De Bruijn's syntax and reductional behaviour of $\boldsymbol{\lambda}$-terms: the typed case*

Fairouz Kamareddine† and Roel Bloo ‡

Abstract

In this paper, we consider the typed versions of the λ-calculus written in a notation which helps describe canonical forms more elegantly than the classical notation, and enables to divide terms into classes according to their reductional behaviour. In this notation, β-reduction can be generalised from a relation on terms to one on equivalence classes. This class reduction covers many known notions of generalised reduction. We extend the Barendregt cube with our class reduction and show that the subject reduction property fails but that this is not unique to our class reduction. We show that other generalisations of reduction (such as the σ-reduction of Regnier) also behave badly in typed versions of the λ-calculus. Nevertheless, solution is at hand for these generalised reductions by adopting the useful addition of definitions in the contexts of type derivations. We show that adding such definitions enables the extensions of type systems with class reduction and σ-reduction to satisfy all the desirable properties of type systems, including subject reduction and strong normalisation. Our proposed typing relation $\vdash^{\mathsf{c}}$ is the most general relation in the literature that satisfies all the desirable properties of type systems. We show that classes contain all the desirable information related to a term with respect to typing, strong normalisation, subject reduction, etc.

Keywords: class reduction, type theory, subject reduction, strong normalisation.

1 Introduction

Generalised reduction has received much attention in the literature in the past decade where various uses and benefits of generalising reduction have been illustrated (cf. [16, 17, 12, 11, 18, 2, 3, 14, 21, 20, 13, 10, 4, 9]). In [5] we gave the most general form of reduction $\leadsto_\beta$ which on one hand is fine grained, and on the other, works on classes rather than terms. We showed that this general notion of class reduction does indeed generalise existing ones.

Yet, if such a reduction is to be useful in practice (and especially in computation), we need to use it in type systems. In particular, we need to show that indeed using $\leadsto_\beta$ with type theory would satisfy the basic requirements of a typed system. In particular, the issues of safety and termination are a priority. A safe type system must give all the intermediate (and final) values of a program, the same type. Similarly, a type systems should not type non terminating programs. And indeed this is the case if we extend simple type theory with $\leadsto_\beta$. But, simple types are easy. The real test is those useful powerful type systems such as dependent or polymorphic type theory.

In this paper, we set out to provide useful type systems which use our general class reduction. Since we are interested in the behaviour of class reduction with all sorts of types (simple, polymorphic and dependent), we take the cube of eight different systems [1] as the basic framework.

*This article builds on and extends the results and proofs of [7] for the typed case. The untyped case can be found in [5]. We are grateful for enlightening discussions and useful feedback and comments received from Henk Barendregt, Twan Laan, Rob Nederpelt and Joe Wells. An anonymous referee provided useful recommendation for which we are grateful.

†School of Mathematical and Computational Sciences, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, Scotland, *fairouz@macs.hw.ac.uk*

‡Mathematics and Computing Science, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, the Netherlands, *c.j.bloo@tue.nl*

The initial step we followed was to simply add class reduction to the cube. Alas, when extending the systems of the cube with $\rightsquigarrow_\beta$, we find that the subject reduction property (SR) which states that if $A \twoheadrightarrow_\beta B$ then B has the same type as A, no longer holds for six of the systems of the cube, although it holds for the weaker systems $\lambda_\rightarrow$ and $\lambda\underline{\omega}$. This problem however can be solved by also extending the cube with *definitions* which avoid the loss of information in the contexts needed to type terms. In this way, subject reduction will hold for all the systems of the cube. Definitions are a useful mechanism in manipulating contexts and we see them already used in programming languages under the name *let expressions*. Hence, in this way, our extension with definitions and class reduction gives type systems which are more useful for computation. We extend the cube with the new typing relation $\vdash^{\mathsf{c}}$, the definition mechanism and class reduction $\rightsquigarrow_\beta$ and show that $\vdash^{\mathsf{c}}$ is the most general relation that satisfies all the desirable properties of type systems. We also show that our notion of classes contains all the desirable information related to a term with respect to typing, strong normalisation, subject reduction, etc. The article is divided as follows:

- In Section 2, we adapt the definitions of [5] needed to introduce our new typing relations.
- In Section 3, we illustrate that our reduction is not unique in losing subject reduction when mixed with powerful type systems. We extend the eight type systems of the cube with reduction $\mapsto_\beta$ modulo σ-equivalence of [17] and show that, in general, subject reduction (SR) fails. [17] showed that σ-quivalence enjoys desirable properties with respect to normalisation and the length of reductions. In [5], we showed that $\rightsquigarrow_\beta$ subsumes reduction modulo σ-equivalence $\mapsto_\beta$. So, even a weaker reduction than $\rightsquigarrow_\beta$ loses SR in the systems of the cube. We explain how SR can be restored to the cube with $\mapsto_\beta$, from the restored cube with $\rightsquigarrow_\beta$.
- In Section 4 we extend the Barendregt cube with $\rightsquigarrow_\beta$ and show that subject reduction holds for $\lambda_\rightarrow$ and $\lambda\underline{\omega}$ but fails for the six other systems.
- In Section 5, in addition to $\rightsquigarrow_\beta$, we add definitions in contexts of the cube. We show that all the desirable properties including SR and strong normalisation hold for all the systems of the extended cube with $\rightsquigarrow_\beta$ and definitions. Using classes means that we cannot use the usual methods for establishing SR. The vital point is that classes preserve types and that the cube with definitions only enjoys SR. As for the proof of strong normalisation (SN), we translate typing judgements of the extended cube into the ordinary cube by removing the definitions from the contexts and unfolding them in terms. In this way, SN of the extended cube is a result of that of the ordinary cube.

2 Adapting the results and notation of [5] for types

Let V be an infinite collection of variables over which $x, y, z, \ldots$ range and let $\pi \in \{\lambda, \Pi\}$. The set of *pseudo-expressions* $\mathcal{T}$ (also called terms) is defined by: $\mathcal{T} = * \mid \Box \mid V \mid (\mathcal{T}\delta)\mathcal{T} \mid (\mathcal{T}\pi_V)\mathcal{T}$.

We take $A, B, C, D, E, M, N, a, b, \ldots$, resp. S, S_1, S_2 to range over $\mathcal{T}$ resp. $\{*, \Box\}$.

We assume familiarity with the λ-calculus and notions like compatibility and reduction (cf. [1]). We use the **item notation** (cf. [8]) where one writes the argument before the function so ab becomes $(b\delta)a$, and one writes $\pi_{x:B}.a$ as $(B\pi_x)a$. This way, a term A is a sequence (possibly empty) of **Π-items** $(B\Pi_x)$, **λ-items** $(B\lambda_x)$ and **δ-items** $(b\delta)$, followed by a variable called the **heart** of A, notation $\heartsuit(A)$. We use $s_1, s_2, \ldots$ to range over items and call b the **body** of the δ-item $(b\delta)$. A sequence of items is called a **segment**. We use $\overline{s}, \overline{s}_1, \overline{s}_2, \ldots$ to range over segments and write $\emptyset$ for the empty segment. If $\overline{s} \equiv s_1 s_2 \ldots s_n$, we call the s_i's (for $1 \leq i \leq n$) the **main items** of $\overline{s}$; these s_i's are also the main items of $\overline{s}x$. The **weight** of a segment $\overline{s}$, $\mathtt{weight}(\overline{s})$, is the number of its main items. We define $\mathtt{weight}(\overline{s}x)$ to be $\mathtt{weight}(\overline{s})$. Terms have now specific forms:

Lemma 1 *Every term has one of the three forms: 1.* $(A_1\delta)\cdots(A_n\delta)x$, *where* $x \in V$ *and* $n \geq 0$, *2.* $(B\pi_x)A$, *and 3.* $(A_1\delta)\cdots(A_n\delta)(B\delta)(D\pi_x)C$, *where* $n \geq 0$.

Well-balanced segments (w-b) are constructed inductively from matching δ- and π-items:

- $\emptyset$ is w-b,
- if $\overline{s}$ is w-b then $(A\delta)\overline{s}(B\pi_x)$ is w-b,
- if $\overline{s_1}$, $\overline{s_2}$, $\ldots$ $\overline{s_n}$ are w-b, then the concatenation $\overline{s_1}\ \overline{s_2}, \cdots \overline{s_n}$ is w-b.

Let $E \equiv \overline{s_1}(A\delta)\overline{s_2}(B\pi_y)\overline{s_3}x$. We say that $(A\delta)$ and $(B\pi_y)$ **match** or are **partners** or **partnered** if $\overline{s_2}$ is w-b. If $\overline{s_2} \equiv \emptyset$, then $(A\delta)$ and $(B\pi_y)$ are $\delta\pi$**-pairs**, else, they are $\delta\pi$**-couples**. Hence, we speak of $\delta\lambda$-pairs/couples, $\delta\Pi$-pairs/couples, and $\delta\pi$-pairs/couples. If an item s has no partner, we say that s is **bachelor**. In item notation, a β-redex is a $\delta\lambda$**-pair**. Bound and free variables and substitution are defined as usual. We write $BV(A)$ and $FV(A)$ to represent the bound and free variables of A respectively. We write $A[x := B]$ to denote the term where all the free occurrences of x in A have been replaced by B. We take terms to be equivalent up to variable renaming and use $\equiv$ to denote syntactical equality of terms. We assume the usual Barendregt variable convention (which says that bound variables are always chosen distinct from free variables and that whenever necessary, variables are renamed to ensure this) (cf. [1]). The next definition is basic for this paper:

Definition 2

- *A term is in canonical form if it has the form:*

  $(B_1\pi_{x_1})\ldots(B_n\pi_{x_n})(C_1\delta)(D_1\pi_{y_1})\ldots(C_m\delta)(D_m\pi_{y_m})(A_1\delta)\ldots(A_l\delta)x$.[1]

- *For $r \in \{\beta, \theta, \gamma, p\}$, we define $\to_r$ as the compatible closure of the rule (r)*[2]*:*

(β)	$(B\delta)(C\lambda_x)A$	$\to_\beta$	$A[x := B]$
(θ)	$(C\delta)(B\delta)(D\pi_x)A$	$\to_\theta$	$(B\delta)(D\pi_x)(C\delta)A$
(γ)	$(B\delta)(D\pi_x)(E\pi_y)A$	$\to_\gamma$	$(E\pi_y)(B\delta)(D\pi_x)A$
(p)	$(A_1\delta)(B_1\pi_{y_1})(A_2\delta)(B_2\pi_{y_2})B$	$\to_p$	$(A_2\delta)(B_2\pi_{y_2})(A_1\delta)(B_1\pi_{y_1})B$ *if* $y_1 \notin FV(A_2) \cup FV(B_2)$

- *We define $\to_{\theta\gamma}$ to be $\to_\theta \cup \to_\gamma$ and $\to_{\theta\gamma p}$ to be $\to_\theta \cup \to_\gamma \cup \to_p$.*
- *We define σ-reduction $\to_\sigma$ as the smallest compatible relation containing (θ) and (γ).*
- *For a reduction relation $\to_r$ where $r \in \{\beta, \theta, \gamma, p, \theta\gamma, \theta\gamma p, \sigma\}$, we write $\twoheadrightarrow_r$ for its reflexive transitive closure and $=_r$ for its equivalence closure. A and B are σ-equivalent if $A =_\sigma B$.*
- *$\mapsto_\beta$ is the least compatible relation generated by: $A \mapsto_\beta B$ iff $\exists C =_\sigma A$ such that $C \to_\beta B$.*
- *$\mapsto\!\!\!\!\twoheadrightarrow_\beta$ is the reflexive transitive closure of $\mapsto_\beta$ and $\cong_\beta$ its reflexive transitive symmetric closure.*
- *The class $[A]$ of terms that are semi reductionally equivalent to A, is $\{B \mid \theta(\gamma(A)) =_p \theta(\gamma(B))\}$. We say that B is semi reductionally equivalent to A, and write $B \approx_{\mathtt{equi}} A$, iff $B \in [A]$.*
- *One-step class-reduction $\rightsquigarrow_\beta$ is the least compatible relation generated by:*
  $A \rightsquigarrow_\beta B$ iff $\exists A' \in [A]$ (i.e., $A' \approx_{\mathtt{equi}} A$) $\exists B' \in [B]$ (i.e., $B' \approx_{\mathtt{equi}} B$) such that $A' \to_\beta B'$.
- *$\rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta$ is the reflexive transitive closure of $\rightsquigarrow_\beta$ and $\approx_\beta$ its reflexive transitive symmetric closure.*

A is strongly normalizing with respect to $\to$ (written $\mathrm{SN}_\to(A)$ or $A \in \mathrm{SN}_\to(A)$) iff every $\to$-reduction path from A terminates. As usual, we use CR for Church Rosser.

Lemma 3

- $\to_\theta, \to_\gamma, =_\gamma, =_\theta, =_p \subsetneq \approx_{\mathtt{equi}} \subsetneq =_\beta$. *Moreover,* $\approx_{\mathtt{equi}} = =_{\theta\gamma p} = =_{\theta\gamma} = =_\sigma$.
- $\mapsto_\beta, \rightsquigarrow_\beta \subsetneq =_\beta$. *Moreover,* $\to_\beta \subsetneq \mapsto_\beta \subsetneq \rightsquigarrow_\beta$ *and* $\twoheadrightarrow_\beta \subsetneq \mapsto\!\!\!\!\twoheadrightarrow_\beta \subsetneq \rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta$. *Also,* $\cong_\beta = \approx_\beta = =_\beta$.
- $\to_\theta$, $\to_\gamma$ *and* $\to_{\theta\gamma}$ *are strongly normalising. Moreover,* $\to_\theta$, $\to_\gamma$, $\mapsto_\beta$ *and* $\rightsquigarrow_\beta$ *are CR.*
- *If $A \rightsquigarrow_\beta B$ then for all $A' \approx_{\mathtt{equi}} A$, for all $B' \approx_{\mathtt{equi}} B$, $A' \rightsquigarrow_\beta B'$.*
- *Let $\to \in \{\to_\beta, \rightsquigarrow_\beta\}$. If $A \in SN_\to$ and $A' \in [A]$ then $A' \in SN_\to$. Moreover, $SN_{\rightsquigarrow_\beta} = SN_{\to_\beta}$.*

[1] Note that, for $1 \le p, q \le m$, $1 \le i \le n$ and $1 \le j \le l$, D_p, C_q, B_i and A_j are not required to be canonical forms themselves, and that $(B_i\pi_{x_i})$ and $(A_j\delta)$ are bachelor.

[2] In (γ), the Barendregt convention on the right hand side ensures that the reductions are only allowed when $x \notin FV(E)$. Moreover, we keep to tradition and do not allow reduction to take place for a $\delta\Pi$-pair although [6, 15] give good reasons why this tradition needs to change and why Π-reduction is useful.

Table 1: Systems of the cube

System	*Set of* (S_1, S_2)*-rules*				*System*	*Set of* (S_1, S_2)*-rules*			
$\lambda_{\rightarrow}$	$(*,*)$				$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
$\lambda 2$	$(*,*)$	$(\Box,*)$			$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$
λP	$(*,*)$		$(*,\Box)$		$\lambda P2$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\lambda P\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$	$\lambda P\omega = \lambda C$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

3 The cube with the reduction modulo σ-equivalence $\mapsto_\beta$

In this section we extend the cube of [1] with β-reduction modulo σ-equivalence, $\mapsto_\beta$, and show that subject reduction fails. First, we give the typing rules of the original cube.

The systems of the cube are based on a the set of terms $\mathcal{T}$ and a set of rules $\mathcal{R} \subseteq \{*, \Box\}^2$. The reduction relation of the cube is $\rightarrow_\beta$. We define a context to be a sequence (possibly empty) of λ-items. We use $\Gamma, \Gamma', \Gamma_1, \ldots$ to denote contexts. We denote the empty context by $<>$.

Definition 4 (The typing rules of the cube in item notation)

(axiom) $<> \vdash * : \Box$

(start) $$\frac{\Gamma \vdash A : S}{\Gamma(A\lambda_x) \vdash x : A} \qquad \textit{if } x \textit{ is fresh}$$

(weak) $$\frac{\Gamma \vdash A : S \qquad \Gamma \vdash D : E}{\Gamma(A\lambda_x) \vdash D : E} \qquad \textit{if } x \textit{ is fresh}$$

(app) $$\frac{\Gamma \vdash F : (A\Pi_x)B \qquad \Gamma \vdash a : A}{\Gamma \vdash (a\delta)F : B[x := a]}$$

(abs) $$\frac{\Gamma(A\lambda_x) \vdash b : B \qquad \Gamma \vdash (A\Pi_x)B : S}{\Gamma \vdash (A\lambda_x)b : (A\Pi_x)B}$$

(conv) $$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : S \qquad B =_\beta B'}{\Gamma \vdash A : B'}$$

(form) $$\frac{\Gamma \vdash A : S_1 \qquad \Gamma(A\lambda_x) \vdash B : S_2}{\Gamma \vdash (A\Pi_x)B : S_2} \qquad \textit{if } (S_1, S_2) \in \mathcal{R}$$

A context or a term is called *legal* with respect to a type system if it occurs as such in a type-derivation in that system.

Each of the eight systems of the cube is obtained by taking the (S_1, S_2) rules from a subset $\mathcal{R}$ of $\{(*,*), (*,\Box), (\Box,*), (\Box,\Box)\}$. The basic system is the one where $(S_1, S_2) = (*,*)$ is the only possible choice. All other systems have this version of the formation rules, plus one or more other combinations of $(*,\Box)$, $(\Box,*)$ and $(\Box,\Box)$ for (S_1, S_2). Table 1 presents those eight systems.

To introduce β-reduction modulo σ-equivalence to the cube, we simply use $\mapsto_\beta$ instead of $\rightarrow_\beta$. This means that none of the typing rules changes and that our extended cube of this subsection is exactly that of Barendregt in [1] with the only difference that we use $\mapsto_\beta$ instead of $\rightarrow_\beta$.

The next two examples show that if our type derivation rules are those of Definition 4 and our reduction relation is $\mapsto_\beta$ instead of $\rightarrow_\beta$, then we lose the subject reduction property (SR) which states that if $\Gamma \vdash A : B$ and $A \twoheadmapsto_\beta A'$ then $\Gamma \vdash A' : B$.

Example 5 (With $\mapsto_\beta$, SR fails in $\lambda 2$, $\lambda P2, \lambda\omega$ and λC)
Let $\Gamma \equiv (*\lambda_\beta)(\beta\lambda_{y'})$, $A \equiv (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x$ *and* $B \equiv (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x$. *Then,* $\Gamma \vdash_{\lambda 2} A : \beta$ *and* $A \mapsto_\beta B$. *Yet,* $\Gamma \not\vdash_{\lambda 2} B : \beta$. *Even,* $\Gamma \not\vdash_{\lambda 2} B : \tau$ *for any* τ.
This is because $(\alpha\lambda_x)x : (\alpha\Pi_x)\alpha$ *and* $y : \beta$ *yet* α *and* β *are unrelated and hence we fail in firing the application rule to find the type of* $(y'\delta)(\alpha\lambda_x)x$. *Looking closer however, one finds that* $(\beta\delta)(*\lambda_\alpha)$

is defining α to be β, yet no such information can be used to combine $(\alpha\Pi_x)\alpha$ with β. Definitions take such information into account, but definitions are not part of the cube. Finally note that failure of SR in $\lambda 2$, means its failure in $\lambda P2, \lambda\omega$ and λC.

Example 6 (With $\mapsto_\beta$, SR fails in λP, $\lambda P2, \lambda P\underline{\omega}$ and λC)
Let $\Gamma \equiv (*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_C)((t\delta)C\lambda_A)$, $A \equiv (A\delta)(t\delta)(\sigma\lambda_x)((x\delta)C\lambda_y)(y\delta)((x\delta)C\lambda_Z)Z$ and $B \equiv (t\delta)(\sigma\lambda_x)(A\delta)((x\delta)C\lambda_Z)Z$. Then, $\Gamma \vdash_{\lambda P} A : (t\delta)C$ and $A \mapsto_\beta B$ but $\Gamma \not\vdash_{\lambda P} B : \tau$ for any τ, since as $A : (t\delta)C, y : (x\delta)C, (t\delta)C \neq (x\delta)C$.
Here again the reason of failure is similar to the above example. At one stage, we need to match $(x\delta)C$ with $(t\delta)C$ but this is not possible even though we do have the definition segment: $(t\delta)(\sigma\lambda_x)$ which defines x to be t. All this calls for the need to use these definitions. Finally note that failure of SR in λP, means its failure in $\lambda P2, \lambda P\underline{\omega}$ and λC.

The above two examples show that SR fails in six systems of the cube when β-reduction modulo σ-equivalence, $\mapsto_\beta$, is used. However, SR does not fail for the other two systems $\lambda_\to$ and $\lambda\underline{\omega}$. Furthermore, SR can be re-established for all the eight systems by allowing definitions in the context. We will not show this for this particular extension, but instead, we show it for a more general extension of the cube, that with class reduction $\rightsquigarrow_\beta$ rather than reduction modulo σ-equivalence. In fact, since $\mapsto_\beta \subset \rightsquigarrow_\beta$ and hence, as definitions restore subject reduction in the cube with $\rightsquigarrow_\beta$ (cf. Section 4), they will also restore subject reduction in the cube with $\mapsto_\beta$.

4 The cube with class reduction $\rightsquigarrow_\beta$

Alas, when extending the systems of the cube with $\rightsquigarrow_\beta$, we find that the subject reduction property which states that if $A \twoheadrightsquigarrow_\beta B$ then B has the same type as A, no longer holds for six of the systems of the cube, although it holds for the systems $\lambda_\to$ and $\lambda\underline{\omega}$. This problem however can be solved by also extending the cube with *definitions* which avoid the loss of information in the contexts needed to type terms. In this way, subject reduction will hold for all the systems of the cube.

In Section 4.1 we extend the cube with class reduction and show that that subject reduction fails for 6 systems of the cube with class reduction. In Section 4.2 we show that subject reduction holds for $\lambda_\to$ and $\lambda\underline{\omega}$ with $\twoheadrightsquigarrow_\beta$ (without definitions). We show furthermore that in $\lambda_\to$ and $\lambda\underline{\omega}$ with $\twoheadrightsquigarrow_\beta$, reductionally equivalent terms have the same type in the sense that if $\Gamma \vdash A : B$ then for all $A' \in [A]$, for all $B' \in [B]$, we have $\Gamma \vdash A' : B'$ (see Theorem 20).

4.1 Extending the cube with $\twoheadrightsquigarrow_\beta$

In this section, we introduce class-reduction to the cube of [1]. This means that our reduction relation now is not $\to_\beta$ but $\rightsquigarrow_\beta$ and that our extended cube of this subsection is exactly that of Barendregt in [1] with the only difference that we use $\rightsquigarrow_\beta$ instead of $\to_\beta$.

The same two examples (Examples 5 and 6) given for $\mapsto_\beta$ show that if our type derivation rules are those of Definition 4 and our reduction relation is $\twoheadrightsquigarrow_\beta$ instead of $\twoheadrightarrow_\beta$, then we lose the subject reduction property (SR) which states that if $\Gamma \vdash A : B$ and $A \twoheadrightsquigarrow_\beta A'$ then $\Gamma \vdash A' : B$.

Lemma 7 *In the cube with class reduction $\twoheadrightsquigarrow_\beta$, we have:*
SR fails in $\lambda 2$, $\lambda P2, \lambda\omega$ and λC and also in λP, $\lambda P2, \lambda P\underline{\omega}$ and λC.

Proof: Examples 5 and 6 also hold for $\twoheadrightsquigarrow_\beta$. □
The rest of this section proves that subject reduction holds for $\lambda_\to$ and $\lambda\underline{\omega}$.

Remark 8 *Because the extension of the cube in this section with $\twoheadrightsquigarrow_\beta$ does not involve any changes to the syntax or typing rules of the cube of [1], we assume the same notational convention of [1]. In particular, we take dom(Γ), subcontexts and $\Gamma \subset \Delta$ to have the usual meaning.*

The first three lemmas and corollary are exactly those of the cube of [1] because $\twoheadrightsquigarrow_\beta$ does not play any role in them. Only $\approx_\beta$ (which is the same as $=_\beta$) is involved.

Lemma 9 (Thinning for $\vdash$ and $\rightsquigarrow_\beta$) *Let Γ and Δ be legal contexts such that $\Gamma \subseteq' \Delta$. Then $\Gamma \vdash A : B \Rightarrow \Delta \vdash A : B$.*

Proof: Induction on the length of derivations $\Gamma \vdash A : B$. □

Lemma 10 (Generation Lemma for $\vdash$ and $\twoheadrightarrow_\beta$)

1. $\Gamma \vdash x : C \Rightarrow \exists S_1, S_2 \in \mathcal{S}\, \exists B =_\beta C[\Gamma \vdash B : S_1 \wedge (B\lambda_x) \in' \Gamma \wedge \Gamma \vdash C : S_2]$.
2. $\Gamma \vdash (A\Pi_x)B : C \Rightarrow \exists S_1, S_2 \in \mathcal{S}\, [\Gamma \vdash A : S_1 \wedge \Gamma(A\lambda_x) \vdash B : S_2 \wedge (S_1, S_2)$ *is a rule* $\wedge$ $C =_\beta S_2 \wedge [C \not\equiv S_2 \Rightarrow \exists S[\Gamma \vdash C : S]]]$
3. $\Gamma \vdash (A\lambda_x)b : C \Rightarrow \exists S, B\, [\Gamma \vdash (A\Pi_x)B : S \wedge \Gamma(A\lambda_x) \vdash b : B \wedge C =_\beta (A\Pi_x)B \wedge$ $C \not\equiv (A\Pi_x)B \Rightarrow \exists S' \in \mathcal{S}[\Gamma \vdash C : S']]$.
4. $\Gamma \vdash (a\delta)F : C \Rightarrow \exists A, B, x[\Gamma \vdash F : (A\Pi_x)B \wedge \Gamma \vdash a : A \wedge C =_\beta B[x := a] \wedge$ $(B[x := a] \not\equiv C \Rightarrow \exists S \in \mathcal{S}[\Gamma \vdash C : S])]$.

Proof: Induction on the derivation rules using thinning. □

Corollary 11 (Generation Corollary for $\vdash$ and $\twoheadrightarrow_\beta$)

1. *Correctness of Types: If $\Gamma \vdash A : B$ then $\exists S[B \equiv S$ or $\Gamma \vdash B : S]$.*
2. *If $\Gamma \vdash A : (B_1\Pi_x)B_2$ then $\exists S[\Gamma \vdash (B_1\Pi_x)B_2 : S]$.*
3. *If A is a $\Gamma^\vdash$-term, then A is $\Box$, a $\Gamma^\vdash$-kind or a Γ-element.*

Lemma 12 (Substitution for $\vdash$ and $\rightsquigarrow_\beta$) *If $\Gamma(B\lambda_x)\Delta \vdash C : D$ and $\Gamma \vdash A : B$, then $\Gamma\Delta[x := A] \vdash C[x := A] : D[x := A]$.*

Proof: By induction on the derivation rules, using the thinning lemma. □

4.2 Subject reduction and preservation of types by classes in $\lambda_\rightarrow$ and $\lambda\underline{\omega}$

Since $\rightsquigarrow_\beta$ is defined on classes instead of terms, we cannot use the usual methods for establishing Subject Reduction. For this, we need to establish that classes preserve types (Theorem 20). Subject Reduction will then be a corollary of the fact that classes preserve types. We start with a definition:

Definition 13 (Context Reduction and equivalence in the cube with $\vdash$ and $\twoheadrightarrow_\beta$)

1. *We define $\Gamma \rightarrow_r \Gamma'$ for $r \in \{\theta, \gamma, \theta\gamma\}$ by $\Gamma \equiv \Gamma_1(A\lambda_x)\Gamma_2$, $\Gamma' \equiv \Gamma_1(A'\lambda_x)\Gamma_2$ and $A \rightarrow_r A'$. We define $\twoheadrightarrow_r$ on contexts to be the reflexive transitive closure of $\rightarrow_r$.*
2. *We define $\Gamma \Rightarrow_{\theta\gamma} \Gamma'$ by $\Gamma \equiv \Gamma_1(A\lambda_x)\Gamma_2$, $\Gamma' \equiv \Gamma_1(A'\lambda_x)\Gamma_2$ and $A \twoheadrightarrow_{\theta\gamma} A'$. Note that on contexts, $=_{\theta\gamma}$ is the equivalence relation of $\Rightarrow_{\theta\gamma}$.*
3. *We say that $\Gamma' \in [\Gamma]$ if Γ' results from Γ by substituting some main items $(C\lambda_x)$ of Γ by $(C'\lambda_x)$ where $C' \in [C]$. Note that $\Gamma' \in [\Gamma]$ iff $\Gamma =_{\theta\gamma} \Gamma'$.*

In order to prove Theorem 20, which works for classes modulo $=_{\theta\gamma}$, we will show Lemmas 15..18 which deal with θ-reduction and γ-reduction. Lemma 14 is a help lemma (proved in the appendix). Item 1 simplifies the proofs by induction on the derivation rules since we can work with B instead of $B[x := C]$ for some C (it is used in the proofs of Item 2 and Lemmas 15..18). Item 2 combines various steps of Lemma 10 and is used in the proofs of Lemmas 16..18). Item 3 eliminates the case that π might be a Π when some typing condition holds. As a result of item 3, in what follows, when a term $(a\delta)(b\pi_x)c$ is typable, we write it as $(a\delta)(b\lambda_x)c$. Items 4 resp. 5 are needed in the proofs of Lemmas 17 and 18 resp. Lemma 17. Throughout, IH stands for Induction Hypothesis.

Lemma 14

1. *Since we only use rules* $(*,*)$ *and* $(\Box,\Box)$, *if* $\Gamma \vdash (A\Pi_x)B : S$ *then* $x \notin FV(B)$.
2. *[Redex Generation] If* $\Gamma \vdash (A\delta)(B\lambda_x)C : D$ *then* $\Gamma(B\lambda_x) \vdash C : D$, $\Gamma \vdash A : B$ *and* $\Gamma \vdash (B\Pi_x)D : S$ *for some sort* S.
3. *If* $\Gamma \vdash (a\delta)(b\pi_x)c : A$ *then* π *is* λ.
4. *[Interchange] If* $\Gamma(A\lambda_x)(B\lambda_y)\Delta \vdash C : D$ *and* $x \notin FV(B)$ *then also* $\Gamma(B\lambda_y)(A\lambda_x)\Delta \vdash C : D$.
5. *If* $\Gamma \vdash A : e$ *and* $e =_\beta S$ *for some sort* S, *then* $e \equiv S$.

Lemmas 15..18 are the basic building blocks to proving Theorem 20. In these lemmas, 1 and 2 are proven simultaneously by a tedious induction on the derivation of $\Gamma \vdash A : B$. See the appendix.

Lemma 15 (One step SR for $\to_\theta$ in the cube with $\vdash$ and $\rightsquigarrow_\beta$) *Let* $\Gamma \vdash A : B$.
1. *If* $A \to_\theta A'$ *then* $\Gamma \vdash A' : B$.
2. *If* $\Gamma \to_\theta \Gamma'$ *then* $\Gamma' \vdash A : B$.

Lemma 16 (One step SR for $\leftarrow_\theta$ in the cube with $\vdash$ and $\rightsquigarrow_\beta$) *Let* $\Gamma \vdash A : B$.
1. *If* $A' \to_\theta A$ *then* $\Gamma \vdash A' : B$.
2. *If* $\Gamma' \to_\theta \Gamma$ *then* $\Gamma' \vdash A : B$.

Lemma 17 (One step SR for $\to_\gamma$ in the cube with $\vdash$ and $\rightsquigarrow_\beta$) *Let* $\Gamma \vdash A : B$.
1. *If* $A \to_\gamma A'$ *then* $\Gamma \vdash A' : B$.
2. *If* $\Gamma \to_\gamma \Gamma'$ *then* $\Gamma' \vdash A : B$.

Lemma 18 (One step SR for $\leftarrow_\gamma$ in the cube with $\vdash$ and $\rightsquigarrow_\beta$) *Let* $\Gamma \vdash A : B$.
1. *If* $A' \to_\gamma A$ *then* $\Gamma \vdash A' : B$.
2. *If* $\Gamma' \to_\gamma \Gamma$ *then* $\Gamma' \vdash A : B$.

Corollary 19 (SR for classes, $\twoheadrightarrow_{\theta\gamma}$, $\twoheadleftarrow_{\theta\gamma}$ in the cube with $\vdash$ and $\rightsquigarrow_\beta$) *Let* $\Gamma \vdash A : B$.
1. *If* $A \twoheadrightarrow_{\theta\gamma} A'$ *then* $\Gamma \vdash A' : B$.
2. *If* $A' \twoheadrightarrow_{\theta\gamma} A$ *then* $\Gamma \vdash A' : B$.
3. *If* $\Gamma \Rightarrow_{\theta\gamma} \Gamma'$ *then* $\Gamma' \vdash A : B$.
4. *If* $\Gamma' \Rightarrow_{\theta\gamma} \Gamma$ *then* $\Gamma' \vdash A : B$.
5. *If* $A' \in [A]$ *then* $\Gamma \vdash A' : B$.
6. *If* $\Gamma \vdash A : B$ *and* $\Gamma' \in [\Gamma]$ *then* $\Gamma' \vdash A : B$.

Proof: Items 1 and 2 are by induction on the length of the reduction $\twoheadrightarrow_{\theta\gamma}$ using Lemmas 15..18. Items 3 and 4 are by induction on the derivation $\Gamma \vdash A : B$ using 1 and 2. For 5, if $A' \in [A]$ then $A' =_{\theta\gamma} A$ by Proposition 3 and $\exists A''$ where $A' \twoheadrightarrow_{\theta\gamma} A''$ and $A \twoheadrightarrow_{\theta\gamma} A''$; by 1, $\Gamma \vdash A'' : B$ (as $A \twoheadrightarrow_{\theta\gamma} A''$ and $\Gamma \vdash A : B$) and hence by 2, $\Gamma \vdash A' : B$. For 6, as $\Gamma' \in [\Gamma]$, there are main items $(C_1\lambda_{x_1}), \ldots, (C_n\lambda_{x_n})$ of Γ (for $n \geq 0$) which are replaced in Γ' by $(C'_1\lambda_{x_1}), \ldots, (C'_n\lambda_{x_n})$ where $C'_i \in [C_i]$ and otherwise, Γ and Γ' are the same. The proof is by induction on n.

- Case $n = 0$ nothing to prove as $\Gamma \equiv \Gamma'$.
- If $n = 1$ then assume $\Gamma' \in [\Gamma]$ is due to $(C'\lambda_x) \in \Gamma'$ and $(C\lambda_x) \in \Gamma$ where $C' \in [C]$ and this is the only difference between Γ and Γ'. Then as for 1. $C' =_{\theta\gamma} C$ and $\exists C''$ such that $C' \twoheadrightarrow_{\theta\gamma} C''$ and $C \twoheadrightarrow_{\theta\gamma} C''$. Let Γ'' be the same as Γ but where $(C\lambda_x)$ is replaced by $(C''\lambda_x)$. Then, $\Gamma \Rightarrow_{\theta\gamma} \Gamma''$ and $\Gamma' \Rightarrow_{\theta\gamma} \Gamma''$. By Item 3, $\Gamma'' \vdash A : B$ (because $\Gamma \Rightarrow_{\theta\gamma} \Gamma''$ and $\Gamma \vdash A : B$). Also, by Item 4, $\Gamma' \vdash A : B$ (because $\Gamma' \Rightarrow_{\theta\gamma} \Gamma''$ and $\Gamma'' \vdash A : B$) and we are done.
- Assume the property holds for some $n \geq 1$ and take Γ and Γ' which differ by $n+1$ such $(C_i\lambda_{x_i})$. Let Γ'' be Γ' but where $(C'_{n+1}\lambda_{x_{n+1}})$ is replaced by the original item $(C_{n+1}\lambda_{x_{n+1}})$ of Γ. Hence Γ'' and Γ differ only in n items. Hence, by IH $\Gamma'' \vdash A : B$. But Γ'' and Γ' differ by 1 item only and hence, again by IH, $\Gamma' \vdash A : B$. □

Theorem 20 (Classes preserve types in the cube with $\vdash$ and $\rightsquigarrow_\beta$)
$\Gamma \vdash A : B \iff \forall \Gamma' \in [\Gamma], \forall A' \in [A], \forall B' \in [B]$, *we have* $\Gamma' \vdash A' : B'$.

Proof: $\Longleftarrow$) is obvious. $\Longrightarrow$) By Corollary 19.6, $\Gamma' \vdash A : B$ and Corollary 19.5, $\Gamma' \vdash A' : B$. By Correctness of Types (Corollary 11.1) $B \equiv \Box$ or $\Gamma' \vdash B : S$ for some sort S.

- If $B \equiv \Box$ then as $B' \in [B]$, we also have $B' \equiv \Box$ and hence $\Gamma' \vdash A' : B'$.

- If $\Gamma' \vdash B : S$, then as $B' \in [B]$ we have by Corollary 19.5, that $\Gamma' \vdash B' : S$. Now, as $=_{\theta\gamma} \subseteq =_\beta$, we use (conv) to get $\Gamma' \vdash A' : B'$. □

Now with Theorem 20, we can establish SR using $\vdash$ with $\rightsquigarrow\!\!\!\!\!\rightarrow_\beta$, via SR of $\vdash$ with $\twoheadrightarrow_\beta$.

Corollary 21 (Subject Reduction for $\vdash$ and $\rightsquigarrow_\beta$)
If $\Gamma \vdash A : B$ and $A \rightsquigarrow\!\!\!\!\!\rightarrow_\beta A'$ then $\Gamma \vdash A' : B$.

Proof: We prove $\Gamma \vdash A : B$, $A \rightsquigarrow_\beta A' \Longrightarrow \Gamma \vdash A' : B$. By definition of $\rightsquigarrow_\beta$, there are A_1, A_1' such that $A_1 \in [A]$, $A_1' \in [A']$ and $A_1 \rightarrow_\beta A_1'$. By Theorem 20, $\Gamma \vdash A_1 : B$. By subject reduction for the usual $\rightarrow_\beta$ we have $\Gamma \vdash A_1' : B$. Again by Theorem 20, $\Gamma \vdash A' : B$. □
Although SR fails for the six remaining systems of the cube with $\vdash$ of Definition 4 and $\rightsquigarrow\!\!\!\!\!\rightarrow_\beta$, strong normalisation holds for all the systems of the cube with $\vdash$ of Definition 4 and $\rightsquigarrow\!\!\!\!\!\rightarrow_\beta$. Instead of proving this here, we move to the version that indeed satisfies SR and all other properties.

5 Extending the cube with $\rightsquigarrow\!\!\!\!\!\rightarrow_\beta$ and definitions

In this section we add definitions to our extension of Section 4 and show in that all the desirable properties including SR hold for all the systems of the extended cube with $\rightsquigarrow_\beta$ and definitions.

Looking back at, for instance, Example 5, one notices that when reducing using $\rightsquigarrow_\beta$, the information that y' has replaced y of type α is lost. All we know after the reduction is that y' has type β. But we need y' of type α to be able to type the subterm $(y'\delta)(\alpha\lambda_x)x$ of the reduct. Definitions enable us to have extra information in our contexts such as "α and β can be identified". We do this by writing in our context: $(\beta\delta)(*\lambda_\alpha)$ which expresses that α is defined to be β and is of type $*$. Next, we give the notion of definitions and how they can be unfolded. A definition identifies a variable with a whole term. The unfolding of the definition, undoes this identification and the variable will be replaced everywhere it occurs free by the term it identifies.

Definition 22 (definitions, unfolding)
- *If $\overline{s}$ is a well-balanced segment not containing $\delta\Pi$-couples, then a segment $(B\delta)\overline{s}(C\lambda_x)$ occurring in a context is called a definition.*
- *For $\overline{s}$ well-balanced segment, we define the unfolding of $\overline{s}$ in A, $|A|_{\overline{s}}$, inductively as follows: $|A|_\emptyset \equiv A$, $|A|_{(B\delta)\overline{s_1}(C\pi_x)} \equiv |A[x := B]|_{\overline{s_1}}$ and $|A|_{\overline{s_1}\,\overline{s_2}} \equiv ||A|_{\overline{s_2}}|_{\overline{s_1}}$. Note that substitution takes place from right to left.*

Lemma 23 *Let $\overline{s}$ be a well-balanced segment not containing main $\delta\Pi$-couples.*
1. *$|(A\delta)A'|_{\overline{s}} \equiv (|A|_{\overline{s}}\delta)|A'|_{\overline{s}}$ and $|A|_{\overline{s}} =_\beta \overline{s}A$.*
2. *If none of the binding variables of $\overline{s}$ is free in A, then $|A|_{\overline{s}} \equiv A$.*
3. *If none of the binding variables of $\overline{s}$ is free in A, then for any segment $\overline{s_1}$, $\overline{s_1}(A\delta)\overline{s}B =_\beta \overline{s_1}\,\overline{s}(A\delta)B$.*

Proof: 1. and 2. are by induction on $\mathtt{weight}(\overline{s})$.
3. is now obvious as $\overline{s_1}(A\delta)\overline{s}B \overset{3}{\equiv} \overline{s_1}(|A|_{\overline{s}}\delta)\overline{s}B \overset{2}{=_\beta} \overline{s_1}(|A|_{\overline{s}}\delta)|B|_{\overline{s}} \overset{1}{\equiv} \overline{s_1}\,\overline{s}(A\delta)B$. □
We now introduce some notions concerning typing rules which coincide with the usual ones when we do not allow definitions in the context (as is the case in the cube). When definitions are present however, the notions are more general. Let $\vdash$ be a typing relation and let $\rightarrow$ be a reduction relation whose reflexive transitive closure is $\twoheadrightarrow$ and whose equivalence closure is $=_\beta$.

Definition 24 (declarations, pseudocontexts, $\subseteq'$, $\rightsquigarrow_\beta$, definitional equality $=_{\mathtt{def}}$)
1. *A declaration d is a λ-item $(A\lambda_x)$; we define $\mathtt{subj}(d)$, $\mathtt{pred}(d)$ and $\underline{d}$ to be x, A and $\emptyset$ resp.*
2. *For a definition $d \equiv (B\delta)\overline{s}(A\lambda_x)$ let $\mathtt{subj}(d)$, $\mathtt{pred}(d)$, $\underline{d}$ and $\mathtt{def}(d)$ be x, A, $\overline{s}$ and B resp.*
3. *We use $d, d_1, d_2, \ldots$ to range over declarations and definitions.*

4. *A pseudocontext* Γ *is a concatenation of declarations and definitions such that if* $(A\lambda_x)$ *and* $(B\lambda_y)$ *are different main items of* Γ *then* $x \not\equiv y$. *We range* $\Gamma, \Delta, \Gamma', \Gamma_1, \ldots$ *over pseudocontexts.*

5. *For* Γ *a pseudocontext, define* $dom(\Gamma) = \{x \in V \mid (A\lambda_x)$ *is a main* λ*-item in* Γ *for some* $A\}$, $\Gamma\texttt{-def} = \{\overline{s} \mid \overline{s} \equiv (A\delta)\overline{s}_1(B\lambda_x)$ *is a main segment of* Γ *where* $\overline{s}_1$ *is well-balanced* $\}$, $\Gamma\texttt{-decl} = \{s \mid s$ *is a bachelor main* λ*-item of* $\Gamma\}$. *Note that* $dom(\Gamma) = \{\texttt{subj}(d) \mid d \in \Gamma\texttt{-decl} \cup \Gamma\texttt{-def}\}$.

6. *Define* $\subseteq'$ *between pseudocontexts as the least reflexive transitive relation satisfying:*

   - $\Gamma\Delta \subseteq' \Gamma(C\lambda_x)\Delta$ *if no* λ*-item in* Δ *matches a* δ*-item in* Γ
   - $\Gamma\underline{d}\Delta \subseteq' \Gamma d\Delta$ *if* d *is a definition*
   - $\Gamma\overline{s}(A\lambda_x)\Delta \subseteq' \Gamma(D\delta)\overline{s}(A\lambda_x)\Delta$ *if* $(A\lambda_x)$ *is bachelor in* $\Gamma\overline{s}(A\lambda_x)\Delta$ *and* $\overline{s}$ *is well-balanced*

7. *Reduction on pseudocontexts is defined by:*

   - $\Gamma(A\omega)\Gamma' \rightsquigarrow_\beta \Gamma(B\omega)\Gamma'$ *if* $A \twoheadrightsquigarrow_\beta B$, *for* $\omega \in \{\delta\} \cup \{\lambda_v : v \in V\}$.
   - $\Gamma(A\omega)\Gamma' \rightarrow_\beta \Gamma(B\omega)\Gamma'$ *if* $A \twoheadrightarrow_\beta B$, *for* $\omega \in \{\delta\} \cup \{\lambda_v : v \in V\}$.
   - $\twoheadrightsquigarrow_\beta$ *(resp.* $\twoheadrightarrow_\beta$*) on contexts is the reflexive transitive closure of* $\rightsquigarrow_\beta$ *(resp.* $\rightarrow_\beta$*).*

8. *We define the binary relation* $\Gamma \vdash \cdot =_{\texttt{def}} \cdot$ *to be the equivalence relation generated by*

   - *if* $A =_\beta B$ *then* $\Gamma \vdash A =_{\texttt{def}} B$
   - *if* $d \in \Gamma\texttt{-def}$ *and* $A, B \in \mathcal{T}$ *such that* B *arises from* A *by substituting one particular occurrence of* $\texttt{subj}(d)$ *in* A *by* $\texttt{def}(d)$, *then* $\Gamma \vdash A =_{\texttt{def}} B$.

Definition 25 (Statement, judgement, $\prec$)

1. *A statement is of the form* $A : B$, A *and* B *are called the subject and the predicate resp.*

2. *For pseudocontext* Γ *and statement* $A : B$, *we call* $\Gamma \vdash A : B$ *a judgement, meaning* $A : B$ *is derivable from the context* Γ, *and we write* $\Gamma \vdash A : B : C$ *to mean* $\Gamma \vdash A : B \wedge \Gamma \vdash B : C$.

3. *For pseudocontext* Γ *and definition/declaration* d, *we say that* Γ *invites* d, *notation* $\Gamma \prec d$, *iff*

   - Γd *is a pseudocontext,*
   - $\Gamma\underline{d} \vdash \texttt{pred}(d) : S$ *for some sort* S,[3] *and*
   - *if* d *is a definition then* $\Gamma\underline{d} \vdash \texttt{def}(d) : \texttt{pred}(d)$ *and* $FV(\texttt{def}(d)) \subseteq dom(\Gamma)$.

4. *For declarations/definitions* $d, d_1, \ldots, d_n$, *define* $\Gamma \vdash d$ *and* $\Gamma \vdash d_1 \cdots d_n$ *simultaneously by:*

   - *If* d *is a definition:* $\Gamma \vdash d$ *iff* $\Gamma \vdash \texttt{subj}(d) : \texttt{pred}(d) \wedge \Gamma \vdash \texttt{def}(d) : \texttt{pred}(d) \wedge \Gamma \vdash \underline{d} \wedge \Gamma \vdash \texttt{subj}(d) =_{\texttt{def}} \texttt{def}(d)$. *If* d *is a declaration:* $\Gamma \vdash d$ *iff* $\Gamma \vdash \texttt{subj}(d) : \texttt{pred}(d)$.
   - $\Gamma \vdash d_1 \cdots d_n$ *iff* $\Gamma \vdash d_i$ *for all* $1 \leq i \leq n$.

5. A *is* $\Gamma^\vdash$*-term if* $\exists B[\Gamma \vdash A : B$ *or* $\Gamma \vdash B : A]$. $\Gamma^\vdash$*-terms* $= \{A \mid \exists B[\Gamma \vdash A : B \vee \Gamma \vdash B : A]\}$. A *is called legal if* $\exists \Gamma[A \in \Gamma^\vdash$*-terms*$]$. Γ *is called legal if* $\exists A, B$ *such that* $\Gamma \vdash A : B$.

6. *We take* $\Gamma^\vdash$*-kinds* $= \{A \mid \Gamma \vdash A : \Box\}$ *and* $\Gamma^\vdash$*-types* $= \{A \in \mathcal{T} \mid \Gamma \vdash A : *\}$.

7. A *is a* Γ*-element if* $\exists B, S[\Gamma \vdash A : B$ *and* $\Gamma \vdash B : S]$.

Now we will in the definition below present the rules of Definition 4 differently. Note that in Definition 26, if one takes d to be a meta-variable for declarations only, $=_{\texttt{def}}$ the same as $=_\beta$ (which is independent of $\vdash$) and the reduction relation as $\rightarrow_\beta$, then one gets the known cube of [1] given in Definition 4. We invite the reader to check this.

[3] Note that binding variables in $\underline{d}$ may occur free in $\texttt{pred}(d)$ but not in $\texttt{def}(d)$ if $\Gamma \prec d$.

Table 2: Definitions solve subject reduction

$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathtt{c}} y' : \beta : * : \Box$	
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathtt{c}} y' : \beta, \alpha : *$	(weakening resp. start)
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathtt{c}} \alpha =_{\mathtt{def}} \beta$	(use the definition in the context)
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathtt{c}} y' : \alpha$	(conversion)
$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x) \vdash^{\mathtt{c}} x : \alpha$	(start)
$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathtt{c}} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \alpha[x := y][\alpha := \beta] \equiv \beta$	(definition rule)

Definition 26 (Axioms and rules of the cube with the $\prec$ notation)

(axiom) $<> \vdash * : \Box$

(start) $$\frac{\Gamma \prec d}{\Gamma d \vdash \mathtt{subj}(d) : \mathtt{pred}(d)}$$

(weak) $$\frac{\Gamma \prec d \qquad \Gamma \underline{d} \vdash D : E}{\Gamma d \vdash D : E}$$

(app) $$\frac{\Gamma \vdash F : (A\Pi_x)B \qquad \Gamma \vdash a : A}{\Gamma \vdash (a\delta)F : B[x := a]}$$

(abs) $$\frac{\Gamma(A\lambda_x) \vdash b : B \qquad \Gamma \vdash (A\Pi_x)B : S}{\Gamma \vdash (A\lambda_x)b : (A\Pi_x)B} \quad \textit{if } (A\lambda_x) \textit{ is bachelor in } \Gamma(A\lambda_x)$$

(conv) $$\frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : S \quad \Gamma \vdash B =_{\mathtt{def}} B'}{\Gamma \vdash A : B'}$$

(form) $$\frac{\Gamma \vdash A : S_1 \quad \Gamma(A\lambda_x) \vdash B : S_2}{\Gamma \vdash (A\Pi_x)B : S_2} \quad \textit{if } (S_1, S_2) \textit{ is a rule and } (A\lambda_x) \textit{ is bachelor in } \Gamma(A\lambda_x)$$

If we did not use the $\prec$ notation, we would have needed for the cube with definitions, two rules for (start) and two rules for (weak) as follows (in (start-def) and (weak-def), take $d \equiv (B\delta)\underline{d}(A\lambda_x)$):

(start-dec) $$\frac{\Gamma \vdash A : S}{\Gamma(A\lambda_x) \vdash x : A} \qquad \text{if } x \text{ is fresh}$$

(start-def) $$\frac{\Gamma \underline{d} \vdash A : S \qquad \Gamma \underline{d} \vdash B : A}{\Gamma(B\delta)\underline{d}(A\lambda_x) \vdash x : A} \qquad \text{if } \Gamma d \text{ is a pseudocontext, } FV(B) \subseteq dom(\Gamma)$$

(weak-dec) $$\frac{\Gamma \vdash A : S \qquad \Gamma \vdash D : E}{\Gamma(A\lambda_x) \vdash D : E} \qquad \text{if } x \text{ is fresh}$$

(weak-def) $$\frac{\Gamma \underline{d} \vdash A : S \quad \Gamma \underline{d} \vdash B : A \quad \Gamma \underline{d} \vdash D : E}{\Gamma(B\delta)\underline{d}(A\lambda_x) \vdash D : E} \quad \text{if } \Gamma d \text{ is a pseudocontext, } FV(B) \subseteq dom(\Gamma)$$

In order to solve the SR problem for the six systems of the cube, we extend the cube with definitions, $\twoheadrightarrow_\beta$ and equivalence classes modulo $=_{\theta\gamma}$. Contexts now have declarations and definitions.

Definition 27 (Axioms and rules of the cube with both $\twoheadrightarrow_\beta$ and definitions) *The typing rules $\vdash^{\mathtt{c}}$ are exactly those of $\vdash$ of Definition 26 but with the addition of the definition rule:*

$$\textit{(def rule)} \qquad \frac{\Gamma d \vdash^{\mathtt{c}} C : D}{\Gamma \vdash^{\mathtt{c}} dC : |D|_d} \qquad \textit{if } d \textit{ is a definition}$$

In this new system, the problem of subject reduction is solved, and all the other desirable properties hold too. The reason that subject reduction holds now whereas it did not hold in Examples 5 and 6 can be intuitively seen by showing that the counterexample given in Example 5 no longer holds. Table 2 shows how the reduct of Example 5 can now be typed.

From the point of view of efficiency, it may seem unsatisfactory that in the (def rule) definitions are being unfolded in D, since this will usually mean a size explosion of the predicate. However, the unfolding is not necessary for non-topsorts (i.e. for $D \not\equiv \Box$) as the following lemma shows:

Lemma 28 *The following rule is a derived rule:*

$$\textit{(derived def rule)} \qquad \frac{\Gamma d \vdash^c C : D \qquad \Gamma d \vdash^c D : S}{\Gamma \vdash^c dC : dD} \textit{ if } d \textit{ is a definition}$$

Proof: If $\Gamma d \vdash^c C : D$ then by the (def rule), $\Gamma \vdash^c dC : |D|_d$; if $\Gamma d \vdash^c D : S$ then by the (def rule) $\Gamma \vdash^c dD : S$. Now by conversion $\Gamma \vdash^c dC : dD$ since $\Gamma \vdash^c dD =_{\mathtt{def}} |D|_d$. □
If D is a sort then of course unfolding d in D is not inefficient since d will disappear.

Due to the possibility of using the (def rule) to type a redex, by using the (derived def rule), in some cases it is even possible to circumvent a size explosion: suppose we want to derive in λC a type for the term $(B\delta)(*\lambda_\beta)(\beta\lambda_x)((\beta\Pi_y)\beta\lambda_f)(x\delta)f$.

In λC without definions, we will have to derive first the type $(*\Pi_\beta)(\beta\Pi_x)((\beta\Pi_y)\beta\Pi_f)\beta$ for the subterm $(*\lambda_\beta)(\beta\lambda_x)((\beta\Pi_x)\beta\lambda_f)(x\delta)f$, and by the application rule we will finally derive the type $(B\Pi_x)((B\Pi_y)B\Pi_f)B$. Note that due to the last applied application rule the term B has been copied four times, which could make the resulting type very large.

Using our type system extended with definitions however, we would first derive the type $(\beta\Pi_x)((\beta\Pi_y)\beta\Pi_f)\beta$ for the term $(\beta\lambda_x)((\beta\Pi_y)\beta\lambda_f)(x\delta)f$, and then by the derived definition rule we would derive the type $(B\delta)(*\lambda_\beta)(\beta\Pi_x)((\beta\Pi_y)\beta\Pi_f)\beta$ and avoid the substitution of B for β. This is a further evidence for the advantage of using definitions.

5.1 Properties of the cube with $\twoheadrightarrow_\beta$ and definitions

Lemma 29 (Free Variable Lemma for $\vdash^c$ and $\twoheadrightarrow_\beta$) *Assume $\Gamma \vdash^c B : C$. The following holds:*

1. *If d and d' are two different elements of Γ-$\mathtt{decl} \cup \Gamma$-$\mathtt{def}$, then $\mathtt{subj}(d) \not\equiv \mathtt{subj}(d')$.*
2. *$FV(B), FV(C) \subseteq dom(\Gamma)$.*
3. *If $\Gamma = \Gamma_1 s_1 \Gamma_2$ then $FV(s_1) \subseteq dom(\Gamma_1)$.*

Proof: All by induction on the derivation of $\Gamma \vdash^c B : C$. □

Lemma 30 (Start Lemma for $\vdash^c$ and $\twoheadrightarrow_\beta$)
Let Γ be a legal context. Then $\Gamma \vdash^c * : \Box$ and $\forall d \in \Gamma[\Gamma \vdash^c d]$.

Proof: Γ is legal $\Rightarrow \exists B, C[\Gamma \vdash^c B : C]$; use induction on the derivation $\Gamma \vdash^c B : C$. □

Lemma 31 (Transitivity Lemma for $\vdash^c$ and $\twoheadrightarrow_\beta$)
Let Γ and Δ be legal contexts and define $\Gamma \vdash^c \Delta$ as usual. Then we have:
$[\Gamma \vdash^c \Delta \wedge \Delta \vdash^c A : B] \Rightarrow \Gamma \vdash^c A : B$.

Proof: Induction on the derivation $\Delta \vdash^c A : B$. By the compatibility of $\Gamma \vdash^c C =_{\mathtt{def}} D$ it follows that if $d \in \Delta$ and D arises from C by substituting one particular free occurrence of $\mathtt{subj}(d)$ in C by $\mathtt{def}(d)$, then $\Gamma \vdash^c C =_{\mathtt{def}} D$ and hence $\Delta \vdash^c C =_{\mathtt{def}} D$ implies $\Gamma \vdash^c C =_{\mathtt{def}} D$. □

By the next lemma, nested definitions like $(A\delta)(B\delta)(C\lambda_x)(D\lambda_y)$ work as linear definitions like $(B\delta)(C\lambda_x)(A\delta)(D\lambda_y)$. Moreover, abstractions can be interchanged with definitions.

Lemma 32 *Let d be a definition and note that $\mathtt{subj}(d) \notin FV(\underline{d})$.*

1. *If $\Gamma d\Delta \vdash^c C =_{\mathtt{def}} D$ then $\Gamma \underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \vdash^c C =_{\mathtt{def}} D$ and $\Gamma(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\underline{d}\Delta \vdash^c C =_{\mathtt{def}} D$.*
2. *If $x \notin FV(d)$ then $\Gamma(A\lambda_x)d\Delta \vdash^c C =_{\mathtt{def}} D$ iff $\Gamma d(A\lambda_x)\Delta \vdash^c C =_{\mathtt{def}} D$.*
3. *Let d' be a definition. If $\Gamma d\Delta \prec d'$ then $\Gamma \underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \prec d'$ and $\Gamma(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\underline{d}\Delta \prec d'$.*

4. *If* $\Gamma d\Delta \vdash^c C : D$ *then* $\Gamma \underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \vdash^c C : D$ *and* $\Gamma(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\underline{d}\Delta \vdash^c C : D$.

5. *Let* d' *be a definition. If* $x \notin FV(d)$ *then* $\Gamma(A\lambda_x)d\Delta \prec d'$ *iff* $\Gamma d(A\lambda_x)\Delta \prec d'$.

6. *If* $x \notin FV(d)$ *then* $\Gamma(A\lambda_x)d\Delta \vdash^c C : D$ *iff* $\Gamma d(A\lambda_x)\Delta \vdash^c C : D$.

Proof: Note that $(A\lambda_x)$ needs not be bachelor. 1. & 2. are by induction on the generation of $=_{\mathtt{def}}$. 3. & 4. are proven simultaneously by induction on the derivation $\Gamma d\Delta \vdash^c C : D$. 5. & 6. are split into implications and proven simultaneously by induction on the derivation ($\Gamma(A\lambda_x)d\Delta \vdash^c C : D$ for one implication and $\Gamma d(A\lambda_x)\Delta \vdash^c C : D$ for the other). 3 ... 6. need 1. & 2. for conversion. □

The following three lemmas and corollary are familiar from [1], but take definitions into account.

Lemma 33 (Thinning for $\vdash^c$ and $\twoheadrightarrow_\beta$)

1. *If* $\Gamma_1\Gamma_2 \vdash^c A =_{\mathtt{def}} B$, $\Gamma_1\Delta\Gamma_2$ *is a legal context, then* $\Gamma_1\Delta\Gamma_2 \vdash^c A =_{\mathtt{def}} B$.

2. *If* Γ *and* Δ *are legal contexts such that* $\Gamma \subseteq' \Delta$ *and* $\Gamma \vdash^c A : B$, *then* $\Delta \vdash^c A : B$.

Proof: 1. is by induction on the derivation $\Gamma_1\Gamma_2 \vdash^c A =_{\mathtt{def}} B$. 2. is done by showing:

- If $\Gamma\Delta \vdash^c A : B$, $\Gamma \vdash^c C : S$, x is fresh, and no λ-item in Δ is partnered by a δ-item in Γ, then also $\Gamma(C\lambda_x)\Delta \vdash^c A : B$. By induction on the derivation $\Gamma\Delta \vdash^c A : B$ using 1. for conversion.

- If $\Gamma\overline{s}\Delta \vdash^c A : B$, $\Gamma\overline{s} \vdash^c C : D : S$, $FV(C) \subseteq dom(\Gamma)$, x is fresh, $\overline{s}$ is well-balanced, then also $\Gamma(C\delta)\overline{s}(D\lambda_x)\Delta \vdash^c A : B$. We show this by induction on $\Gamma\overline{s}\Delta \vdash^c A : B$. For (start) for instance where $\Gamma(A\delta)\overline{s}(B\lambda_y) \vdash^c y : A$ comes from $\Gamma\overline{s} \vdash^c A : B : S$, y fresh and $FV(A) \subseteq dom(\Gamma)$, then $\Gamma(C\delta)\overline{s}(D\lambda_x) \vdash^c A : B : S$ by IH so again by (start), $\Gamma(C\delta)(A\delta)\overline{s}(B\lambda_y)(D\lambda_x) \vdash^c x : A$.

- If $\Gamma\overline{s}(A\lambda_x)\Delta \vdash^c B : C$, $(A\lambda_x)$ bachelor, $\overline{s}$ well-balanced, $\Gamma\overline{s} \vdash^c D : A$, $FV(D) \subseteq dom(\Gamma$, then $\Gamma(D\delta)\overline{s}(A\lambda_x)\Delta \vdash^c B : C$. We show this by induction on $\Gamma\overline{s}(A\lambda_x)\Delta \vdash^c B : C$. □

Lemma 34 (Generation Lemma for $\vdash^c$ and $\twoheadrightarrow_\beta$)

1. *If* $\Gamma \vdash^c x : A$ *then for some* B, S: $(B\lambda_x) \in \Gamma$, $\Gamma \vdash^c B : S$, $\Gamma \vdash^c A =_{\mathtt{def}} B$ *and* $\Gamma \vdash^c A : S'$ *for some* S'.

2. *If* $\Gamma \vdash^c (A\lambda_x)B : C$ *then for some* D, S: $\Gamma(A\lambda_x) \vdash^c B : D$, $\Gamma \vdash^c (A\Pi_x)D : S$, $\Gamma \vdash^c (A\Pi_x)D =_{\mathtt{def}} C$ *and if* $(A\Pi_x)D \not\equiv C$ *then* $\Gamma \vdash^c C : S'$ *for a sort* S'.

3. *If* $\Gamma \vdash^c (A\Pi_x)B : C$ *then for some* S_1, S_2: $\Gamma \vdash^c A : S_1$, $\Gamma \vdash^c B : S_2$, (S_1, S_2) *is a rule,* $\Gamma \vdash^c C =_{\mathtt{def}} S_2$ *and if* $S_2 \not\equiv C$ *then* $\Gamma \vdash^c C : S$ *for some* S.

4. *If* $\Gamma \vdash^c (A\delta)B : C$, $(A\delta)$ *bachelor in* B , *then for some* D, E, x: $\Gamma \vdash^c A : D$, $\Gamma \vdash^c B : (D\Pi_x)E$, $\Gamma \vdash^c E[x := A] =_{\mathtt{def}} C$ *and if* $E[x := A] \not\equiv C$ *then* $\Gamma \vdash^c C : S$ *for some sort* S.

5. *If* $\Gamma \vdash^c \overline{s}A : B$, *then* $\Gamma\overline{s} \vdash^c A : B$ *for well balanced* $\overline{s}$.

Proof: 1., 2., 3. and 4. follow by a tedious but straightforward induction on the derivations (use Thinning Lemma 33). As to 5., use induction on $\mathtt{weight}(\overline{s})$. □

Lemma 35 (Substitution Lemma for $\vdash^c$ and $\twoheadrightarrow_\beta$) *Let* d *be a definition.*

1. *If* $\Gamma d\Delta \vdash^c A =_{\mathtt{def}} B$, A *and* B *are* $\Gamma d\Delta$*-legal terms, then* $\Gamma \underline{d}\Delta[\mathtt{subj}(d) := \mathtt{def}(d)] \vdash^c A[\mathtt{subj}(d) := \mathtt{def}(d)] =_{\mathtt{def}} B[\mathtt{subj}(d) := \mathtt{def}(d)]$.

2. *If* B *is a* Γd*-legal term, then* $\Gamma d \vdash^c B =_{\mathtt{def}} |B|_d$.

3. *If* $\Gamma(A\lambda_x)\Delta \vdash^c B : C$, $\Gamma \vdash^c D : A$ *and* $(A\lambda_x)$ *bachelor in* $\Gamma(A\lambda_x)\Delta$ *then* $\Gamma\Delta[x := D] \vdash^c B[x := D] : C[x := D]$.

4. *If* $\Gamma(D\delta)\overline{s}(A\lambda_x)\Delta \vdash^c B : C$ *and* $\overline{s}$ *well-balanced then* $\Gamma\overline{s}\Delta[x := D] \vdash^c B[x := D] : C[x := D]$.

5. *If* $\Gamma d\Delta \vdash^c C : D$, *then* $\Gamma|\Delta|_d \vdash^c |C|_d : |D|_d$.

Proof: 1. Induction on the derivation rules of $=_{\mathtt{def}}$. 2. Induction on the structure of B. 3. and 4. Induction on the derivation rules, using 1., 2. and Lemma 33. Finally, 5. is a corollary of 3. □

Corollary 36 (Correctness of Types for $\vdash^c$ and $\twoheadrightarrow_\beta$)
If $\Gamma \vdash^c A : B$ *then* $B \equiv \Box$ *or* $\Gamma \vdash^c B : S$ *for some sort* S.

Proof: Induction on the derivation rules. The interesting rules are application and definition:

- Case $\Gamma \vdash^c dA : |B|_d$ results from $\Gamma d \vdash^c A : B$, then by IH $B \equiv \Box$ or $\Gamma d \vdash^c B : S$ for some S. In the first case $|B|_d \equiv \Box$, in the second case by the Substitution Lemma $\Gamma \vdash^c |B|_d : |S|_d \equiv S$.
- Case $\Gamma \vdash^c (a\delta)F : B[x := a]$ results from $\Gamma \vdash^c F : (A\Pi_x)B$, $\Gamma \vdash^c a : A$, then by IH $\Gamma \vdash^c (A\Pi_x)B : S$ for some S and hence by Generation $\Gamma(A\lambda_x) \vdash^c B : S$. Then by Lemma 33 $\Gamma(a\delta)(A\lambda_x) \vdash^c B : S$, so by the definition rule $\Gamma \vdash^c (a\delta)(A\Pi_x)B : S[x := a] \equiv S$. □

5.2 Subject reduction and preservation of types by classes for $\vdash^c$ and $\twoheadrightarrow_\beta$

Similarly to our earlier extension of the cube with class reduction ($\vdash$ and $\twoheadrightarrow_\beta$), we cannot use the usual methods for establishing Subject Reduction for $\vdash^c$ and $\twoheadrightarrow_\beta$. For this, we need to establish that classes preserve types (Theorem 45) and that Subject Reduction holds for $\vdash^c$ and $\twoheadrightarrow_\beta$ (Theorem 37). Subject Reduction for $\vdash^c$ and $\twoheadrightarrow_\beta$ will then be a corollary of Theorems 37 and 45.

Theorem 37 (Subject Reduction for $\vdash^c$ and $\twoheadrightarrow_\beta$)
If $\Gamma \vdash^c A : B$ *and* $A \twoheadrightarrow_\beta A'$ *then* $\Gamma \vdash^c A' : B$.

Proof: We show by simultaneous induction on the derivation rules that:
1. If $\Gamma \vdash^c A : B$ and $\Gamma \to_\beta \Gamma'$ then $\Gamma' \vdash^c A : B$ and
2. If $\Gamma \vdash^c A : B$ and $A \to_\beta A'$ then $\Gamma \vdash^c A' : B$

using Lemmas 34.5 and 35 when reduction is at the root. □

Similarly to Theorem 20, in order to prove Theorem 45 we need to establish four lemmas which will be the basic blocks for the proof of Theorem 45. We start with a definition:

Definition 38 (Context Reduction and equivalence for the cube with $\vdash^c$ and $\twoheadrightarrow_\beta$)

1. *Let* $r \in \{\theta, \gamma, \theta\gamma\}$.
   - *We say* $\Gamma \to_r \Gamma'$ *if* $\Gamma \equiv \Gamma_1\overline{s}\Gamma_2$, $\Gamma' \equiv \Gamma_1\overline{s}'\Gamma_2$ *where*
     - *Either* $\overline{s} \equiv (A\lambda_x)$, $\overline{s}' \equiv (A'\lambda_x)$ *and* $A \to_r A'$ *or*
     - $\overline{s}$ *is well-balanced and* $\overline{s} \to_r \overline{s}'$.
   - *We say that* $\Gamma \Rightarrow_r \Gamma'$ *if* $\Gamma \equiv \Gamma_1\overline{s}\Gamma_2$ *and* $\Gamma' \equiv \Gamma_1\overline{s}'\Gamma_2$ *where*
     - *Either* $\overline{s} \equiv (A\lambda_x)$, $\overline{s}' \equiv (A'\lambda_x)$ *and* $A \twoheadrightarrow_r A'$ *or*
     - $\overline{s}$ *is well-balanced and* $\overline{s} \twoheadrightarrow_r \overline{s}'$.
   - *We define* $\twoheadrightarrow_r$ *(resp.* $\Rightarrow\!\!\!\!\twoheadrightarrow_r$*) as the reflexive transitive closure of* $\to_r$ *(resp.* $\Rightarrow_r$*).*
     It is easy to show that on contexts, the equivalence relation based on $\Rightarrow\!\!\!\!\twoheadrightarrow_{\theta\gamma}$ *is* $=_{\theta\gamma}$.
2. *We say that* $\Gamma' \in [\Gamma]$ *iff* $\Gamma =_{\theta\gamma} \Gamma'$.[4]

[4] Note that this implies that Γ' and Γ are the same except that both items below hold:

- There are $d_1 \ldots d_n$ $(n \geq 0)$ declarations/definitions in Γ which are replaced in Γ' by declarations/definitions $d_1' \ldots d_n'$ such that $d_i' \in [d_i]$.
- There are main well-balanced segments $\overline{s_1} \ldots \overline{s}_n$ $(n \geq 0)$ in Γ which are replaced in Γ' by main well-balanced segments $\overline{s_1}' \ldots \overline{s_n}'$ such that $\overline{s_i}' \in [\overline{s_n}]$.

Note here that we are treating contexts like terms. If you have any problem with this, use any sort S say, and write $\Gamma S =_{\theta\gamma} \Gamma' S$.

The following lemma will be used in the proofs of Lemmas 40...43.

Lemma 39

1. *If* $B \in [A]$ *then* $FV(A) = FV(B)$.
2. *If* $B \in [A]$ *and* A *doesn't contain partnered* Π*-items then* B *doesn't contain partnered* Π*-items.*
3. *For well balanced segments* d, d'*, if* $d =_{\theta\gamma} d'$ *then* $|C|_d \equiv |C|_{d'}$.

Proof: 1. Induction on the structure of A. 2. Induction on the number of symbols in A. 3. Direct consequence of: $A[x := B][y := C] \equiv A[y := C][x := B]$ if $y \notin FV(C)$ and $x \notin FV(B)$. □
Here are now the four lemmas which form the basic blocks for the proof of Theorem 45. As for Lemmas 15...18, the proof of these lemmas is by a tedious simultaneous induction on the length of the derivation, distinguishing cases according to the last rule in 1 and 2. See the appendix.

Lemma 40 (One step SR for $\to_\theta$ in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$)

1. *Let* $\Gamma \vdash^c A : B$. *a) If* $A \to_\theta A'$ *then* $\Gamma \vdash^c A' : B$. *b) If* $\Gamma \to_\theta \Gamma'$ *then* $\Gamma' \vdash^c A : B$.
2. *If* $\Gamma \prec d$ *and* $\Gamma \to_\theta \Gamma'$ *then* $\Gamma' \prec d$.
3. *If* $\Gamma \prec d$ *and* $d \to_\theta d'$ *then either* $\Gamma \prec d'$ *or (there exists* $\overline{s}, d''$ *such that* $d' \equiv \overline{s}d'', \overline{s}$ *well balanced,* d'' *is a definition and* $\Gamma\overline{s} \prec d''$*).*

Lemma 41 (One step SR for $\leftarrow_\theta$ in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$)

1. *Let* $\Gamma \vdash^c A : B$. *a) If* $A' \to_\theta A$ *then* $\Gamma \vdash^c A' : B$. *b) If* $\Gamma' \to_\theta \Gamma$ *then* $\Gamma' \vdash^c A : B$.
2. *Let* $\Gamma \prec d$. *a) If* $\Gamma' \to_\theta \Gamma$ *then* $\Gamma' \prec d$. *b) If* $d' \to_\theta d$ *then* $\Gamma \prec d'$.

Lemma 42 (One step SR for $\to_\gamma$ in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$)

1. *Let* $\Gamma \vdash^c A : B$. *a) If* $A \to_\gamma A'$ *then* $\Gamma \vdash^c A' : B$. *b) If* $\Gamma \to_\gamma \Gamma'$ *then* $\Gamma' \vdash^c A : B$.
2. *If* $\Gamma \prec d$ *and* $\Gamma \to_\gamma \Gamma'$ *then* $\Gamma' \prec d$.
3. *If* $\Gamma \prec d$ *and* $d \to_\gamma d'$ *then there exists* $\overline{s}, d''$ *such that* $d' \equiv d''\overline{s}$, $\overline{s}$ *is well balanced,* d'' *is a definition and* $\Gamma \prec d''$.

Lemma 43 (One step SR for $\leftarrow_\gamma$ in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$)

1. *Let* $\Gamma \vdash^c A : B$. *a) If* $A' \to_\gamma A$ *then* $\Gamma \vdash^c A' : B$. *b) If* $\Gamma' \to_\gamma \Gamma$ *then* $\Gamma' \vdash^c A : B$.
2. *Let* $\Gamma \prec d$ *a) If* $\Gamma' \to_\gamma \Gamma$ *then* $\Gamma' \prec d$. *b) If* $d' \to_\gamma d$ *then* $\Gamma \prec d'$.

Corollary 44 (SR for classes, $\twoheadrightarrow_{\theta\gamma}$, $\twoheadleftarrow_{\theta\gamma}$ in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$) *Let* $\Gamma \vdash^c A : B$.

1. *If* $A \twoheadrightarrow_{\theta\gamma} A'$ *then* $\Gamma \vdash^c A' : B$.
2. *If* $A' \twoheadrightarrow_{\theta\gamma} A$ *then* $\Gamma \vdash^c A' : B$.
3. *If* $\Gamma \Rightarrow\!\!\!\!\!\twoheadrightarrow_{\theta\gamma} \Gamma'$ *then* $\Gamma' \vdash^c A : B$.
4. *If* $\Gamma' \Rightarrow\!\!\!\!\!\twoheadrightarrow_{\theta\gamma} \Gamma$ *then* $\Gamma' \vdash^c A : B$.
5. *If* $A' \in [A]$ *then* $\Gamma \vdash^c A' : B$.
6. *If* $\Gamma' \in [\Gamma]$ *then* $\Gamma' \vdash^c A : B$.

Proof: Items 1 and 2 are by induction on on the length of the reduction $\twoheadrightarrow_{\theta\gamma}$ using Lemmas 40...43. Item 3: We only show it for $\Rightarrow_{\theta\gamma}$ because the proof for $\Rightarrow\!\!\!\!\!\twoheadrightarrow_{\theta\gamma}$ is by induction on the length of $\Rightarrow\!\!\!\!\!\twoheadrightarrow_{\theta\gamma}$. If $\Gamma \Rightarrow_{\theta\gamma} \Gamma'$ comes from $(C\lambda_x) \twoheadrightarrow_{\theta\gamma} (C'\lambda_x)$ then the proof is by induction on the length of the derivation $(C\lambda_x) \twoheadrightarrow_{\theta\gamma} (C'\lambda_x)$ using Lemmas 40..43. If $\Gamma \Rightarrow_{\theta\gamma} \Gamma'$ comes from $\overline{s} \twoheadrightarrow_{\theta\gamma} \overline{s}'$ then similar. Item 4 is similar to Item 3. Item 5: Let $A' \in [A]$. Then $A' =_{\theta\gamma} A$ by Proposition 3 and hence $\exists A''$ such that $A' \twoheadrightarrow_{\theta\gamma} A''$ and $A \twoheadrightarrow_{\theta\gamma} A''$. By 1, $\Gamma \vdash^c A'' : B$ (because $A \twoheadrightarrow_{\theta\gamma} A''$ and $\Gamma \vdash^c A : B$). By 2, $\Gamma \vdash^c A' : B$ (because $A' \twoheadrightarrow_{\theta\gamma} A''$ and $\Gamma \vdash^c A'' : B$). Item 6: As $\Gamma' \in [\Gamma]$ then $\Gamma' =_{\theta\gamma} \Gamma$ and hence there is Γ'' such that $\Gamma' \Rightarrow\!\!\!\!\!\twoheadrightarrow_{\theta\gamma} \Gamma''$ and $\Gamma \Rightarrow\!\!\!\!\!\twoheadrightarrow_{\theta\gamma} \Gamma''$. Now use items 3 and 4 to derive that $\Gamma' \vdash^c A : B$. □

Theorem 45 (Classes preserve types in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$)
$\Gamma \vdash^c A : B \iff \forall \Gamma' \in [\Gamma], \forall A' \in [A], \forall B' \in [B]$ *we have* $\Gamma' \vdash^c A' : B'$.

Proof: $\Longleftarrow$) is obvious. $\Longrightarrow$) By Corollary 44.6, $\Gamma' \vdash A : B$ and by Corollary 44.5, $\Gamma' \vdash A' : B$. By Corollary 36, $B \equiv \Box$ or $\Gamma' \vdash B : S$ for some sort S.

- If $B \equiv \Box$ then as $B' \in [B]$, we also have $B' \equiv \Box$ and hence $\Gamma' \vdash A' : B'$.
- If $\Gamma' \vdash B : S$, then as $B' \in [B]$ we have by Corollary 44.5, that $\Gamma' \vdash B' : S$. Now, as $=_{\theta\gamma} \subseteq =_\beta$, we use (conv) to get $\Gamma' \vdash A' : B'$. □

Here is now the proof of SR using $\vdash^c$ and $\rightsquigarrow\!\!\!\!\rightarrow_\beta$, via the SR of $\vdash^c$ and $\twoheadrightarrow_\beta$.

Corollary 46 (Subject Reduction for $\vdash^c$ and $\rightsquigarrow\!\!\!\!\rightarrow_\beta$)
If $\Gamma \vdash^c A : B$ *and* $A \rightsquigarrow\!\!\!\!\rightarrow_\beta A'$ *then* $\Gamma \vdash^c A' : B$.

Proof: We only prove $\Gamma \vdash^c A : B$, $A \rightsquigarrow_\beta A' \Longrightarrow \Gamma \vdash^c A' : B$. By definition of $\rightsquigarrow_\beta$, there are A_1, A_1' such that $A_1 \in [A]$, $A_1' \in [A']$ and $A_1 \to_\beta A_1'$. By Theorem 45, $\Gamma \vdash^c A_1 : B$. By subject reduction for $\vdash^c$ and $\to_\beta$ (Theorem 37), $\Gamma \vdash^c A_1' : B$. Again by Theorem 45, $\Gamma \vdash^c A' : B$. □

Lemma 47 (Unicity of Types for $\vdash^c$ and $\rightsquigarrow_\beta$)

1. *If* $\Gamma \vdash^c A : B$ *and* $\Gamma \vdash^c A : B'$ *then* $\Gamma \vdash^c B =_{\mathtt{def}} B'$
2. *If* $\Gamma \vdash^c A : B$ *and* $\Gamma \vdash^c A' : B'$ *and* $A =_\beta A'$ *then* $\Gamma \vdash^c B =_{\mathtt{def}} B'$

Proof: 1. By induction on the structure of A using the Generation Lemma. 2. By Church-Rosser and Subject Reduction using 1. □

Remark 48 *We didn't prove the property* $\Gamma \vdash^c B : S$, $\Gamma \vdash^c A : B'$, $B =_\beta B' \Rightarrow \Gamma \vdash^c B' : S$. *It seems difficult to prove because if* $\Gamma \vdash^c B' : S'$ *then by Unicity of Types* $\Gamma \vdash^c S =_{\mathtt{def}} S'$ *and it is unclear whether* $S \equiv S'$. *Also, it would be interesting whether* $\Gamma \vdash^c A : B$, $\Gamma \vdash^c A' : B'$, $\Gamma \vdash^c A =_{\mathtt{def}} A'$ *implies* $\Gamma \vdash^c B =_{\mathtt{def}} B'$, *but to prove this we face similar problems. We claim that one can prove it by showing first that* $\Gamma \vdash^c A : B$ *implies* $\Gamma \vdash^c |A|_\Gamma : |B|_\Gamma$, *where* $|A|_\Gamma$ *means that all definitions in* Γ *are unfolded in* A.

Fact 49 *Subtyping does not hold for* $\vdash^c$. *Consider the following derivable judgement:*

$$(*\lambda_\alpha) \vdash^c (\alpha\delta)(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z : (\alpha\Pi_y)\alpha$$

The subterm $(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z$ *is not typable: suppose* $\Gamma \vdash^c (*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z : A$, *then by the Generation Lemma,* $\Gamma' \vdash^c z : \alpha'$ *where* $\Gamma' \equiv \Gamma(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)$ *and* α' *satisfies* $\Gamma' \vdash^c \alpha =_{\mathtt{def}} \alpha'$ *and* $\Gamma' \vdash^c \alpha' : S$.

Since Γ *cannot contain bachelor* δ*-items, we know that* $(*\lambda_\beta)$ *is not partnered in* Γ', *hence* $\Gamma' \not\vdash^c \alpha =_{\mathtt{def}} \beta$. *But since* $(y\delta)(\alpha\lambda_z) \in \Gamma'$-`def` *we know that* $\Gamma(*\lambda_\beta)(\beta\lambda_y) \vdash^c y : \alpha : S$, *also* $\Gamma(*\lambda_\beta)(\beta\lambda_y) \vdash^c y : \beta$ *so by Unicity of Types,* $\Gamma(*\lambda_\beta)(\beta\lambda_y) \vdash^c \alpha =_{\mathtt{def}} \beta$, *contradiction.*

The reason for failure of subtyping is that when we typed the term $(\alpha\delta)(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z$, we used the context $(*\lambda_\alpha)(\alpha\delta)(*\lambda_\beta)$ to type $(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z$. In this context, β is defined to be α. Now, to type $(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z$, the definition $(\alpha\delta)(*\lambda_\beta)$ cannot be used. Hence, we don't have all the information necessary to derive the type of $(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\alpha\lambda_z)z$. We do however have a partial result concerning subtyping:

Lemma 50 (Restricted Subtyping in the cube with $\vdash^c$ and $\rightsquigarrow_\beta$) *If* $\Gamma \vdash^c A : B$, A' *is a subterm of* A *such that all bachelor items in* A' *are also bachelor in* A, *then* A' *is legal.*

Proof: We prove by induction on the derivations: if A' is a subterm of Γ or A such that all bachelor items in A' are also bachelor items in Γ respectively A, then A' is legal. Note that in the case of the (def rule) subterms $\overline{s_2}C$ where $d \equiv \overline{s_1}\,\overline{s_2}$ and $\overline{s_1}$ is not the empty segment, do not satisfy the restrictions, since at least one item of $\overline{s_2}$ is bachelor in $\overline{s_2}C$ but partnered in dC. □
Subterms satisfying the bachelor restriction as in Lemma 50, seem to be more important than those not satisfying it. The reason for this is that the latter terms have an extra abstraction (the newly bachelor λ-item) and hence are Π-types which makes them more involved, whereas the subterm property is useful because it tells something about less involved terms.

5.3 Strong Normalisation of the cube with $\rightsquigarrow_\beta$ and definitions

Finally, we establish strong normalisation for the cube extended with definitions and class-reduction. The absence of a stepwise definition unfolding reduction (in contrast to the work on definitions to the λ-cube in [19]) makes it possible to base the proof on a translation to the cube without definitions and class-reduction. This way, we avoid the complications [19] meets. We still need strong normalisation of λC in order to prove strong normalisation for all systems of the cube extended with definitions and class-reduction, but this is a weaker requirement than that of [19].

We start by defining the translation of judgements in the extended cube. In contexts, definitions must be removed. In terms and types, definitions from the context must be unfolded similar to definition unfolding in Definition 22.

Definition 51 *For contexts Γ, $||\Gamma||$ is defined inductively by*
$||\Gamma(A\lambda_x)|| \equiv ||\Gamma||(A\lambda_x), \qquad ||\Gamma d|| \equiv ||\Gamma||.$
For terms D and contexts Γ, $||D||_\Gamma$ is defined inductively by
$||D||_{\Gamma(A\lambda_x)} \equiv ||D||_\Gamma, \qquad ||D||_{\Gamma(B\delta)\overline{s}(A\lambda_x)} \equiv ||D[x := B]||_{\Gamma\overline{s}}.$

We first prove some auxiliary lemmas.

Lemma 52 *For all terms A, B and context Γ we have $||B[x := A]||_\Gamma \equiv ||B||_\Gamma[x := ||A||_\Gamma]$.*

Proof: Induction on the length of Γ using the substitution lemma which says that $A[x := B][y := C] \equiv A[y := C][x := B[y := C]]$. □

Lemma 53 *For all terms B, B' and context Γ, if $\Gamma \vdash^{\mathtt{c}} B =_{\mathtt{def}} B'$ then $||B||_\Gamma =_\beta ||B'||_\Gamma$.*

Proof: Induction on the length of Γ. □

Lemma 54 *If $\Gamma \vdash^{\lambda C} C[x := B] : D$, and $\Gamma \vdash^{\lambda C} B : A$ then also $\Gamma, x : A \vdash^{\lambda C} C : D$.*

Proof: Tedious induction on the structure of C. □

Lemma 55 *If $A[x := b] \in SN_{\rightarrow_\beta}$ then also $A \in SN_{\rightarrow_\beta}$.*

Proof: By contraposition. Suppose $A \notin SN_{\rightarrow_\beta}$, say $A \rightarrow_\beta A_1 \rightarrow_\beta A_2 \rightarrow_\beta A_3 \rightarrow_\beta \cdots$ is an infinite reduction. Then also $A[x := B] \rightarrow_\beta A_1[x := B] \rightarrow_\beta A_2[x := B] \rightarrow_\beta \cdots$, so $A[x := B] \notin SN_{\rightarrow_\beta}$. □

Now we prove that we have a translation from derivable $\vdash^{\mathtt{c}}$-judgements to derivable judgements in ordinary λC. We need the strong type system λC since redexes in terms as a result of the (def rule) can have arbitrary abstractions.

Theorem 56 *For all terms A, B and contexts Γ, if $\Gamma \vdash^{\mathtt{c}} A : B$ then $||\Gamma|| \vdash^{\lambda C} ||A||_\Gamma : ||B||_\Gamma$.*

Proof: Induction on the derivation of $\Gamma \vdash^{\mathtt{c}} A : B$. We consider two cases.

(app) $\Gamma \vdash^{\mathtt{c}} (a\delta)F : B[x := a]$ from $\Gamma \vdash^{\mathtt{c}} F : (A\Pi_x)B$ and $\Gamma \vdash^{\mathtt{c}} a : A$.

By IH we know that $||\Gamma|| \vdash^{\lambda C} ||F||_\Gamma : ||(A\Pi_x)B||_\Gamma$ and $||\Gamma|| \vdash^{\lambda C} ||a||_\Gamma : ||A||_\Gamma$.

Note that $||(A\Pi_x)B||_\Gamma \equiv (||A||_\Gamma\Pi_x)||B||_\Gamma$ since by the variable convention, $x \notin dom(\Gamma)$. Hence by (app) in λC we get $||\Gamma|| \vdash^{\lambda C} (||a||_\Gamma\delta)||F||_\Gamma : ||B||_\Gamma[x := ||a||_\Gamma]$. Now we have $(||a||_\Gamma\delta)||F||_\Gamma \equiv ||(a\delta)F||_\Gamma$ and $||B||_\Gamma[x := ||a||_\Gamma] \equiv ||B[x := a]||_\Gamma$ by Lemma 52.

(def rule) $\Gamma \vdash^{\mathtt{c}} dC : ||D||_d$ as a consequence of $\Gamma d \vdash^{\mathtt{c}} C : D$. By IH we have $||\Gamma d|| \vdash^{\lambda C} ||C||_{\Gamma d} : ||D||_{\Gamma d}$.

We show by induction on the length of d that now $|\Gamma| \vdash^{\lambda C} ||dC||_\Gamma : |||D|_d||_\Gamma$.

Suppose d is not empty, say $d \equiv (B\delta)\overline{d}(A\lambda_x)$. First note that $|||D|_d||_\Gamma \equiv ||D||_{\Gamma d}$, $||\Gamma d|| \equiv ||\Gamma\overline{d}||$ $(\equiv ||\Gamma||)$ and that $||dC||_\Gamma \equiv ||C[x := B]||_{\Gamma\overline{d}} \equiv ||C||_{\Gamma\overline{d}}[x := ||B||_{\Gamma\overline{d}}]$.

Since Γd is $\vdash^{\mathsf{c}}$-legal we have $\Gamma\overline{d} \vdash^{\mathsf{c}} B : A$ so by IH we have $||\Gamma\overline{d}|| \vdash^{\lambda C} ||B||_{\Gamma\overline{d}} : ||A||_{\Gamma\overline{d}}$. Then by Lemma 54 we have $||\Gamma\overline{d}||, x : ||A||_{\Gamma\overline{d}} \vdash^{\lambda C} ||C||_{\Gamma\overline{d}} : ||D||_{\Gamma d}$, so by the (abs) rule (since we are in λC the necessary formation is allowed) we have $||\Gamma\overline{d}|| \vdash^{\lambda C} (||A||_{\Gamma\overline{d}}\lambda_x)||C||_{\Gamma\overline{d}} : (||A||_{\Gamma\overline{d}}\Pi_x)||D||_{\Gamma d}$, which is equivalent to $||\Gamma\overline{d}|| \vdash^{\lambda C} ||(A\lambda_x)C||_{\Gamma\overline{d}} : (||A||_{\Gamma\overline{d}}\Pi_x)||D||_{\Gamma d}$. Note that $x \notin FV(||D||_{\Gamma d})$. Now by the second IH we get $||\Gamma|| \vdash^{\lambda C} ||\overline{d}(A\lambda_x)C||_{\Gamma} : (||A||_{\Gamma\overline{d}}\Pi_x)||D||_{\Gamma d}$, and since $||B||_{\Gamma\overline{d}} \equiv ||B||_{\Gamma}$, we also have $||\Gamma|| \vdash^{\lambda C} ||B||_{\Gamma} : ||A||_{\Gamma\overline{d}}$. Therefore by the (app) rule, $||\Gamma|| \vdash^{\lambda C} (||B||_{\Gamma}\delta)||\overline{d}(A\lambda_x)C||_{\Gamma} : ||D||_{\Gamma d}$ which is $||\Gamma|| \vdash^{\lambda C} ||dC||_{\Gamma} : ||D||_{\Gamma d}$. □

Now we conclude our list of properties of the cube with $\vdash^{\mathsf{c}}$ and $\twoheadrightarrow_\beta$.

Corollary 57 (Strong Normalisation for $\vdash^{\mathsf{c}}$ and $\twoheadrightarrow_\beta$) *If $\Gamma \vdash^{\mathsf{c}} A : B$ then $A \in SN_{\twoheadrightarrow_\beta}$.*

Proof: Suppose $\Gamma \vdash^{\mathsf{c}} A : B$. By Theorem 56, $||\Gamma|| \vdash^{\lambda C} ||A||_{\Gamma} : ||B||_{\Gamma}$ and since λC is strongly normalising, $||A||_{\Gamma} \in SN_{\to_\beta}$. Now $||A||_{\Gamma} \equiv A[x_1 := A_1]\cdots[x_n := A_n]$ for some $n, x_1, \ldots, x_n$ and terms $A_1, \ldots, A_n$. Therefore, by Lemma 55, also $A \in SN_{\to_\beta}$. Hence, by Lemma 3, $A \in SN_{\twoheadrightarrow_\beta}$. □

6 Conclusion

In this paper, we extended the cube of eight type systems with class reduction and showed that subject reduction fails for six of the eight extended systems. We then established that subject reduction can be regained by adding definitions. The importance of definitions (also known as "let expressions") is witnessed by their extensive use in programming languages and theorem provers. Intuitively, definitions repair the problem of subject reduction because they save the type information that otherwise would have been lost as a result of reduction.

Our typing relation $\vdash^{\mathsf{c}}$ is the most general relation so far in the literature that satisfies all the desirable properties of type systems and which we have shown to be more general than all the rest. We defined a notion of classes of terms which contain all the desirable information related to a term with respect to normalisation, subject reduction, etc., and we showed that this notion is more general than any classification of terms that exists in the literature. We showed that if $A' \in [A]$ then A' and A have the same normalisation behaviour and that if $\Gamma \vdash^{\mathsf{c}} A : B$ then for any $\Gamma' \in [\Gamma]$, for any $A' \in A$ and for any $B' \in B$, $\Gamma' \vdash^{\mathsf{c}} A' : B'$. We believe that our type system based on classes is a non-trivial extension of the usual typing relations and that it deserves attention.

References

[1] H.P. Barendregt. λ-calculi with types. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 118–310. Oxford University Press, 1992.

[2] Olivier Danvy and Lasse R. Nielsen. CPS transformation of beta-redexes. In Amr Sabry, editor, *Proceedings of the Third ACM SIGPLAN Workshop on Continuations*, Technical report 545, Computer Science Department, Indiana University, pages 35–39, London, England, January 2001. Also available as the technical report BRICS RS-00-35.

[3] P. de Groote. The conservation theorem revisited. In *International Conference on Typed Lambda Calculi and Applications, LNCS*, volume 664. Springer-Verlag, 1993.

[4] F. Kamareddine. Postponement, conservation and preservation of strong normalisation for generalised reduction. *Logic and Computation*, 10(5):721–738, 2000.

[5] F. Kamareddine and R. Bloo. De Bruijn's syntax and reductional equivalence of lambda terms: the untyped case. Logic and Algebraic Programming, 2004.

[6] F. Kamareddine, R. Bloo, and R. Nederpelt. On Π-conversion in the λ-cube and the combination with abbreviations. *Annals of Pure and Applied Logic*, 97(1–3):27–45, 1999.

[7] F. Kamareddine, R. Bloo, and R. P. Nederpelt. De Bruijn's syntax and reductional equivalence of lambda terms. In *Proc. 3rd Int'l Conf. Principles & Practice Declarative Programming*, pages 16–27, 5–7 September 2001.

[8] F. Kamareddine and R. Nederpelt. A useful λ-notation. *Theoretical Computer Science*, 155:85–109, 1996.

[9] F. Kamareddine, A. Ríos, and J.B. Wells. Calculi of generalised β_e-reduction and explicit substitution: Type free and simply typed versions. *Journal of Functional and Logic Programming*, 1998.

[10] M. Karr. Delayability in proofs of strong normalizability in the typed λ-calculus. In *Mathematical Foundations of Computer Software, LNCS*, volume 185. Springer-Verlag, 1985.

[11] A.J. Kfoury, J. Tiuryn, and P. Urzyczyn. An analysis of ML typability. *ACM*, 41(2):368–398, 1994.

[12] A.J. Kfoury and J.B. Wells. Addendum to new notions of reduction and non-semantic proofs of β-strong normalisation in typed λ-calculi. Technical report, Boston University, 1995.

[13] Z. Khasidashvili. The longest perpetual reductions in orthogonal expression reduction systems. *Proc. of the 3^{rd} International Conference on Logical Foundations of Computer Science, Logic at St Petersburg*, 813, 1994.

[14] J. W. Klop. Combinatory Reduction Systems. *Mathematical Center Tracts*, 27, 1980. CWI.

[15] S. Peyton-Jones and E. Meijer. Henk: a typed intermediate language. *Types in Compilation Workshp*, 1997.

[16] L. Regnier. *Lambda calcul et réseaux*. PhD thesis, University Paris 7, 1992.

[17] L. Regnier. Une équivalence sur les lambda termes. *Theoretical Computer Science*, 126:281–292, 1994.

[18] A. Sabry and M. Felleisen. Reasoning about programs in continuation-passing style. *Proceedings of the 1992 ACM Conference on LISP and Functional Programming*, pages 288–298, 1992.

[19] P. Severi and E. Poll. Pure type systems with definitions. In A. Nerode and Yu.V. Matiyasevich, editors, *Proceedings of LFCS'94 (LNCS* **813***)*, pages 316–328, New York, 1994. LFCS'94, St. Petersburg, Russia, Springer Verlag.

[20] M. H. Sørensen. Strong normalisation from weak normalisation in typed λ-calculi. *Information and Computation*, 133(1), 1997.

[21] H. Xi. On weak and strong normalisations. Technical Report 96-187, Carnegie Mellon University, 1996.

A Proofs of the lemmas

Proof:[Lemma 14]

1. We first prove (by induction on the derivations) that $\Gamma \not\vdash \Box : A$ for any A, and that if $\Gamma \vdash A : \Box$ then $FV(A) = \emptyset$. Then if $\Gamma \vdash (A\Pi_x)B : \Box$, by Generation, $\Gamma \vdash B : \Box$, so $FV(B) = \emptyset$.

   For the case $\Gamma \vdash A : *$, we first prove by induction on the derivations that if $\Gamma \vdash A : A' : \Box$, then $FV(A) \subseteq \{x \mid \exists A'' : \Gamma \vdash x : A'' : \Box\}$. Now we prove that if $\Gamma \vdash (A\Pi_x)B : *$ then $x \notin FV(B)$: since we are in $\lambda\underline{\omega}$, $\Gamma(A\lambda_x) \vdash B : *$ and $\Gamma \vdash A : *$. Since $\Gamma(A\lambda_x)$ is a legal context, we have $\Gamma \vdash x : A : *$, but $FV(B) \subseteq \{y \mid \exists A'' : \Gamma(A\lambda_x) \vdash x : A'' : \Box\}$, so $x \notin FV(B)$.

2. Applying the Generation Lemma 10 twice we get first $\Gamma \vdash (B\lambda_x)C : (A'\Pi_y)B'$, $\Gamma \vdash A : A'$, $B'[y := A] =_\beta D$ and then $\Gamma(B\lambda_x) \vdash C : B''$, $(A'\Pi_y)B' =_\beta (B\Pi_x)B''$ and $\Gamma \vdash (B\Pi_x)B'' : S$. Now, it follows that $B' =_\beta B''$ and by Lemma 14.1, $B'[y := A] \equiv B'$ so $B'' =_\beta D$.
   Then by (conv) we have $\Gamma(B\lambda_x) \vdash C : D$. But $\Gamma \vdash (B\Pi_x)B'' : S$ implies by Generation Lemma 10 that $\Gamma \vdash B'' : S$. Also, $A' =_\beta B$ so by (conv) $\Gamma \vdash A : B$.

3. Note that by Generation Lemma 10, $\Gamma \vdash (b\pi_x)c : (B\Pi_y)D$. If $\pi \equiv \Pi$ then again use Generation Lemma 10 to get that $(B\Pi_y)D =_\beta S$ for some S, which is absurd.

4. By induction on the derivations. (axiom): easy. (app), (abs), (conv), (form): use IH.

   - (start): if interchanging is in Γ, use IH. Otherwise, $\Gamma(A\lambda_x)(B\lambda_y) \vdash y : B$ as a conclusion of $\Gamma(A\lambda_x) \vdash B : S$ and $x \notin FV(B)$. We must prove that $\Gamma(B\lambda_y)(A\lambda_x) \vdash y : B$. By the converse of Lemma 9 (for ordinary λ-calculus), which is proven by van Benthem-Jutting and listed in [1], we have $\Gamma \vdash B : S$, so by (start) $\Gamma(B\lambda_y) \vdash y : B$. Now, since $\Gamma(A\lambda_x)$ is a legal context, $\Gamma \vdash A : S'$ for some sort S' and thus by (weak) $\Gamma(B\lambda_y)(A\lambda_x) \vdash y : B$.
   - (weak): if interchanging is in Γ, use IH. Otherwise, $\Gamma(A\lambda_x)(B\lambda_y) \vdash D : E$ as a conclusion of $\Gamma(A\lambda_x) \vdash B : S$, $\Gamma(A\lambda_x) \vdash D : E$ and $y \notin FV(B)$. We must prove $\Gamma(B\lambda_y)(A\lambda_x) \vdash D : E$. Since, $\Gamma(A\lambda_x) \vdash D : E$ and $y \notin FV(A) \cup FV(D) \cup FV(E)$, by Thinning Lemma 9 for the ordinary λ-calculus we get $\Gamma(B\lambda_y)(A\lambda_x) \vdash D : E$.

5. Suppose $e =_\beta S$ and $e \not\equiv S$. Then $e \twoheadrightarrow_\beta^+ S$ so there is an e' such that $e \twoheadrightarrow_\beta e' \to_\beta S$. Now by Corollary 11.1, $\Gamma \vdash e : S'$ for some sort S' and by Subject Reduction for $\twoheadrightarrow_\beta$, $\Gamma \vdash e' : S'$. But, $e' \to_\beta S$ means that $e' \equiv (J\delta)(H\lambda_x)I$ for some H, I, J such that $I[x := J] \equiv S$. But then either $I \equiv S$ or ($I \equiv x$ and $J \equiv *$). It is easy to check by Lemma 10 that such application of and abstraction over a sort are impossible for any system of the cube. □

Proof:[Lemma 15] We prove 1 and 2 simultaneously by induction on the derivation of $\Gamma \vdash A : B$.

- Case (axiom): No θ-reduction is possible.
- Case (start) where $\Gamma(A\lambda_x) \vdash x : A$ comes from $\Gamma \vdash A : S$ and x is fresh, then the only possible θ-reduction is in Γ or $(A\lambda_x)$.
  - If θ-reduction is in Γ, use IH.
  - If θ-reduction is in $(A\lambda_x)$, by IH, $\Gamma \vdash A' : S$ and hence by (start) $\Gamma(A'\lambda_x) \vdash x : A'$. By (conv), $A =_\beta A'$ (Proposition 3) and $\Gamma(A'\lambda_x) \vdash A : S$ (weak), we get $\Gamma(A'\lambda_x) \vdash x : A$.
- Case (weak), (conv), (abs) or (form), use IH. For (abs), also use (conv).
- Case (app) where $\Gamma \vdash (a\delta)F : B[x := a]$ comes from $\Gamma \vdash F : (A\Pi_x)B$ and $\Gamma \vdash a : A$:
  - If θ-reduction is in Γ or F, use IH.
  - If θ-reduction is in a, then by Correctness of Types (Corollary 11.1), $\Gamma \vdash (A\Pi_x)B : S$ and by Lemma 14.1 $x \notin FV(B)$. Hence, $B[x := a] \equiv B$. Now, use IH.

– If $F \equiv (b\delta)(c\lambda_y)d$ and $(a\delta)F \to_\theta (b\delta)(c\lambda_y)(a\delta)d$, we must show that $\Gamma \vdash (b\delta)(c\lambda_y)(a\delta)d : B$ (note again by Lemma 14.1, $B[x := a] \equiv B$). By Lemma 10 we get from $\Gamma \vdash F : (A\Pi_x)B$ that $\Gamma(c\lambda_y) \vdash d : (A'\Pi_x)B'$ for some A', B' such that $(A'\Pi_x)B'[y := b] =_\beta (A\Pi_x)B$. Since we are in $\lambda_\to$ or $\lambda\underline{\omega}$, $y \notin FV((A'\Pi_x)B')$ by Lemma 14.1 so $A' =_\beta A$ and $B' =_\beta B$, and by (conv) $\Gamma(c\lambda_y) \vdash d : (A\Pi_x)B$. Now, by (weak) also $\Gamma(c\lambda_y) \vdash a : A$ so by (app) $\Gamma(c\lambda_y) \vdash (a\delta)d : B[x := a]$, i.e., $\Gamma(c\lambda_y) \vdash (a\delta)d : B$. But $\Gamma \vdash (c\Pi_y)B : S$ for some sort S, since B and $(A\Pi_x)B$ have the same type, and so by (abs) $\Gamma \vdash (c\lambda_y)(a\delta)d : (c\Pi_y)B$ and by (app) $\Gamma \vdash (b\delta)(c\lambda_y)(a\delta)d : B[y := b] \equiv B$. □

Proof:[Lemma 16] We prove 1 and 2 simultaneously by induction on the derivation of $\Gamma \vdash A : B$. As all cases are similar to Lemma 15, we only consider (app) where $\Gamma \vdash (a\delta)F : e[y := a]$ comes from $\Gamma \vdash F : (A\Pi_y)e$ and $\Gamma \vdash a : A$. The cases where θ-reduction is to either Γ or F or a are similar to the corresponding cases in the proof of Lemma 15. We consider the crucial case where $F \equiv (b\lambda_x)(c\delta)d$ and $(c\delta)(a\delta)(b\lambda_x)d \to_\theta (a\delta)F$. By Lemma 14.1, $y \notin FV(e)$ so $e[y := a] \equiv e$. We must therefore show that $\Gamma \vdash (c\delta)(a\delta)(b\lambda_x)d : e$. By Lemma 14.2, $\Gamma(b\lambda_x) \vdash (c\delta)d : e$, $\Gamma \vdash a : b$ and $\Gamma \vdash (b\Pi_x)e : S$ for a sort S. By Lemma 10 on $\Gamma(b\lambda_x) \vdash (c\delta)d : e$, there are B, C such that $\Gamma(b\lambda_x) \vdash d : (C\Pi_y)B$, $\Gamma(b\lambda_x) \vdash c : C$, $e =_\beta B[y := c]$ and $B[y := c] \not\equiv e$ implies $\Gamma \vdash e : S'$ for some sort S'. Now by Lemma 14.1, $y \notin FV(B)$ so $e =_\beta B$ and if $e \not\equiv B$ then $\Gamma \vdash e : S'$.
In order to show that $\Gamma \vdash (b\Pi_x)(C\Pi_y)B : S''$ for some sort S'', note from above that $\Gamma \vdash (b\Pi_x)e : S''$ for some sort S'' and both $(A\Pi_y)e$ and $(C\Pi_y)B$ are legal. Since the only formation rules are $(*, *)$ and $(\Box, \Box)$, this implies that b, e, C, and B are all typable and have type S''. Then, also $(b\Pi_x)(C\Pi_y)B$ has type S''. Hence, we can apply (abs) on $\Gamma(b\lambda_x) \vdash d : (C\Pi_y)B$ to get $\Gamma \vdash (b\lambda_x)d : (b\Pi_x)(C\Pi_y)B$. Since $\Gamma \vdash a : b$, by (app) we get $\Gamma \vdash (a\delta)(b\lambda_x)d : ((C\Pi_y)B)[x := a]$. Since $\Gamma(b\lambda_x) \vdash c : C$, by the Substitution Lemma 12 we have $\Gamma \vdash c[x := a] : C[x := a]$.
But, $c[x := a] := c$ because $(c\delta)(a\delta)(b\lambda_x)d \to_\theta (a\delta)(b\lambda_x)(c\delta)d$. So we get by (app) that $\Gamma \vdash (c\delta)(a\delta)(b\lambda_x)d : B[x := a][y := c]$ and since $x, y \notin FV(B)$ and $e =_\beta B$ we are done by (conv). □
Proof:[Lemma 17] We prove 1 and 2 simultaneously by induction on the derivation of $\Gamma \vdash A : B$.

- Case (axiom): nothing to prove.
- Case (start) where $\Gamma(A\lambda_x) \vdash x : A$ comes from $\Gamma \vdash A : S$ and x is fresh, then the only possible γ-reduction is in Γ or $(A\lambda_x)$. In the first case use IH, in the second, use IH, (start), (conv), (weak) and Proposition 3.
- Case (weak), (conv), (abs) or (form), use IH. For (abs), use also (conv).
- Case (app) where $\Gamma \vdash (a\delta)F : e[x := a]$ comes from $\Gamma \vdash F : (A\Pi_x)e$ and $\Gamma \vdash a : A$. If γ-reduction is in a, F or Γ, apply IH (if γ-reduction is in a, note that by Lemma 14.1, $x \notin FV(e)$ so $e[x := a] \equiv e$). Now we consider the crucial case where $\Gamma \vdash (a\delta)(b\lambda_x)(c\pi_y)d : e$ (i.e., $F \equiv (b\lambda_x)(c\pi_y)d$) with $\pi \in \{\Pi, \lambda\}$ and $x \notin FV(c)$. We must prove that $\Gamma \vdash (c\pi_y)(a\delta)(b\lambda_x)d : e$. By Lemma 14.2, $\Gamma(b\lambda_x) \vdash (c\pi_y)d : e$, $\Gamma \vdash a : b$ and $\Gamma \vdash (b\Pi_x)e : S$ for some sort S.
  - Suppose $\pi \equiv \Pi$. By Lemma 10 on $\Gamma(b\lambda_x) \vdash (c\pi_y)d : e$ we get $\Gamma(b\lambda_x)(c\lambda_y) \vdash d : S_2$, $\Gamma(b\lambda_x) \vdash c : S_1$, (S_1, S_2) is a rule and $e =_\beta S_2$ (and if $e \not\equiv S_2$ then $\Gamma(b\lambda_x) \vdash e : S'$). By Lemma 14.4 (note that $x \notin FV(c)$) we also have $\Gamma(c\lambda_y)(b\lambda_x) \vdash d : S_2$. If needed, we use (conv) to get $\Gamma(c\lambda_y)(b\lambda_x) \vdash d : e$ and by (abs) $\Gamma(c\lambda_y) \vdash (b\lambda_x)d : (b\Pi_x)e$. As $\Gamma \vdash a : b$ then by (weak) $\Gamma(c\lambda_y) \vdash a : b$ and by (app) $\Gamma(c\lambda_y) \vdash (a\delta)(b\lambda_x)d : e[x := a] \equiv e$. By Lemma 14.5, $e \equiv S_2$ so we can use formation to get that $\Gamma \vdash (c\Pi_y)(a\delta)(b\lambda_x)d : e$.
  - Suppose $\pi \equiv \lambda$. By Lemma 10 on $\Gamma(b\lambda_x) \vdash (c\lambda_y)d : e$ we have for some f that $\Gamma(b\lambda_x)(c\lambda_y) \vdash d : f$, $\Gamma(b\lambda_x) \vdash (c\Pi_y)f : S'$ and $(c\Pi_y)f =_\beta e$. By Lemma 14.4, $\Gamma(c\lambda_y)(b\lambda_x) \vdash d : f$. By Corollary 11.1, $\Gamma \vdash (b\Pi_x)e : S$ for some S. Hence by Lemma 10, $\Gamma \vdash b : S_1$, $\Gamma(b\lambda_x) \vdash e : S$ for some S_1 where (S_1, S) is a rule. Similarly, $\Gamma(b\lambda_x) \vdash c : S_3$, $\Gamma(b\lambda_x)(c\lambda_y) \vdash f : S'$ for some S_3 where (S_3, S') is a rule. Since $e =_\beta (c\Pi_y)f$, then by the termination of all the cube systems for $\to_\beta$ we have $S =_\beta S'$ and hence $S \equiv S'$. Also by Lemma 14.4, $\Gamma(c\lambda_y)(b\lambda_x) \vdash f : S' \equiv S$. By (weak) $\Gamma(c\lambda_y) \vdash b : S_1$. Hence since (S_1, S) is a rule, (form) gives $\Gamma(c\lambda_y) \vdash (b\Pi_x)f : S$. So by (abs) $\Gamma(c\lambda_y) \vdash (b\lambda_x)d : (b\Pi_x)f$.

Now by (weak) $\Gamma(c\lambda_y) \vdash a : b$ so by (app) $\Gamma(c\lambda_y) \vdash (a\delta)(b\lambda_x)d : f[x := a]$. Since by Lemma 14.1 $x \notin FV(f)$, $\Gamma(c\lambda_y) \vdash (a\delta)(b\lambda_y)d : f$. From $\Gamma(b\lambda_x) \vdash (c\Pi_y)f : S'$, $S \equiv S'$, $x \notin FV(c) \cup FV(f)$ and by the reverse of Thinning (Jutting, see [1]) it follows that $\Gamma \vdash (c\Pi_y)f : S$. So by (abs) we get: $\Gamma \vdash (c\lambda_y)(a\delta)(b\lambda_x)d : (c\Pi_y)f$. Now use (conv) to get $\Gamma \vdash (c\lambda_y)(a\delta)(b\lambda_x)d : e$. □

Proof:[Lemma 18] We prove 1 and 2 simultaneously by induction on the derivation of $\Gamma \vdash A : B$.

- Case (axiom): nothing to prove.
- Case (start) where $\Gamma(A\lambda_x) \vdash x : A$ comes from $\Gamma \vdash A : S$ and x is fresh, then the only possible γ-reduction is to Γ or $(A\lambda_x)$. In the first case use IH. In the second, use IH, (start), (conv), (weak) and Proposition 3.
- Case (weak) or (conv), use IH.
- Case (app) where $\Gamma \vdash (a\delta)F : e[x := a]$ comes from $\Gamma \vdash F : (A\Pi_x)e$ and $\Gamma \vdash a : A$. If γ-reduction is to a, F or Γ, apply IH (for the case where γ-reduction is in a, note that by Lemma 14.1, $x \notin FV(e)$ so $e[x := a] \equiv e$).
- Case (abs) where $\Gamma \vdash (A\lambda_x)b : (A\Pi_x)B$ comes from $\Gamma(A\lambda_x) \vdash b : B$ and $\Gamma \vdash (A\Pi_x)B : S$. If reduction is to Γ or b use IH. If reduction is to A use IH and (conv). Now take the crucial case where $b \equiv (c\delta)(d\lambda_y)e$ (recall Lemma 14.3) and $(c\delta)(d\lambda_y)(A\lambda_x)e \to_\gamma (A\lambda_x)(c\delta)(d\lambda_y)e$. We must show that $\Gamma \vdash (c\delta)(d\lambda_y)(A\lambda_x)e : (A\Pi_x)B$. Note that $y \notin FV(A)$ and $y \notin FV(B)$ (the latter holds since $FV(B) \subseteq dom(\Gamma)$ and by the Barendregt Convention, as y is bound in b we would not choose it in $dom(\Gamma)$). By Lemma 14.2 on $\Gamma(A\lambda_x) \vdash b \equiv (c\delta)(d\lambda_y)e : B$ we get: $\Gamma(A\lambda_x)(d\lambda_y) \vdash e : B$, $\Gamma(A\lambda_x) \vdash c : d$ and $\Gamma(A\lambda_x) \vdash (d\Pi_y)B : S'$ for a sort S'. Note by γ-reduction that $x \notin FV(c) \cup FV(d)$ and hence by the converse of the Thinning Lemma on $\Gamma(A\lambda_x) \vdash c : d$ we get $\Gamma \vdash c : d$. Note by γ-reduction that $x \notin FV(d)$ and hence by Lemma 14.4 on $\Gamma(A\lambda_x)(d\lambda_y) \vdash e : B$ we get $\Gamma(d\lambda_y)(A\lambda_x) \vdash e : B$. But also by Lemma 9 on $\Gamma \vdash (A\Pi_x)B : S$ we get $\Gamma(d\lambda_y) \vdash (A\Pi_x)B : S$. Hence, by (abs) we get $\Gamma(d\lambda_y) \vdash (A\lambda_x)e : (A\Pi_x)B$. Recall that we have $\Gamma \vdash (A\Pi_x)B : S$ and $\Gamma(A\lambda_x) \vdash (d\Pi_y)B : S'$. We want to show that $\Gamma \vdash (d\Pi_y)(A\Pi_x)B : S''$ for some sort S''. By Lemma 10 on $\Gamma(A\lambda_x) \vdash (d\Pi_y)B : S'$ we get that $\Gamma(A\lambda_x) \vdash d : S_1$, $\Gamma(A\lambda_x)(d\lambda_y) \vdash B : S'$ and (S_1, S') is a rule. As $y \notin FV(B)$ then by the converse of Thinning, $\Gamma(A\lambda_x) \vdash B : S'$. By Lemma 10 on $\Gamma \vdash (A\Pi_x)B : S$ we get $\Gamma(A\lambda_x) \vdash B : S$. Hence, $S \equiv S'$. Also from $\Gamma \vdash (A\Pi_x)B : S$ we get by Lemma 9 $\Gamma(d\lambda_y) \vdash (A\Pi_x)B : S$. But $\Gamma \vdash d : S_1$ and (S_1, S) is a rule, hence we get by (form) that $\Gamma \vdash (d\Pi_y)(A\Pi_x)B : S$. Now as $\Gamma(d\lambda_y) \vdash (A\lambda_x)e : (A\Pi_x)B$ and $\Gamma \vdash (d\Pi_y)(A\Pi_x)B : S$ we get by (abs) that $\Gamma \vdash (d\lambda_y)(A\lambda_x)e : (d\Pi_y)(A\Pi_x)B$. Finally, recall that $y \notin FV(A) \cup FV(B)$ and hence $((A\Pi_x)B)[y := c] \equiv (A\Pi_x)B$. Now, as $\Gamma \vdash c : d$ and $\Gamma \vdash (d\lambda_y)(A\lambda_x)e : (d\Pi_y)(A\Pi_x)B$ we use (app) to get the conclusion that $\Gamma \vdash (c\delta)(d\lambda_y)(A\lambda_x)e : (A\Pi_x)B$.
- Case (form) where $\Gamma \vdash (A\Pi_x)B : S_2$ comes from $\Gamma \vdash A : S_1$, $\Gamma(A\lambda_x) \vdash B : S_2$ and (S_1, S_2) rule. If γ-reduction is in either Γ or A or B then use IH. Now take the crucial case where $B \equiv (C\delta)(D\lambda_y)E$ (recall Lemma 14.3) and $(C\delta)(D\lambda_y)(A\Pi_x)E \to_\gamma (A\Pi_x)(C\delta)(D\lambda_y)E$, we must show that $\Gamma \vdash (C\delta)(D\lambda_y)(A\Pi_x)E : S_2$. By Lemma 14.2 on $\Gamma(A\lambda_x) \vdash (C\delta)(D\lambda_y)E : S_2$ we get $\Gamma(A\lambda_x)(D\lambda_y) \vdash E : S_2$, $\Gamma(A\lambda_x) \vdash C : D$ and $\Gamma(A\lambda_x) \vdash (D\Pi_y)S_2 : S'$ for some sort S'.
  As $x \notin FV(D)$, we get by Lemma 14.4 that $\Gamma(D\lambda_y)(A\lambda_x) \vdash E : S_2$. By Lemma 9 on $\Gamma \vdash A : S_1$, we have $\Gamma(D\lambda_y) \vdash A : S_1$. Now use the fact that (S_1, S_2) is a rule to get by (form) that $\Gamma(D\lambda_y) \vdash (A\Pi_x)E : S_2$. By converse of Thinning Lemma, as $x \notin FV(D)$ we get from $\Gamma(A\lambda_x) \vdash (D\Pi_y)S_2 : S'$ that $\Gamma \vdash (D\Pi_y)S_2 : S'$. $\Gamma \vdash (D\Pi_y)S_2 : S'$ and $\Gamma(D\lambda_y) \vdash (A\Pi_x)E : S_2$ give by (abs) $\Gamma \vdash (D\lambda_y)(A\Pi_x)E : (D\Pi_y)S_2$. As $x \notin FV(C) \cup FV(D)$, we get by the converse of Thinning Lemma on $\Gamma(A\lambda_x) \vdash C : D$ that $\Gamma \vdash C : D$. Now use (app) on $\Gamma \vdash C : D$ and $\Gamma \vdash (D\lambda_y)(A\Pi_x)E : (D\Pi_y)S_2$ to get $\Gamma \vdash (C\delta)(D\lambda_y)(A\Pi_x)E : S_2[y := C] \equiv S_2$. □

Proof: [Lemma 40] By simultaneous induction on the length of the derivation. We distinguish cases according to the last rule in 1.

- (axiom): nothing to prove.
- (conv), (abs) or (form): use IH. For (abs) use also (conv).
- (start): Assume $\Gamma d \vdash^{\mathsf{c}} \mathtt{subj}(d) : \mathtt{pred}(d)$ comes from $\Gamma \prec d$. If reduction is in Γ or in $\underline{d}$ where d is a definition then use IH. If d is a declaration and reduction is in d then use IH and (conv). Else, if $d \equiv (a\delta)(c\delta)(e\lambda_y)(b\lambda_x)$ and $d \to_\theta d' \equiv (c\delta)(e\lambda_y)(a\delta)(b\lambda_x)$ then by IH $\Gamma \prec d'$, so by (start) $\Gamma d' \vdash^{\mathsf{c}} \mathtt{subj}(d') : \mathtt{pred}(d')$. Now it is easy to show that $\Gamma(c\delta)(e\lambda_y) \prec (a\delta)(b\lambda_x)$ so by (start) $\Gamma(c\delta)(e\lambda_y)(a\delta)(b\lambda_x) \vdash x : b$.
- (weak): Assume $\Gamma d \vdash^{\mathsf{c}} D : E$ comes from $\Gamma\underline{d} \vdash^{\mathsf{c}} D : E$ and $\Gamma \prec d$. If reduction is in $\Gamma\underline{d}$, in D, or in a main item of d, use IH and (conv). Otherwise, $d \equiv (a\delta)(b\delta)(c\lambda_x)\overline{s}(e\lambda_y)$ for some $a, b, c, e, \overline{s}$ well balanced and $d' \equiv (b\delta)(c\lambda_x)(a\delta)\overline{s}(e\lambda_y)$ where $d \to_\theta d'$. We must show that $\Gamma(b\delta)(c\lambda_x)(a\delta)\overline{s}(e\lambda_y) \vdash^{\mathsf{c}} D : E$. Since $\Gamma \prec d$ we have that Γd is a pseudocontext, $\Gamma(b\delta)(c\lambda_x)\overline{s} \vdash^{\mathsf{c}} e : S$ for some sort S and $\Gamma(b\delta)(c\lambda_x)\overline{s} \vdash^{\mathsf{c}} a : e$ and $FV(a) \subseteq dom(\Gamma)$. Now it follows that also $\Gamma(b\delta)(c\lambda_x) \prec (a\delta)\overline{s}(e\lambda_y)$, so by (weak) $\Gamma(b\delta)(c\lambda_x)(a\delta)\overline{s}(e\lambda_y) \vdash^{\mathsf{c}} D : E$.
- (app): If reduction is in Γ, a or F, use IH (also (conv), Corollary 36 and Lemma 35 for a). Otherwise, $\Gamma \vdash^{\mathsf{c}} (a\delta)(b\delta)(c\lambda_x)F : B[y := a]$ is a conclusion from $\Gamma \vdash^{\mathsf{c}} (b\delta)(c\lambda_x)F : (A\Pi_y)B$ and $\Gamma \vdash a : A$. Then by Lemma 34, $\Gamma(b\delta)(c\lambda_x) \vdash^{\mathsf{c}} F : (A\Pi_y)B$ and by Lemma 33 on $\Gamma \vdash^{\mathsf{c}} a : A$ we have $\Gamma(b\delta)(c\lambda_x) \vdash^{\mathsf{c}} a : A$. So by (app) $\Gamma(b\delta)(c\lambda_x) \vdash^{\mathsf{c}} (a\delta)F : B[y := a]$ and by (def rule) $\Gamma \vdash^{\mathsf{c}} (b\delta)(c\lambda_x)(a\delta)F : |B[y := a]|_{(b\delta)(c\lambda_x)}$. Since $x \notin FV(B[y := a])$ we are done.
- (def rule) where $\Gamma \vdash^{\mathsf{c}} dC : |D|_d$ comes from $\Gamma d \vdash^{\mathsf{c}} C : D$ where d is a definition. Since d is well-balanced, and Γ contains only partnered δ-items, reduction must be in Γ, or d or C.

  † If reduction is in C use IH. † If reduction is in Γ where $\Gamma \to_\theta \Gamma'$ then:

  - Use IH to deduce by (def rule) that $\Gamma' \vdash^{\mathsf{c}} dC : |D|_d$.
  - Now suppose $\Gamma \prec d$ and $\Gamma \to_\theta \Gamma'$. Then $\Gamma' d$ is a pseudocontext, $\Gamma' d \vdash^{\mathsf{c}} \mathtt{pred}(d) : S$ and $\Gamma' d \vdash^{\mathsf{c}} \mathtt{def}(d) : \mathtt{pred}(d)$ follow from IH. Moreover, $FV(\mathtt{def}(d')) \subseteq dom(\Gamma)$ since $\to_\theta$ does not change binders in Γ. Hence, $\Gamma' \prec d$.

  † If reduction is in d, say $d \to_\theta d'$, then:

  - Either d' is still a definition or $d \equiv (a\delta)(b\delta)(c\lambda_x)\overline{s}(e\lambda_y)$ and $d' \equiv (b\delta)(c\lambda_x)(a\delta)\overline{s}(e\lambda_y)$. In the first case, $\Gamma \vdash^{\mathsf{c}} d'C : |D|_d$ follows from IH and (conv). In the second case, since by IH, $\Gamma d' \vdash^{\mathsf{c}} C : D$, we get by (def rule) $\Gamma(b\delta)(c\lambda_x) \vdash^{\mathsf{c}} (a\delta)\overline{s}(e\lambda_y)C : |D|_{(a\delta)\overline{s}(e\lambda_y)}$. By applying (def rule) again, we get $\Gamma \vdash^{\mathsf{c}} d'C : |D|_{d'}$, and by Lemma 39 we are done.
  - Lastly, suppose $\Gamma \prec d$. Then $\Gamma d'$ is a pseudocontext. If the reduction is in $\underline{d}$ or not on the main items of d, use IH and (conv) to get $\Gamma \prec d'$. Otherwise, $d \equiv (a\delta)(b\delta)(c\lambda_x)\overline{s}(e\lambda_y)$ and $d' \equiv (b\delta)(c\lambda_x)(a\delta)\overline{s}(e\lambda_y)$ for some a, b, c, e and well balanced $\overline{s}$. It is easy to show that $\Gamma d'$ is a pseudocontext and that $FV(a) \subseteq dom(\Gamma(b\delta)(c\lambda_x))$. Also, because $\Gamma \prec d$ we have that $\Gamma(b\delta)(c\lambda_x)\overline{s} \vdash^{\mathsf{c}} e : S$ and $\Gamma(b\delta)(c\lambda_x)\overline{s} \vdash^{\mathsf{c}} a : e$. Hence, $\Gamma(b\delta)(c\lambda_x) \prec (a\delta)\overline{s}(e\lambda_y)$. □

Proof:[Lemma 41] By simultaneous induction on the length of the derivation. We distinguish cases according to the last rule in the derivation for 1.

- (axiom): nothing to prove.
- (conv), (abs) or (form): use IH.

- (start): Assume $\Gamma d \vdash^{\mathsf{c}} \mathtt{subj}(d) : \mathtt{pred}(d)$ comes from $\Gamma \prec d$. If reduction is to Γ or to $\underline{d}$ where d is a definition then use IH. If d is a declaration and reduction is to d then use IH and (conv). Else, if $d \equiv (a\delta)\overline{s}(b\lambda_x)$, $\Gamma \equiv \Gamma_1(c\delta)(e\lambda_y)$ where $(a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x) \to_\theta (c\delta)(e\lambda_y)(a\delta)\overline{s}(b\lambda_x)$, and $\Gamma_1(c\delta)(e\lambda_y)(a\delta)\overline{s}(b\lambda_x) \vdash^{\mathsf{c}} x : a$ comes from $\Gamma_1(c\delta)(e\lambda_y) \prec (a\delta)\overline{s}(b\lambda_x)$ then we need to show that $\Gamma_1(a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x) \vdash^{\mathsf{c}} x : a$. It is easy to show that $\Gamma_1 \prec (a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x)$ and hence by (start) $\Gamma_1(a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x) \vdash^{\mathsf{c}} x : a$.

- (weak): Assume $\Gamma d \vdash^{\mathsf{c}} D : E$ comes from $\Gamma\underline{d} \vdash^{\mathsf{c}} D : E$ and $\Gamma \prec d$. If reduction is either to Γ, $\underline{d}$, D, or to a main item of d, use IH and (conv) if needed. Else, if $d \equiv (a\delta)\overline{s}(b\lambda_x)$, $\Gamma \equiv \Gamma_1(c\delta)(e\lambda_y)$ where $(a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x) \to_\theta (c\delta)(e\lambda_y)(a\delta)\overline{s}(b\lambda_x)$, and $\Gamma_1(c\delta)(e\lambda_y)(a\delta)\overline{s}(b\lambda_x) \vdash^{\mathsf{c}} D : E$ comes from $\Gamma_1(c\delta)(e\lambda_y) \prec (a\delta)\overline{s}(b\lambda_x)$ and $\Gamma_1(c\delta)(e\lambda_y)\overline{s} \vdash^{\mathsf{c}} D : E$, then use IH to show $\Gamma_1 \prec (a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x)$ and use $\Gamma_1(c\delta)(e\lambda_y)\overline{s} \vdash^{\mathsf{c}} D : E$ to conclude by (weak) that $\Gamma_1(a\delta)(c\delta)(e\lambda_y)\overline{s}(b\lambda_x) \vdash^{\mathsf{c}} D : E$.

- (app): $\Gamma \vdash^{\mathsf{c}} (a\delta)F : B[x := a]$ comes from $\Gamma \vdash^{\mathsf{c}} F : (A\Pi_x)B$ and $\Gamma \vdash^{\mathsf{c}} a : A$.

  - If θ-reduction is to Γ or F, use IH.
  - If θ-reduction is to a use IH, Correctness of Types Corollary 36 and (conv).
  - If $F \equiv (b\lambda_x)(c\delta)F'$ and $(c\delta)(a\delta)(b\lambda_x)F' \to_\theta (a\delta)F$ (where $x \notin FV(c)$), we must show that $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)F' : B[x := a]$. By Lemma 34 on $\Gamma \vdash^{\mathsf{c}} (a\delta)(b\lambda_x)(c\delta)F' : B[x := a]$ we have $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} (c\delta)F' : B[x := a]$. Again by Lemma 34 $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} F' : (C\Pi_y)D$ and $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} c : C$ for some C, y, D such that $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} D[y := c] =_{\mathtt{def}} B[x := a]$. By (def rule), $\Gamma \vdash^{\mathsf{c}} (a\delta)(b\lambda_x)F' : ((C\Pi_y)D)[x := a]$ and by Lemma 35 on $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} c : C$ (note $x \notin FV(c)$), we get $\Gamma \vdash^{\mathsf{c}} c : C[x := a]$. Now by (app) on $\Gamma \vdash^{\mathsf{c}} (a\delta)(b\lambda_x)F' : ((C\Pi_y)D)[x := a]$ and $\Gamma \vdash^{\mathsf{c}} c : C[x := a]$ we get $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)F' : D[x := a][y := c]$.
    Since $x \notin FV(c)$, $D[x := a][y := c] \equiv D[y := c][x := a] =_\beta B[x := a][x := a] \equiv B[x := a]$. Therefore by (conv), $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)F' : B[x := a]$.

- (def rule): where $\Gamma \vdash^{\mathsf{c}} dC : |D|_d$ comes from $\Gamma d \vdash^{\mathsf{c}} C : D$ where d is a definition. Since d is well-balanced, and Γ contains only partnered δ-items, reduction must be to Γ, to d or to C or we must have $C \equiv (c\delta)e$ and $d \equiv (a\delta)(b\lambda_x)$ where $(c\delta)(a\delta)(b\lambda_x)e \to_\theta (a\delta)(b\lambda_x)(c\delta)e$. The first three cases are an easy application of IH. In the last case, $\Gamma \vdash^{\mathsf{c}} (a\delta)(b\lambda_x)(c\delta)e : |D|_{(a\delta)(b\lambda_x)}$ comes from $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} (c\delta)e : D$. We need to show that $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)e : |D|_{(a\delta)(b\lambda_x)}$. By Lemma 34 on $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} (c\delta)e : D$ we have, $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} e : (G\Pi_y)E$, $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} c : G$ and $\Gamma(a\delta)(b\lambda_x) \vdash^{\mathsf{c}} E[y := c] =_{\mathtt{def}} D$. Then by the (def rule), $\Gamma \vdash^{\mathsf{c}} (a\delta)(b\lambda_x)e : ((G\Pi_y)E)[x := a]$ and by Lemma 35.4, $\Gamma \vdash^{\mathsf{c}} c[x := a] : G[x := a]$. But $c[x := a] \equiv c$ and hence by (app) we get $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)e : E[x := a][y := c]$. But as $y \notin FV(a)$, we have $E[x := a][y := c] \equiv E[y := c][x := a] =_\beta D[x := a]$ and now use (conv) on $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)e : E[x := a][y := c]$ and Corollary 36 to obtain $\Gamma \vdash^{\mathsf{c}} (c\delta)(a\delta)(b\lambda_x)e : D[x := a]$.

  Now suppose $\Gamma \prec d$ and $\Gamma' \to_\theta \Gamma$ then by IH, also $\Gamma' \prec d$.
  Lastly, suppose $\Gamma \prec d$ and $d' \to_\theta d$. If d is a definition, then reduction must be in a main item of d' or on the main items of $\underline{d'}$. Hence by IH, we get $\Gamma \prec d'$. □

Proof:[Lemma 42] By simultaneous induction on the length of the derivation. We distinguish cases according to the last rule in the derivation for 1.

- (axiom): nothing to prove. (conv), (abs) or (form): use IH. For (abs) also use (conv).

- (start): Assume $\Gamma d \vdash^{\mathsf{c}} \mathtt{subj}(d) : \mathtt{pred}(d)$ comes from $\Gamma \prec d$. If reduction is in Γ or in $\underline{d}$ where d is a definition then use IH. If d is a declaration and reduction is in d then use IH and (conv). Else, if $d \equiv (a\delta)\overline{s}(c\delta)(e\lambda_y)(b\lambda_x)$ and $d \to_\gamma d' \equiv (a\delta)\overline{s}(b\lambda_x)(c\delta)(e\lambda_y)$ then by Lemma 32, $\Gamma(a\delta)\overline{s}(b\lambda_x)(c\delta)(e\lambda_y) \vdash^{\mathsf{c}} \mathtt{subj}(d) : \mathtt{pred}(d)$ because $x \notin FV(c) \cup FV(e)$.

- (weak): Assume $\Gamma d \vdash^c D : E$ comes from $\Gamma \underline{d} \vdash^c D : E$ and $\Gamma \prec d$. If reduction is in $\Gamma \underline{d}$, in D, or in a main item of d, use IH and (conv). Otherwise, $d \equiv (a\delta)\overline{s}(b\delta)(c\lambda_x)(e\lambda_y)$ for some $a, b, c, e, \overline{s}$ and $d' \equiv (a\delta)\overline{s}(e\lambda_y)(b\delta)(c\lambda_x)$ where $d \to_\gamma d'$. Now, use Lemma 32.

- (app): Assume $\Gamma \vdash^c (a\delta)F : B[x := a]$ comes from $\Gamma \vdash^c F : (A\Pi_x)B$ and $\Gamma \vdash^c a : A$. If reduction is in Γ, a or F, use IH (also (conv), Corollary 36 and Lemma 35 for a). Else if $\Gamma \vdash^c (a\delta)(b\lambda_x)(c\lambda_y)F' : B[x := a]$ comes from $\Gamma \vdash^c (b\lambda_x)(c\lambda_y)F' : (A\Pi_x)B$ and $\Gamma \vdash^c a : A$ where $(a\delta)(b\lambda_x)(c\lambda_y)F' \to_\gamma (c\lambda_y)(a\delta)(b\lambda_x)F'$ then we need to show that $\Gamma \vdash^c (c\lambda_y)(a\delta)(b\lambda_x)F' : B[x := a]$. By Lemma 34 on $\Gamma \vdash^c (a\delta)(b\lambda_x)(c\lambda_y)F' : B[x := a]$ we have that $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\lambda_y)F' : B[x := a]$ and again by Lemma 34, for some term D and sort S, $\Gamma(a\delta)(b\lambda_x)(c\lambda_y) \vdash^c F' : D$, $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)D : S$, $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)D =_{\mathtt{def}} B[x := a]$ and if $(c\Pi_y)D \not\equiv B[x := a]$ then $\Gamma(a\delta)(b\lambda_x) \vdash^c B[x := a] : S'$ for some S'. By Lemma 32 on $\Gamma(a\delta)(b\lambda_x)(c\lambda_y) \vdash^c F' : D$, $\Gamma(c\lambda_y)(a\delta)(b\lambda_x) \vdash^c F' : D$ and hence by (def rule) we have $\Gamma(c\lambda_y) \vdash^c (a\delta)(b\lambda_x)F' : D[x := a]$. But $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)D : S$ gives by Lemma 35.4, $\Gamma \vdash^c ((c\Pi_y)D)[x := a] : S$. But $x \notin FV(c)$ and hence $\Gamma \vdash^c (c\Pi_y)D[x := a] : S$. Now use (abs) on $\Gamma \vdash^c (c\Pi_y)D[x := a] : S$ and $\Gamma(c\lambda_y) \vdash^c (a\delta)(b\lambda_x)F' : D[x := a]$ to get $\Gamma \vdash^c (c\lambda_y)(a\delta)(b\lambda_x)F' : (c\Pi_y)D[x := a]$. But as $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)D =_{\mathtt{def}} B[x := a]$ and $x \notin FV(c)$, Lemma 35.1 gives $\Gamma \vdash^c (c\Pi_y)D[x := a] =_{\mathtt{def}} B[x := a][x := a] \equiv B[x := a]$. We treat two cases:

  - If $(c\Pi_y)D \equiv B[x := a]$ then $(c\Pi_y)D[x := a] \equiv B[x := a]$ and we are done.
  - If $(c\Pi_y)D \not\equiv B[x := a]$ then $\Gamma \vdash^c B[x := a] : S'$. By (conv) $\Gamma \vdash^c (c\lambda_y)(a\delta)(b\lambda_x)F' : B[x := a]$.

- (def rule) where $\Gamma \vdash^c dC : |D|_d$ comes from $\Gamma d \vdash^c C : D$ where d is a definition.

  - If reduction is in C or in Γ use IH.
  - If reduction is in d, say $d \to_\theta d'$, then if d' is still a definition use IH and (conv).
  - If reduction is in d, say $d \to_\theta d'$ and d' is not a definition then $d \equiv (a\delta)\overline{s}(b\delta)(c\lambda_x)(e\lambda_y)$ for some $a, b, c, e, \overline{s}$ well balanced and $d' \equiv (a\delta)\overline{s}(e\lambda_y)(b\delta)(c\lambda_x)$ where $d \to_\gamma d'$. Since by IH, $\Gamma d' \vdash^c C : D$, we get by (def rule) $\Gamma(a\delta)\overline{s}(e\lambda_y) \vdash^c (b\delta)(c\lambda_x)C : |D|_{(b\delta)(c\lambda_x)}$. By (def rule) again, $\Gamma \vdash^c (a\delta)\overline{s}(e\lambda_y)(b\delta)(c\lambda_x)C : ||D|_{(b\delta)(c\lambda_x)}|_{(a\delta)\overline{s}(e\lambda_y)} \equiv |D|_{d'}$. Now use Lemma 39.
  - If reduction is in dC where $\Gamma \vdash^c (a\delta)(b\lambda_x)(c\lambda_y)e : D[x := a]$ comes from $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\lambda_y)e : D$ and $(a\delta)(b\lambda_x)(c\lambda_y)e \to_\gamma (c\lambda_y)(a\delta)(b\lambda_x)e$. We must show $\Gamma \vdash^c (c\lambda_y)(a\delta)(b\lambda_x)e : D[x := a]$. By Lemma 34 on $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\lambda_y)e : D$ we get $\Gamma(a\delta)(b\lambda_x)(c\lambda_y) \vdash^c e : E$, $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)E : S$, $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)E =_{\mathtt{def}} D$, and if $(c\Pi_y)E \not\equiv D$ then $\Gamma(a\delta)(b\lambda_x) \vdash^c D : S'$. By Lemma 32 on $\Gamma(a\delta)(b\lambda_x)(c\lambda_y) \vdash^c e : E$ we get $\Gamma(c\lambda_y)(a\delta)(b\lambda_x) \vdash^c e : E$. By (def rule) we get $\Gamma(c\lambda_y) \vdash^c (a\delta)(b\lambda_x)e : E[x := a]$. As $x \notin FV(c)$, we get by Substitution Lemma 35 on $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)E : S$ that $\Gamma \vdash^c (c\Pi_y)E[x := a] : S$. Hence, by (abs) $\Gamma \vdash^c (c\lambda_y)(a\delta)(b\lambda_x)e : (c\Pi_y)E[x := a]$.
    - If $(c\Pi_y)E \equiv D$ then $(c\Pi_y)E[x := a] \equiv D[x := a]$ and we are done.
    - If $(c\Pi_y)E \not\equiv D$ then $\Gamma(a\delta)(b\lambda_x) \vdash^c D : S'$ and hence by Lemma 35 $\Gamma \vdash^c D[x := a] : S'$. As $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)E =_{\mathtt{def}} D$, then by Lemma 35 $\Gamma(a\delta)(b\lambda_x) \vdash^c (c\Pi_y)E[x := a] =_{\mathtt{def}} D[x := a]$. Now use (conv) to get that $\Gamma \vdash^c (c\lambda_y)(a\delta)(b\lambda_x)e : D[x := a]$.

  Lastly, suppose $\Gamma \prec d$. Then $\Gamma d'$ is a pseudocontext. If the reduction is in $\underline{d}$ or not on the main items of d, use IH and (conv) to get $\Gamma \prec d'$. Otherwise, $d \equiv (a\delta)\overline{s}(b\delta)(c\lambda_x)(e\lambda_y)$ for some $a, b, c, e, \overline{s}$ well balanced and $d' \equiv (a\delta)\overline{s}(e\lambda_y)(b\delta)(c\lambda_x)$ where $d \to_\gamma d'$. Take $d'' \equiv (a\delta)\overline{s}(e\lambda_y)$. It is easy to show that $\Gamma d''$ is a pseudocontext and that $FV(a) \subseteq dom(\Gamma)$. Also, because $\Gamma \prec d$ we have that $\Gamma\overline{s}(b\delta)(c\lambda_x) \vdash^c e : S$ and $\Gamma\overline{s}(b\delta)(c\lambda_x) \vdash^c a : e$ and so by converse of thinning $\Gamma\overline{s} \vdash^c e : S$ and $\Gamma\overline{s} \vdash^c a : e$. Hence, $\Gamma \prec (a\delta)\overline{s}(e\lambda_y) \equiv d''$. □

Proof:[Lemma 43] By simultaneous induction on the length of the derivation. We distinguish cases according to the last rule in the derivation for 1.

- (axiom): nothing to prove. (conv): use IH.
- (abs): Assume $\Gamma \vdash^c (A\lambda_x)b : (A\Pi_x)B$ comes from $\Gamma(A\lambda_x) \vdash^c b : B$ and $\Gamma \vdash^c (A\Pi_x)B : S$. If reduction is to Γ or b use IH. If reduction is to A use IH and (conv). Now take the crucial case where $b \equiv (c\delta)(d\lambda_y)e$ and $(c\delta)(d\lambda_y)(A\lambda_x)e \to_\gamma (A\lambda_x)(c\delta)(d\lambda_y)e$. We must show that $\Gamma \vdash^c (c\delta)(d\lambda_y)(A\lambda_x)e : (A\Pi_x)B$. By Generation Lemma 34 on $\Gamma(A\lambda_x) \vdash^c (c\delta)(d\lambda_y)e : B$ we get $\Gamma(A\lambda_x)(c\delta)(d\lambda_y) \vdash^c e : B$. By Lemma 32 $\Gamma(c\delta)(d\lambda_y)(A\lambda_x) \vdash^c e : B$. By Lemma 33 on $\Gamma \vdash^c (A\Pi_x)B : S$ we get $\Gamma(c\delta)(d\lambda_y) \vdash^c (A\Pi_x)B : S$. By (abs) we get $\Gamma(c\delta)(d\lambda_y) \vdash^c (A\lambda_x)e : (A\Pi_x)B$. by (def rule) $\Gamma \vdash^c (c\delta)(d\lambda_y)(A\lambda_x)e : ((A\Pi_x)B)[y := c]$. But by γ-reduction, $y \notin FV(A)$. Also, $y \notin FV(B)$ because by Lemma 29 on $\Gamma(A\lambda_x) \vdash^c (c\delta)(d\lambda_y)e : B$, $FV(B) \subseteq dom(\Gamma(A\lambda_x))$ and as $\Gamma(A\lambda_x)(c\delta)(d\lambda_y)$ is a pseudocontext, $y \notin dom(\Gamma(A\lambda_x))$. Hence, $((A\Pi_x)B)[y := c] \equiv (A\Pi_x)B$ and we have $\Gamma \vdash^c (c\delta)(d\lambda_y)(A\lambda_x)e : (A\Pi_x)B$.
- (form): If $\Gamma \vdash^c (A\Pi_x)B : S_2$ comes from $\Gamma \vdash^c A : S_1$, $\Gamma(A\lambda_x) \vdash^c B : S_2$ and (S_1, S_2) rule. If γ-reduction is in either Γ or A or B then use IH. Now take the crucial case where $\Gamma \vdash^c (A\Pi_x)(a\delta)(b\lambda_y)C : S_2$ comes from $\Gamma \vdash^c A : S_1$, $\Gamma(A\lambda_x) \vdash^c (a\delta)(b\lambda_y)C : S_2$, (S_1, S_2) and $(a\delta)(b\lambda_y)(A\Pi_x)C \to_\gamma (A\Pi_x)(a\delta)(b\lambda_y)C$. We need to show that $\Gamma \vdash^c (a\delta)(b\lambda_y)(A\Pi_x)C : S_2$. By Lemma 34 on $\Gamma(A\lambda_x) \vdash^c (a\delta)(b\lambda_y)C : S_2$, we get $\Gamma(A\lambda_x)(a\delta)(b\lambda_y) \vdash^c C : S_2$. By Lemma 32 we get $\Gamma(a\delta)(b\lambda_y)(A\lambda_x) \vdash^c C : S_2$ because $x \notin FV(a) \cup FV(b)$. By Lemma 33 on $\Gamma \vdash^c A : S_1$, we get $\Gamma(a\delta)(b\lambda_y) \vdash^c A : S_1$. Now use (form) to get $\Gamma(a\delta)(b\lambda_y) \vdash^c (A\Pi_x)C : S_2$. Finally, by (def rule) $\Gamma \vdash^c (a\delta)(b\lambda_y)(A\Pi_x)C : S_2$.
- (start): Assume $\Gamma d \vdash^c \mathtt{subj}(d) : \mathtt{pred}(d)$ comes from $\Gamma \prec d$. If reduction is to Γ or to $\underline{d}$ where d is a definition then use IH. If d is a declaration and reduction is to d then use IH and (conv). Else, if $\Gamma_1(a\delta)(b\lambda_x)(c\delta)(e\lambda_y) \vdash^c y : c$ comes from $\Gamma_1(a\delta)(b\lambda_x) \prec (c\delta)(e\lambda_y)$ where $(a\delta)(c\delta)(e\lambda_y)(b\lambda_x) \to_\gamma (a\delta)(b\lambda_x)(c\delta)(e\lambda_y)$ then by Lemma 32 $\Gamma_1(a\delta)(c\delta)(e\lambda_y)(b\lambda_x) \vdash^c y : c$.
- (weak): Assume $\Gamma d \vdash^c D : E$ comes from $\Gamma\underline{d} \vdash^c D : E$ and $\Gamma \prec d$. If reduction is either to Γ, $\underline{d}$, D, or to a main item of d, use IH and (conv) if needed. Else, if $\Gamma_1(a\delta)(b\lambda_x)(c\delta)(e\lambda_y) \vdash^c D : E$ comes from $\Gamma_1(a\delta)(b\lambda_x) \prec (c\delta)(e\lambda_y)$ and $\Gamma_1(a\delta)(b\lambda_x) \vdash^c D : E$ where $(a\delta)(c\delta)(e\lambda_y)(b\lambda_x) \to_\gamma (a\delta)(b\lambda_x)(c\delta)(e\lambda_y)$ then by Lemma 32, $\Gamma_1(a\delta)(c\delta)(e\lambda_y)(b\lambda_x) \vdash^c D : E$.
- (app): If reduction is to Γ, a or F, use IH (also (conv) Corollary 36 and Lemma 35 for a).
- (def rule): Assume $\Gamma \vdash^c dC : |D|_d$ comes from $\Gamma d \vdash^c C : D$ where d is a definition.
  - If reduction is to C or to Γ use IH.
  - If reduction is to d, say $d \to_\theta d'$, then d' must be a definition and we use IH.
  - Assume reduction is to dC where $\Gamma \vdash^c (a\delta)\overline{s}(e\lambda_y)(b\delta)(d\lambda_x)f : |D|_{(a\delta)\overline{s}(e\lambda_y)}$ comes from $\Gamma(a\delta)\overline{s}(e\lambda_y) \vdash^c (b\delta)(d\lambda_x)f : D$ with $(a\delta)\overline{s}(b\delta)(d\lambda_x)(e\lambda_y)f \to_\gamma (a\delta)\overline{s}(e\lambda_y)(b\delta)(d\lambda_x)f$. We show $\Gamma \vdash^c (a\delta)\overline{s}(b\delta)(d\lambda_x)(e\lambda_y)f : |D|_{(a\delta)\overline{s}(e\lambda_y)}$. By Lemma 34, $\Gamma(a\delta)\overline{s}(e\lambda_y)(b\delta)(d\lambda_x) \vdash^c f : D$. By Lemma 32, $\Gamma(a\delta)\overline{s}(b\delta)(d\lambda_x)(e\lambda_y) \vdash^c f : D$ because $y \notin FV(b) \cup FV(d)$. The segment $(a\delta)\overline{s}(b\delta)(d\lambda_x)(e\lambda_y)$ is well balanced. Case $(e\lambda_y)$ is partnered by $(a\delta)$, use the (def rule) and Lemma 39. Else, assume $(a\delta)\overline{s}(b\delta)(d\lambda_x)(e\lambda_y) \equiv (a\delta)\overline{s_1}(h\delta)\overline{s_2}(b\delta)(d\lambda_x)(e\lambda_y)$ where $(h\delta)$ is the partner of $(e\lambda_y)$. Apply the (def rule) to get $\Gamma(a\delta)\overline{s_1} \vdash^c (h\delta)\overline{s_2}(b\delta)(d\lambda_x)(e\lambda_y)f : |D|_{(h\delta)\overline{s_2}(b\delta)(d\lambda_x)(e\lambda_y)}$. As $(a\delta)\overline{s_1}$ is well balanced, continue applying the (def rule) and using Lemma 39 until you get $\Gamma \vdash^c (a\delta)\overline{s}(b\delta)(d\lambda_x)(e\lambda_y)f : |D|_{(a\delta)\overline{s}(e\lambda_y)}$. □