The Calculus of Linear Constructions — Technical Report

Anonymous Author(s)

January 3, 2022

Contents

1 Introduction

This extended report is meant to accompany our paper of the same title. Here, we describe the meta-theory of CLC and its proofs in detail. All the results presented here have been formalized and proven correct in the Coq Proof Assistant.

2 Syntax of CLC (`clc_ast.v`)

i	$::= 0 \mid 1 \mid 2 \ldots$	universe levels
s, t	$::= U \mid L$	sorts
m, n, A, B, M	$::= U_i \mid L_i \mid x$	expressions
	$\mid\ (x :_s A) \to B$	
	$\mid\ (x :_s A) \multimap B$	
	$\mid\ \lambda x :_s A.n$	
	$\mid\ m\ n$	

3 Reduction and Equality of CLC (`clc_ast.v`)

$$\frac{m_1 \leadsto^* n \qquad m_2 \leadsto^* n}{m_1 \equiv m_2 : A}\textsc{Join} \qquad \frac{}{(\lambda x :_s A.m)\ n \leadsto m[n/x]}\textsc{Step-}\beta \qquad \frac{A \leadsto A'}{\lambda x :_s A.m \leadsto \lambda x :_s A'.m}\textsc{Step-}\lambda\textsc{L}$$

$$\frac{m \leadsto m'}{\lambda x :_s A.m \leadsto \lambda x :_s A.m'}\textsc{Step-}\lambda\textsc{R} \qquad \frac{A \leadsto_p A'}{(x :_s A) \to B \leadsto (x :_s A') \to B}\textsc{Step-L}{\to}$$

$$\frac{B \leadsto_p B'}{(x :_s A) \to B \leadsto (x :_s A) \to B'}\textsc{Step-R}{\to} \qquad \frac{A \leadsto_p A'}{(x :_s A) \multimap B \leadsto (x :_s A') \multimap B}\textsc{Step-L}{\multimap}$$

$$\frac{B \leadsto_p B'}{(x :_s A) \multimap B \leadsto (x :_s A) \multimap B'}\textsc{Step-R}{\multimap} \qquad \frac{m \leadsto m'}{m\ n \leadsto m'\ n}\textsc{Step-AppL} \qquad \frac{n \leadsto n'}{m\ n \leadsto m\ n'}\textsc{Step-AppR}$$

4 Confluence of CLC (`clc_confluence.v`)

4.1 Parallel Reduction

To prove the confluence property of CLC, we employ the standard technique utilizing parallel reductions.

$$\frac{}{x \rightsquigarrow_p x}\textsc{PStep-Var} \qquad \frac{}{s_i \rightsquigarrow_p s_i}\textsc{PStep-Sort} \qquad \frac{A \rightsquigarrow_p A' \qquad m \rightsquigarrow_p m'}{\lambda x :_s A.m \rightsquigarrow_p \lambda x :_s A'.m'}\textsc{PStep-}\lambda$$

$$\frac{m \rightsquigarrow_p m' \qquad n \rightsquigarrow_p n'}{m\ n \rightsquigarrow_p m'\ n'}\textsc{PStep-App} \qquad \frac{m \rightsquigarrow_p m' \qquad n \rightsquigarrow_p n'}{(\lambda x :_s A.m)\ n \rightsquigarrow_p m'[n'/x]}\textsc{PStep-}\beta$$

$$\frac{A \rightsquigarrow_p A' \qquad B \rightsquigarrow_p B'}{(x :_s A) \to B \rightsquigarrow_p (x :_s A') \to B'}\textsc{PStep}\to \qquad \frac{A \rightsquigarrow_p A' \qquad B \rightsquigarrow_p B'}{(x :_s A) \multimap B \rightsquigarrow_p (x :_s A') \multimap B'}\textsc{PStep}\multimap$$

4.2 Reduction Lemmas

Here, we prove some simple lemmas concerning $\rightsquigarrow$, $\rightsquigarrow^*$ and substitution.

Definition 4.1. For a term m and a map σ from variables to terms, let $m[\sigma]$ be the term obtained by applying σ uniformly to all free variables in m.

Definition 4.2. For maps σ, τ from variables to terms, we say that σ reduces to τ if for any variable x there exists a reduction $(\sigma\ x) \rightsquigarrow^* (\tau\ x)$. We write $\sigma \rightsquigarrow^* \tau$ when it is clear from context that σ, τ are maps and not terms.

Lemma 4.1. *For terms m, n and a map σ from variables to terms, if there exist a step $m \rightsquigarrow n$, then there exists a step $m[\sigma] \rightsquigarrow n[\sigma]$.*

Proof. By induction on the derivation of $m \rightsquigarrow n$. □

Lemma 4.2. *For terms m_1, m_2, n_1, n_2, if these exists reductions $m_1 \rightsquigarrow^* m_2$ and $n_1 \rightsquigarrow^* n_2$, then there exists reduction $(m_1\ n_1) \rightsquigarrow^* (m_2\ n_2)$.*

Proof. By transitivity of $\rightsquigarrow^*$ and applying rules Step-AppL, Step-AppR. □

Lemma 4.3. *For terms A_1, A_2, m_1, m_2 and sort s, if there exists reductions $A_1 \rightsquigarrow^* A_2$ and $m_1 \rightsquigarrow^* m_2$, then there exists reduction $\lambda x :_s A_1.m_1 \rightsquigarrow^* \lambda x :_s A_2.m_2$.*

Proof. By transitivity of $\rightsquigarrow^*$ and applying rules Step-λL, Step-λR. □

Lemma 4.4. *For terms A_1, A_2, B_1, B_2 and sort s, if there exists reductions $A_1 \rightsquigarrow^* A_2$ and $B_1 \rightsquigarrow^* B_2$, then there exists reduction $(x :_s A_1) \to B_1 \rightsquigarrow^* (x :_s A_2) \to B_2$.*

Proof. By transitivity of $\rightsquigarrow^*$ and applying rules Step-L$\to$, Step-R$\to$. □

Lemma 4.5. *For terms A_1, A_2, B_1, B_2 and sort s, if there exists reductions $A_1 \rightsquigarrow^* A_2$ and $B_1 \rightsquigarrow^* B_2$, then there exists reduction $(x :_s A_1) \multimap B_1 \rightsquigarrow^* (x :_s A_2) \multimap B_2$.*

Proof. By transitivity of $\rightsquigarrow^*$ and applying rules Step-L$\multimap$, Step-R$\multimap$. □

Lemma 4.6. *For terms m, n and a map σ from variables to terms, if there exist a reduction $m \rightsquigarrow^* n$, then there exists a reduction $m[\sigma] \rightsquigarrow^* n[\sigma]$.*

Proof. By induction on the derivation of $\rightsquigarrow^*$, the transitivity of $\rightsquigarrow^*$ and Lemma 4.1. □

Lemma 4.7. *For maps σ, τ from variables to terms, if there is a map reduction $\sigma \rightsquigarrow^* \tau$, then for any term m these is a reduction $m[\sigma] \rightsquigarrow^* m[\tau]$.*

Proof. By induction on the structure of m, applying Lemmas 4.2, 4.3, 4.4, 4.5. □

4.3 Equality Lemmas

Here, we prove some simple lemmas concerning $\leadsto^*$, $\equiv$ and substitution.

Definition 4.3. For maps σ, τ from variables to terms, we say that σ is equal to τ if for any variable x there exists an equality $(\sigma\ x) \equiv (\tau\ x)$. We write $\sigma \equiv \tau$ when it is clear from context that σ, τ are maps and not terms.

Lemma 4.8. *For any map f from terms to terms, if for any terms m, n such that $m \leadsto n$ implies $f\ m \equiv f\ n$, then for any terms m, n equality $m \equiv n$ implies $f\ m \equiv f\ n$.*

Proof. By the properties of the transitive reflexive closure $\leadsto^*$ and that $\equiv$ is an equivalence relation. □

Lemma 4.9. *For terms m_1, m_2, n_1, n_2, if there exists equalities $m_1 \equiv m_2$ and $n_1 \equiv n_2$, then there exists equality $(m_1\ n_1) \equiv (m_2\ n_2)$.*

Proof. By transitivity of $\equiv$ and applying rules Join, Step-AppL, Step-AppR. □

Lemma 4.10. *For terms A_1, A_2, m_1, m_2 and sort s, if there exists equalities $A_1 \equiv A_2$ and $m_1 \equiv m_2$, then there exists equality $\lambda x :_s A_1.m_1 \equiv \lambda x :_s A_2.m_2$.*

Proof. By transitivity of $\equiv$ and applying rules Join, Step-λL, Step-λR. □

Lemma 4.11. *For terms A_1, A_2, B_1, B_2 and sort s, if there exists equalities $A_1 \equiv A_2$ and $B_1 \equiv B_2$, then there exists equality $(x :_s A_1) \to B_1 \equiv (x :_s A_2) \to B_2$.*

Proof. By transitivity of $\equiv$ and applying rules Join, Step-L$\to$, Step-R$\to$. □

Lemma 4.12. *For terms A_1, A_2, B_1, B_2 and sort s, if there exists equalities $A_1 \equiv A_2$ and $B_1 \equiv B_2$, then there exists equality $(x :_s A_1) \multimap B_1 \equiv (x :_s A_2) \multimap B_2$.*

Proof. By transitivity of $\equiv$ and applying rules Join, Step-L$\multimap$, Step-R$\multimap$. □

Lemma 4.13. *For terms m, n and map σ from variables to terms, if there is equality $m \equiv n$, then there is equality $m[\sigma] \equiv n[\sigma]$.*

Proof. By Lemmas 4.8 and 4.1. □

Lemma 4.14. *For maps σ, τ from variables to terms and term m, if these is map equality $\sigma \equiv \tau$, then there is equality $m[\sigma] \equiv m[\tau]$.*

Proof. By induction on the structure of m, applying Lemmas 4.9, 4.10, 4.11, 4.12. □

Lemma 4.15. *For terms m_1, m_2, n, if there is equality $m_1 \equiv m_2$, then there is equality $n[m_1/x] \equiv n[m_2/x]$ for any variable $x \in FV(n)$.*

Proof. This is a special case of Lemma 4.14 where σ maps x to m_1 and τ maps x to m_2. □

4.4 Parallel Reduction Lemmas

Definition 4.4. For maps σ, τ from variables to terms, we say σ parallel reduces to τ if for any variable x there exists a parallel reduction $(\sigma\ x) \leadsto_p (\tau\ x)$. We write $\sigma \leadsto_p \tau$ when it is clear from context that σ, τ are maps and not terms.

Lemma 4.16. *For any term m, there exists a reflexive parallel reduction $m \leadsto_p m$.*

Proof. By induction on the structure of m. □

Lemma 4.17. *For any map σ from variables to terms, there exists a reflexive parallel map reduction $\sigma \leadsto_p \sigma$.*

Proof. By Definition 4.4 and Lemma 4.16. □

Lemma 4.18. *For any terms m, n, if there exists step $m \leadsto n$, then there exists a parallel reduction $m \leadsto_p n$.*

Proof. By induction on the derivation of $m \rightsquigarrow n$ and Lemma 4.16. $\square$

Lemma 4.19. *For terms m, n, if there exists parallel reduction $m \rightsquigarrow_p n$, then there exists a reduction $m \rightsquigarrow^* n$.*

Proof. By induction on the derivation of $m \rightsquigarrow_p n$, utilizing the transitive property of $\rightsquigarrow^*$ and Lemmas 4.2, 4.3, 4.4, 4.5, 4.6, 4.7. $\square$

Lemma 4.20. *For terms m, n and map σ from variables to terms, if there exists parallel reduction $m \rightsquigarrow_p n$, there exists parallel reduction $m[\sigma] \rightsquigarrow_p n[\sigma]$.*

Proof. By induction on the derivation of $m \rightsquigarrow_p n$ and Lemma 4.16. $\square$

Lemma 4.21. *For terms m, n and maps σ, τ from variables to terms, if there exists parallel reduction $m \rightsquigarrow_p n$ and parallel map reduction $\sigma \rightsquigarrow_p \tau$, there exists parallel reduction $m[\sigma] \rightsquigarrow_p n[\tau]$.*

Proof. By induction on the derivation of $m \rightsquigarrow_p n$. $\square$

Lemma 4.22. *For terms m_1, m_2, n, if there is parallel reduction $m_1 \rightsquigarrow_p m_2$, then there is parallel reduction $n[m_1/x] \rightsquigarrow_p n[m_2/x]$ for any variable $x \in FV(n)$.*

Proof. By Lemma 4.16, this is a special case of Lemma 4.21 where σ maps x to m_1 and τ maps x to m_2. $\square$

4.5 Confluence Theorem

We first show that $\rightsquigarrow_p$ satisfies the diamond property. Using the diamond property, we ultimately prove the confluence theorem.

Lemma 4.23. *CLC term reduction has the diamond property. For terms m, m_1, m_2, if there are parallel reductions $m \rightsquigarrow_p m_1$ and $m \rightsquigarrow_p m_2$, then there exists term m' such that $m_1 \rightsquigarrow_p m'$ and $m_2 \rightsquigarrow_p m'$.*

Proof. By induction on the derivation of $m \rightsquigarrow_p m_1$. Each case in the induction specializes m appearing in $m \rightsquigarrow_p m_2$, allowing one to invert its derivation in a syntax directed way and apply the induction hypothesis. The difficult cases are due to PStep-β as it concerns substitution, so Lemma 4.21 is used to push these cases through. $\square$

Lemma 4.24. *Strip lemma. For terms m, m_1, m_2, if there is parallel reduction $m \rightsquigarrow_p m_1$ and reduction $m \rightsquigarrow^* m_2$, then there exists term m' such that $m_1 \rightsquigarrow^* m'$ and $m_2 \rightsquigarrow_p m'$.*

Proof. By induction on the derivation of $m \rightsquigarrow_p m_1$, utilizing transitivity of $\rightsquigarrow^*$ and Lemmas 4.18, 4.19, 4.23. $\square$

Theorem 4.25. *CLC term reduction is confluent. For terms m, m_1, m_2, if there are reductions $m \rightsquigarrow^* m_1$ and $m \rightsquigarrow^* m_2$, then there exists term m' such that $m_1 \rightsquigarrow^* m'$ and $m_2 \rightsquigarrow^* m'$.*

Proof. By induction on the derivation of $m \rightsquigarrow^* m_1$, utilizing transitivity of $\rightsquigarrow^*$ and Lemmas 4.18, 4.19, 4.24. $\square$

4.6 Corollaries of Confluence

The following results are all corollaries of confluence, proven using a combination of induction, transitivity and confluence. These corollaries allow us to refute false reductions and equalities in future proofs.

Corollary 4.25.1. *For a universe s_i and term m, if there is reduction $s_i \rightsquigarrow^* m$, then $m = s_i$.*

Corollary 4.25.2. *For variable x and term m, if there is reduction $x \rightsquigarrow^* m$, then $m = x$.*

Corollary 4.25.3. *For terms A, B, m and sort s, if there is reduction $(x :_s A) \to B \rightsquigarrow^* m$, then there exists A', B' such that there are reductions $A \rightsquigarrow^* A'$, $B \rightsquigarrow^* B'$ and $m = (x :_s A') \to B'$.*

Corollary 4.25.4. *For terms A, B, m and sort s, if there is reduction $(x :_s A) \multimap B \rightsquigarrow^* m$, then there exists A', B' such that there are reductions $A \rightsquigarrow^* A'$, $B \rightsquigarrow^* B'$ and $m = (x :_s A') \multimap B'$.*

Corollary 4.25.5. *For terms A, m, n and sort s, if there is reduction $\lambda x :_s A.m \rightsquigarrow^* n$, then there exists A', m' such that there are reductions $A \rightsquigarrow^* A'$, $m \rightsquigarrow^* m'$ and $n = \lambda x :_s A'.m'$.*

Corollary 4.25.6. *For sorts s, t and levels i, j, if there is equality $s_i \equiv t_j$, then there is $s = t$ and $i = j$.*

Corollary 4.25.7. *For terms A_1, A_2, B_1, B_2 and sorts s, t, if there is equality $(x :_s A_1) \to B_1 \equiv (x :_t A_2) \to B_2$, then there are equalities $A_1 \equiv A_2$, $B_1 \equiv B_2$ and $s = t$.*

Corollary 4.25.8. *For terms A_1, A_2, B_1, B_2 and sorts s, t, if there is equality $(x :_s A_1) \multimap B_1 \equiv (x :_t A_2) \multimap B_2$, then there are equalities $A_1 \equiv A_2$, $B_1 \equiv B_2$ and $s = t$.*

5 Context of CLC (`clc_context.v`)

Contexts of CLC are of the form $x_1 :_{s_1} A_1, x_2 :_{s_2} A_2, ...x_k :_{s_k} A_k$ where each free variable x_i is assigned a type A_i and sort s_i. Contexts will be referred to by meta variables Γ and Δ.

$$\frac{}{\epsilon \vdash}\textsc{Wf-}\epsilon \qquad \frac{\Gamma \vdash \qquad \overline{\Gamma} \vdash A : U_i}{\Gamma, x :_U A \vdash}\textsc{Wf-U} \qquad \frac{\Gamma \vdash \qquad \overline{\Gamma} \vdash A : L_i}{\Gamma, x :_L A \vdash}\textsc{Wf-L}$$

$$\frac{}{|\epsilon|}\textsc{Pure-}\epsilon \qquad \frac{|\Gamma| \qquad \Gamma \vdash A : U_i}{|\Gamma, x :_U A|}\textsc{Pure-U}$$

$$\frac{}{\mathit{merge}\ \epsilon\ \epsilon\ \epsilon}\textsc{Merge-}\epsilon \qquad \frac{\mathit{merge}\ \Gamma_1\ \Gamma_2\ \Gamma}{\mathit{merge}\ (\Gamma_1, x :_U A)\ (\Gamma_2, x :_U A)\ (\Gamma, x :_U A)}\textsc{Merge-U}$$

$$\frac{\mathit{merge}\ \Gamma_1\ \Gamma_2\ \Gamma \qquad x \notin \Gamma_2}{\mathit{merge}\ (\Gamma_1, x :_L A)\ \Gamma_2\ (\Gamma, x :_L A)}\textsc{Merge-L1} \qquad \frac{\mathit{merge}\ \Gamma_1\ \Gamma_2\ \Gamma \qquad x \notin \Gamma_1}{\mathit{merge}\ \Gamma_1\ (\Gamma_2, x :_L A)\ (\Gamma, x :_L A)}\textsc{Merge-L2}$$

5.1 Merge Lemmas

Since weakening and contraction rules will not be allowed on restricted variables, it is necessary to have lemmas that enable the manipulation of contexts.

Lemma 5.1. *For contexts $\Gamma_1, \Gamma_2, \Gamma$, if there is merge $\Gamma_1\ \Gamma_2\ \Gamma$, then there is merge $\Gamma_2\ \Gamma_1\ \Gamma$.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. □

Lemma 5.2. *For any context Γ, if there is $|\Gamma|$, then there is merge $\Gamma\ \Gamma\ \Gamma$.*

Proof. By induction on the derivation of $|\Gamma|$. □

Lemma 5.3. *For any context Γ, there is merge $\overline{\Gamma}\ \Gamma\ \Gamma$.*

Proof. By induction on the structure of Γ. □

Lemma 5.4. *For any context Γ, there is merge $\Gamma\ \overline{\Gamma}\ \Gamma$.*

Proof. By induction on the structure of Γ. □

Lemma 5.5. *For contexts $\Gamma_1, \Gamma_2, \Gamma$, if there is merge $\Gamma_1\ \Gamma_2\ \Gamma$ and $|\Gamma|$, then there is $|\Gamma_1|$ and $|\Gamma_2|$.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. □

Lemma 5.6. *For contexts* $\Gamma_1, \Gamma_2, \Gamma$ *, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$ *and* $|\Gamma_1|$*, then there is* $\Gamma = \Gamma_2$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

Lemma 5.7. *For contexts* $\Gamma_1, \Gamma_2, \Gamma$ *, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$ *and* $|\Gamma_2|$*, then there is* $\Gamma = \Gamma_1$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

Lemma 5.8. *For contexts* $\Gamma_1, \Gamma_2, \Gamma$*, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$*, and also* $|\Gamma_1|$*,* $|\Gamma_2|$*, then there is* $|\Gamma|$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

Lemma 5.9. *For contexts* $\Gamma_1, \Gamma_2, \Gamma$*, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$*, and also* $|\Gamma_1|$*,* $|\Gamma_2|$*, then there is* $\Gamma_1 = \Gamma_2$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

Lemma 5.10. *For contexts* $\Gamma_1, \Gamma_2, \Gamma$*, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$*, then there is* $\overline{\Gamma_1} = \overline{\Gamma}$ *and* $\overline{\Gamma_2} = \overline{\Gamma}$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

Lemma 5.11. *For any context* Γ*, there is merge* $\overline{\Gamma}\ \overline{\Gamma}\ \overline{\Gamma}$*.*

Proof. By induction on the structure of Γ. ☐

5.2 Restriction and Purity Lemmas

Lemma 5.12. *For any context* Γ*, there is* $\overline{\Gamma} = \overline{\overline{\Gamma}}$*.*

Proof. By induction on the structure of Γ. ☐

Lemma 5.13. *For any context* Γ*, if there is* $|\Gamma|$*, then there is* $\Gamma = \overline{\Gamma}$*.*

Proof. By induction on the structure of Γ. ☐

Lemma 5.14. *For any context* Γ *, there is* $|\overline{\Gamma}|$*.*

Proof. By induction on the structure of Γ. ☐

Lemma 5.15. *For any context* Γ*, variable* x *and type* A*, if there is* $x :_U A \in \Gamma$*, then there is* $x :_U A \in \overline{\Gamma}$*.*

Proof. By induction on the derivation of $x :_U A \in \Gamma$. ☐

Lemma 5.16. *For any context* Γ*, variable* x *and type* A*, there is* $x :_L A \notin \overline{\Gamma}$*.*

Proof. By induction on the structure of Γ. ☐

Lemma 5.17. *For contexts* $\Gamma_1, \Gamma_2, \Gamma, \Delta_1, \Delta_2$*, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$ *and merge* $\Delta_1\ \Delta_2\ \Gamma_1$*, then there exists* Δ *such that merge* $\Delta_1\ \Gamma_2\ \Delta$ *and merge* $\Delta\ \Delta_2\ \Gamma$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

Lemma 5.18. *For contexts* $\Gamma_1, \Gamma_2, \Gamma, \Delta_1, \Delta_2$*, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$ *and merge* $\Delta_1\ \Delta_2\ \Gamma_1$*, then there exists* Δ *such that merge* $\Delta_2\ \Gamma_2\ \Delta$ *and merge* $\Delta_1\ \Delta\ \Gamma$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$. ☐

6 Subtyping of CLC (`clc_subtype.v`)

The cumulativity relation ($\preceq$) is the smallest binary relation over terms such that

1. $\preceq$ is a partial order with respect to equality.
   (a) If $A \equiv B$, then $A \preceq B$.
   (b) If $A \preceq B$ and $B \preceq A$, then $A \equiv B$.
   (c) If $A \preceq B$ and $B \preceq C$, then $A \preceq B$.
2. $U_0 \preceq U_1 \preceq U_2 \preceq \cdots$
3. $L_0 \preceq L_1 \preceq L_2 \preceq \cdots$
4. If $A_1 \equiv A_2$ and $B_1 \preceq B_2$,
   then $(x :_s A_1) \to B_1 \preceq (x :_s A_2) \to B_2$
5. If $A_1 \equiv A_2$ and $B_1 \preceq B_2$,
   then $(x :_s A_1) \multimap B_1 \preceq (x :_s A_2) \multimap B_2$

Here, we give an inductive definition of the cumulativity relation ($\preceq$) that is suitable for writing proofs.

$$\frac{}{A \prec A}\prec\text{-}\textsc{Refl} \qquad \frac{i_1 \leq i_2}{s_{i_1} \prec s_{i_2}}\prec\text{-}\textsc{Sort} \qquad \frac{B_1 \prec B_2}{(x :_s A) \to B_1 \prec (x :_s A) \to B_2}\prec\text{-}\to$$

$$\frac{B_1 \prec B_2}{(x :_s A) \multimap B_1 \prec (x :_s A) \multimap B_2}\prec\text{-}\multimap \qquad \frac{A' \prec B' \qquad A \equiv A' \qquad B \equiv B'}{A \preceq B}\prec\text{-}\preceq$$

6.1 Subtyping Lemmas

Lemma 6.1. *For terms A, B, if there is $A \prec B$, then there is $A \preceq B$.*

Proof. By $\prec$-$\preceq$ and the reflexivity of equality $\equiv$. □

Lemma 6.2. *For terms A, B, C, if there is $A \prec B$ and $B \equiv C$, then there is $A \preceq C$.*

Proof. By $\prec$-$\preceq$ and the transitivity of equality $\equiv$. □

Lemma 6.3. *For terms A, B, C, if there is $A \equiv B$ and $B \prec C$, then there is $A \preceq C$.*

Proof. By $\prec$-$\preceq$ and the transitivity of equality $\equiv$. □

Lemma 6.4. *For terms A, B, if there is $A \equiv B$, then there is $A \preceq B$.*

Proof. By Lemma 6.3 and $\prec$-Refl. □

Lemma 6.5. *For term A, there is $A \preceq A$.*

Proof. By Lemma 6.1 and $\prec$-Refl. □

Lemma 6.6. *For natural numbers i, j and sort s such that $i \leq j$, there is $s_i \preceq s_j$.*

Proof. By Lemma 6.1 and $\prec$-Sort. □

Lemma 6.7. *For terms A, B, C, D, if there is $A \prec B$, $B \equiv C$ and $C \prec D$, then there is $A \preceq D$.*

Proof. By induction on the derivation of $A \prec B$, definition of $\prec$ and Lemmas 6.1, 6.2, 6.3. □

Lemma 6.8. *For terms A, B, C, if there is $A \preceq B$ and $B \preceq C$, then there is $A \preceq C$.*

Proof. By transitivity of $\equiv$, rule $\prec$-$\preceq$ and Lemma 6.7. □

Lemma 6.9. *For sorts s, t and natural numbers i, j, if there is $s_i \preceq t_j$, then there is $s = t$ and $i \leq j$.*

Proof. By transitivity of $\equiv$ and Corollary 4.25.6. $\square$

Lemma 6.10. *For terms A_1, A_2, B_1, B_2 and sorts s, t, if there is $(x :_s A_1) \to B_1 \preceq (x :_t A_2) \to B_2$, then there is $A_1 \equiv A_2$ and $B_1 \preceq B_2$ and $s = t$.*

Proof. By transitivity of $\equiv$ and Corollary 4.25.7. $\square$

Lemma 6.11. *For terms A_1, A_2, B_1, B_2 and sorts s, t, if there is $(x :_s A_1) \multimap B_1 \preceq (x :_t A_2) \multimap B_2$, then there is $A_1 \equiv A_2$ and $B_1 \preceq B_2$ and $s = t$.*

Proof. By transitivity of $\equiv$ and Corollary 4.25.8. $\square$

Lemma 6.12. *For terms A, B and map σ from variables to terms, if there is $A \prec B$, then there is $A[\sigma] \prec B[\sigma]$.*

Proof. By induction on the derivation of $A \prec B$ and the definition of $\prec$. $\square$

Lemma 6.13. *For terms A, B and map σ from variables to terms, if there is $A \preceq B$, then there is $A[\sigma] \preceq B[\sigma]$.*

Proof. By rule $\prec$-$\preceq$ and Lemmas 4.13, 6.12. $\square$

7 Typing of CLC (`clc_typing.v`)

The following rules define well-formed contexts.

$$\frac{}{\epsilon \vdash}\,\epsilon\text{-}\textsc{Ok} \qquad \frac{\Gamma \vdash \qquad \overline{\Gamma} \vdash A : U_i}{\Gamma, x :_U A \vdash}\,\text{U-}\textsc{Ok} \qquad \frac{\Gamma \vdash \qquad \overline{\Gamma} \vdash A : L_i}{\Gamma, x :_L A \vdash}\,\text{L-}\textsc{Ok}$$

The typing rules of CLC are presented below.

$$\frac{|\Gamma|}{\Gamma \vdash s_i : U_{i+1}}\,\textsc{Sort-Axiom} \qquad \frac{|\Gamma| \qquad \Gamma \vdash A : U_i \qquad \Gamma, x :_U A \vdash B : s_i}{\Gamma \vdash (x :_U A) \to B : U_i}\,\text{U}{\to}$$

$$\frac{|\Gamma| \qquad \Gamma \vdash A : L_i \qquad \Gamma \vdash B : s_i \qquad x \notin \Gamma}{\Gamma \vdash (x :_L A) \to B : U_i}\,\text{L}{\to} \qquad \frac{|\Gamma| \qquad \Gamma \vdash A : U_i \qquad \Gamma, x :_U A \vdash B : s_i}{\Gamma \vdash (x :_U A) \multimap B : L_i}\,\text{U}{\multimap}$$

$$\frac{|\Gamma| \qquad \Gamma \vdash A : L_i \qquad \Gamma \vdash B : s_i \qquad x \notin \Gamma}{\Gamma \vdash (x :_L A) \multimap B : L_i}\,\text{L}{\multimap} \qquad \frac{|\Gamma_1, \Gamma_2|}{\Gamma_1, x :_U A, \Gamma_2 \vdash x : A}\,\text{U-}\textsc{Var} \qquad \frac{|\Gamma_1, \Gamma_2|}{\Gamma_1, x :_L A, \Gamma_2 \vdash x : A}\,\text{L-}\textsc{Var}$$

$$\frac{|\Gamma| \qquad \Gamma \vdash (x :_s A) \to B : t_i \qquad \Gamma, x :_s A \vdash n : B}{\Gamma \vdash \lambda x :_s A.n : (x :_s A) \to B}\,\lambda{\to} \qquad \frac{\overline{\Gamma} \vdash (x :_s A) \multimap B : t_i \qquad \Gamma, x :_s A \vdash n : B}{\Gamma \vdash \lambda x :_s A.n : (x :_s A) \multimap B}\,\lambda{\multimap}$$

$$\frac{\Gamma_1 \vdash m : (x :_U A) \to B \qquad |\Gamma_2| \qquad \Gamma_2 \vdash n : A \qquad \mathit{merge}\ \Gamma_1\ \Gamma_2\ \Gamma}{\Gamma \vdash m\ n : B[n/x]}\,\textsc{App}\text{-U}{\to}$$

$$\frac{\Gamma_1 \vdash m : (x :_L A) \to B \qquad \Gamma_2 \vdash n : A \qquad \mathit{merge}\ \Gamma_1\ \Gamma_2\ \Gamma}{\Gamma \vdash m\ n : B[n/x]}\,\textsc{App}\text{-L}{\to}$$

$$\frac{\Gamma_1 \vdash m : (x :_U A) \multimap B \qquad |\Gamma_2| \qquad \Gamma_2 \vdash n : A \qquad merge\ \Gamma_1\ \Gamma_2\ \Gamma}{\Gamma \vdash m\ n : B[n/x]}\text{App-U-}\multimap$$

$$\frac{\Gamma_1 \vdash m : (x :_L A) \multimap B \qquad \Gamma_2 \vdash n : A \qquad merge\ \Gamma_1\ \Gamma_2\ \Gamma}{\Gamma \vdash m\ n : B[n/x]}\text{App-L-}\multimap$$

$$\frac{\Gamma \vdash m : A \qquad \overline{\Gamma} \vdash B : s_i \qquad A \preceq B}{\Gamma \vdash m : B}\text{Conversion}$$

8 Inversion Lemmas of CLC (`clc_inversion.v`)

Lemma 8.1. *For any context Γ and terms A, B, s, if there is $\Gamma \vdash (x :_U A) \to B : s$, then there exists sort t and natural number i such that $\Gamma \vdash A : U_i$ and $\Gamma, x :_U A \vdash B : t_i$.*

Proof. By induction on the derivation of $\Gamma \vdash (x :_U A) \to B : s$. ☐

Lemma 8.2. *For any context Γ and terms A, B, s, if there is $\Gamma \vdash (x :_L A) \to B : s$, then there exists sort t and natural number i such that $\Gamma \vdash A : L_i$ and $\Gamma \vdash B : t_i$.*

Proof. By induction on the derivation of $\Gamma \vdash (x :_L A) \to B : s$. ☐

Lemma 8.3. *For any context Γ and terms A, B, s, if there is $\Gamma \vdash (x :_U A) \multimap B : s$, then there exists sort t and natural number i such that $\Gamma \vdash A : U_i$ and $\Gamma, x :_U A \vdash B : t_i$.*

Proof. By induction on the derivation of $\Gamma \vdash (x :_U A) \multimap B : s$. ☐

Lemma 8.4. *For any context Γ and terms A, B, s, if there is $\Gamma \vdash (x :_L A) \multimap B : s$, then there exists sort t and natural number i such that $\Gamma \vdash A : L_i$ and $\Gamma \vdash B : t_i$.*

Proof. By induction on the derivation of $\Gamma \vdash (x :_L A) \multimap B : s$. ☐

Lemma 8.5. *For any context Γ, terms A, n, C and sort s, if there is $\Gamma \vdash \lambda x :_s A.n : C$, then for all terms A', B, sorts s', t and natural number i such that $C \preceq (x :_{s'} A') \to B$ and $\overline{\Gamma, x :_{s'} A'} \vdash B : t_i$, there is $\Gamma, x :_{s'} A' \vdash n : B$.*

Proof. By induction on the derivation of $\Gamma \vdash \lambda x :_s A.n : C$ and Lemmas 8.1, 8.2. ☐

Lemma 8.6. *For any context Γ, terms A, n, C and sort s, if there is $\Gamma \vdash \lambda x :_s A.n : C$, then for all terms A', B, sorts s', t and natural number i such that $C \preceq (x :_{s'} A') \multimap B$ and $\overline{\Gamma, x :_{s'} A'} \vdash B : t_i$, there is $\Gamma, x :_{s'} A' \vdash n : B$.*

Proof. By induction on the derivation of $\Gamma \vdash \lambda x :_s A.n : C$ and Lemmas 8.3, 8.4. ☐

Lemma 8.7. *For any context Γ, terms A, A', B, n, sorts s, s', t and natural number i, if there is $\overline{\Gamma} \vdash (x :_{s'} A') \to B : t_i$ and $\Gamma \vdash \lambda x :_s A.n : (x :_{s'} A') \to B$, then there is $\Gamma, x :_{s'} A' \vdash n : B$.*

Proof. Direct consequence of Lemmas 8.1, 8.2 and 8.5. ☐

Lemma 8.8. *For any context Γ, terms A, A', B, n, sorts s, s', t and natural number i, if there is $\overline{\Gamma} \vdash (x :_{s'} A') \multimap B : t_i$ and $\Gamma \vdash \lambda x :_s A.n : (x :_{s'} A') \multimap B$, then there is $\Gamma, x :_{s'} A' \vdash n : B$.*

Proof. Direct consequence of Lemmas 8.3, 8.4 and 8.6. ☐

9 Weakening Lemmas of CLC (`clc_weakening.v`)

Weakening for non-linear types is admissible in CLC. To prove this, we first define an *agreeR* relation between two contexts Γ, Γ' and a mapping ξ from variables to variables.

$$\frac{}{agreeR\ \xi\ \epsilon\ \epsilon}\text{AGREER-}\epsilon \qquad \frac{agreeR\ \xi\ \Gamma\ \Gamma' \qquad x \notin FV(\Gamma) \cup FV(\Gamma')}{agreeR\ (\xi \cup (x,x))\ (\Gamma, x :_U A)(\Gamma', x :_U A[\xi])}\text{AGREER-U}$$

$$\frac{agreeR\ \xi\ \Gamma\ \Gamma' \qquad x \notin FV(\Gamma) \cup FV(\Gamma')}{agreeR\ (\xi \cup (x,x))\ (\Gamma, x :_L A)(\Gamma', x :_L A[\xi])}\text{AGREER-L} \qquad \frac{agreeR\ \xi\ \Gamma\ \Gamma' \qquad x \notin FV(\Gamma) \cup FV(\Gamma')}{agreeR\ \xi\ \Gamma\ (\Gamma', x :_U A)}\text{AGREER-WK}$$

9.1 Properties of *agreeR*

Lemma 9.1. *For any context Γ and the identity map id from variables to variables, agreeR id Γ Γ is always true.*

Proof. By induction on the structure of Γ and the definition of *agreeR*. □

Lemma 9.2. *For contexts Γ, Γ' and mapping ξ, if there is agreeR ξ Γ Γ' and $|\Gamma|$, then there is $|\Gamma'|$.*

Proof. By induction on the derivation of *agreeR* $\xi\ \Gamma\ \Gamma'$. □

Lemma 9.3. *For contexts Γ, Γ' and mapping ξ, if there is agreeR ξ Γ Γ', then there is agreeR ξ $|\Gamma|$ $|\Gamma'|$.*

Proof. By induction on the derivation of *agreeR* $\xi\ \Gamma\ \Gamma'$. □

9.2 Weakening Theorem

Lemma 9.4. *For contexts $\Gamma, \Gamma', \Gamma_1, \Gamma_2$ and mapping ξ, if there is agreeR ξ Γ Γ' and merge Γ_1 Γ_2 Γ, then there exists Γ'_1, Γ'_2 such that merge Γ'_1 Γ'_2 Γ', and agreeR ξ Γ_1 Γ'_1 and agreeR ξ Γ_2 Γ'_2.*

Proof. By induction on the derivation of *agreeR* $\xi\ \Gamma\ \Gamma'$ and lemmas in Section 9.1. □

Lemma 9.5. *For context Γ, Γ', terms m, A and mapping ξ, if there is $\Gamma \vdash m : A$ and agreeR ξ Γ Γ', then there is $\Gamma' \vdash m[\xi] : A[\xi]$.*

Proof. By induction on the derivation of $\Gamma \vdash m : A$. We shall only discuss the application case in detail, as the other cases are proven by application of the induction hypothesis and the lemmas in Section 9.1.

- For the App-U$\to$ case, Lemma 9.4 is applied to split the context Γ into two contexts Γ'_1 and Γ'_2 such that there is *merge* $\Gamma'_1\ \Gamma'_2\ \Gamma'$ and *agreeR* $\xi\ \Gamma_1\ \Gamma'_1$ and *agreeR* $\xi\ \Gamma_2\ \Gamma'_2$. From $|\Gamma_2|$ and Lemma 9.2 we know that there is $|\Gamma'_2|$. At this point, the induction hypothesis allows us to apply App-U$\to$ to prove the goal.
- For the App-L$\to$ case, Lemma 9.4 is applied to split the context Γ into two contexts Γ'_1 and Γ'_2 such that there is *merge* $\Gamma'_1\ \Gamma'_2\ \Gamma'$ and *agreeR* $\xi\ \Gamma_1\ \Gamma'_1$ and *agreeR* $\xi\ \Gamma_2\ \Gamma'_2$. At this point, the induction hypothesis allows us to apply App-L$\to$ to prove the goal.
- For the App-U$\multimap$ case, Lemma 9.4 is applied to split the context Γ into two contexts Γ'_1 and Γ'_2 such that there is *merge* $\Gamma'_1\ \Gamma'_2\ \Gamma'$ and *agreeR* $\xi\ \Gamma_1\ \Gamma'_1$ and *agreeR* $\xi\ \Gamma_2\ \Gamma'_2$. From $|\Gamma_2|$ and Lemma 9.2 we know that there is $|\Gamma'_2|$. At this point, the induction hypothesis allows us to apply App-U$\multimap$ to prove the goal.
- For the App-L$\multimap$ case, Lemma 9.4 is applied to split the context Γ into two contexts Γ'_1 and Γ'_2 such that there is *merge* $\Gamma'_1\ \Gamma'_2\ \Gamma'$ and *agreeR* $\xi\ \Gamma_1\ \Gamma'_1$ and *agreeR* $\xi\ \Gamma_2\ \Gamma'_2$. At this point, the induction hypothesis allows us to apply App-L$\multimap$ to prove the goal.

□

Theorem 9.6. *Weakening is admissible for CLC variables of non-linear type. For context Γ and terms m, A, B, if there is $\Gamma \vdash m : A$, then there is $\Gamma, x :_U B \vdash m : A$.*

Proof. Using AGREER-WK and Lemma 9.1 a proof of *agreeR id* Γ $(\Gamma, x :_U B)$ can be constructed. Then by Lemma 9.5, the theorem can be proven. □

10 Substitution Lemmas of CLC (`clc_substitution.v`)

Similar to the proof of weakening, we first define an *agreeS* relation between two contexts Γ, Δ and a mapping σ from variables to terms.

$$\frac{}{\mathit{agreeS}\ \sigma\ \epsilon\ \epsilon}\textsc{agreeS-}\epsilon \qquad \frac{\mathit{agreeS}\ \sigma\ \Delta\ \Gamma \qquad x \notin FV(\Delta) \cup FV(\Gamma)}{\mathit{agreeS}\ (\sigma \cup (x,x))\ (\Delta, x :_{U} A[\sigma])\ (\Gamma, x :_{U} A)}\textsc{agreeS-U}$$

$$\frac{\mathit{agreeS}\ \sigma\ \Delta\ \Gamma \qquad x \notin FV(\Delta) \cup FV(\Gamma)}{\mathit{agreeS}\ (\sigma \cup (x,x))\ (\Delta, x :_{L} A[\sigma])\ (\Gamma, x :_{L} A)}\textsc{agreeS-L}$$

$$\frac{\mathit{agreeS}\ \sigma\ \Delta\ \Gamma \qquad \overline{\Delta} \vdash n : A[\sigma] \qquad x \notin FV(\Delta) \cup FV(\Gamma)}{\mathit{agreeS}\ (\sigma \cup (x,n))\ \Delta\ (\Gamma, x :_{U} A)}\textsc{agreeS-wkU}$$

$$\frac{\mathit{merge}\ \Delta_1\ \Delta_2\ \Delta \qquad \mathit{agreeS}\ \sigma\ \Delta_1\ \Gamma \qquad \Delta_2 \vdash n : A[\sigma] \qquad x \notin FV(\Delta) \cup FV(\Gamma)}{\mathit{agreeS}\ (\sigma \cup (x,n))\ \Delta\ (\Gamma, x :_{L} A)}\textsc{agreeS-wkL}$$

$$\frac{A \preceq B \qquad \overline{\Delta} \vdash B[\sigma] : U_i \qquad \mathit{agreeS}\ \sigma\ \Delta\ (\Gamma, x :_{U} A)}{\mathit{agreeS}\ \sigma\ \Delta\ (\Gamma, x :_{U} B)}\textsc{agreeS-convU}$$

$$\frac{A \preceq B \qquad \overline{\Delta} \vdash B[\sigma] : L_i \qquad \overline{\Gamma} \vdash B : L_i \qquad \mathit{agreeS}\ \sigma\ \Delta\ (\Gamma, x :_{L} A)}{\mathit{agreeS}\ \sigma\ \Delta\ (\Gamma, x :_{L} B)}\textsc{agreeS-convL}$$

10.1 Properties of *agreeS*

Lemma 10.1. *For any context Γ and identity mapping id, there is agreeS id Γ Γ.*

Proof. By induction on the structure of Γ. ☐

Lemma 10.2. *For contexts Δ, Γ and mapping σ, if there is agreeS σ Δ Γ, then there is agreeS σ $\overline{\Delta}$ $\overline{\Gamma}$.*

Proof. By induction on the derivation of *agreeS* $\sigma\ \Delta\ \Gamma$. ☐

10.2 Substitution Lemma

Lemma 10.3. *For contexts $\Delta, \Gamma, \Gamma_1, \Gamma_2$ and mapping σ, if there is agreeS σ Δ Γ and merge Γ_1 Γ_2 Γ, then there exists contexts Δ_1, Δ_2 such that merge Δ_1 Δ_2 Δ and agreeS σ Δ_1 Γ_1 and agreeS σ Δ_2 Γ_2.*

Proof. By induction on the derivation of *agreeS* $\sigma\ \Delta\ \Gamma$ and lemmas in Section 10.1. ☐

Lemma 10.4. *Generalized Substitution Lemma. For context Γ, Δ, terms m, A and mapping σ, if there is $\Gamma \vdash m : A$ and agreeS σ Δ Γ, then there is $\Delta \vdash m[\sigma] : A[\sigma]$.*

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash m : A$. Similar to the proof of Lemma 9.5, the interesting cases are the application cases where Lemma 10.3 must be utilized to split the *merge* $\Gamma_1\ \Gamma_2\ \Gamma$ judgments for use in the induction hypothesis. ☐

10.3 Corollaries of Substitution

Corollary 10.4.1. *For contexts $\Gamma_1, \Gamma_2, \Gamma$ and terms A, B, m, n, if there is $\Gamma_1, x :_{U} A \vdash m : B$ and $|\Gamma_2|$ and merge Γ_1 Γ_2 Γ and $\Gamma_2 \vdash n : A$, then there is $\Gamma \vdash m[n/x] : B[n/x]$.*

Corollary 10.4.2. *For contexts $\Gamma_1, \Gamma_2, \Gamma$ and terms A, B, m, n, if there is $\Gamma_1, x :_{L} A \vdash m : B$ and merge Γ_1 Γ_2 Γ and $\Gamma_2 \vdash n : A$, then there is $\Gamma \vdash m[n/x] : B[n/x]$.*

Corollary 10.4.3. *For context* Γ*, terms* m, A, B, C *and natural number* i*, if there is* $B \equiv A$ *and* $\overline{\Gamma} \vdash A : U_i$ *and* $\Gamma, x :_U A \vdash m : C$*, then there is* $\Gamma, x :_U B \vdash m : C$*.*

Corollary 10.4.4. *For context* Γ*, terms* m, A, B, C *and natural number* i*, if there is* $B \equiv A$ *and* $\overline{\Gamma} \vdash A : L_i$ *and* $\Gamma, x :_L A \vdash m : C$*, then there is* $\Gamma, x :_L B \vdash m : C$*.*

11 Typing Validity of CLC (`clc_validity.v`)

In this section, we prove that the types of all CLC terms are themselves well-sorted.

Lemma 11.1. *For contexts* $\Gamma_1, \Gamma_2, \Gamma$*, if there is merge* $\Gamma_1\ \Gamma_2\ \Gamma$ *and* $\Gamma \vdash$*, then there is* $\Gamma_1 \vdash$ *and* $\Gamma_2 \vdash$*.*

Proof. By induction on the derivation of *merge* $\Gamma_1\ \Gamma_2\ \Gamma$ and the properties of *merge* discussed in Section 5.1. □

Theorem 11.2. *The validity of typing theorem. For any context* Γ *and terms* m, A*, if there is* $\Gamma \vdash$ *and* $\Gamma \vdash m : A$*, then there exists sort* s *and natural number* i *such that* $\overline{\Gamma} \vdash A : s_i$*.*

12 Subject Reduction of CLC (`clc_soundness.v`)

Theorem 12.1. *For any context* Γ *and terms* m, n, A*, if* $\Gamma \vdash$ *and* $\Gamma \vdash m : A$ *and* $m \rightsquigarrow n$*, then there is* $\Gamma \vdash n : A$*.*

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash m : A$. The interesting cases are the application cases which we shall discuss in detail.

- For the App-U$\rightarrow$ case, from assumptions *merge* $\Gamma_1\ \Gamma_2\ \Gamma$ and $|\Gamma_2|$ and Lemmas 5.7, 5.10, we can conclude that $\overline{\Gamma_1} = \overline{\Gamma}$ and $\overline{\Gamma_2} = \overline{\Gamma}$. Applying Lemma 11.1 to assumptions *merge* $\Gamma_1\ \Gamma_2\ \Gamma$ and $\Gamma \vdash$ obtains $\Gamma_1 \vdash$ and $\Gamma_2 \vdash$. Now by the induction hypothesis, we can conclude there exists sorts s, t and natural numbers i, j such that there are $\overline{\Gamma_1} \vdash (x :_U A) \rightarrow B : s_i$ and $\overline{\Gamma_2} \vdash A : t_j$. Applying Lemma 8.1 to assumption $\overline{\Gamma_1} \vdash (x :_U A) \rightarrow B : s_i$ allows us to derive $\overline{\Gamma_1} \vdash A : U_{i'}$ and $\overline{\Gamma_1}, x :_U A \vdash B : s'_{j'}$ where s' is a sort and i', j' are natural numbers. The goal can finally be proven by applying the substitution Lemma 10.4.1 on assumptions $\Gamma_2 \vdash n : A$ and $\overline{\Gamma_1}, x :_U A \vdash B : s'_{j'}$.

- For the App-L$\rightarrow$ case, from assumption *merge* $\Gamma_1\ \Gamma_2\ \Gamma$ and Lemmas 5.10, we can conclude that $\overline{\Gamma_1} = \overline{\Gamma}$ and $\overline{\Gamma_2} = \overline{\Gamma}$. Applying Lemma 11.1 to assumptions *merge* $\Gamma_1\ \Gamma_2\ \Gamma$ and $\Gamma \vdash$ obtains $\Gamma_1 \vdash$ and $\Gamma_2 \vdash$. Now by the induction hypothesis, we can conclude that there exists sorts s, t and natural numbers i, j such that there are $\overline{\Gamma_1} \vdash (x :_L A) \rightarrow B : s_i$ and $\overline{\Gamma_2} \vdash A : t_j$. Applying Lemma 8.2 to assumption $\overline{\Gamma_1} \vdash (x :_L A) \rightarrow B : s_i$ allows us to derive $\overline{\Gamma_1} \vdash A : L_{i'}$ and $\overline{\Gamma_1} \vdash B : s'_{j'}$. Due to the fact that variable x is not a free variable in B, the substitution occurring in goal $\exists s \in sort, i \in \mathbb{N}, \overline{\Gamma} \vdash B[n/x] : s_i$ is trivial, thus the judgment $\overline{\Gamma_1} \vdash B : s'_{j'}$ that we have proven shows the existence of the goal.

- For the App-U$\multimap$ case, the proof is similar to the App-U$\rightarrow$ case, the only difference is that the inversion lemmas used correspond to $\multimap$ instead of $\rightarrow$.

- For the App-L$\multimap$ case, the proof is similar to the App-L$\rightarrow$ case, the only difference is that the inversion lemmas used correspond to $\multimap$ instead of $\rightarrow$.

□

13 Linearity Theorems of CLC (`clc_linearity.v`)

13.1 Linearity

We introduce a meta-function *occurs* that counts the number of times a given variable occurs in a term.

$$
\begin{aligned}
occurs\ x\ y &= \begin{cases} 1 & x =_\alpha y \\ 0 & x \neq_\alpha y \end{cases} \\
occurs\ x\ s_i &= 0 \\
occurs\ x\ ((y :_s A) \to B) &= occurs\ x\ A + occurs\ x\ B \\
occurs\ x\ ((y :_s A) \multimap B) &= occurs\ x\ A + occurs\ x\ B \\
occurs\ x\ (\lambda x :_s A.n) &= occurs\ x\ A + occurs\ x\ n \\
occurs\ x\ (m\ n) &= occurs\ x\ m + occurs\ x\ n
\end{aligned}
$$

Lemma 13.1. *For contexts $\Gamma_1, \Gamma_2, \Gamma$, if there is merge $\Gamma_1\ \Gamma_2\ \Gamma$, then for any variable with linear type $x \in \Gamma$ there is $x \in \Gamma_1$ and $x \notin \Gamma_2$ or $x \in \Gamma_2$ and $x \notin \Gamma_1$.*

Proof. By induction on the derivation of $merge\ \Gamma_1\ \Gamma_2\ \Gamma$. □

Lemma 13.2. *For contexts $\Gamma_1, \Gamma_2, \Gamma$, if there is merge $\Gamma_1\ \Gamma_2\ \Gamma$, then for any variable $x \notin \Gamma$ there is $x \notin \Gamma_1$ and $x \notin \Gamma_2$.*

Proof. By induction on the derivation of $merge\ \Gamma_1\ \Gamma_2\ \Gamma$. □

Lemma 13.3. *For context Γ, terms m, A, if there is $\Gamma \vdash m : A$, then for any variable $x \notin \Gamma$ there is $occurs\ x\ m = 0$*

Proof. By induction on the derivation of $\Gamma \vdash m : A$. □

Theorem 13.4. *Linearity. For context Γ, terms m, A, if there is $\Gamma \vdash m : A$, then for any variable with linear type $x \in \Gamma$ there is $occurs\ x\ m = 1$.*

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash m : A$, we will discuss the application cases in detail.

- For case App-U→, by assumptions $merge\ \Gamma_1\ \Gamma_2\ \Gamma$ and $(x :_L A) \in \Gamma$ and Lemma 13.1 we can conclude that $x \in \Gamma_1$ and $x \notin \Gamma_2$ or $x \in \Gamma_2$ and $x \notin \Gamma_1$. In both cases, applying the induction hypothesis and Lemma 13.3 proves the goal.
- For case App-L→, the proof is the same as App-U→.
- For case App-U⊸, the proof is the same as App-U→.
- For case App-L⊸, the proof is the same as App-U→.

□

13.2 Promotion

Theorem 13.5. *Promotion. For context Γ, terms m, A, B and sort s, if there is $|\Gamma|$ and $\Gamma \vdash$ and $\Gamma \vdash m : (x :_s A) \multimap B$, then there exists term n such that $\Gamma \vdash n : (x :_s A) \to B$.*

Proof. Set $n = \lambda x :_s A.(m\ x)$. The proof proceeds by case analysis on the sort s.

- If $s = U$, then we may apply Theorem 11.2 to assumption $\Gamma \vdash m : (x :_U A) \multimap B$ to show that there exists sort t and natural number i such that there is $\overline{\Gamma} \vdash (x :_U A) \multimap B : t_i$. Now applying Lemma 8.3 to $\overline{\Gamma} \vdash (x :_U A) \multimap B : t_i$ shows that there exists sort t' and natural number i' such that $\overline{\Gamma} \vdash A : U_{i'}$ and $\overline{\Gamma}, x :_U A \vdash B : t'_{i'}$. Now by U→ and Lemma 5.13 the goal is proven.
- If $s = L$, then we may apply Theorem 11.2 to assumption $\Gamma \vdash m : (x :_L A) \multimap B$ to show that there exists sort t and natural number i such that $\overline{\Gamma} \vdash (x :_L A) \multimap B : t_i$. Now applying Lemma 8.4 to $\overline{\Gamma} \vdash (x :_L A) \multimap B : t_i$ shows that there exists sort t' and natural number i' such that $\overline{\Gamma} \vdash A : U_{i'}$ and $\overline{\Gamma} \vdash B : t'_{i'}$. Now by L→ and Lemma 5.13 the goal is proven.

□

13.3 Dereliction

Theorem 13.6. *Dereliction. For context Γ, terms m, A, B and sort s, if there is $\Gamma \vdash$ and $\Gamma \vdash m : (x :_s A) \to B$, then there exists term n such that $\Gamma \vdash n : (x :_s A) \multimap B$.*

Proof. Set $n = \lambda x :_s A.(m\ x)$. The proof proceeds by case analysis on the sort s.

- If $s = U$, then we may apply Theorem 11.2 to $\Gamma \vdash m : (x :_U A) \to B$ showing that there exists sort t and natural number i such that there is $\overline{\Gamma} \vdash (x :_U A) \to B : t_i$. Now applying Lemma 8.1 to $\overline{\Gamma} \vdash (x :_U A) \to B : t_i$ shows that there exists sort t' and natural number i' such that $\overline{\Gamma} \vdash A : U_{i'}$ and $\overline{\Gamma}, x :_U A \vdash B : t'_{i'}$. By U$\multimap$ and Lemma 5.14 we can prove $\overline{\Gamma} \vdash (x :_U A) \multimap B : L_{i'}$. By rule $\lambda$$\multimap$, the rest of the goal can be proven in a straightforward manner.
- If $s = L$, then we may apply Theorem 11.2 to $\Gamma \vdash m : (x :_L A) \to B$ showing that there exists sort t and natural number i such that there is $\overline{\Gamma} \vdash (x :_L A) \to B : t_i$. Now applying Lemma 8.2 to $\overline{\Gamma} \vdash (x :_L A) \to B : t_i$ shows that there exists sort t' and natural number i' such that $\overline{\Gamma} \vdash A : U_{i'}$ and $\overline{\Gamma} \vdash B : t'_{i'}$. By L$\multimap$ and Lemma 5.14 we can prove $\overline{\Gamma} \vdash (x :_L A) \multimap B : L_{i'}$. By rule $\lambda$$\multimap$, the rest of the goal can be proven in a straightforward manner.

□

14 Logical Consistency of CLC `clc_consistent.v`

14.1 Strong Normalization

The proof of the logical consistency of CLC proceeds by construction of a reduction preserving erasure from CLC to CCω. As CCω is consistent, CLC must be consistent as well.

The erasure procedure is recursively defined as follows.

$$\begin{aligned}
[\![x]\!] &= x \\
[\![U_i]\!] &= Type_i \\
[\![L_i]\!] &= Type_i \\
[\![(x :_s A) \to B]\!] &= (x : [\![A]\!]) \to [\![B]\!] \\
[\![(x :_s A) \multimap B]\!] &= (x : [\![A]\!]) \to [\![B]\!] \\
[\![\lambda x :_s A.n]\!] &= \lambda x : [\![A]\!].[\![n]\!] \\
[\![m\ n]\!] &= [\![m]\!]\ [\![n]\!]
\end{aligned}$$

With slight overloading of notation, we define erasure for CLC contexts recursively.

$$\begin{aligned}
[\![\epsilon]\!] &= \epsilon \\
[\![\Gamma, x :_s A]\!] &= [\![\Gamma]\!], x : [\![A]\!]
\end{aligned}$$

Lemma 14.1. *For CLC term m, map σ from variables to CLC terms, map τ from variables to CCω terms, if for all variables x there is $[\![\sigma\ x]\!] = \tau\ x$, then $[\![m[\sigma]]\!] = [\![m]\!][\tau]$.*

Proof. By induction on the structure of term m. □

For the following lemmas, we will index relations and judgments with subscript CLC or CCω to emphasize the language it is defined over.

Lemma 14.2. *For any CLC terms m and n, if there is $m \leadsto_{CLC} n$, then there is $[\![m]\!] \leadsto_{CC\omega} [\![n]\!]$.*

Proof. By induction on the derivation of $m \leadsto_{\text{CLC}} n$. □

Lemma 14.3. *For any CLC terms m and n, if there is $m \equiv_{CLC} n$, then there is $[\![m]\!] \equiv_{CC\omega} [\![n]\!]$.*

Proof. By induction on the derivation of $m \equiv_{\text{CLC}} n$ and Lemma 14.2. $\square$

Lemma 14.4. *For any CLC terms m and n, if there is $m \prec_{CLC} n$, then there is $[\![m]\!] \prec_{CC\omega} [\![n]\!]$.*

Proof. By induction on the derivation of $m \prec_{\text{CLC}} n$. $\square$

Lemma 14.5. *For any CLC terms m and n, if there is $m \preceq_{CLC} n$, then there is $[\![m]\!] \preceq_{CC\omega} [\![n]\!]$.*

Proof. By case analysis on the derivation of $m \preceq_{\text{CLC}} n$ and the properties of subtyping proven in Section 6. $\square$

Theorem 14.6. *Embedding. For any CLC context Γ and CLC terms m, A, if there is $\Gamma \vdash_{CLC} m : A$, then there is $[\![\Gamma]\!] \vdash_{CC\omega} [\![m]\!] : [\![A]\!]$.*

Proof. By induction on the derivation of $\Gamma \vdash_{\text{CLC}} m : A$. $\square$

Corollary 14.6.1. *For any CLC context Γ, if there is $\Gamma \vdash_{CLC}$, then there is $[\![\Gamma]\!] \vdash_{CC\omega}$.*

Proof. Direct consequence of applying Theorem 14.6 to all types in context Γ. $\square$

Theorem 14.7. *Strong normalization of CLC.*

Proof. Suppose there exists a well-typed CLC term m with an infinite sequence of reductions. Theorem 14.6 shows that there must exist some term $[\![m]\!]$ that is well-typed in CCω. Additionally, this infinite sequence of reductions on m can be translated in CCω step-wise by Lemma 14.2. This shows that we have constructed a non-normalizing CCω term $[\![m]\!]$, which a contradiction to the strong normalization property of CCω, thus CLC must be strongly normalizing as well. $\square$

14.2 Embedding of CCω

To show that CLC is compatible with the predicative fragment of CCω, we construct a lifting procedure that lifts CCω terms into CLC in a straightforward way.

$$
\begin{aligned}
(\!|x|\!) &= x \\
(\!|Type_i|\!) &= U_i \\
(\!|(x : A) \to B|\!) &= (x :_U (\!|A|\!)) \to (\!|B|\!) \\
(\!|\lambda x : A.n|\!) &= \lambda x :_U (\!|A|\!).(\!|n|\!) \\
(\!|m\ n|\!) &= (\!|m|\!)\ (\!|n|\!)
\end{aligned}
$$

With slight overloading of notation, we define lifting for CCω recursively.

$$
\begin{aligned}
(\!|\epsilon|\!) &= \epsilon \\
(\!|\Gamma, x : A|\!) &= (\!|\Gamma|\!), x :_U (\!|A|\!)
\end{aligned}
$$

Lemma 14.8. *For CCω context Γ, there is $|(\!|\Gamma|\!)|$.*

Proof. By induction on the structure of Γ. $\square$

Lemma 14.9. *For CCω term m, map σ from variables to CCω terms, map τ from variable to CLC terms, if for all variables x there is $(\!|\sigma\ x|\!) = \tau\ x$, then $(\!|m[\sigma]|\!) = (\!|m|\!)[\tau]$.*

Proof. By induction on the structure of term m. $\square$

For the following lemmas, we will index relations and judgments with subscript CLC of CCω to emphasize the language it is defined over.

Lemma 14.10. *For any CCω terms m and n, if there is $m \leadsto_{CC\omega} n$, then there is $(\!|m|\!) \leadsto_{CLC} (\!|n|\!)$.*

Proof. By induction on the derivation of $m \leadsto_{\text{CC}\omega} n$. $\square$

Lemma 14.11. *For any CCω terms m and n, if there is* $m \equiv_{CC\omega} n$*, then there is* $(\!|m|\!) \equiv_{CLC} (\!|n|\!)$.

Proof. By induction on the derivation of $m \equiv_{\mathrm{CC}\omega} n$ and Lemma 14.10. $\square$

Lemma 14.12. *For any CCω terms m and n, if there is* $m \prec_{CC\omega} n$*, then there is* $(\!|m|\!) \prec_{CLC} (\!|n|\!)$.

Proof. By induction on the derivation of $m \prec_{\mathrm{CC}\omega} n$. $\square$

Lemma 14.13. *For any CCω terms m and n, if there is* $m \preceq_{CC\omega} n$*, then there is* $(\!|m|\!) \preceq_{CLC} (\!|n|\!)$.

Proof. By case analysis on the derivation of $m \preceq_{\mathrm{CC}\omega} n$ and the properties of subtyping proven in Section 6. $\square$

Theorem 14.14. *Lifting. For any CCω context* Γ *and CCω terms* m, A*, if there is* $\Gamma \vdash_{CC\omega} m : A$*, then there is* $(\!|\Gamma|\!) \vdash_{CLC} (\!|m|\!) : (\!|A|\!)$.

Proof. By induction on the derivation of $\Gamma \vdash_{\mathrm{CC}\omega} m : A$. $\square$