phi. \]

**Proof.** By induction on the unary type.

**Lemma 39 (Transitivity of Unary Algorithmic Subtyping in RelCost)**

If \( \Delta; \Phi_a \models^A A_1 \subseteq A_2 \Rightarrow \Phi_1 \) and \( \Delta; \Phi_a \models^A A_2 \subseteq A_3 \Rightarrow \Phi_2 \) and \( \Delta; \Phi_a \models \Phi_1 \land \Phi_2 \), then \( \Delta; \Phi_a \models^A A_1 \subseteq A_3 \Rightarrow \Phi_3 \) for some \( \Phi_3 \) such that \( \Delta; \Phi_a \models \Phi_3 \).

**Proof.** By induction on the first subtyping derivation.

**Theorem 40 (Soundness of the Algorithmic Unary Subtyping in RelCost)**

Assume that

1. \( \Delta; \psi_1; \Phi_a \models^A A' \subseteq A \Rightarrow \Phi \)
2. \( \text{FIV}(\Phi_a, A, A') \subseteq \Delta, \psi_a \)
3. \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) is provable s.t \( \Delta \btriangleright \theta_a : \psi_a \) is derivable.

Then \( \Delta; \Phi_a[\theta_a] \models^A A'[\theta_a] \subseteq A[\theta_a] \).

**Proof.** By induction on the algorithmic unary subtyping derivation.

**Theorem 41 (Completeness of the Unary Algorithmic Subtyping in RelCost)**

Assume that \( \Delta; \Phi_a \models^A A' \subseteq A \). Then \( \exists \Phi \) such that \( \Delta; \Phi_a \models^A A' \subseteq A \Rightarrow \Phi \) and \( \Delta; \Phi_a \models \Phi \).

**Proof.** By induction on the unary subtyping derivation.

**Theorem 42 (Soundness of the Algorithmic Binary Type Equality in RelCost)**

Assume that

1. \( \Delta; \psi_1; \Phi_a \models \tau' \equiv \tau \Rightarrow \Phi \)
2. \( \text{FIV}(\Phi_a, \tau, \tau') \subseteq \Delta, \psi_a \)
3. \( \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \) is provable s.t \( \Delta \btriangleright \theta_a : \psi_a \) is derivable.

Then \( \Delta; \Phi_a[\theta_a] \models \tau'[\theta_a] \equiv \tau[\theta_a] \).

**Proof.** By induction on the algorithmic binary type equivalence derivation.
Theorem 43 (Completeness of the Binary Algorithmic Type Equivalence in RelCost)
Assume that $\Delta; \Phi_a \vdash \tau' \equiv \tau$. Then $\exists \Phi$, such that $\Delta; \Phi_a \vdash \tau' \equiv \tau \Rightarrow \Phi$ and $\Delta; \Phi_a \vdash \Phi$.

*Proof.* By induction on the binary subtyping derivation. \hfill \blacksquare

Theorem 44 (Soundness of RelCostCore & Type Preservation of Embedding)
The following holds.

1. If $\Delta; \Phi; \Omega \vdash e \leadsto e^* : A$, then $\Delta; \Phi; \Omega \vdash \tau^* \leadsto A$ and $\Delta; \Phi; \Omega \vdash e : A$.
2. If $\Delta; \Phi; \Gamma \vdash e_1 \ominus e_2 \leadsto e_1^* \ominus e_2^* \triangleleft t : \tau$, then $\Delta; \Phi; \Gamma \vdash e_1 \ominus e_2 \leq t : \tau$.

*Proof.* Proof is by simultaneous induction on the embedding derivations. The proof follows from the embedding rules presented in Figures 80 to 85. We show a few representative cases.

**Case** 
\[
\begin{array}{l}
\Delta; \Phi_a; |\Gamma| \vdash \Sigma \ e_1 \leadsto e_1^* : A \quad (\ast) \\
\Delta; \Phi_a; |\Gamma| \vdash \Sigma \ e_2 \leadsto e_2^* : A \quad (\circ)
\end{array}
\]
\[\Delta; \Phi_a; \Gamma \vdash e_1 \ominus e_2 \leadsto _{\text{switch}} e_1^* \ominus e_2^* \leq t_1 - t_2 : U A\]

By Theorem 44.1 on (\ast), we get $\Delta; \Phi; \Omega \vdash \Sigma \ e_1^* : A_1 \quad (**)$.

By Theorem 44.1 on (\circ), we get $\Delta; \Phi; \Omega \vdash e_2^* : A_2 \quad (\infty)$.

Then, we conclude as follows:
\[
\begin{array}{l}
\Delta; \Phi_a; |\Gamma| \vdash \Sigma \ e_1^* : A \\
\Delta; \Phi_a; |\Gamma| \vdash \Sigma \ e_2^* : A
\end{array}
\][e-switch]
\[\Delta; \Phi_a; \Gamma \vdash _{\text{switch}} e_1 \ominus e_2 \leq t_1 \ominus t_2 : U A\]

**Case** 
\[\Delta; \Phi_a; \Gamma \vdash e_1 \ominus e_2 \leadsto e_1^* \ominus e_2^* \leq t : \tau\]

By Theorem 44.2 on (\ast), $\Delta; \Phi_a; \Gamma \vdash e_1 \ominus e_2 \leadsto e_1^* \ominus e_2^* \leq t : \tau'$. By Theorem 36 using (\circ), we know that $\Delta; \Phi; \Gamma \vdash e_1 \ominus e_2 \leq 0 : \tau \Rightarrow (\infty)$. By applying **c-r-app** rule in Figure 82 tp (\ast\ast) and (\infty), we get $\Delta; \Phi; \Gamma \vdash e_1^* \ominus e_2^* \leq t : \tau' \quad (\spadesuit)$.

By reflexivity of binary type equivalence, we know $\Delta; \Phi_a \vdash \tau' \equiv \tau' \quad (\spadesuit\spadesuit)$.

Then, we conclude as follows:
\[\Delta; \Phi_a; \Gamma \vdash e_1^* \ominus e_2^* \leq t : \tau' \quad (\spadesuit)\]
\[\Delta; \Phi_a \vdash t \leq t' \quad (\dagger)\]
\[\Delta; \Phi_a; \Gamma \vdash e_1^* \ominus e_2^* \leq t' : \tau' \quad (\spadesuit\spadesuit)\]
\[\Delta; \Phi_a \vdash t \leq t' \quad (\dagger)\]
\[\Delta; \Phi_a; \Gamma \vdash e_1^* \ominus e_2^* \leq t' : \tau' \quad (\spadesuit\spadesuit)\]

**Theorem 45 (Completeness of RelCostCore)**
The following holds.

1. If $\Delta; \Phi; \Omega \vdash e : A$ then, $\exists e^*$ such that $\Delta; \Phi; \Omega \vdash e^* : A$.
2. If $\Delta; \Phi; \Gamma \vdash e_1 \ominus e_2 \leq t : \tau$ then, $\exists e_1^*$ and $\exists e_2^*$ such that $\Delta; \Phi; \Gamma \vdash e_1 \ominus e_2 \leadsto e_1^* \ominus e_2^* \leq t : \tau$.
Proof. Proof is by simultaneous induction on the typing derivations. The proof follows from the embedding rules presented in Figures 73-75. We show a few representative cases.

### Proof of Theorem 45.1:

\[
\frac{
\Delta; \Phi_a; \Omega \vdash k e : A \quad (\ast) \quad \Delta; \Phi_a \models A \subseteq A' \quad (\circ) \quad \Delta; \Phi_a \models k' \leq k \quad (\dagger) \quad \Delta; \Phi_a \models t \leq t' \quad \dagger\dagger}
{\Delta; \Phi_a; \Omega \vdash t e : A'}
\]

By Theorem 45.1 on (\ast), we get \( \exists e^* \) such that \( \Delta; \Phi; \Omega \vdash k e \leadsto e^* : A \) (\ast\ast).

By e-u-\( \theta \) rule using (\ast\ast), (\circ), (\dagger) and (\dagger\dagger), we conclude as follows:

\[
\frac{
\Delta; \Phi_a; \Omega \vdash k e \leadsto e^* : A \quad \Delta; \Phi_a \models A \subseteq A' \quad \Delta; \Phi_a \models k' \leq k \quad \Delta; \Phi_a \models t \leq t' \quad \dagger\dagger}
{\Delta; \Phi_a; \Omega \vdash t e \Rightarrow e^* : A'}
\]

### Proof of Theorem 45.2:

\[
\frac{
\Delta; \Phi_a; \| \vdash k_1 e_1 : A \quad (\ast) \quad \Delta; \Phi_a; \| \vdash k_2 e_2 : A \quad (\circ) \quad \Delta; \Phi_a; \varnothing \models \text{switch} \quad (\dagger)}
{\Delta; \Phi_a; \| \vdash \text{e-switch} \quad (\dagger\dagger)}
\]

By Theorem 45.1 on (\ast), we get \( \exists e^* \) such that \( \Delta; \Phi; \| \vdash k_1 e_1 \Rightarrow e^* : A \) (\ast\ast).

By Theorem 45.1 on (\circ), we get \( \exists e^*_2 \) such that \( \Delta; \Phi; \| \vdash k_2 e_2 \Rightarrow e^*_2 : A \) (\ast\ast\ast).

By e-switch embedding rule using (\ast\ast) and (\ast\ast\ast), we can conclude as follows:

\[
\frac{
\Delta; \Phi_a; \| \vdash k_1 e_1 \Rightarrow e^* : A \quad \Delta; \Phi_a; \| \vdash k_2 e_2 \Rightarrow e^*_2 : A \quad (\ast\ast\ast)}
{\Delta; \Phi_a; \| \models \text{e-switch} \quad (\ast\ast\ast\ast)}
\]

\[
\frac{
\forall x \in \text{dom}(\|). \quad \Delta; \Phi_a \models \| (x) \subseteq \| \text{e}(x) \quad (\circ)}
{\Delta; \Phi_a; \varnothing \models e \odot e \leq t : \tau \quad (\ast)}
\]

By Theorem 45.2 on (\circ), we get \( \exists e^* \) such that \( \Delta; \Phi; \varnothing \models e \odot e \Rightarrow e^* \odot e^* \leq t : \tau \) (\dagger).

By e-nochange embedding rule using (\dagger) and (\dagger\dagger), we can conclude as follows:

\[
\frac{
\Delta; \Phi_a; \| \models e \odot e \Rightarrow e^* \odot e^* \leq t : \tau \quad (\dagger) \quad \forall x \in \text{dom}(\|), \; e_i = \text{coerce}_{\|} (x_i, \| (x_i)) \quad (\dagger\dagger)}
{\Delta; \Phi_a; \| \models \text{e-nochange} \quad (\dagger\dagger\dagger)}
\]

\[
\frac{
\Delta; \Phi_a; \| \models e \odot e \Rightarrow e^* \odot e^* \leq t : \tau \quad (\ast)}
{\Delta; \Phi_a; \| \models \text{r-caseL} \quad (\ast\ast)}
\]

By Theorem 45.2 on (\ast\ast), we get \( \exists e^* \) and \( \exists e^{**} \) s.t. \( \Delta; \Phi; \| \models e \odot e^* \Rightarrow e^* \odot e^* \leq t : \text{list}[n]^\alpha \tau \) (\ast\ast\ast).
\[ i : S, \beta :: S, \Delta; n \equiv i + 1 \land \alpha \equiv \beta + 1 \land \Phi; h : \tau, \theta : \text{list}\langle i \rangle^\beta \tau, \Gamma \vdash e_2 \sqcup e_2' \leadsto e_2^* \sqcup e_2'^* \leq t : \tau' \quad (\triangleleft) \]

By **e-caseL** embedding rule using (**), (**∞), and (\triangleleft), we can conclude as follows:

\[
\begin{array}{llll}
\Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leadsto e^* \circ e'^* \leq t : \text{list}\langle n \rangle^\alpha \tau & \Delta ; \Phi_a \land n = 0 ; \Gamma \vdash e_1 \circ e'_1 \leadsto e_1^* \circ e'_1^* \leq t' : \tau' & \\
i, \Delta ; \Phi_a \land n = i + 1 ; h : \Box \tau, \theta : \text{list}\langle i \rangle^\alpha \tau, \Gamma \vdash e_2 \circ e'_2 \leadsto e_2^* \circ e'_2^* \leq t' : \tau' & i, \beta, \Delta ; \Phi_a \land n = i + 1 \land \alpha = \beta + 1 ; h : \tau, \theta : \text{list}\langle i \rangle^\beta \tau, \Gamma \vdash e_2 \circ e'_2 \leadsto e_2^* \circ e'_2^* \leq t' : \tau' & \\
\end{array}
\]

<table>
<thead>
<tr>
<th>case e of nil</th>
<th>case e of nil</th>
<th>case e' of nil</th>
<th>case e'' of nil</th>
</tr>
</thead>
</table>
| \(\Delta ; \Phi_a ; \Gamma \vdash e_1 \circ e'_1\) \quad \(\circ \) | \(h :: \text{NC} \) \(\to \) \(e_2^* \circ \) | \(h :: \text{NC} \) \(\to \) \(e_2^* \) \(\leq t + t' : \tau'\) | \(h :: \text{C} \) \(\to \) \(e_3^* \)
| \(\Delta ; \Phi_a ; \Gamma \vdash e_2 \circ e'_2\) \quad \(\circ \) | \(h :: \text{C} \) \(\to \) \(e_2' \) | \(h :: \text{C} \) \(\to \) \(e_3' \) |

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash \tau_1 & \quad \text{diff}(\tau_1) \quad \text{wf} \\
\Delta ; \Phi_a ; x : \tau_1, f : \Box (\tau_1 \text{diff}(\tau_2)), \Gamma \vdash e \circ e' \leq t : \tau_2 & \quad (\star) \\
\forall x \in \text{dom}(\Gamma), \ e_i = \text{coerce}_{\Gamma(x_i), \Box \Gamma(x_i)} & \quad \text{for all } x_i \in \text{dom}(\Gamma) \quad (\bigcirc)
\end{align*}
\]

By Theorem 45.2 on (\star), we get \(\exists e^*\) such that \(\Delta ; \Phi_a ; x : \tau_1, f : \Box (\tau_1 \text{diff}(\tau_2)), \Gamma \vdash e \circ e' \leadsto e^* \circ e^* \leq t : \tau_2 \quad (\star\star)\).

By Lemma 36 on (\circ), we get \(\exists e_i = \text{coerce}_{\Gamma(x_i), \Box \Gamma(x_i)}\) for all \(x_i \in \text{dom}(\Gamma) \quad (\bigcirc)\).

By **e-fixNC** embedding rule using (**\star\star\star**) and (**\bigcirc\bigcirc\bigcirc**), we can conclude as follows:

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash \tau_1 & \quad \text{diff}(\tau_1) \quad \text{wf} \\
\Delta ; \Phi_a ; x : \tau_1, f : \Box (\tau_1 \text{diff}(\tau_2)), \Gamma \vdash e \circ e' \leq t : \tau_2
\end{align*}
\]

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash e \circ f(x) \circ f(x) & \quad \text{eFixNC} \\
\forall x \in \text{dom}(\Gamma), \ e_i = \text{coerce}_{\Gamma(x_i), \Box \Gamma(x_i)} & \quad \text{e-fixNC.}
\end{align*}
\]

\[
\begin{array}{ll}
i ; S, \Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leq t : \tau & (\star) \\
i \notin \text{FIV}(\Phi_a ; \Gamma) & \\
\end{array}
\]

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash \Lambda e \circ \Lambda e' \leq 0 : \forall i \text{diff}(\tau) :: S, \tau \\
\end{align*}
\]

By Theorem 45.2 on (\star), we get \(\exists e^*\) such that \(i ; S, \Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leadsto e^* \circ e^* \leq t : \tau \quad (\star\star)\).

By **e-Lam** embedding rule using (**\star\star\star**) , we can conclude as follows:

\[
\begin{align*}
i ; S, \Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leadsto e^* \circ e^* \leq t : \tau & \quad (\star) \\
i \notin \text{FIV}(\Phi_a ; \Gamma) & \\
\end{align*}
\]

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash I : S & \quad (\bigcirc) \\
\end{align*}
\]

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash e[i] \circ e'[i] \leq t + t'[I/i] : \tau(I/i) & \\
\end{align*}
\]

By Theorem 45.2 on (\star), we get \(\exists e^*\) such that \(\Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leadsto e^* \circ e^* \leq t : \forall i \text{exec}(t', \tau) :: S \quad (\star\star)\).

By **e-Lam** embedding rule using (**\star\star\star**) and (\bigcirc), we can conclude as follows:

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leadsto e^* \circ e^* \leq t : \forall i \text{exec}(t', \tau) :: S, \tau \\
\Delta ; \Phi_a ; \Gamma \vdash I : S & \quad (\star) \\
\end{align*}
\]

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash e \circ e' \leq t : \tau(I/i) & \quad (\star) \\
\Delta ; \Phi_a ; \Gamma \vdash I : S & \quad (\bigcirc) \\
\end{align*}
\]

**Case**

\[
\begin{align*}
\Delta ; \Phi_a ; \Gamma \vdash e \circ pack e' \leq t : \exists i : S, \tau & \\
\end{align*}
\]

133
By Theorem 45.2 on (⋆), we get \( \exists e^* \) and \( \exists e'^* \) such that \( \Delta; \Phi; \Gamma \vdash e \circ e' \rightarrow e^* \circ e'^* \preceq t : \tau \{ I/i \} \quad (\star) \).

By \textbf{e-pack} embedding rule using (⋆) and (ο), we can conclude as follows:

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e \circ e' \rightarrow e^* \circ e'^* \preceq t : \tau \{ I/i \} & \quad \Delta \vdash I : S \\
\text{\textbf{e-pack}}
\end{align*}
\]

By Theorem 45.2 on (⋆), we get \( \exists e_1^* \) and \( \exists e_1'^* \) such that \( \Delta; \Phi; \Gamma \vdash e_1 \circ e_1' \rightarrow e_1^* \circ e_1'^* \preceq t : \exists i : S, \tau_1 \quad (\star) \).

By Theorem 45.2 on (ο), we get \( \exists e_2^* \) and \( \exists e_2'^* \) such that \( i :: S, \Delta; \Phi; x : \tau_1, \Gamma \vdash e_2 \circ e_2' \rightarrow e_2^* \circ e_2'^* \preceq t : \tau_2 \quad (\infty) \).

By \textbf{e-unpack} embedding rule using (⋆) and (∞), we can conclude as follows:

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e_1 \circ e_2 \preceq t : \tau \quad (\star) & \quad \Delta; \Phi_a \models \tau \subseteq \tau' \quad (\circ) \\
\text{\textbf{e-unpack}}
\end{align*}
\]

By Theorem 45.2 on (⋆), we get \( \exists e_1^* \) and \( \exists e_2^* \) such that \( \Delta; \Phi; \Gamma \vdash e_1 \circ e_2 \rightarrow e_1^* \circ e_2^* \preceq t : \tau \quad (\star) \).

By Lemma 36 on (ο), we can show that \( \exists e' = \text{coerce}_{\tau, \tau'} \quad (\infty) \).

By \textbf{e-r-\text{\textasciitilde}} rule using (⋆), (∞) and (†), we can conclude as follows

\[
\begin{align*}
\Delta; \Phi_a; \Gamma \vdash e_1 \circ e_2 \preceq e' \circ e_1^* \circ e_2^* \preceq t' : \tau' \quad \Delta; \Phi_a \models t \preceq t' \\
\text{\textbf{e-r-\text{\textasciitilde}}}
\end{align*}
\]

\( \square \)

**Theorem 46 (Invariant of the Algorithmic Typechecking)**

We have the following.

1. Assume that \( \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k, t \Rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Omega; A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \). Then \( \text{FIV}(\Phi) \subseteq \text{dom}(\Delta, \psi_a) \).

2. Assume that \( \Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow A \rightarrow [\psi], k, t, \Phi \) and \( \text{FIV}(\Phi_a, \Omega) \subseteq \text{dom}(\Delta, \psi_a) \). Then \( \text{FIV}(A, k, t, \Phi) \subseteq \text{dom}(\Delta, \psi_a) \).

3. Assume that \( \Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \downarrow \tau, t \Rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Gamma, \tau, t) \subseteq \text{dom}(\Delta, \psi_a) \). Then \( \text{FIV}(\Phi) \subseteq \text{dom}(\Delta, \psi_a) \).

4. Assume that \( \Delta; \psi_a; \Phi_a; \Gamma \vdash e \circ e' \uparrow \tau \Rightarrow [\psi], t, \Phi \) and \( \text{FIV}(\Phi_a, \Gamma) \subseteq \text{dom}(\Delta, \psi_a) \). Then \( \text{FIV}(\tau, t, \Phi) \subseteq \text{dom}(\Delta, \psi_a) \).

**Theorem 47 (Soundness of the Algorithmic Typechecking in RelCost)**

We have the following.

1. Assume that \( \Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k, t \Rightarrow \Phi \) and \( \text{FIV}(\Phi_a, \Omega; A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \)

   1.1. \( \text{FIV}(\Phi_a, \Omega; A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \)
1.2. $\Delta; \Phi_a[\theta_a] = \Phi[\theta_a]$ is provable for some $\theta_a$ such that $\Delta \triangleright \theta_a : \psi_a$ is derivable

Then $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \triangleright e \vdash \Delta; \Phi_a[\theta_a]$.

2. Assume that $\Delta; \psi_a; \Phi_a[\theta_a] ; \Omega \vdash e \uparrow A = [\psi], k, t, \Phi$ and

2.1. $\text{FIV}(\Phi_a, \Omega) \subseteq \text{dom}(\Delta, \psi_a)$

2.2. $\forall \theta \forall \theta_a. \Delta; \Phi_a[\theta_a] = \Phi[\theta_a]$ is provable s.t. $\Delta \triangleright \theta : \psi$ and $\Delta \triangleright \theta_a : \psi_a$ are derivable

Then $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \triangleright e \vdash \Delta; \Phi_a[\theta_a]$.

3. Assume that $\Delta; \psi_a; \Phi_a[\theta_a]; \Gamma \vdash e \circ e' \downarrow \tau \vdash \Phi$ and

3.1. $\text{FIV}(\Phi_a, \Gamma, \tau, t) \subseteq \text{dom}(\Delta, \psi_a)$

3.2. $\Delta; \Phi_a[\theta_a] = \Phi[\theta_a]$ is provable for some $\theta_a$ such that $\Delta \triangleright \theta_a : \psi_a$ is derivable

Then $\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \circ |e'| \leq \triangleright t[\theta_a] \vdash t[\theta_a]$.

4. Assume that $\Delta; \psi_a; \Phi_a[\theta_a]; \Gamma \vdash e \circ e' \uparrow \tau = [\psi], t, \Phi$ and

4.1. $\text{FIV}(\Phi_a, \Gamma) \subseteq \text{dom}(\Delta, \psi_a)$

4.2. $\forall \theta \forall \theta_a. \Delta; \Phi_a[\theta_a] = \Phi[\theta_a]$ is provable s.t. $\Delta \triangleright \theta : \psi$ and $\Delta \triangleright \theta_a : \psi_a$ are derivable

Then $\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \circ |e'| \leq \triangleright t[\theta_a] \vdash t[\theta_a]$.

Proof. Statements (1—4) follow from simultaneous structural induction on the algorithmic typing derivations. We present several cases below.

Proof of Theorem 47.1:

Case

\begin{align*}
\Delta; k_1, t_1, k_2, t_2 & \in \fresh(R) \\
\Delta; k_1, t_1, \psi_a; \Phi_a[\theta_a]; \Omega & \vdash e_1 \downarrow A_1, k_1, t_1 \Rightarrow \Phi_1 \\
\Delta; k_2, t_2, \psi_a; \Phi_a[\theta_a]; \Omega & \vdash e_2 \downarrow A_1, k_2, t_2 \Rightarrow \Phi_2
\end{align*}

alg-

u-prod-

TS: $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \triangleright e_1, e_2 \vdash \Delta; \Phi_a[\theta_a]$.

By the main assumptions, we have $\text{FIV}(\Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a)$. (*)&

Using (*)&'s derivation must be in a form such that we have

a) $\Delta \vdash K_1 :: R$ and $\Delta \vdash T_1 :: R$

b) $\Delta \vdash K_2 :: R$ and $\Delta \vdash T_2 :: R$

c) $\Delta; \Phi_a[\theta_a] \vdash \Phi_1[\theta_a] \vdash K_1, t_1 \Rightarrow T_1$

d) $\Delta; \Phi_a[\theta_a] \vdash \Phi_2[\theta_a] \vdash K_2, t_2 \Rightarrow T_2$

e) $\Delta; \Phi_a[\theta_a] \vdash (T_1 + T_2) \vdash t[\theta_a] \land (K_1 + K_2) \vdash k[\theta_a]$

By Theorem 47.1 on the first premise using (*)& and c), we can show that

\begin{equation}
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \triangleright e_1 \vdash A_1[\theta_a]
\end{equation}
By Theorem 47.1 on the second premise using (⋆) and d), we can show that

$$\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \frac{T_1}{K_2} \mid e_1 \mid e_2 \mid : c \vdash A_2[\theta_a]$$

(5.5)

Combining eqs. (5.4) and (5.5) with c-prod rule, we get

$$\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \frac{T_1}{K_1 + T_2} \mid e_1 \mid e_2 \mid : c \vdash A_1[\theta_a] \times A_2[\theta_a].$$

Then, by using c) with the c-exec rule, we can conclude that $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \frac{T(a)}{K_1 + T_2} \mid e_1 \mid e_2 \mid : c \vdash A_1[\theta_a] \times A_2[\theta_a].$

By Theorem 47.1 on the second premise using (⋆) eqs. (5.6) and (5.7) and (d) that $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \frac{T(a)}{K_1 + T_2} \mid e_1 \mid e_2 \mid : c \vdash A_1[\theta_a] \times A_2[\theta_a].$

Case

$$\Delta; \psi_a; \Phi_a; \Omega \vdash e \uparrow A' \Rightarrow [\psi], k', t', \Phi_1 \quad \Delta; \psi_a; \Phi_a \vdash^\Delta A' \subseteq A \Rightarrow \Phi_2$$

alg-↑↓

TS: $\Delta; \psi_a; \Phi_a; \Omega \vdash e \downarrow A, k, t \Rightarrow \exists(\psi). \Phi_1 \cap \Phi_2 \wedge t' \leq t \wedge k \leq k'$

By the main assumptions, we have FIV($\Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a)$ (⋆) and $\Delta; \Phi_a[\theta_a] = (\exists(\psi). \Phi_1 \cap \Phi_2 \wedge t' \leq t \wedge k \leq k')[\theta_a]$ (★★)

By Theorem 46 using (⋆) and the first premise, we get FIV($A', k', t', \Phi_1) \subseteq \text{dom}(\Delta, \psi_a)$ (♣).

Using (⋆) and (♣), (★★)'s derivation must be in a form such that we have

a) $\Delta \vdash \theta_a : \psi_a$

b) $\Delta; \Phi_a[\theta_a] = \Phi_1[\theta_a, \theta_a]$

c) $\Delta; \Phi_a[\theta_a] = \Phi_2[\theta_a, \theta_a]$

d) $\Delta; \Phi_a[\theta_a] = t'[\theta_a] \leq t[\theta_a] \wedge k[\theta_a] \leq k'[\theta_a]$

By Theorem 47.2 on the first premise using (⋆), a) and b), we can show that

$$\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \frac{T(a)}{K_1 + T_2} \mid e_1 \mid e_2 \mid : c \vdash A'[\theta_a]$$

(5.6)

By Theorem 40 using the second premise and c), we obtain

$$\Delta; \Phi_a[\theta_a] \vdash^\Delta A'[\theta_a] \subseteq A[\theta_a]$$

(5.7)

Note that due to (⋆), we have $A[\theta_a] = A[\theta_a]$. Then we can conclude by the c-exec rule using eqs. (5.6) and (5.7) and (d) that $\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \frac{T(a)}{K_1 + T_2} \mid e_1 \mid e_2 \mid : c \vdash A[\theta_a].$

Case

$$\Delta; \psi_a; \Phi_a; f : A_1 \xrightarrow{\text{exec}(k', t')} A_2, x : A_1, \Omega \vdash e \downarrow A_2, k', t' \Rightarrow \Phi$$

alg-u-fix-↓

TS: $\Delta; \psi_a; \Phi_a; f : A_1 \xrightarrow{\text{exec}(k', t')} A_2, k, t \Rightarrow \Phi \wedge k = 0 \wedge 0 = t$

By the main assumptions, we have FIV($\Phi_a, \Omega, A_1 \xrightarrow{\text{exec}(k', t')} k, t) \subseteq \text{dom}(\Delta, \psi_a)$ (⋆) and $\Delta; \Phi_a[\theta_a] = \Phi \wedge 0 = k \wedge 0 = t'[\theta_a]$ (★★)

Using (⋆), we can show that

a) FIV($\Phi_a, \Omega, A_1, A_2 \xrightarrow{\text{exec}(k', t')} A_2, k', t' \subseteq \text{dom}(\Delta, \psi_a)$.

136
We also can show that (⋆⋆)’s derivation must be in a form such that we have

b) \( \Delta; \Phi_\alpha[\theta_\alpha] \models \Phi[\theta_\alpha] \)

c) \( \Delta; \Phi_\alpha[\theta_\alpha] \models 0 \vdash k[\theta_\alpha] \)

d) \( \Delta; \Phi_\alpha[\theta_\alpha] \models 0 \vdash l[\theta_\alpha] \)

By Theorem 47.1 on the first premise using a) and b), we can show that

\[ \Delta; \Phi_\alpha[\theta_\alpha]; x : A_1[\theta_\alpha], f : A_1[\theta_\alpha] \xrightarrow{\text{exec}(k[\theta_\alpha], t[\theta_\alpha])} A_2[\theta_\alpha], \Omega[\theta_\alpha] \vdash t'[\theta_\alpha] \mid e \vdash A_2[\theta_\alpha] \quad (5.8) \]

By the c-fix rule using eq. (5.8), we obtain

\[ \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash^0 f(x).e \vdash^\ast A_1[\theta_\alpha] \xrightarrow{\text{exec}(k[\theta_\alpha], t[\theta_\alpha])} A_2[\theta_\alpha]. \]

By c-exec rule using (c) and (d), we obtain \( \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash_{k[\theta_\alpha]} f(x).e \vdash_{A_1[\theta_\alpha]} \xrightarrow{\text{exec}(k'[\theta_\alpha], t'[\theta_\alpha])} A_2[\theta_\alpha] \).

**Case**

\[ i :: S; \Delta; \psi_\alpha; \Phi_\alpha; \Omega \vdash e \downarrow A, k_e, t_e \Rightarrow \Phi \quad \text{alg-u-iLam-↓} \]

TS: \( \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash i \vdash (\forall i :: S.\Phi) \land k \not= 0 \land 0 \not= t \)

By the main assumptions, we have FIV(\( \Phi_\alpha, \Omega, \forall i :: S.\Phi \) \) \( \subseteq \text{dom}(\Delta, \psi_\alpha) \) (⋆) and

\( \Delta; \Phi_\alpha[\theta_\alpha] \models (\forall i :: S.\Phi) \land 0 \not= k \land 0 \not= t) \theta_\alpha \) (⋆⋆)

Using (⋆), we can show that

a) \( \text{FIV}(\Phi_\alpha, \Omega, A, k_e, t_e) \subseteq i, \text{dom}(\Delta, \psi_\alpha) \)

We can also show that (⋆⋆)’s derivation must be in a form such that we have

b) \( i :: S; \Delta; \Phi_\alpha[\theta_\alpha] \models \Phi[\theta_\alpha] \)

c) \( \Delta; \Phi_\alpha[\theta_\alpha] \models 0 \vdash k[\theta_\alpha] \)

d) \( \Delta; \Phi_\alpha[\theta_\alpha] \models 0 \vdash l[\theta_\alpha] \)

By Theorem 47.1 on the premise using a) and b), we can show that

\[ i :: S; \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash_{k'[\theta_\alpha]} t[\theta_\alpha] \mid e \vdash A[\theta_\alpha] \quad (5.9) \]

By the c-iLam rule using eq. (5.9), we obtain \( \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash^0 \Delta_i.e \vdash^\ast \forall i :: S.\Phi \vdash_{k[\theta_\alpha]} A[\theta_\alpha]. \)

By c-exec rule using (c) and (d), we obtain \( \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash_{k[\theta_\alpha]} \Delta_i.e \vdash_{\forall i :: S.\Phi} \vdash_{A[\theta_\alpha]} \).

**Case**

\[ \Delta; \psi_\alpha; \Phi_\alpha; \Omega \vdash e \downarrow A[I/i], k, t \Rightarrow \Phi \quad \Delta \vdash I :: S \quad \text{alg-u-pack-↓} \]

TS: \( \Delta; \Phi_\alpha[\theta_\alpha]; \Omega[\theta_\alpha] \vdash_{k[\theta_\alpha]} \text{pack} e \) with \( I \downarrow \exists i :: S.A, k, t \Rightarrow \Phi \)

By the main assumptions, we have FIV(\( \Phi_\alpha, \exists i :: S.A, k, t) \) \( \subseteq \text{dom}(\Delta, \psi_\alpha) \) (⋆) and

\( \Delta; \Phi_\alpha[\theta_\alpha] \models \Phi[\theta_\alpha] \) (⋆⋆)
Using (⋆) and the second premise, we can show that

a) \( \text{FIV}(\Phi_a;\Omega,A\{I;i\},k,t) \subseteq \text{dom}(\Delta,\psi_a) \).

By Theorem 47.1 on the premise using a) and (⋆⋆), we can show that

\[
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash_{k[\theta_a]}^{t[\theta_a]} \mid e \mid \quad : \quad \text{A}[\theta_a]\{I/i\}
\]  
(5.10)

By the \texttt{c-pack} rule using eq. (5.10) and the second premise, we obtain

\[
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash_{k[\theta_a]}^{t[\theta_a]} \text{pack} \mid e \mid \text{with } I \vdash \exists i::S. A[\theta_a].
\]

\[
\Delta; \psi_a; \Phi_a; \Omega \vdash_{k_1}^{e_1} \exists i::S. A_1 \Rightarrow [\psi], k_1, t_1, \Phi_1 \quad k_2, t_2 \in \text{fresh}(\mathbb{R})
\]

\[
i \vdash S, \Delta; k_2, t_2, \psi, \psi_a; \Phi_a; x: A_1, \Omega \vdash_{e_2} e_2 \downarrow A_2, k_2, t_2 \Rightarrow \Phi_2 \quad i \notin \text{FV}(\Phi_a; \Omega, A_2, k_2, t_2)
\]

\[
\Phi = \exists(\psi).(\Phi_1 \land \exists k_2, t_2 :: \mathbb{R}. \forall i :: S. \Phi_2 \land t = (k_1 + k_2 + c_{\text{unpack}} \land t_1 + t_2 + c_{\text{unpack}} \Rightarrow t)) \quad (\star)
\]

\[
\Delta; \psi_a; \Phi_a; \Omega \vdash_{k_2}^{e_1} \text{unpack} e_1 \text{ as } (x, i) \text{ in } e_2 \downarrow A_2, k, t \Rightarrow \Phi
\]

\text{alg-unpack-↓}

TS: \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash_{k[\theta_a]}^{t[\theta_a]} \text{unpack} \mid e \mid \text{ with } (x, i) \text{ in } \mid e \mid \vdash \exists A_2[\theta_a] \).

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Omega, A_2, k, t) \subseteq \text{dom}(\Delta, \psi_a) \) (⋆) and

\[
\Delta; \Phi_a[\theta_a] = (\exists(\psi).(\Phi_1 \land \exists k_2, t_2 :: \mathbb{R}. \forall i :: S. \Phi_2 \land t = (k_1 + k_2 + c_{\text{unpack}} \land t_1 + t_2 + c_{\text{unpack}} \Rightarrow t)) [\theta_a] \quad (\star\star)
\]

By Theorem 46 using the first premise and (⋆), we get \( \text{FIV}(A_1, k_1, t_1, \Phi_1) \subseteq \text{dom}(\Delta, \psi; \psi_a) \) (⋆).

Using (⋆), (⋆) and the 4th premise, (⋆⋆)’s derivation must be in a form such that we have

a) \( \Delta \vdash \theta : \psi \)

b) \( \Delta; \Phi_a[\theta_a] \models \Phi_1[\theta \theta_a] \)

c) \( i :: S, \Delta; \Phi_a[\theta_a] \models \Phi_2[\theta_a, \theta, k_2 \mapsto K_2, t_2 \mapsto T_2] \)

d) \( \Delta; \Phi_a[\theta_a] \models t_1[\theta \theta_a] + T_2 + c_{\text{unpack}} \Rightarrow t[\theta_a] \)

e) \( \Delta; \Phi_a[\theta_a] \models k[\theta_a] \models k_1[\theta \theta_a] + K_2 + c_{\text{unpack}} \)

By Theorem 47.2 on the first premise using (⋆), a) and b), we can show that

\[
\Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash_{k_1[\theta_a]}^{t_1[\theta_a]} \mid e_1 \mid \quad : \quad \exists i::S. A_1[\theta \theta_a]
\]  
(5.11)

From (⋆) and (⋆), we can show that

f) \( \text{FIV}(\Phi_a, A_1, \Omega, A_2, k_2, t_2) \subseteq i, k_2, t_2, \text{dom}(\Delta, \psi, \psi_a) \)

By Theorem 47.1 on the second premise using c), f), (⋆) and (⋆), we obtain

\[
i :: S, \Delta; \Phi_a[\theta_a]; x: A_1[\theta \theta_a], \Omega[\theta_a] \vdash_{T_2}^{K_2} \mid e_2 \mid \quad \vdash A_2[\theta \theta_a]
\]  
(5.12)

Note that due to (⋆), we have \( A_2[\theta \theta_a] = A_{2[\theta]} \). Then by the \texttt{c-unpack} rule using eqs. (5.11) and (5.12), we can show that \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash_{k_1[\theta_a]}^{t_1[\theta_a] + T_2 + c_{\text{unpack}} + K_1 + c_{\text{unpack}}} \text{unpack} \mid e_1 \mid \text{ with } (x, i) \text{ in } \mid e_2 \mid \vdash A_2[\theta_a] \).
By \( \subseteq \) and (e), we obtain \( \Phi_\alpha[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \) with (x, i) in |e2| : \( A[\theta_a] \).

\[
\begin{align*}
\Delta; \psi_a; C \land \Phi_a; \Omega \vdash e_1 \downarrow A, k, t \Rightarrow \Phi_1 \\
\Delta; \psi_a; \neg C \land \Phi_a; \Omega \vdash e_2 \downarrow A, k, t \Rightarrow \Phi_2 \\
\Delta \vdash C \wedge \Phi_1 \wedge \Phi_2
\end{align*}
\]

alg-u-split

Case 

\[
\begin{align*}
\Delta; \psi_a; C \land \Phi_a; \Omega \vdash e_1 \downarrow A, k, t \Rightarrow \Phi_1 \\
\Delta; \psi_a; \neg C \land \Phi_a; \Omega \vdash e_2 \downarrow A, k, t \Rightarrow \Phi_2 \\
\Delta \vdash C \land \Phi_1 \wedge \Phi_2
\end{align*}
\]

alg-u-c-impl

TS: \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \) and (5.15) \( \Box \) with (x, i) in |e2| : \( A[\theta_a] \).

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \) (⋆) and \( \Delta; \Phi_a[\theta_a] \models \boxed{(C \rightarrow \Phi_1 \land \neg C \rightarrow \Phi_2)[\theta_a]} \) (⋆⋆)

Using (⋆) and the third premise, we can show that

a) \( \text{FIV}(C \land \Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \).

b) \( \text{FIV}(\neg C \land \Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \).

Using (⋆⋆) and the third premise, we can show that

c) \( \Delta; C \land \Phi_a[\theta_a] \models \Phi_1[\theta_a] \)
d) \( \Delta; \neg C \land \Phi_a[\theta_a] \models \Phi_2[\theta_a] \)

By Theorem 47.1 on the first premise using (⋆) and c), we can show that

\[
\begin{align*}
\Delta; C \land \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \boxed{\text{I}[\theta_a]/i} & : C[\theta_a] \subseteq A[\theta_a] \\
\text{(5.13)}
\end{align*}
\]

By Theorem 47.1 on the second premise using (⋆) and d), we can show that

\[
\begin{align*}
\Delta; \neg C \land \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \boxed{\text{I}[\theta_a]/i} & : C[\theta_a] \subseteq A[\theta_a] \\
\text{(5.14)}
\end{align*}
\]

By the c-split rule using eqs. (5.13) and (5.14) and the third premise, we obtain

\( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \) with \( C \subseteq A[\theta_a] \).

Case 

\[
\begin{align*}
\Delta; \psi_a; C \land \Phi_a; \Omega \vdash e_1 \downarrow A, k, t \Rightarrow \Phi \\
\Delta; \psi_a; \neg C \land \Phi_a; \Omega \vdash e_2 \downarrow A, k, t \Rightarrow \Phi \\
\Delta \vdash C \land \Phi
\end{align*}
\]

alg-u-c-impl \downarrow

TS: \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \) and (5.15) \( \Box \) with (x, i) in |e2| : \( A[\theta_a] \).

By the main assumptions, we have \( \text{FIV}(\Phi_a, \Omega, C \supset A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \) (⋆) and \( \Delta; \Phi_a[\theta_a] \models \boxed{(C \rightarrow \Phi)[\theta_a]} \) (⋆⋆)

Using (⋆), we can show that

a) \( \text{FIV}(C \land \Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a) \).

By Theorem 47.1 on the premise using (⋆) and a), we can show that

\[
\begin{align*}
\Delta; C[\theta_a] \land \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \boxed{\text{I}[\theta_a]/i} & : C[\theta_a] \subseteq A[\theta_a] \\
\text{(5.15)}
\end{align*}
\]

By the c-cimp rule using eq. (5.15), we obtain \( \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash C[\theta_a] \) with (x, i) in \boxed{A[\theta_a]}.
Proof of Theorem 47.2:

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash c \downarrow A, k, t \Rightarrow \Phi \quad \Delta; \Phi_a \vdash A \text{ uf} \quad \text{FIV}(A, k, t) \in \Delta \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash (e : A, k, t) \uparrow A \Rightarrow [\cdot], k, t, \Phi \]

TS: \[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{\theta_a\}_k[\theta_a] \mid (\cdot, e : A, k, t) \vdash c \text{ A}[\theta_a]. \]

By the main assumptions, we have FIV(\Phi_a, \Omega) \subseteq dom(\Delta, \psi_a) \ (\ast) and \[ \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a] \ (\ast\ast) \]

Using the third premise, we can show that

a) \[ \text{FIV}(\Phi_a, \Omega, A, k, t) \subseteq \text{dom}(\Delta, \psi_a). \]

By Theorem 47.1 on the first premise using \( \ast\ast \) and a), we can conclude that
\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{\theta_a\}_k[\theta_a] \mid |e| : c \text{ A}[\theta_a]. \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash e_1 e_2 \uparrow A \Rightarrow [k_2, k_2, \psi, k_1, \Phi_1] \]

\[ k_2, t_2 \in \text{fresh}(\mathbb{R}) \]

\[ \Delta; k_2, t_2, \psi, \psi_a; \Phi_a; \Omega \vdash e_2 \downarrow A \Rightarrow 1 \quad k_2, \psi, k_1, \Phi_1 \]

Case \[ \Delta; \psi_a; \Phi_a; \Omega \vdash e_1 e_2 \uparrow A : \text{alg-u-anno-}\uparrow \]

TS: \[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{\theta_a\}_k[\theta_a] \mid (k_2 + k_2 + c_{\text{app}})[\theta_a, \theta_2] \mid |e_1| \mid |e_2| : c \text{ A}[\theta_a, \theta_2]. \]

By the main assumptions, we have FIV(\Phi_a, \Omega) \subseteq \text{dom}(\Delta, \psi_a) \ (\ast) and \[ \Delta; \Phi_a[\theta_a] \models (\Phi_1 \land \Phi_2)[\theta_a, \theta_2] \ (\ast\ast) \] such that \[ \Delta \vdash \theta_a : k_2, t_2, \psi (\circ) \] and \[ \Delta \vdash \theta_a : \psi_a \text{ are derivable.} \]

By (\circ), we can show that \[ t_2 = k_2, t_2, \psi \text{ such that} \]

a) \[ \theta(k_2) = K_2 \text{ and } \theta(t_2) = T_2 \text{ for some } K_2 \text{ and } T_2. \]

b) \[ \Delta \vdash \theta : \psi. \]

By Theorem 47.2 on the first premise using (\ast) and (b), we obtain
\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{\theta_a\}_k[\theta_a] \mid |e_1| : c \text{ A}[\theta_a] \]

\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{\theta_a\}_k[\theta_a] \mid |e_2| : c \text{ A}[\theta_a] \]

(5.16)

By Theorem 46.2 on the first premise and (\ast), we get
\[ \text{c} \text{-app} \]

c) \[ \text{FIV}(A_1 \downarrow k_2, k_1, \Phi_1) \subseteq \text{dom}(\Delta, \psi, \psi_a). \]

By (\ast) and c), we get
\[ \text{d} \]

D) \[ \text{FIV}(\Phi_a, \Omega, A_2, k_2, t_2) \subseteq k_2, t_2, \text{dom}(\Delta, \psi, \psi_a). \]

By Theorem 47.2 on the third premise using (c), (d) and (\ast\ast), we obtain
\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{e_2\}_k[\theta_a] \mid |e_2| : c \text{ A}[\theta_a] \]

(5.17)

Then, by using \text{c-app} \ rule using eqs. (5.16) and (5.17), we can show that
\[ \Delta; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash \{e_2\}_k[\theta_a] + T_2[\theta_a] \mid |e_2| : c \text{ A}[\theta_a, \theta_2]. \]
Note that we have $k_2[\theta_a, \theta_2] = K_2$ and $k_2[\theta_a, \theta_2] = K_2$. Moreover, $t_1[\theta_a, \theta_2] = t_1[\theta_a]$ and $k_1[\theta_a, \theta_2] = k_1[\theta_a]$ (similarly for $c_e$ and $c_e$) since $k_2, t_2$ are fresh variables.

**Case**

$\Delta; \psi_3; \Phi_3; \Omega \vdash e \uparrow \forall i \quad \text{exec}(k, t_e) \quad S. A' \Rightarrow [\psi], [k], t, \Phi \quad \Delta \vdash I :: S$ \quad \text{alg-u-iApp}$^*$

$\Delta; \psi_3; \Phi_3; \Omega \vdash e[I] \uparrow A'[I/i] \Rightarrow [\psi], k + k_e[I/i], t + t_e[I/i], \Phi$

**TS:** $\Delta; \Phi_3[\theta_a]; \Omega[\theta_a] \vdash (I[i] + t_e[I/i])|\theta_a| \vdash [I] : (A'[I/i])|\theta_a|$

By the main assumptions, we have $\text{FIV}(\Phi_a, \Omega) \subseteq \text{dom}(\Delta, \psi_a)$ \text{(*)} and $\Delta; \Phi_3[\theta_a] \vdash \Phi_2[\theta_a]$ \text{(**)} such that $\Delta \vdash \theta : \psi$ \text{(*)} and $\Delta \vdash \theta : \psi$ \text{(**)} are derivable.

By Theorem 47.2 on the first premise using \text{*} and \text{(**)}, we obtain

$$\Delta; \Phi_3[\theta_a]; \Omega[\theta_a] \vdash t[I] \quad \vdash \forall i \quad \text{exec}(k, t_e) \quad S. A'[\theta_a]$$ (5.18)

Then, by \text{c-iApp} rule using eq. (5.18) and the second premise, we can conclude that $\Delta; \Phi_3[\theta_a]; \Omega[\theta_a] \vdash t[I] + t_e[I/i] \vdash [I] : A'[\theta_a][I/i]$.

**Proof of Theorem 47.3:**

**Case**

$t' \in \text{fresh}(R)$ \quad $\Delta; t', \psi_3; \Phi_3; \Box \Gamma \vdash e \ominus e \downarrow \tau, t' \Rightarrow \Phi$ \quad \text{alg-r-nochange}$^*$

$\Delta; \psi_3; \Phi_3; \Gamma' \vdash e \ominus \Box \Gamma \vdash 0 \Leftrightarrow \exists \tau :: R. \Phi$

**TS:** $\Delta; \Phi_3[\theta_a]; \Gamma'[\theta_a], \Box \Gamma[\theta_a] \vdash e \ominus \Box \Gamma \vdash 0 :: \Box \Gamma[\theta_a]$.

By the main assumptions, we have $\text{FIV}(\Phi_a, \Gamma', \Box \Gamma, \tau, t) \subseteq \text{dom}(\Delta, \psi_a)$ \text{(*)} and $\Delta; \Phi_3[\theta_a] \vdash (0 \Leftrightarrow \exists \tau :: R. \Phi)[\theta_a]$ \text{(**)}

Using \text{*} and the first premise, we can show that

a) $\text{FIV}(\Phi_a, \Gamma, \tau, t') \subseteq t', \text{dom}(\Delta, \psi_a)$.

Using \text{*}, \text{(**)}’s derivation must be in a form such that we have

b) $\Delta; \Phi_3[\theta_a] \vdash 0 \square \tau[\theta_a]$

c) $\Delta \vdash T' :: R$ for some $T'$

d) $\Delta; \Phi_3[\theta_a] \vdash \Phi[\theta_a, t' \Rightarrow T']$

By Theorem 47.3 on the premise using a), d) and \text{*}, we can show that

$$\Delta; \Phi_3[\theta_a]; \Box \Gamma[\theta_a] \vdash |e| \ominus |e| \lesssim T' : \tau[\theta_a]$$ (5.19)

By the \text{c-nochange} rule using eq. (5.19), we obtain $\Delta; \Phi_3[\theta_a]; \Gamma'[\theta_a], \Box \Gamma[\theta_a] \vdash e \ominus \Box \Gamma \vdash 0 :: \Box \tau[\theta_a]$.

By the \text{c-r-□} rule using this and b), we obtain $\Delta; \Phi_3[\theta_a]; \Gamma'[\theta_a], \Box \Gamma[\theta_a] \vdash e \ominus \Box \Gamma \vdash t[\theta_a] : \Box \tau[\theta_a]$. 

141
Case

By Theorem 47.3 on the second premise using d) and (

By Theorem 47.4 on the first premise using b) and (

By Theorem 47.3 on the third premise using e) and (

Using (⋆), we get FIV(list[\alpha, \beta], t, t) \subseteq dom(\Delta, \psi_a) (⋆)

By Theorem 46 using the first premise and (⋆), we get FIV(list[\alpha, \beta], t, t) \subseteq dom(\Delta, \psi_a) (⋆)

Using (⋆) and (⋆⋆), (⋆⋆⋆)’s derivation must be in a form such that we have

a) \Delta \triangleright \theta : \psi
b) \Delta; \Phi_a[\theta_a] \models \Phi[\theta_a]

c) \Delta \vdash T_2 :: R

By Theorem 47.4 on the first premise using b) and (⋆), we can show that

\Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e] \lor [e'] \subseteq \Delta[\theta_a : t_1[\theta_a]] : \text{list}[\alpha, \theta_a] : \tau[\theta_a] (5.20)

By Theorem 47.3 on the second premise using d) and (⋆), we can show that

\Delta; n[\theta_a] \equiv 0 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e_1] \lor [e_1'] \subseteq T_2 : \tau[\theta_a] (5.21)

By Theorem 47.3 on the third premise using e) and (⋆), we can show that

i :: S, \Delta; n[\theta_a] \equiv i + 1 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash [e_2] \lor [e_2'] \subseteq T_2 : \tau[\theta_a] (5.22)
By Theorem 47.3 on the fourth premise using f) and (⋆), we can show that

\[ i : S, \beta :: S; n[\theta_a] \doteq i + 1 \land \alpha[\theta_a] \doteq \beta + 1 \land \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e_3| \uplus |e_5| \subseteq T_2 : \tau'[\theta_a] \quad (5.23) \]

Then by c-r-caseL rule using eqs. (5.20) to (5.23), we can show that

\[
\begin{align*}
\text{case } e \text{ of nil } & \rightarrow |e_1| & \text{case } e' \text{ of nil } & \rightarrow |e'_1| \\
\Delta ; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash & h :: N \ t_l \rightarrow |e_2| \uplus & h :: N \ t_l \rightarrow |e'_2| \subseteq t_1[\theta_a] + T_2 : \tau'[\theta_a] \\
& h :: C \ t_l \rightarrow |e_3| & h :: C \ t_l \rightarrow |e'_3| \\
\end{align*}
\]

We conclude by applying c-r-cons rule to this using g).

Case

\[
\begin{align*}
t_1, t_2 \in \text{fresh}(\mathbb{R}) & \quad i \in \text{fresh}(\mathbb{N}) & \Delta ; t_1, \psi_a; \Phi_a; \Gamma \vdash e_1 \uplus e'_1 \downarrow \quad \text{alg-} \\
\Delta ; i, t_2, \psi_a; \Phi_a; \Gamma \vdash e_2 \uplus e'_2 \downarrow \text{list}[\iota]^{n} \tau, t_2 \Rightarrow \Phi_2 & \quad \Phi_2' = \Phi_2 \land n \doteq (i + 1) \land t_1 + t_2 \doteq t \\
\end{align*}
\]

By the main assumptions, we have FIV(\Phi_a, \Gamma, \text{list}[n]^{a} \tau, t) \subseteq \text{dom}(\Delta, \psi_a) \quad (⋆)

Using (⋆), (⋆⋆)'s derivation must be in a form such that we have

a) \ \Delta \vdash T_1 :: \mathbb{R} \\
b) \ \Delta \vdash T_2 :: \mathbb{R} \\
c) \ \Delta ; \Phi_a[\theta_a] \vdash \Phi_1[\theta_a], t_1 \rightarrow T_1 \\
d) \ \Delta \vdash I :: \mathbb{N} \\
e) \ \Delta ; \Phi_a[\theta_a] \vdash \Phi_2[\theta_a], t_2 \rightarrow T_2, i \rightarrow I \\
f) \ \Delta ; \Phi_a[\theta_a] \vdash (I + 1) \doteq n[\theta_a] \\
g) \ \Delta ; \Phi_a[\theta_a] \vdash (T_1 + T_2) \doteq t[\theta_a] \\

By Theorem 47.3 on the third premise using (⋆) and e), we can show that

\[ \Delta ; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_1| \uplus |e'_1| \subseteq T_1 : \Box \tau[\theta_a] \quad (5.24) \]

By Theorem 47.3 on the fourth premise using (⋆) and e), we can show that

\[ \Delta ; \Phi_a[\theta_a]; \Omega[\theta_a] \vdash |e_2| \uplus |e'_2| \subseteq T_2 : \text{list}[I]^{n[\theta_a]} \tau[\theta_a] \quad (5.25) \]

By c-r-cons typing rule using eqs. (5.24) and (5.25), we obtain

\[ \Delta ; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \text{cons}_{NC}(|e_1|, |e_2|) \uplus \text{cons}_{NC}(|e'_1|, |e'_2|) \subseteq T_1 + T_2 : \text{list}[I + 1]^{n[\theta_a]} \tau[\theta_a]. \]

We conclude by applying c-r-cons rule to this using f) and g).
Case

\[ \Delta; \psi_a; \Phi; [\Gamma]_1 \vdash e_1 \uparrow A_1 \xrightarrow{\text{exec}(k_i, t_e)} A_2 \Rightarrow [\psi], k_1, t_1, \Phi_1 \]

\[ t_2 \in \text{fresh}(\mathcal{R}) \]

\[ \Delta; t_2, \psi, \psi_a; \Phi_a; \Gamma \vdash e_2 \oplus e'_2 \downarrow U (A_1, A'_2), t_2 \Rightarrow \Phi_2 \]

TS: \( \Delta; \Phi_a[\theta_a]; [\Gamma][\theta_a] \vdash [e_1] | [e_2] \circ [e'_2] \subseteq t[\theta_a] : U (A[\theta_a], A'[\theta_a]) \).

By the main assumptions, we have FIV(\( \Phi_a, \Gamma, U (A, A'), t \)) \( \subseteq \text{dom}(\Delta, \psi_a) \) (\( \ast \)) and

\[ \Delta; \Phi_a[\theta_a] = ([\exists (\psi). \Phi_1 \land (\exists t_2 :: \mathcal{R}. \Phi_2 \land t_1 + t_2 + t_c + c_{app} = t) \land \Phi_4])[\theta_a] \] (\( \ast \ast \)) such that \( \Delta \triangleright \theta_a : \psi_a \) is derivable.

By Theorem 46 using (\( \ast \)) and the first premise, we get

\[ \text{FIV}(A_1 \xrightarrow{\text{exec}(k_i, t_e)} A_2, k_1, t_1, \Phi_1) \subseteq \text{dom}(\Delta, \psi; \psi_a) \] (\( \diamond \)).

Using (\( \ast \)) and (\( \diamond \)), (\( \ast \ast \))’s derivation must be in a form such that we have

a) \( \Delta \triangleright \theta : \psi \)

b) \( \Delta; \Phi_a[\theta_a] = \Phi_1[\theta \theta_a] \)

c) \( \Delta; \Phi_a[\theta_a] = \Phi_2[\theta_a, \theta, t_2 \triangleright T_2] \)

d) \( \Delta; \Phi_a[\theta_a] = \Phi_3[\theta \theta_a] \)

e) \( \Delta; \Phi_a[\theta_a] = t_1[\theta \theta_a] + t_c[\theta \theta_a] + T_2 + c_{app} = t[\theta_a] \)

By Theorem 47.2 on the first premise using (\( \ast \)), a) and b), we can show that

\[ \Delta; \Phi_a[\theta_a]; [\Gamma][\theta_a] ; [e_1] :: A_1[\theta \theta_a] \xrightarrow{\text{exec}(k_i[\theta \theta_a], t_e[\theta \theta_a])} A_2[\theta \theta_a] \] (5.26)

From (\( \ast \)) and (\( \diamond \)), we can show that

f) \( \text{FIV}(\Phi_a, \Gamma, U (A_1, A'_2), t_2) \subseteq t_2, \text{dom}(\Delta, \psi; \psi_a) \)

By Theorem 47.3 on the third premise using c), (\( \ast \)) and (\( \diamond \)), we obtain

\[ \Delta; \Phi_a[\theta_a]; [\Gamma][\theta_a] \vdash [e_2] \circ [e'_2] \subseteq t_2 : U (A_1[\theta \theta_a], A'[\theta \theta_a]) \] (5.27)

By Theorem 40 on the fourth premise using (\( \diamond \)), (\( \ast \)) and (d), we obtain

\[ \Delta; \Phi_a[\theta_a] \models A_2[\theta \theta_a] \subseteq A[\theta_a] \] (5.28)

By \textbf{U} subtyping rule using eq. (5.28) and \textbf{u-refl} subtyping rule, we obtain

\[ \Delta; \Phi_a[\theta_a] \models U (A_2[\theta \theta_a], A'[\theta_a]) \subseteq U (A[\theta_a], A'[\theta_a]) \] (5.29)

Then by the \textbf{c-r-app-e} rule using eqs. (5.26) and (5.27), we can show that

\[ \Delta; \Phi_a[\theta_a]; [\Gamma][\theta_a] \vdash [e_1] | [e_2] \circ [e'_2] \subseteq t_1[\theta \theta_a] + t_c[\theta \theta_a] + T_2 + c_{app} : U (A_2[\theta_a], A'[\theta_a]) \]

Applying \textbf{c-r-\( \varnothing \)} rule to this using (e) and eq. (5.29), we can conclude as
Case \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash e \odot e' \downarrow \tau, t \Rightarrow \Phi \)

By Theorem 47.4 on the first premise using (\( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \tau \uparrow \)).

Case \( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash \tau \uparrow \Phi \) \n
\( \text{alg-r-anno-}\uparrow \)

\( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash (e : \tau, t) \odot (e' : \tau, t) \uparrow \tau \Rightarrow [\cdot], t, \Phi \)

By the main assumptions, we have \( FIV(\Phi_a[\theta_a]) \subseteq dom(\Delta, \psi_a) \) (\( \ast \)) and \( \Delta; \Phi_a[\theta_a] \vdash \Phi[\theta_a] \) (\( \ast \ast \)).

Using the third premise, we can show that

a) \( FIV(\Phi_a, \Gamma, \tau, k, t) \subseteq dom(\Delta, \psi_a) \).

By Theorem 47.4 on the first premise using (\( \ast \ast \)) and a), we can conclude that

\( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| \subseteq t[\theta_a] : \tau[\theta_a] \).

Case \( \Delta; \psi_a[\Phi_a]; \Gamma[\theta_a] \vdash e_1 \odot e_2 \uparrow \Box \tau \Rightarrow [\psi], t, \Phi \)

\( \text{alg-r-der-}\uparrow \)

\( \Delta; \psi_a[\Phi_a]; \Gamma[\theta_a] \vdash \text{der } e_1 \odot \text{der } e_2 \uparrow \tau \Rightarrow [\psi], t, \Phi \)

By the main assumptions, we have \( FIV(\Phi_a, \Gamma) \subseteq dom(\Delta, \psi_a) \) (\( \ast \)) and

\( \Delta; \Phi_a[\theta_a] \vdash \Phi[\theta_a] \) (\( \ast \ast \)) such that \( \Delta \triangleright \theta : \psi (\ast) \) and \( \Delta \triangleright \theta : \psi_a \) are derivable.

By Theorem 47.4 on the first premise using (\( \ast \)) and (\( \ast \)), we obtain

\[ \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| \subseteq t[\theta a] : \Box \tau[\theta_a] \tag{5.30} \]

Then, by \( \text{c-der} \) rule using eq. (5.30) and the second premise, we can conclude that

\( \Delta; \Phi_a[\theta_a]; \Gamma[\theta_a] \vdash |e| \odot |e'| \subseteq t[\theta a] : \tau[\theta_a] \).

\( \Box \)

Theorem 48 (Completeness of the Algorithmic Typechecking in RelCost)
We have the following.

1. Assume that \( \Delta; \Phi_a; \Omega \vdash_k e : e \). Then, \( \exists e' \) such that
   1.1. \( \Delta; \vdash \Phi_a; \Omega \vdash e' \downarrow A, k, t \Rightarrow \Phi \)
   1.2. \( \Delta; \Phi_a \vdash \Phi \)
   1.3. \( |e'| = e \)

2. Assume that \( \Delta; \Phi_a; \Gamma \vdash e_1 \odot e_2 \preceq t : e \). Then, \( \exists e'_1, e'_2 \) such that
   2.1. \( \Delta; \vdash \Phi_a, \Gamma \vdash e'_1 \odot e'_2 \downarrow \tau, t \Rightarrow \Phi \)
   2.2. \( \Delta; \Phi_a \vdash \Phi \)
   2.3. \( |e'_1| = e_1 \) and \( |e'_2| = e_2 \)

145
Proof. Proof is by simultaneous induction on the RelCostCore typing derivations.

Proof of Theorem 48.1:

Case $\Omega(x) = A$

\[ \Delta; \Phi_a; \Omega \vdash \ell_b x : {^c} A \]

We can conclude as follows

\[ \Omega(x) = A \]

\[ \Delta; \psi_a; \Omega \vdash x \triangleright A \Rightarrow [\cdot], 0, 0, T \]

alg-u-var-\dagger

\[ \Delta; \Phi_a \models^\Delta A \subseteq A \Rightarrow \Phi \] by lemma 38

alg-r-\dagger

Case $\Delta; \Phi_a; \Omega \vdash_{l_{k_1}} e_1 : {^c} A$

$\Delta; \Phi_a; \Omega \vdash_{l_{k_2}} e_2 : {^c} \text{list}[n] A$

$\Delta; \Phi_a; \Omega \vdash_{l_{k_1 + k_2}} \text{cons}_C(e_1, e_2) : {^c} \text{list}[n + 1] A$

By Theorem 48.2 on the first premise, $\exists e_1'$ such that

a) $\Delta; \psi_a; \Omega \vdash e_1' \triangleright A, k_1, t_1 \Rightarrow \Phi_1$

b) $\Delta; \Phi_a \models e_1$

c) $|e_1'| = e_1$

By a), we can show that for $k_1', t_1' \in \text{fresh}(\mathbb{R})$ where $\Phi_1' = \Phi_1 \land k_1 = k_1' \land t_1 = t_1'$

\[ \Delta; k_1', t_1'; \Phi_a; \Omega \vdash e_1' \triangleright A, k_1', t_1' \Rightarrow \Phi_1' \] (5.31)

By Theorem 48.2 on the second premise, $\exists e_2'$ such that

d) $\Delta; \psi_a; \Omega \vdash e_2' \triangleright \text{list}[n] A, k_2, t_2 \Rightarrow \Phi_2$

e) $\Delta; \Phi_a \models e_2$

f) $|e_2'| = e_2$

By a), we can show that for $i, k_2', t_2' \in \text{fresh}(\mathbb{R})$ where $\Phi_2' = \Phi_2 \land k_2 = k_2' \land t_2 = t_2' \land i = n$

\[ \Delta; i, k_2', t_2'; \Phi_a; \Omega \vdash e_2' \triangleright \text{list}[i] A, k_2', t_2' \Rightarrow \Phi_2' \] (5.32)

Then, we can conclude as follows

1.

\[ k_1', t_1', k_2', t_2' \in \text{fresh}(\mathbb{R}) \land i \in \text{fresh}(\mathbb{N}) \land \Delta; k_1', t_1', i; \psi_a; \Phi_a; \Omega \vdash e_1' \triangleright A, k_1', t_1' \Rightarrow \Phi_1' \text{ eq. (5.31)} \]

\[ \Delta; i, k_2', t_2'; \psi_a; \Phi_a; \Omega \vdash e_2' \triangleright \text{list}[i] A, k_2', t_2' \Rightarrow \Phi_2' \text{ eq. (5.32)} \]

\[ \Phi_2'' = (\Phi_2' \land n + 1 \equiv (i + 1) \land k_1 + k_2 \equiv k_1' + k_2' \land t_1' + t_2' \equiv t_1 + t_2) \]

\[ \Delta; \psi_a; \Phi_a; \Omega \vdash \text{cons}_C(e_1', e_2') \downarrow \text{list}[n + 1] A, k_1 + k_2, t_1 + t_2 \Rightarrow \exists k_1', t_1' :: \mathbb{R}. (\Phi_1' \land \exists k_2', t_2' :: \mathbb{R}. \exists i :: \mathbb{N}. \Phi_2'') \]

alg-u-cons-\dagger

146
2. Using b) and e) for the substitutions $k'_i = k_i$ and $t'_i = t_i$ for the fresh costs and $i = n$ for the size of the tail.

3. Using c) and f), $|\text{cons}_C(e'_1, e'_2)| = \text{cons}_C(e_1, e_2)$

**Proof of Theorem 48.2:**

**Case**

\[
\begin{array}{c}
\Delta; \Phi_a; \Gamma \vdash e_1 \in \tau \subseteq \tau \\
\Delta; \Phi_a; \Gamma \vdash \text{der } e_1 \in \text{der } e_2 \subseteq \tau \\
\hline
\Delta; \Phi_a; \Gamma \vdash c-\text{der}
\end{array}
\]

By Theorem 48.2 on the premise, $\exists e'_1, e'_2$ such that

a) $\Delta; \Phi_a; \Gamma \vdash e'_1 \in \tau \subseteq \tau \Rightarrow \Phi$

b) $\Delta; \Phi_a \models \Phi$

c) $|e'_1| = e_1$ and $|e'_2| = e_2$

Then, we can conclude by using a), b) and c) as follows:

\[
\begin{array}{c}
\Delta; \Phi_a; \Gamma \vdash \text{der } e'_1 \in \text{der } e'_2 \subseteq \tau \Rightarrow \Phi \\
\hline
\Delta; \Phi_a; \Gamma \vdash \text{der } e'_1 \in \text{der } e'_2 \subseteq \tau \Rightarrow \Phi
\end{array}
\]

1. By c), $|(\text{der } e'_1 : \tau, t)| = \text{der } |e'_1|$.

2. By b) and Lemma 37.

**Case**

\[
\begin{array}{c}
\Delta; \Phi_a; \Gamma \vdash k'_1 : A \\
\Delta; \Phi_a; \Gamma \vdash k_2 : A \\
\hline
\Delta; \Phi_a; \Gamma \vdash \text{switch } e_1 \in \text{switch } e_2 \subseteq t_1 - k_2 : U A
\end{array}
\]

By Theorem 48.1 on the first premise, $\exists e'_1$ such that

a) $\Delta; \Phi_a; \Gamma_1 \vdash e'_1 \in A_1, k_1, t_1 \Rightarrow \Phi_1$

b) $\Delta; \Phi_a \models \Phi_1$

c) $|e'_1| = e_1$

By a), we can show that for $k'_1, t'_1 \in \text{fresh}(\mathbb{R})$ where $\Phi'_1 = \Phi_1 \land k_1 \equiv k'_1 \land t_1 \equiv t'_1$

\[
\Delta; k'_1, t'_1; \Phi_a; \Gamma_1 \vdash e'_1 \in A_1, k'_1, t'_1 \Rightarrow \Phi'_1
\]

(5.33)
By Theorem 48.1 on the second premise, \( \exists e'_2 \) such that

d) \( \Delta; \cdot \Phi_a ; [\Gamma] \vdash e'_2 \downarrow A_2, k_2, t_2 \Rightarrow \Phi_2 \)
e) \( \Delta; \Phi_a \models \Phi_2 \)
f) \( |e'_2| = e_2 \)

By d), we can show that for \( k'_2, t'_2 \in \text{fresh}(\mathbb{R}) \) where \( \Phi'_2 = \Phi_2 \land k_2 \equiv k'_2 \land t_2 \equiv t'_2 \)

\[
\Delta; k'_2, t'_2; \Phi_a; [\Gamma] \vdash e'_2 \downarrow A_2, k'_2, t'_2 \Rightarrow \Phi'_2 \tag{5.34}
\]

Then, we can conclude as follows

1. By using eqs. (5.33) and (5.34):

\[
\begin{array}{c}
k'_1, t'_1, k'_2, t'_2 \in \text{fresh}(\mathbb{R}) \\
\Delta; k'_1, t'_1, \psi_a ; [\Gamma] \vdash e'_1 \downarrow A, k'_1, t'_1 \Rightarrow \Phi'_1 \\
\Delta; k'_2, t'_2, \psi_a ; [\Gamma] \vdash e'_2 \downarrow A, k'_2, t'_2 \Rightarrow \Phi'_2 \\
\end{array}
\]

\[\text{alg-r-switch} \downarrow.\]

2. By using b) and c) and the substitutions \( k'_1 = k_1 \) and \( t'_1 = t_1 \) for the fresh costs where \( t = t_1 - k_2 \).

3. By c) and f), we get \( |\text{switch} e'_1| = \text{switch} |e_1| \)

\[
\Delta; \Phi_a; [\Gamma] \vdash e \equiv e' \leq t : \tau \\
\Delta; \Phi_a \models t \ll t'
\]

Case

\[\Delta; \Phi_a; [\Gamma] \vdash t \ll t' \Rightarrow \Phi \]

By Theorem 48.2 on the first premise, \( \exists e'_1, e'_2 \) such that

a) \( \Delta; ; \Phi_a ; [\Gamma] \vdash e'_1 \downarrow \tau, t \Rightarrow \Phi_1 \)
b) \( \Delta; \Phi_a \models \Phi_1 \)
c) \( |e'_1| = e \) and \( |e'_2| = e' \)

By Theorem 43 on the second premise,

d) \( \Delta; \Phi_a \models \tau \equiv \tau' \Rightarrow \Phi_2 \)
e) \( \Delta; \Phi_a \models \Phi_2. \)

Then, we can conclude as follows

1. By using a) and d)

\[
\begin{array}{c}
\Delta; ; \Phi_a ; [\Gamma] \vdash e'_1 \downarrow \tau, t \Rightarrow \Phi_1 \\
\Delta; \Phi_a ; [\Gamma] \vdash (e'_1 : \tau, t) \downarrow (e'_2 : \tau, t) \uparrow \tau \Rightarrow \Phi_1 [\Gamma, t, \Phi_1] \\
\Delta; \Phi_a ; [\Gamma] \vdash (e'_1 : \tau, t) \downarrow (e'_2 : \tau, t) \downarrow \tau' \Rightarrow \Phi_1 \land \Phi_2 \land t \ll t' \\
\end{array}
\]

\[\text{alg-r-anno} \uparrow, \text{alg-r-anno} \downarrow.\]

2. By using b), e) and the third premise, we can show that \( \delta; \Phi_a \models \Phi_1 \land \Phi_2 \land t \ll t' \)

3. By c), \( |(e'_1 : \tau, t)| = e \) and \( (e'_2 : \tau, t) = e' \)
By Theorem 48.2 on the first premise, \( \exists \bar{e}_1, \bar{e}_2 \) such that

a) \( \Delta; \Phi_a; \Gamma \vdash \bar{e}_1 \odot \bar{e}_2 \iff \bar{t}_1 \vdash \tau_1 \mathrel{\text{diff}(t)} \tau_2 \)

By Theorem 48.2 on the second premise, \( \exists \bar{e}_2, \bar{e}_3 \) such that

d) \( \Delta; \Phi_a; \Gamma \vdash \bar{e}_2 \odot \bar{e}_3 \downarrow \tau_1, t_2 \Rightarrow \Phi_2 \)

e) \( \Delta; \Phi_a \models \Phi_2 \)

By d), we can show that for \( \bar{t}_2' \in \text{fresh}(\mathbb{R}) \) where \( \Phi_2' = \Phi_2 \land t_2 = t_2' \)

\[
\Delta; \bar{t}_2'; \Phi_a; \Gamma \vdash \bar{e}_2 \odot \bar{e}_3' \downarrow \tau_1, \bar{t}_2' \Rightarrow \Phi_2'
\]  

(5.35)

Then, we can conclude as follows

1. 

\[
\Delta; \Phi_a; \Gamma \vdash \bar{e}_1 \odot \bar{e}_2 \downarrow \tau_1 \mathrel{\text{diff}(t)} \tau_2, t_1 \Rightarrow \Phi_1
\]

\[
\Delta; \Phi_a; \Gamma \vdash (\bar{e}_1 : \tau_1 \mathrel{\text{diff}(t)} \tau_2, t_1) \odot (\bar{e}_2 : \tau_2, t_1) \uparrow \tau_1 \mathrel{\text{diff}(t)} \tau_2 \Rightarrow [\cdot], t_1, \Phi_1
\]

\[
t_2' \in \text{fresh}(\mathbb{R}) \quad \Delta; \bar{t}_2'; \Phi_a; \Gamma \vdash \bar{e}_2 \odot \bar{e}_2' \downarrow \tau_1, t_2' \Rightarrow \Phi_2'
\]

where \( E_1 = (\bar{e}_1 : \tau_1 \mathrel{\text{diff}(t)} \tau_2, t_1) \bar{e}_2 \) and \( E_2 = (\bar{e}_2' : \tau_1 \mathrel{\text{diff}(t)} \tau_2, t_1) \bar{e}_2'. \)

2. By using b) and e) and the substitution \( t_2' = t_2 \) for the fresh cost.

3. Using c) and f), \( [(\bar{e}_1 : \tau_1 \mathrel{\text{diff}(t)} \tau_2, t_2)] \bar{e}_2 = e_1 e_2 \) and \( [(\bar{e}_1' : \tau_1 \mathrel{\text{diff}(t)} \tau_2, t_1)] \bar{e}_2' = e_1' e_2'. \)

\[
\Delta; \Phi_a; \Gamma \vdash e_1 \odot e_2 \iff t_1 \vdash \exists i :: S. \tau_1
\]

\[
i :: S, \Delta; \Phi_a; x : \tau_1, \Gamma \vdash e_2 \odot e_2' \iff t_2 \vdash e_2', i \not\in \text{FV}(\Phi_a; \tau_2, t_2)
\]

By Theorem 48.2 on the first premise, \( \exists \bar{e}_1, \bar{e}_2 \) such that

a) \( \Delta; \Phi_a; \Gamma \vdash \bar{e}_1 \odot \bar{e}_2 \downarrow \exists i :: S. \tau_1, t_1 \Rightarrow \Phi_1 \)

b) \( \Delta; \Phi_a \models \Phi_1 \)

c) \( |\bar{e}_1| = e_1 \) and \( |\bar{e}_2| = e_2' \)
By Theorem 48.2 on the second premise, $\exists \tau_2, \tau'_2$ such that

d) $i :: S; \Delta; \Phi_a; x : \tau_1, \Gamma \vdash \tau_2 \cup \tau'_2 \downarrow \tau_2, t_2 \Rightarrow \Phi_2$

e) $i :: S; \Delta; \Phi_a \models \Phi_2$
f) $|\tau_2| = e_2$ and $|\tau'_2| = e'_2$

By d), we can show that for $t'_2 \in \text{fresh}(\mathbb{R})$ where $\Phi'_2 = \Phi_2 \land t_2 \equiv t'_2$

$$i :: S; \Delta; t'_2; \Phi_a; x : \tau_1, \Gamma \vdash \tau_2 \cup \tau'_2 \downarrow \tau_2, t'_2 \Rightarrow \Phi'_2$$ (5.36)

Then, we can conclude as follows

1. 

$$
\begin{align*}
\Delta; :: \Phi_a; \Gamma \vdash \tau_1 \cup \tau'_1 \downarrow \exists i :: S. \tau_1, t_1 \Rightarrow \Phi_1 \quad \text{(a) alg-r-anno-\uparrow} \\
\Delta; :: \Phi_a; \Gamma \vdash E_1 \downarrow E'_1 \uparrow \exists i :: S. \tau_1 \Rightarrow \{ \}, t_1, \Phi_1 \\
i :: S; \Delta; t'_2; \Phi_a; x : \tau_1, \Gamma \vdash \tau_2 \cup \tau'_2 \downarrow \tau_2, t'_2 \Rightarrow \Phi'_2 \quad \text{(eq. (5.36))}
\end{align*}

$$
\Phi' = \exists \tau_2 :: \exists \Phi_1, \Phi_2 \land t_1 + t'_2 \equiv t_1 + t_2

\Delta; :: \Phi_a; \Gamma \vdash \text{unpack } E_1 \text{ as } (x, i) \text{ in } \tau_2 \cup \text{unpack } E'_1 \text{ as } (x, i) \text{ in } \tau'_2 \downarrow \tau_2, t_1 + t_2 \Rightarrow \Phi'$$

where $E_1 = (\tau_1 : \exists i :: S. \tau_1, t_1)$ and $E_2 = (\tau'_1 : \exists i :: S. \tau_1, t_1)$

2. By using b) and e) and the substitution $t'_2 = t_2$ for the fresh cost.

3. Using c) and f), $\text{unpack } (\tau_1 : \exists i :: S. \tau_1, t_1)$ as $(x, i)$ in $\tau_2] = \text{unpack } e_1$ as $(x, i)$ in $e_2$ and $|\text{unpack } (\tau'_1 : \exists i :: S. \tau_1, t_1)$ as $(x, i)$ in $\tau'_2] = \text{unpack } e'_1$ as $(x, i)$ in $e'_2$.

\square

6 Experimental Evaluation

In this section, we present example programs that we have typechecked with BiRelCost.

**Example (map)** Consider the standard list map function that applies the mapping function map to all the elements of the list.

$\text{\text{fix} map}(f)$. $\text{\text{fix} map}(f)$.  

$\lambda$. $\lambda$. $\lambda$. $\lambda$.  

\text{\text{case} l of nil} \rightarrow \text{nil} 

\text{\text{| h :: tl} \rightarrow \text{cons}(f \ h, \ \text{map f[] tl})} 

Two runs of map can be typed with 0 cost and the following type:

$$\vdash \text{\text{map} \circ map} \subseteq 0 : \forall t. (\exists \tau_1 \quad \text{\text{diff}(t) \rightarrow \text{\text{\tau}_2)}) \rightarrow \forall \alpha. \text{\text{list[n]}^\wedge \tau_1 \quad \text{\text{diff}(t)} \rightarrow \text{\text{\text{list[n]}^\wedge \tau_2}}.$$ (6.1)

**Example (filter)** Consider the standard list filter function that goes over the list and only returns the list elements that satisfy the filtering function $f$.

$\text{\text{fix} filter}(f)$. $\text{\text{fix} filter}(f)$.  

$\lambda$. $\lambda$. $\lambda$. $\lambda$.  

\text{\text{case} l of nil} \rightarrow \text{\text{pack} nil} 

\text{| h :: tl} \rightarrow \text{\text{let} r' = filter f[]} tl \text{ in} 

\text{\text{let} b = f \ h \ \text{\text{in \ unpack} r' \ as \ r \ in \ \text{\text{if \ b \ then \ pack cons}(h, r) \ \text{else} \ \text{\text{pack} r})}
Two runs of $\text{filter}$ can be typed with 0 cost and the following type:
\[
\vdash \text{filter} \uplus \text{filter} \preceq 0 : \forall t. (\square (r \xrightarrow{\text{diff}(t)} \text{bool})) \to \forall n, \alpha. \text{list}[n]^{\alpha} r \xrightarrow{\text{diff}(t \cdot \alpha)} \exists j : S.U \ (\text{list}[j] \mid |r|).
\] (6.2)

Example (square-and-multiply) Consider the square-and-multiply algorithm, a fast way of computing the positive powers of a number based on the observation that $x^m = x \cdot (x^2)^{\frac{m-1}{2}}$ when $m$ is odd, and $x^m = (x^2)^{\frac{m}{2}}$ when $m$ is even. The following function, $\text{sam}$, implements this idea, assuming that $m$ is stored in binary form in a list $l$ of 0s and 1s, with the least significant bit at the head.

\[
\text{fix } \text{sam}(x) = \Lambda . \Lambda . \Lambda . \lambda l. \text{case } l \text{ of } \text{nil} \to \text{contra} | b :: bs \to \text{case } bs \text{ of } \text{nil} \to \text{if } x = 0 \text{ then } 1 \text{ else } x | . \vdash \text{sam} x \mid \mid \mid \mid \text{bs in } \text{if } b = 0 \text{ then } r^2 \text{ else } x \cdot r^2
\]

Assuming that multiplication ($\cdot$) and equality ($=$) are primitives with types $(\text{int} \times \text{int}) \xrightarrow{\text{exec}(1,1)} \text{int}$ and $(\text{int} \times \text{int}) \xrightarrow{\text{exec}(1,1)} \text{bool}$, respectively, two runs of $\text{sam}$ can be typed with 0 cost and the following type:
\[
\vdash \text{sam} \uplus \text{sam} \preceq 0 : \text{int} \xrightarrow{\text{diff}(0)} \forall n > 0, \alpha :: \text{N.} \text{list}[n]^{\alpha} \text{int} \xrightarrow{\text{diff}(\alpha)} \text{U int}
\] (6.3)

Example (constant-time comparison) Consider the following comparison function $\text{comp}$ that checks the equality of two passwords represented as equal-length lists of bits. In BiRelCost, we can show that $\text{comp}$ is constant-time, i.e. its relative cost wrt. itself is 0.

\[
\text{fix } \text{comp}(\cdot) = \Lambda . \Lambda . \Lambda . \lambda x. \lambda M. \text{case } l \text{ of } \text{nil} \to \text{true} | h :: tl \to \text{case } l_2 \text{ of } \text{nil} \to \text{false} | h_2 :: tl_2 \to \text{boolAnd} (\text{comp} (\cdot) \mid \mid \mid \mid tl_1, tl_2, \text{eq} (h_1, h_2))
\]

Assuming that $\text{boolAnd}$ and $\text{eq}$ are constant-time primitives with the same type $(\text{U bool} \times \text{U bool}) \xrightarrow{\text{diff}(0)} \text{U bool}$, two runs of $\text{comp}$ can be typed with 0 cost and the following type:
\[
\vdash \text{comp} \uplus \text{comp} \preceq 0 : \text{unit} \xrightarrow{\text{diff}(0)} \forall n > 0, \alpha :: \text{N.} \text{list}[n]^{\alpha} \text{U int} \times \text{list}[n]^{\alpha} \text{U int} \xrightarrow{\text{diff}(0)} \text{U bool}
\] (6.4)

Example (two-dimensional count) Consider the following function $\text{2Dcount}$ that counts the number of rows of a matrix $M$ (represented as a list of lists in row-major form) that both contain a key $x$ and satisfy a predicate $p$. The function takes as argument another function $\text{find}$ that returns 1 when a given row $l$ contains $x$, else returns 0.

\[
\text{fix } \text{2Dcount}(\text{find}) = \Lambda . \lambda x. \lambda M. \text{case } M \text{ of } \text{nil} \to 0 | l :: M' \to \text{let } r = \text{2Dcount} (\mid ) \mid \mid \mid \text{find} x M' \text{ in } \text{let } r' = \text{find} x l \text{ in } \text{if } p l \text{ then } r + r' \text{ else } r
\]

Consider the following two different implementations of $\text{find}$.
\[
\begin{align*}
\text{fix } \text{find1}(x) &. \Lambda \lambda l. \text{case } l \text{ of} \\
\quad \text{nil} & \rightarrow 0 \\
\quad h :: tl & \rightarrow \text{if } h = x \text{ then } 1 \text{ else } \text{find1 } x \] tl
\end{align*}
\]

\[
\begin{align*}
\text{fix } \text{find2}(x) &. \Lambda \lambda l. \text{case } l \text{ of} \\
\quad \text{nil} & \rightarrow 0 \\
\quad h :: tl & \rightarrow \text{if } (\text{find2 } x \] tl) = 1 \text{ then } 1 \text{ else if } (h = x) \text{ then } 1 \text{ else } 0
\end{align*}
\]

where \text{find1} and \text{find2} can be given the following weakest relational type:

\[
\vdash \text{find1} \preceq \text{find2} \preceq 0 : U(\text{int} + \forall n :: N. (\text{list}[n] \text{ int}) \text{ exec}(\text{cons}(x), 4 \cdot n) \rightarrow \text{int})
\]

\[
(6.5)
\]

Assuming that the predicate \(p\) is a constant-time primitive with type \(U(\forall n, t \text{ exec}(\text{list}[n] \cdot n) :: S. \text{ int} \text{ exec}(t, t) \rightarrow \text{bool})\), then we can type \(2\text{Dcount find1}\) and \(2\text{Dcount find2}\) with 0 cost as follows:

\[
\vdash 2\text{Dcount find1} \preceq 2\text{Dcount find2} \preceq 0 : \text{unit} \rightarrow \forall i, j :: N. (\text{list}[n] \cdot 0) \text{ diff}(0) \rightarrow U(\text{int})
\]

\[
(6.6)
\]

\textbf{Example (Mergesort)} Consider the standard mergesort function.

\[
\begin{align*}
\text{fix } \text{msort}(\cdot) &. \Lambda \lambda x. \text{case } x \text{ of} \\
\quad \text{nil} & \rightarrow \text{nil} \\
\quad h :: tl & \rightarrow \text{case } tl \text{ of} \\
\quad \text{nil} & \rightarrow \text{cons}(h, \text{nil}) \\
\quad r & \rightarrow \text{let } r = \text{bsplit } tl \text{ in } \text{unpack } r \text{ as } r' \text{ as } r'' \\
\quad & \text{clet } r'' \text{ as } z \text{ in } \text{merge } [] [] (\text{msort } [] [] (\pi_1 z), \text{msort } [] [] (\pi_2 z))
\end{align*}
\]

There are two helper functions: \text{bsplit} and \text{merge}. The helper function \text{bsplit} splits an input list into two nearly equal lists by alternating the input’s elements to the two outputs.

\[
\begin{align*}
\text{fix } \text{bsplit}(\cdot) &. \Lambda \lambda l. \text{case } l \text{ of} \\
\quad \text{nil} & \rightarrow (\text{nil}, \text{nil}) \\
\quad h :: tl & \rightarrow \text{case } tl \text{ of} \\
\quad \text{nil} & \rightarrow \langle \text{cons}(h, \text{nil}), \text{nil} \rangle \\
\quad r & \rightarrow \text{let } r = \text{bsplit } tl \text{ in } \text{unpack } r \text{ as } r' \text{ as } r'' \\
\quad & \text{clet } r'' \text{ as } z \text{ in } \text{merge } [] [] (\text{msort } [] [] (\pi_1 z), \text{msort } [] [] (\pi_2 z))
\end{align*}
\]

It can be given the following relational type:

\[
\begin{align*}
\text{bsplit} : (\text{unit} \rightarrow \forall n, \alpha :: N. \text{list}[n] \cdot \tau) & \rightarrow \exists \beta :: N. (\text{list}[\lfloor n/2 \rfloor] \cdot \beta) \times \text{list}[\lceil n/2 \rceil] \cdot \tau)
\end{align*}
\]

\[
(6.7)
\]

The helper function \text{merge} takes two sorted lists and merges them into a new sorted list.

\[
\begin{align*}
\text{fix } \text{merge}(\cdot) &. \Lambda \lambda y \text{ case } x \text{ of} \\
\quad \text{nil} & \rightarrow y \\
\quad a :: as & \rightarrow \text{case } y \text{ of} \\
\quad \text{nil} & \rightarrow x \\
\quad b :: bs & \rightarrow \text{if } a \leq b \text{ then } \text{cons}(a, \text{merge } [] [] as y) \text{ else } \text{cons}(b, \text{merge } [] [] x bs)
\end{align*}
\]
Assuming that the comparison operation (≤) is unit cost primitive with type (int × int) \(\xrightarrow{\text{exec}(1.1, \text{bool})}\), then we can type \(\text{merge}\) as follows:

\[
\vdash^0 \text{merge} : \text{int} \to \forall n, m : \mathbb{N}. (\text{list}[n] \times \text{list}[m]) \text{int} \quad (6.8)
\]

Using the unary type of \(\text{merge}\), two runs of \(\text{merge}\) can be given a relational type by encapsulating into an unrelated type. Then, we can give \(\text{msort}\) the following relational type

\[
\text{msort} : \square (\forall n, \alpha : \mathbb{N}. \text{list}[n] \alpha \times \text{list}[n] \alpha) \text{int} \quad (6.9)
\]

Example (selection sort) Consider the standard selection sort algorithm that finds the smallest among a value and the elements in a list using the function \(\text{select}\) and then sorts the remaining list recursively. In BiRelCost, we can show that \(\text{ssort}\) is a constant time algorithm, i.e. its relative cost is 0.

\[
\vdash \text{ssort} : \text{unit} \quad (6.10)
\]

Example (approximate sum) Consider two implementations of the mean calculation of a list of numbers. The first function computes the sum of a list of numbers and divides the sum by the length of the list whereas the second function (its approximate version) only computes the sum of the half of the elements, divides this sum by the total length of the list and then doubles the result afterwards.

\[
\vdash \text{sum} : \text{unit} \quad (6.11)
\]
fix \text{sumAppr}(\text{acc}) \cdot \Lambda\cdot \Lambda\cdot \lambda\text{l}. \text{case l of}
| \text{nil} \rightarrow \text{acc}
| \text{h :: tl} \rightarrow \text{case tl of}
| \text{nil} \rightarrow \text{h + acc}
| \text{h' :: tl'} \rightarrow \text{sum} (\text{h' + acc})[[[]]] \text{ tl'}

\vdash \text{sum} \odot \text{sumAppr} \preceq 0 : (U \text{ int}) \xrightarrow{\text{diff}(0)} \forall n::\mathbb{N}. \text{list}[n]^\alpha U \text{ int} \xrightarrow{\text{diff}(n)} U \text{ int}.

\textbf{Example (balanced fold)}\ The following balanced fold function is often used in incremental computing. Given an \textit{associative and commutative} binary function \textit{f}, a list can be folded by splitting it into two nearly equal sized lists, folding the sublists recursively and then applying \textit{f} to the two results.

\text{fix bfold}(\cdot) \cdot \Lambda\cdot \Lambda\cdot \lambda\text{l}. \text{case l of}
| \text{nil} \rightarrow 0
| \text{h :: tl} \rightarrow \text{case tl of}
| \text{nil} \rightarrow \text{h}
| \text{r :: _} \rightarrow r = \text{bsplit }[[[]]] l \text{ in unpack } r \text{ as } r' \text{ in clet } r' \text{ as } z \text{ in } f (\text{bfold }[[[]]()] ) (\text{bfold }[[[]]()] )

The helper function \text{bsplit} splits an input list into two nearly equal lists by alternating the input’s elements to the two outputs. Its relational type is shown at eq. (6.7). Then, like \text{msort}, we can give \textbf{bfold} the following relational type

\vdash \text{bfold} \odot \text{bfold} \preceq 0 : \Box (\Box (U ((\text{int} \times \text{int}) \xrightarrow{\text{exec}(1,2)} \text{int})) \xrightarrow{\text{diff}(0)} \forall n,\alpha::\mathbb{N}. \text{list}[n]^\alpha U \text{ int} \xrightarrow{\text{diff}(P(n,\alpha))} U \text{ int}) \xrightarrow{\text{P}(n,\alpha)} U \text{ int}

\text{where } P(n,\alpha) = \sum_{i=0}^{\lfloor \log_2(n) \rfloor} \min(\alpha, 2^\lfloor \log_2(n) \rfloor - i) \in O(\alpha + \alpha \cdot \log_2(n/\alpha))

Example (list append) Consider the standard list \textbf{append} function that appends a list to another list.

\text{fix append}(\cdot)\cdot \Lambda\cdot \Lambda\cdot \Lambda\cdot \lambda l_1. \lambda l_2. \text{case l_1 of}
| \text{nil} \rightarrow l_2
| \text{h :: tl} \rightarrow \text{cons}(\text{h}, \text{append }[[[]]] [] [] [] t l l_2)

We can type two runs of \textbf{append} as follows:

\vdash \text{append} \odot \text{append} \preceq 0 : \text{unit}_r \xrightarrow{\text{diff}(r)} \forall i,\alpha,\beta::\mathbb{N}. \text{list}[i]^\alpha U \text{ int} \xrightarrow{\text{diff}(0)} \text{list}[j]^\beta U \text{ int} \xrightarrow{\text{diff}(0)} \text{list}[i + j]^\alpha + \beta U \text{ int} .
Example (list reverse)  Consider the standard list reverse function that reverses the list.

\[
\text{fix } \texttt{rev}(\_).\lambda\_\_\_.\lambda\texttt{acc}.\texttt{case } l_1 \text{ of } \\
\text{nil } \rightarrow l_2 \\
| \ h :: tl \ → \ \texttt{rev} ([]) [\_\_\_] [\_\_] tl \ \texttt{cons}(h, acc) \\
\]

We can type two runs of \texttt{rev} as follows:

\[\vdash \texttt{rev} \circ \texttt{rev} \lesssim 0 : \texttt{unit,r} \xrightarrow{\text{diff}(0)} \forall i,j,\alpha,\beta::\texttt{N}.\texttt{list}[\_][\_\_\_] \texttt{U} \int \xrightarrow{\text{diff}(0)} \texttt{list}[\_][\_\_\_] \texttt{U} \int \xrightarrow{\text{diff}(0)} \texttt{list}[i+j]^{\alpha+\beta} \texttt{U} \int .\]

Example (list flatten)  Consider the standard list flatten function that flattens a lists of lists.

\[
\text{fix } \texttt{flatten}(\_).\lambda\_\_\_.\lambda\texttt{M}.\texttt{case } M \text{ of } \\
\text{nil } \rightarrow \text{nil} \\
| l :: M' \ → \ \texttt{let } r = \texttt{flatten} ([]) [\_\_\_] [\_\_] M' \ \texttt{in} \\
\texttt{append} ([]) [\_\_\_] [\_\_] l r \\
\]

We can type two runs of \texttt{flatten} as follows:

\[\vdash \texttt{flatten} \circ \texttt{flatten} \lesssim 0 : \texttt{unit,r} \xrightarrow{\text{diff}(0)} \forall i,j,\alpha,\beta::\texttt{N}.\texttt{list}[\_][\_\_\_] (\texttt{list}[\_][\_\_\_] \texttt{U} \int) \xrightarrow{\text{diff}(0)} \texttt{list}[i \cdot j]^+j \texttt{U} \int .\]

Example (list zip)  Consider the standard list zip function as follows:

\[
\text{fix } \texttt{zip}(\_).\lambda\_\_\_.\lambda x.\texttt{case } (\pi_1 x) \text{ of } \\
\text{nil } \rightarrow \text{nil} \\
| h_1 :: tl_1 \ → \ \texttt{case } (\pi_2 x) \text{ of } \texttt{nil } \rightarrow \cdots \\
| h_2 :: tl_2 \ → \ \texttt{let } rest = \texttt{zip} f[\_\_\_] [\_\_] (tl_1, tl_2) \ \texttt{in} \\
\texttt{let } f h = f \langle h, h' \rangle \ \texttt{in} \ \texttt{cons}(f h, rest) \\
\]

It can be given the following relational type:

\[\vdash \texttt{zip} \circ \texttt{zip} \lesssim 0 : (\forall U \langle (\tau_1 \times \tau_2) \xrightarrow{\text{exec}(1.1)} \tau_3 \rangle) \xrightarrow{\text{diff}(0)} \forall n,\alpha,\beta::\texttt{N}.(\texttt{list}[n]^\alpha \tau_1 \times \texttt{list}[n]^\beta \tau_2) \xrightarrow{\text{diff}(0)} \texttt{list}[n]^{\min(n,\alpha+\beta)} \tau_3 \]

(6.14)

Example (multi_sort)  Consider the standard list multi_sort function.

\[
\text{fix } \texttt{multi_sort}(\_).\lambda\_\_\_.\lambda l_1 \lambda l_2 \lambda l_3 \lambda l_4 \lambda l_5 \lambda l_6 \lambda l_7 \lambda l_8 \lambda l_9 \lambda l_7. \\
\texttt{let } l_1 = \texttt{sSORT} ([]) [\_\_\_] [\_\_] l \ \texttt{in} \\
\texttt{unpack } l_1 \ \texttt{as} l_2 \ \texttt{in} \\
\texttt{let } l_3 = \texttt{rev} ([]) [\_\_\_] [\_\_] [\_\_] l_2 \ \texttt{nil} \ \texttt{in} \\
\texttt{let } l_4 = \texttt{sSORT} ([]) [\_\_\_] [\_\_] [\_\_] l_3 \ \texttt{in} \\
\texttt{unpack } l_4 \ \texttt{as} l_5 \ \texttt{in} \\
\texttt{let } l_5 = \texttt{rev} ([]) [\_\_\_] [\_\_] [\_\_] [\_\_] l_4 \ \texttt{nil} \ \texttt{in} \\
\texttt{let } l_6 = \texttt{sSORT} ([]) [\_\_\_] [\_\_] [\_\_] [\_\_] l_5 \ \texttt{nil} \ \texttt{in} \\
\texttt{let } l_7 = \texttt{msORT} ([]) [\_\_\_] [\_\_] l_6 \ \texttt{in} \\
\texttt{let } l_8 = \texttt{msORT} ([]) [\_\_\_] l_7 \ \texttt{in} \\
\texttt{let } l_9 = \texttt{msORT} ([]) [\_\_] l_8 \ \texttt{in} \\
\]

We can give \texttt{multi_sort} the following relational type

\[\vdash \texttt{multi_sort} \circ \texttt{multi_sort} \lesssim 0 : (\forall U \langle (\forall n,\alpha::\texttt{N}.\texttt{list}[n]^\alpha \texttt{U} \int) \xrightarrow{\text{diff}(3+Q(n,\alpha))} U \langle \texttt{list}[n] \int \rangle \rangle) \]

(6.15)

where \(Q(n,\alpha) = \sum_{i=0}^{\lceil \log_2(n) \rceil} 5 \cdot \left( \frac{\alpha}{2^i} \right) \cdot \min(\alpha, 2^{\lceil \log_2(n) \rceil} - i) \in O(n \cdot (1 + \log_2(\alpha)))\)
Example (ssort_list)

fix ssort_list(\Lambda.\Lambda.\Lambda.M).
let \(l_1 = \text{ssort}()\)\ \[\]\ \[\]\ \(l\) in
unpack \(l_1\) as \(l_2\) in
let \(l_3 = \text{rev}()\)\ \[\]\ \[\]\ \[\]\ \(l_2\) \(\)\ni in
let \(l_4 = \text{ssort}()\)\ \[\]\ \(l_3\) in
unpack \(l_4\) as \(l_5\) in
let \(l_6 = \text{rev}()\)\ \[\]\ \[\]\ \[\]\ \[\]\ \(l_5\) \(\)\ni in
let \(l_7\) = \(l_9\) \(\)\ni

We can give \(\text{ssort\_list}\) the following relational type

\[\vdash \text{ssort\_list} \subseteq 0 : \text{unit} \xrightarrow{\text{diff}(0)} \forall n, \alpha :: \text{N. list}[n]^{\alpha} \xrightarrow{\text{int}} \exists \beta :: \text{N. list}[2 \times n]^{\beta} \xrightarrow{\text{int}} \text{int} \] (6.16)

6.1 Experimental results

Table 1 demonstrates experimental evaluation results for all the benchmark programs described above. In all cases, the total typechecking (including existential elimination and SMT solving) takes less than 1s. A "-" indicates a negligible value. Our experiments were performed on a 3.10GHz 2-core Intel Core i5 processor with 8 GB of RAM.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Total time(s)</th>
<th>Type-checking</th>
<th>Existential elim.</th>
<th>Constraint solving</th>
<th>Loc</th>
<th># of Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>0.167</td>
<td>0.005</td>
<td>-</td>
<td>0.162</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>append</td>
<td>0.173</td>
<td>0.005</td>
<td>-</td>
<td>0.167</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>rev</td>
<td>0.170</td>
<td>0.005</td>
<td>-</td>
<td>0.164</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>map</td>
<td>0.172</td>
<td>0.005</td>
<td>-</td>
<td>0.167</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>comp</td>
<td>0.171</td>
<td>0.005</td>
<td>-</td>
<td>0.166</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>sam</td>
<td>0.174</td>
<td>0.005</td>
<td>-</td>
<td>0.169</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>find</td>
<td>0.172</td>
<td>0.005</td>
<td>-</td>
<td>0.167</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>2Dcount</td>
<td>0.168</td>
<td>0.005</td>
<td>-</td>
<td>0.163</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>ssort</td>
<td>0.181</td>
<td>0.005</td>
<td>-</td>
<td>0.175</td>
<td>26</td>
<td>3</td>
</tr>
<tr>
<td>bsplit</td>
<td>0.180</td>
<td>0.005</td>
<td>-</td>
<td>0.175</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>flatten</td>
<td>0.173</td>
<td>0.005</td>
<td>-</td>
<td>0.167</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>appSum</td>
<td>0.171</td>
<td>0.005</td>
<td>-</td>
<td>0.166</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>merge</td>
<td>0.207</td>
<td>0.005</td>
<td>-</td>
<td>0.201</td>
<td>26</td>
<td>3</td>
</tr>
<tr>
<td>zip</td>
<td>0.187</td>
<td>0.005</td>
<td>-</td>
<td>0.181</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>msort</td>
<td>0.384</td>
<td>0.008</td>
<td>0.003</td>
<td>0.373</td>
<td>29</td>
<td>3</td>
</tr>
<tr>
<td>bfold</td>
<td>0.216</td>
<td>0.006</td>
<td>0.001</td>
<td>0.208</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>multi_sort</td>
<td>0.958</td>
<td>0.012</td>
<td>0.015</td>
<td>0.930</td>
<td>81</td>
<td>9</td>
</tr>
<tr>
<td>ssort_list</td>
<td>0.207</td>
<td>0.006</td>
<td>0.004</td>
<td>0.197</td>
<td>72</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1: Statistics for BiRelCost example programs. All times are in seconds.