Term Reshuffling in the Barendregt Cube*

Roel Bloo[†] Fairouz Kamareddine[‡] Rob Nederpelt[§]

April 16, 1997

1 Brief Synopsis

Automath was invented by de Bruijn ([Nederpelt 73]) with the basic goal of automating Mathematics. The language and theory of Automath were designed to deal with very basic questions of which only now the computing community is becoming aware. Of these topics, we mention explicit substitution and definitions ([1] and [SP 93]). In the beginning, it was claimed that the notation of Automath is too difficult. Now, it became clear that the influence of Automath on various theorem provers is invaluable.

In Automath, $(\lambda_{x:A}.B)$ and (AB) are written as $(A\lambda_x B)$ and $(B\delta A)$ respectively. We propose to change slightly the Automath notation so that the above two terms would be written in our notation (*item notation*), as $(A\lambda_x)B$ and $(B\delta)A$ respectively. This slight change has been studied for explicit substitution in [KN 93], generalised reduction and definitions in [BKN 9x] and was shown to bear attractive advantages over both the classical and the Automath notations. This paper will concentrate on a new feature related to *reshuffling terms* so that more redexes become visible. The idea is explained as follows:

Assume a redex is a '[' next to a ']'. What will happen in a term of the form '[[ ] [ ]]'? We know that the two internal '[ ]' are redexes, but classical notation does not allow us to say that the outside '[' and ']' form a redex. In [BKN 9x], we generalised the notion of a redex from a pair of adjacent matching parentheses to simply a pair of matching parentheses. Hence, with generalised reduction all the three redexes are visible in '[[ ] [ ]]'. In this paper, we propose to *reshuffle* '[[ ] [ ]]' to '[ ] [ ] [ ]' where the first '[' has been moved next to the last ']'. The item notation enables us to see the matching parentheses and to reshuffle terms so that all matching paretheses become adjacent.

We show that term reshuffling is correct in that it preserves the semantical meaning and the type of a term. Moreover, when definitions are added, the Cube with term reshuffling, would satisfy all its original properties including Church Rosser, Subject Reduction and Strong Normalisation.

*We are grateful for the Netherlands Computer Science Research Foundation (SION), the Netherlands Organisation for Scientific Research (NWO), the universities of Glasgow and Eindhoven and to the Basic Action for Research ESPRIT project "Types for Proofs and Programs", for their financial support.

†Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, the Netherlands, fax: +31 40 43 66 85, *email*: bloo@win.tue.nl

‡Department of Computing Science, 17 Lilybank Gardens, University of Glasgow, Glasgow G12 8QQ, Scotland, fax: +44 41 330 4913, *email*: fairouz@dcs.glasgow.ac.uk

§same address as Bloo. `email`: wsinrpn@win.tue.nl

Contents

Based on these observations related to reduction and definitions, we divide the paper into the following sections:

- In Section 2, we start by introducing the item notation and the formal machinery of the Cube as in [Barendregt 92] using this notation.
- In Section 3, we introduce the ordinary typing rules of the cube and the properties that will be shown for our extended typing with definitions and shuffle reduction.
- In Section 4, we introduce term reshuffling and study their characteristics.
- In Section 5, we introduce shuffle reduction $\rightsquigarrow\!\!\!\!\rightarrow_\beta$, and show that it is a generalisation of $\twoheadrightarrow_\beta$ such that $=_\beta$ and $\approx_\beta$ are the same and hence $\rightsquigarrow\!\!\!\!\rightarrow_\beta$ is Church Rosser. We show moreover that $A \rightsquigarrow_\beta B$ then $\exists B' \in [B][TS(A) \rightarrow_\beta B']$.
- In Section 6, we study the Cube as in [Barendregt 92] with term reshuffling using shuffle reduction and adding definitions. We show that this extension of the Cube preserves its original properties. In particular, we show that SR, SN and CR hold. We show moreover that term reshuffling preserves typing in the sense that $\Gamma \vdash^{\mathtt{sh}} A : B$ then $\Gamma \vdash^{\mathtt{sh}} TS(A) : B$.

2 The formal machinery and item notation

Assume a translation function $\mathcal{I}$ from terms in classical notation to terms in item notation such that:

$$\begin{array}{lcll} \mathcal{I}(A) & = & A & \text{if } A \text{ is a variable or a constant} \\ \mathcal{I}(\mathcal{O}_{x:A}.B) & = & (\mathcal{I}(A)\mathcal{O}_x)\mathcal{I}(B) & \mathcal{O} = \lambda \text{ or } \Pi \\ \mathcal{I}(AB) & = & (\mathcal{I}(B)\delta)\mathcal{I}(A) & \end{array}$$

With this notation, a redex is a term that starts with a δ-item next to a λ-item. An extended redex is a term that starts with a δ-item followed by a sequence of matching $\delta\lambda$-items followed by a λ-item. Term reshuffling amounts to moving δ-items in the term through sequences of definitions in order to occupy a place next to their matching λ-item.

Example 2.1 $\mathcal{I}((\lambda_{x:A\to(B\to C)}.\lambda_{y:A}.xy)z) \equiv (z\delta)(A \to (B \to C)\lambda_x)(A\lambda_y)(y\delta)x$. The items are $(z\delta)$, $(A \to (B \to C)\lambda_x)$, $(A\lambda_y)$ and $(y\delta)$and the whole term is a redex. Note that the translation into item notation of a redex $(\lambda_{x:B}.A)C$ becomes $(\mathcal{I}(C)\delta)(\mathcal{I}(B)\lambda_x)\mathcal{I}(A)$ and that the scope of a λ is precisely the term to the right of it.

Let us explain here why this notation enables us to see more redexes and to reshuffle terms enabling one to contract any visible redex independently of other redexes. Let us start first by rewriting the axiom β in item notation:

Definition 2.2 *(Classical redexes and β-reduction in item notation)*
*In the item notation of the λ-calculus, a classical redex is of the form $(C\delta)(B\lambda_x)A$. We call the pair $(C\delta)(B\lambda_x)$, a $\delta\lambda$-**pair**, or a $\delta\lambda$-**segment**. The classical β-reduction axiom is: $(C\delta)(B\lambda_x)A \rightarrow_\beta A[x := C]$. Many step β-reduction $\twoheadrightarrow_\beta$, is the reflexive transitive closure of $\rightarrow_\beta$, and $=_\beta$ is the least equivalence relation closed under $\rightarrow_\beta$.*

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. Furthermore, we take terms to be equivalent up to variable renaming. For example, we take $\lambda_{x:A}.x \equiv \lambda_{y:A}.y$ where $\equiv$ is used to denote syntactical equality of terms. We assume moreover, the Barendregt variable convention which is formally stated as follows:

Convention 2.3 *(BC: Barendregt's Convention)*
Names of bound variables will always be chosen such that they differ from the free ones in a term. Moreover, different λ's have different variables as subscript. Hence, we will not have $(\lambda_{x:A}.x)x$, but $(\lambda_{y:A}.y)x$ instead.

Now, let us look at $A \equiv ((\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)v)w)u$ and $B \equiv (\lambda_{x:P}.(\lambda_{y:Q}.(\lambda_{z:R}.xyz)u)v)w$. Note that $A =_\beta wvu$ and $B =_\beta wvu$. In other words, A and B are semantically equivalent. There is an even closer relation between A and B. Namely, a relation between the redexes.

Example 2.4 In $A \equiv ((\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)v)w)u$, we have the following redexes which are all needed to get the normal form of A:

1. $(\lambda_{y:Q}.\lambda_{z:R}.xyz)v$
2. $(\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)v)w$
3. $(\lambda_{z:R}.wvz)u$ which appears as $((\lambda_{z:R}.xyz)[y := v][x := w])u$

The first and second redexes are classical redexes, immediately visible and subject to contraction. The third redex is neither a classical redex nor is immediately visible, nor is subject to contraction without having unfolded in $\lambda_{z:R}.xyz$ the two definitions that y is v and x is w. It will only be a proper visible classical redex and subject to contraction, after we have contracted the first two redexes (we will not discuss the order here). For example, assume we contract the second redex in the first step, and the first redex in the second step, then

$$
\begin{array}{ll}
((\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)v)w)u & \to_\beta \\
((\lambda_{y:Q}.\lambda_{z:R}.xyz)v)u & \to_\beta \\
(\lambda_{z:R}.wvz)u & \to_\beta wvu
\end{array}
$$

There is however a need to make as many needed redexes visible as possible (see [BKKS 87]). In fact, even though the notion of a needed redex is undecidable, much work has been carried out in order to study some classes of needed redexes (as in [BKKS 87] and [Gardena 94]). In $B \equiv (\lambda_{x:P}.(\lambda_{y:Q}.(\lambda_{z:R}.xyz)u)v)w$, the redexes are:

1. $(\lambda_{y:Q}.(\lambda_{z:R}.xyz)u)v$
2. $(\lambda_{x:P}.(\lambda_{y:Q}.(\lambda_{z:R}.xyz)u)v)w$
3. $(\lambda_{z:R}.xyz)u$

All the three redexes of B are classical, immediately visible and subject to contraction.

Hence, for A, there is a semantically equivalent term B where more redexes of A become visible, and even subject to contraction before any other redexes.

Looking again at A and B, we see that not only A has a semantically equivalent term B where more redexes become visible and subject to contraction, but also we can find that there is a relation between the redexes of A and B.

Basic to our study in this paper will be a new notation *the item notation* and a term rewriting called *term reshuffling*. The term reshuffling of a term will rewrite it so that as many redexes as possible become visible.

With the presence of more visible redexes, and with the fact that in the reshuffled version of a term all visible redexes are classical, we generalise reduction and instead of reducing a term, we reduce its reshuffled version.

Example 2.5 Let $A \equiv ((\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.za)b)c)d$ and $B \equiv (\lambda_{x:P}.(\lambda_{y:Q}.(\lambda_{z:R}.za)d)b)c$. We denote the reshuffled version of A by $TS(A)$. Now, $TS(A) \equiv TS(B) \equiv B$ and it is obvious that $A =_\beta B$. Hence, A and B are semantically equivalent. Moreover, it is evident that all extended redexes of A, namely: $(\lambda_{x:P}.-)c$, $(\lambda_{y:Q}.-)b$ and $(\lambda_{z:R}.-)d$, are classical redexes of B. Furthermore, these redexes can be contracted independently of each other.

Of course here, there will be complaints that this reshuffling is not so easy or obvious. We agree and this is what we are trying to say. The classical notation which we have used so far cannot extend redexes or enable reshuffling in an easy way. Our notation however, the *item notation* will solve these problems. We call this reduction which works with the reshuffled version of the term, *shuffle reduction*.

Extending redexes and enabling newly visible redexes to be contracted before other ones, and studying the classes of terms that are semantically equivalent, may act as a powerful tool in the study of some programming languages. For example, in lazy evaluation ([Launchbury 93]), some redexes get frozen while other ones are being contracted. Now, if we had the ability of choosing which redex to contract out of all visible redexes, rather than waiting for some redex to be evaluated before we can proceed with the rest, then we can say that we have achieved a flexible system where we have control over what to contract rather than letting reductions force themselves in some order. This may lead to some advantages concerning optimal reductions as in [Lévy 80].

Moreover, we may avoid explosion if we had the choice of making more redexes visible and the ability of contracting any visible redex before any other ones:

Example 2.6 Let $M \equiv (\lambda_{x:u}.\lambda_{y:u}.y(Cxx\ldots x))B(\lambda_{z:u}.u)$ where B is a BIG term. Then $M \to_\beta (\lambda_{y:u}.y(CBB\ldots B))(\lambda_{z:u}.u) \to_\beta (\lambda_{z:u}.u)(CBB\ldots B) \to_\beta u$ and u is in normal form. Now the first and second reduct both contain the segment $CBB\cdots B$, so they are very, very long terms. Shuffle reduction however allows us to reduce M in the following way: $TS(M) \equiv (\lambda_{x:u}.(\lambda_{y:u}.y(Cxx\ldots x))\lambda_{z:u}.u)B \to_\beta (\lambda_{x:u}.(\lambda_{z:u}.u)(Cxx\ldots x))B \to_\beta (\lambda_{x:u}.u)B \to_\beta u$, and in this reduction all the terms are of equal or smaller size than M! So shuffle reduction might allow us to define clever strategies that reduce terms via paths of relatively small terms.

Let us assure the reader again here that one must not be anxious that it is not obvious how to reshuffle the term or to work with classes of terms. The notation that we will provide will make term reshuffling a straightforward operation. Furthermore, reshuffling terms makes us realise that there is a certain part of the term which passes through another part which can be viewed as a *definition*. In fact, look at how we rewrote $((\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)w)v)u$ to $((\lambda_{x:P}.(\lambda_{y:Q}.(\lambda_{z:R}.xyz)u)v)w$. u went through two definitions (or redexes) $(\lambda_{x:P}.-)w$ and $(\lambda_{y:Q}.-)v$ to occupy a place next to its matching λ_z.

Example 2.7 A of Example 2.4 is written $(u\delta)(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(R\lambda_z)(z\delta)(y\delta)x$ in item notation (for convenience sake, we assume u, v, w, P, Q, R are variables). Here, the first two redexes, the classical redexes, correspond to $\delta\lambda$-pairs as follows:

1. $(\lambda_{y:Q}.\lambda_{z:R}.xyz)v$ corresponds to $(v\delta)(Q\lambda_y)$. $(R\lambda_z)(z\delta)(y\delta)x$ is omitted as it is easily retrievable in item notation. It is the maximal subterm of A to the right of $(Q\lambda_y)$.

2. $(\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)v)w$ corresponds to $(w\delta)(P\lambda_x)$.
   Again $(v\delta)(Q\lambda_y)(R\lambda_z)(z\delta)(y\delta)x$ is ignored for the same reason as above.

If one looks more closely at A written in item notation however, one sees that the third redex can be obtained by just matching δ- and λ-items. The third redex $(\lambda_{z:R}.xyz)u$ is visible as it corresponds to the matching $(u\delta)(R\lambda_z)$ where $(u\delta)$ and $(R\lambda_z)$ are separated by the segment $(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)$. Hence, by extending the notion of a redex from being a δ-item adjacent to a λ-item, to being a matching pair of δ- and λ-items, we can make more redexes visible. This extension furthermore is simple, as in $(C\delta)\overline{s}(B\lambda_x)$, we say that $(C\delta)$ and $(B\lambda_x)$ match if $\overline{s}$ has the same structure as a matching composite of opening and closing brackets, each δ-item corresponding to an opening bracket and each λ-item corresponding to a closing bracket. For example, in A above, $(u\delta)$ and $(R\lambda_z)$ match as $(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)$ has the bracketing structure [ ][ ] (see Figure 1).

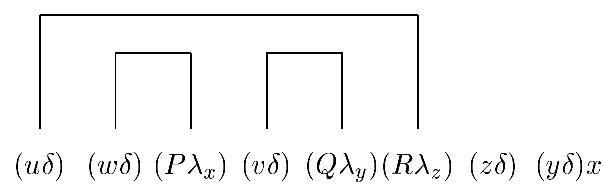

Figure 1: Extended redexes in item notation

Now, when we see a δ-item which matches a λ-item, we move the δ-item to occur next to its matching λ-item. With this extension of redexes and term reshuffling, we refine one-step β-reduction by making it a sequence of two operations: a reshuffling of the original term (so that all matching $\delta\lambda$-couples occur adjacent) followed by a classical one-step β-reduction. Hence A of Example 2.7 will be reshuffled to $(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(u\delta)(R\lambda_z)(z\delta)(y\delta)x$ and Figure 1 changes to Figure 2. Note that the item $(u\delta)$ is being shuffled into the scope of $(P\lambda_x)$ and $(Q\lambda_y)$, so we have to make sure by variable-renaming that no unwanted bindings are being introduced. Note also that no items are being shuffled outside scopes of λ-items they previously were in.

We use $TS(A)$ to describe the term reshuffled version of A. Now, we apply classical β-reduction to $TS(A)$ and we contract the classical redex $(u\delta)(R\lambda_z)$. We use $\rightsquigarrow_\beta$ for one step shuffle reduction which is the sequence of term reshuffling followed by one-step ordinary reduction $\rightarrow_\beta$. The following example summarizes all this.

Example 2.8 Back to Example 2.4, $A \equiv (u\delta)(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(R\lambda_z)(z\delta)(y\delta)x$. Now, $TS(A) \equiv (w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(u\delta)(R\lambda_z)(z\delta)(y\delta)x$. As $TS(A) \rightarrow_\beta (w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(u\delta)(y\delta)x$,

$$(w\delta)\ (P\lambda_x)\ (v\delta)\ (Q\lambda_y)\ (w\delta)\ (R\lambda_z)\ (z\delta)\ (y\delta)x$$

Figure 2: Term reshuffling in item notation

we get that $A \rightsquigarrow_\beta (w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(u\delta)(y\delta)x$. I.e. one-step $\rightsquigarrow_\beta$ amounts to a term reshuffling followed by one-step $\to_\beta$.

It is this shuffle reduction that we will put on the top of the Cube and we will investigate its properties. This reduction will be introduced in Section 5.

Notation 2.9 *Throughout the whole paper, we take $\mathcal{O}$ to range over $\{\lambda, \Pi\}$ and ω over $\{\delta, \lambda_x, \Pi_x\}$.*

2.1 Pseudo-Expressions in item notation

The Cube is a generalisation of some type systems which are explicitly typed à la Church (see [Barendregt 92]). The system $\lambda_\to$ of [Church 40] is one of the systems of the Cube. Now the systems of the Cube are based on a set of *pseudo-expressions* $\mathcal{T}$ defined by the following abstract syntax (again see [Barendregt 92]):

$$\mathcal{T} = V \mid C \mid (\mathcal{T}\delta)\mathcal{T} \mid (\mathcal{T}\mathcal{O}_V)\mathcal{T}$$

where V and C are infinite collections of variables and constants respectively. We assume that $x, y, z, \ldots$ range over V and we take two special constants $*$ and $\Box$. These constants are called sorts and the meta-variables $S, S_1, S_2, \ldots$ are used to range over the set of sorts $\mathcal{S} = \{*, \Box\}$. We take $A, B, C, a, b, \ldots$ to range over pseudo-expressions. Note furthermore that there is no distinction between term- and type-variables and that there are two notions of abstraction: λ- and Π-abstraction. Parentheses will be omitted when no confusion occurs.

For convenience sake, we divide V in two disjoint sets V^* and $V^\Box$, the sets of object respectively constructor variables. We take $x^*, y^*, z^*, \ldots$ to range over V^* and $x^\Box, y^\Box, z^\Box, \ldots$ to range over $V^\Box$.

Definition 2.10 *(Compatibility)*
Let ω range over $\{\delta\} \cup \{\mathcal{O}_x \mid x \in V\}$. We say that a relation $\to$ on terms is compatible iff the following holds:

$$\frac{A_1 \to A_2}{(A_1\omega)B \to (A_2\omega)B} \qquad \frac{B_1 \to B_2}{(A\omega)B_1 \to (A\omega)B_2}$$

Basically compatibility means that if $A \to B$ then $T[A] \to T[B]$ where $T[\ ]$ is a "pseudo-expression with a hole in it".

Definition 2.11 *(β-reduction $\to_\beta$ for the Cube)*
In the Cube, β-reduction $\to_\beta$, is the least compatible relation generated out of the following axiom:

$$(\beta) \qquad (C\delta)(B\lambda_x)A \to_\beta A[x := C]$$

We take $\twoheadrightarrow_\beta$ to be the reflexive transitive closure of $\to_\beta$ and we take $=_\beta$ to be the least equivalence relation generated by $\twoheadrightarrow_\beta$.

Note that in the Cube, β-reduction is only assumed for λ-expressions and not for Π-expressions. That is, we do not have $(C\delta)(B\Pi_x)A \to_\beta A[x := C]$. For an extension of (β) to Π-expressions, see [KN 9y].

Definition 2.12 *((main) items, (main, $\delta\mathcal{O}$-)segments, end variable, weight)*

- *If x is a variable and A is a pseudo-expression, then $(A\lambda_x), (A\Pi_x)$ and $(A\delta)$ are items (called λ-item, Π-item and δ-item respectively). We use $s, s_1, s_i, \ldots$ to range over items.*
- *A concatenation of zero or more items is a* **segment**. *We use $\overline{s}, \overline{s}_1, \overline{s}_i, \ldots$ as meta-variables for segments. We write $\emptyset$ for the empty segment.*
- *Each pseudo-expression A is the concatenation of zero or more items and a variable or constant: $A \equiv s_1 s_2 \cdots s_n x$. These items $s_1, s_2, \ldots, s_n$ are called the* **main items** *of A, x is called the* **end variable** *of A, notation* $\mathtt{endvar}(A)$.
- *Analogously, a segment $\overline{s}$ is a concatenation of zero or more items: $\overline{s} \equiv s_1 s_2 \cdots s_n$; again, these items $s_1, s_2, \ldots, s_n$ (if any) are called the* **main items**, *this time of $\overline{s}$.*
- *A concatenation of adjacent main items (in A or $\overline{s}$), $s_m \cdots s_{m+k}$, is called a* **main segment** *(in A or $\overline{s}$).*
- *A* **$\delta\mathcal{O}$-segment** *is a δ-item immediately followed by an $\mathcal{O}$-item.*
- *The* **weight** *of a segment $\overline{s}$, $\mathtt{weight}(\overline{s})$, is the number of main items that compose the segment. Moreover, we define $\mathtt{weight}(\overline{s}x) = \mathtt{weight}(\overline{s})$.*

When one desires to start a β-reduction on the basis of a certain δ-item and a λ-item occurring in one segment (recall, no reductions are based on δ- and Π-items), the *matching* of the δ and the λ in question is the important thing, even when the δ- and λ-items are separated by other items. I.e., the relevant question is whether they *may* together become a $\delta\lambda$-segment after a number of β-steps. This depends solely on the structure of the intermediate segment. If such an intermediate segment is well-balanced then the δ-item and the λ-item match and β-reduction based on these two items may take place. Some well-balanced segments also play an important role. They may act as a definition. For example, $(A\delta)(B\lambda_x)C$ means define x of type B to be A in C. Sometimes, definitions are interleaved as in $(A_1\delta)(B_1\delta)(B_2\lambda_x)(A_2\lambda_y)D$ where the definition "x becomes B_1" is used inside the definition "y becomes A_1". We will assume definitions not to contain Π-items in this paper. Extending this work to the case where for example $(A\delta)(B\Pi_x)$ is a definition will be investigated in [**?**]. (TOEVOEGEN in literatuurlijst: artikel over $\beta\Pi$-reduction) Here is the definition of well-balanced/definitional segments and applying definitions:

Definition 2.13 *(well-balanced segments, definitions, definition application)*

- *The empty segment $\emptyset$ is a well-balanced segment.*
- *If $\overline{s}$ is well-balanced, then $(A\delta)\overline{s}(B\mathcal{O}_x)$ is well-balanced.*
- *If $\overline{s}$ is well-balanced which does not contain main Π-items, then $(A\delta)\overline{s}(B\lambda_x)$ is a definition.*
- *The concatenation of well-balanced segments is a well-balanced segment.*
- *Let $\overline{s}$ be a well-balanced segment which is a sequence of definitions and $A \in \mathcal{T}$. We define the application of the definition $\overline{s}$ in A, $[A]_{\overline{s}}$ inductively as follows: $[A]_{\emptyset} \equiv A$, $[A]_{(B\delta)\overline{s_1}(C\lambda_x)} \equiv [A[x := B]]_{\overline{s_1}}$ 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 and that when none of the binding variables of $\overline{s}$ are free in A, then $[A]_{\overline{s}} \equiv A$.*

Lemma 2.14 *If $\overline{s_2}$ is a definition, none of the binding variables in $\overline{s_2}$ is free in A, and $(A\delta)$ does not match a Π-item in B, then*

$$\overline{s_1}(A\delta)\overline{s_2}B =_\beta \overline{s_1 s_2}(A\delta)B$$

Proof: *induction on* `weight`$(\overline{s_2})$*:*

- $\overline{s_2} \equiv \emptyset$*: by definition of β-equality.*
- $\overline{s_2} \equiv (D\delta)(E\lambda_x)$ *then*

$$\begin{array}{llll} \overline{s_1}(A\delta)\overline{s_2}B & \equiv & \overline{s_1}(A\delta)(D\delta)(E\lambda_y)B & =_\beta \\ \overline{s_1}(A\delta)(B[y := D]) & \equiv^{VC} & \overline{s_1}((A\delta)B[y := D]) & =_\beta \\ \overline{s_1}(D\delta)(E\lambda_y)(A\delta)B & & & \end{array}$$

- $\overline{s_2} \equiv (D\delta)\overline{s_3}(E\lambda_y)$, $\overline{s_3}$ *well-balanced, then*

$$\begin{array}{llll} \overline{s_1}(A\delta)\overline{s_2}B & \equiv & \overline{s_1}(A\delta)(D\delta)\overline{s_3}(E\lambda_y)B & \stackrel{IH}{=}_\beta \\ \overline{s_1}(A\delta)\overline{s_3}(D\delta)(E\lambda_y)B & \equiv & \overline{s_1 s_3}(A\delta)(D\delta)(E\lambda_y)B & \stackrel{IH}{=}_\beta \\ \overline{s_1 s_3}(D\delta)(E\lambda_y)(A\delta)B & \stackrel{IH}{=}_\beta & \overline{s_1 s_3}(D\delta)(E\lambda_y)(A\delta)B & \stackrel{IH}{=}_\beta \\ \overline{s_1}(D\delta)\overline{s_3}(E\lambda_y)(A\delta)B & & & \end{array}$$

□

Corollary 2.15 *If $\overline{s_2}$ is a sequence of definitions, none of the binding variables in $\overline{s_2}$ is free in A, and $(A\delta)$ does not match a Π-item in B, then*

$$\overline{s_1}(A\delta)\overline{s_2}B =_\beta \overline{s_1 s_2}(A\delta)B$$

□

Remark 2.16 Note that this does not hold in case $\overline{s_2}$ is well-balanced but neither a definition nor a sequence of definitions. The reason for this failure is that we have no way of reducing $\delta\Pi$-segments. For example, $(u\delta)(x\delta)(x\Pi_y)(y\lambda_z)z \neq_\beta (x\delta)(x\Pi_y)(u\delta)(y\lambda_z)z$. This will not be a problem we face as in *legal* terms of the cube, all Π-items are bachelor.

Lemma 2.17

1. *If none of the binding variables of the sequence of definitions $\overline{s}$ is free in A, then $[A]_{\overline{s}} \equiv A$.*

2. $[A]_{\overline{s}} =_\beta \overline{s}A$.

Proof:

1. Obvious.

2. Induction on $\mathtt{weight}(\overline{s})$*:*

- *If* $\overline{s} \equiv \emptyset$*, then* $[A]_{\overline{s}} \equiv \overline{s}A$ *by definition.*
- *If* $\overline{s} \equiv (B\delta)\overline{s_1}(C\lambda_x)$*, then* $[A]_{\overline{s}} \equiv [A[x := B]]_{\overline{s_1}} \stackrel{IH}{=}_\beta \overline{s_1}(A[x := B]) =_\beta \overline{s_1}(B\delta)(C\lambda_x)A \stackrel{Lemma 2.14}{=}_\beta (B\delta)\overline{s_1}(C\lambda_x)A$ *as none of the binding variables of* $\overline{s_1}$ *is free in* B *by* VC*.*
- $\overline{s} \equiv \overline{s_1 s_2}$*:* $[A]_{\overline{s}} \equiv [[A]_{\overline{s_2}}]_{\overline{s_1}} \stackrel{IH}{=}_\beta \overline{s_1}[A]_{\overline{s_2}} \stackrel{IH}{=}_\beta \overline{s_1 s_2}A$*.*

□

A well-balanced segment has the same structure as a matching composite of opening and closing brackets, each δ- (or $\mathcal{O}$-)item corresponding with an opening (resp. closing) bracket. In a definition, the first [matches the last] and no Π-items are allowed.

Remark 2.18 Note that the definition of well-balanced segments and definitions is equivalent to saying that

1. $\emptyset$ is well-balanced.
2. If $\overline{s_1}, \overline{s_2}$ are well-balanced, then $(A\delta)\overline{s_1}(B\mathcal{O}_x)\overline{s_2}$ is well-balanced.
3. If $\overline{s}$ is well-balanced and does not contain Π-items, then $(A\delta)\overline{s}(B\lambda_x)$ is a definition.

Sometimes we use this definition in proofs by induction.

Now we can easily define what matching $\delta\mathcal{O}$-couples are, given a segment $\overline{s}$. Namely, they are a main δ-item and a main $\mathcal{O}$-item separated by a well-balanced segment. Such couples are reducible couples in case $\mathcal{O} = \lambda$. The δ-item and $\mathcal{O}$-item of the $\delta\mathcal{O}$-couple are said to match and each of them is called a partner or a partnered item. The items in a segment that are not partnered are called bachelor items. The following definition summarizes all this:

Definition 2.19 *(match, $\delta\mathcal{O}$- (reducible) couple, partner, partnered item, bachelor item) Let $A \in \mathcal{T}$. Let $\overline{s} \equiv s_1 \cdots s_n$ be a segment occurring in A.*

- *We say that s_i and s_j* **match***, when $1 \leq i < j \leq n$, s_i is a δ-item, s_j is a $\mathcal{O}$-item, and the sequence $s_{i+1}, \ldots, s_{j-1}$ forms a well-balanced segment.*
- *When s_i and s_j match, we call $s_i s_j$ a $\delta\mathcal{O}$-***couple***. If $\mathcal{O} = \lambda$ and $s_{i+1} \cdots s_{j-1}$ contains no Π-item then $s_i s_j$ is a* **reducible couple***.*
- *When s_i and s_j match, we call both s_i and s_j the* **partners** *in the $\delta\mathcal{O}$-couple. We also say that s_i and s_j are* **partnered** *items.*
- *All the $\mathcal{O}$- (or δ-)items s_k in A that are not partnered, are called* **bachelor** *$\mathcal{O}$- (resp. δ-)items.*

Example 2.20 In $\overline{s} \equiv (a\lambda_x)(b\lambda_y)(c\delta)(d\lambda_z)(e\lambda_u)(f\delta)(g\delta)(h\delta)(i\lambda_v)(j\lambda_w)(k\delta)$:

- $(c\delta)$ matches with $(d\lambda_z)$, $(h\delta)$ matches with $(i\lambda_v)$ and $(g\delta)$ with $(j\lambda_w)$. The segments $(c\delta)(d\lambda_z)$ and $(h\delta)(i\lambda_v)$ are $\delta\lambda$-segments (and $\delta\lambda$-couples). There is another $\delta\lambda$-couple in $\overline{s}$, viz. the couple of $(g\delta)$ and $(j\lambda_w)$.

- $(c\delta)$, $(d\lambda_z)$, $(g\delta)$, $(h\delta)$, $(i\lambda_v)$ and $(j\lambda_w)$, are the partnered main items of $\overline{s}$. $(a\lambda_x)$, $(b\lambda_y)$, $(e\lambda_u)$, $(f\delta)$ and $(k\delta)$, are bachelor items.

- $(g\delta)(h\delta)(i\lambda_v)(j\lambda_w)$ is a well-balanced segment.

2.2 Background for Typing in item notation

In this section, we let $\vdash$ range over the typing relations of Sections $3 \cdots 7$ and $\twoheadrightarrow$ range over both $\twoheadrightarrow_\beta$ and $\rightsquigarrow\!\!\!\!\twoheadrightarrow_\beta$.

Definition 2.21 *(declarations, statements, pseudocontexts, $\vdash_\smile$, judgements, $\subseteq'$)*

1. *A declaration is of the form $\overline{s}(A\lambda_x)$ where $\overline{s} \equiv \emptyset$ or $\overline{s} \equiv (B\delta)\overline{s_1}$ with $\overline{s_1}$ well-balanced not containing main Π-items. Hence declarations are either definitions or of the form $(A\lambda_x)$. We take $d, d_1, \ldots$ to range over declarations.*

2. *In a declaration $d \equiv \overline{s}(A\lambda_x)$, we define $\mathtt{subj}(d)$ and $\mathtt{pred}(d)$ to be x and A respectively. $\underline{d}$ and $\mathtt{def}(d)$ are defined to be $\emptyset$ if $\overline{s} \equiv \emptyset$ and to be $\overline{s_1}$, B respectively if $\overline{s} \equiv (B\delta)\overline{s_1}$.*

3. *We define $dom(d)$ to be $\{x \mid (A\lambda_x)$ is a main item in $d\}$.*

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

5. *A pseudocontext is a concatenation of declarations such that if $(A\lambda_x)$ and $(B\lambda_y)$ are two different main items of the pseudocontext, then $x \not\equiv y$. We use $\Gamma, \Delta, \Gamma', \Gamma_1, \Gamma_2, \ldots$ to range over pseudocontexts.*

6. *If $\Gamma \equiv d_1 \cdots d_k$ then $dom(\Gamma) = \cup_{1\leq i\leq k} dom(d_i)$ and $d \in' \Gamma$ iff $\exists i[d \equiv d_i]$.*

7. *If $\Gamma \equiv d_1 \cdots d_n$, we define the set of subdeclarations of Γ, Γ-$\mathtt{decl}$ inductively as follows:*

   - *$\{d_1, \ldots, d_n\} \subseteq \Gamma$-$\mathtt{decl}$.*
   - *If $d \in \Gamma$-$\mathtt{decl}$ and $\underline{d} \not\equiv \emptyset$ then for all $d' \in \underline{d}$-$\mathtt{decl}$, $d' \in \Gamma$-$\mathtt{decl}$.*

   Note that $dom(\Gamma) = \{\mathtt{subj}(d) \mid d \in \Gamma$-$\mathtt{decl}\}$. We define the set of definitions of Γ by Γ-$\mathtt{def} = \{d \in \Gamma$-$\mathtt{decl} \mid d$ is a definition$\}$.

8. *Let Γ be a pseudocontext and d be a declaration. We say that Γ invites d with respect to $\vdash$, notation $\Gamma \vdash_\smile d$ iff*

   - *Case $d \equiv (A\lambda_x)$ then $\Gamma \vdash A : S$ for some sort S, x is fresh in Γ, A, case $S \equiv *$ then $x \in V^*$ and case $S \equiv \Box$ then $x \in V^\Box$.*
   - *Case $d \equiv (A\delta)\underline{d}(B\lambda_x)$ then $\Gamma\underline{d} \vdash A : B$, $\Gamma\underline{d} \vdash B : S$ for some sort S, x is fresh in $\Gamma\underline{d}, A, B$, $[A]_{\underline{d}} \equiv A$, case $S \equiv *$ then $x \in V^*$ and case $S \equiv \Box$ then $x \in V^\Box$.*

9. *When Γ is a pseudocontext and $A : B$ is a statement, we call $\Gamma \vdash A : B$, a judgement, and write $\Gamma \vdash A : B : C$ to mean $\Gamma \vdash A : B \wedge \Gamma \vdash B : C$.*

10. *We define $\subseteq'$ between pseudocontexts to be the least reflexive transitive relation which satisfies:*
    - $\Gamma\Delta \subseteq' \Gamma(C\lambda_x)\Delta$ *if x is fresh in Γ, Δ, C and no λ-item in Δ matches a δ-item in Γ and $FV(C) \subseteq dom(\Gamma)$.*
    - $\Gamma\underline{d}\Delta \subseteq' \Gamma d\Delta$ *if d is a definition, $\mathtt{subj}(d)$ is fresh in $\Gamma\underline{d}\Delta, \mathtt{def}(d), \mathtt{pred}(d)$ and $FV(\mathtt{def}(d)) \subseteq dom(\Gamma)$, $FV(\mathtt{pred}(d)) \subseteq dom(\Gamma\underline{d})$,*
    - $\Gamma\overline{s}(A\lambda_x)\Delta \subseteq' \Gamma(D\delta)\overline{s}(A\lambda_x)\Delta$ *if $(A\lambda_x)$ is bachelor, $\overline{s}$ is well-balanced and $FV(D) \subseteq dom(\Gamma)$.*

Definition 2.22 *(Definitional β-equality) For all legal contexts Γ we define the binary relation $\Gamma \vdash \cdot =_{\mathtt{def}} \cdot$ to be the equivalence relation generated by*

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

Remark 2.23 If no definitions are present in Γ then $\Gamma \vdash A =_{\mathtt{def}} B$ is the same as $A =_\beta B$.

Definition 2.24 *Let Γ be a pseudocontext and A be a pseudo-expression.*

1. *Let $d, d_1, \ldots, d_n$ be declarations. We define $\Gamma \vdash d$ and $\Gamma \vdash d_1 \cdots d_n$ simultaneously as follows:*
    - $\Gamma \vdash d$ *iff* $\Gamma \vdash \mathtt{subj}(d) : \mathtt{pred}(d) \wedge \Gamma \vdash \mathtt{def}(d) : \mathtt{pred}(d) \wedge \Gamma \vdash \underline{d} \wedge \Gamma \vdash \mathtt{subj}(d) =_{\mathtt{def}} \mathtt{def}(d)$.
    - $\Gamma \vdash d_1 \cdots d_n$ *iff* $\Gamma \vdash d_i$ *for all* $1 \leq i \leq n$.
2. *Γ is called $\vdash$-legal if $\exists P, Q \in \mathcal{T}$ such that $\Gamma \vdash P : Q$.*
3. *$A \in \mathcal{T}$ is called a $\Gamma^{\vdash}$-term if $\exists B \in \mathcal{T}[\Gamma \vdash A : B$ or $\Gamma \vdash B : A]$.*
   We take $\Gamma^{\vdash}$-terms $= \{A \in \mathcal{T} \mid \exists B \in \mathcal{T}[\Gamma \vdash A : B \vee \Gamma \vdash B : A]\}$.
4. *We take $\Gamma^{\vdash}$-kinds $= \{A \mid \Gamma \vdash A : \square\}$ and $\Gamma^{\vdash}$-types $= \{A \in \mathcal{T} \mid \Gamma \vdash A : *\}$.*
5. *$A \in \mathcal{T}$ is called a $\Gamma^{\vdash}$-element if $\exists B \in \mathcal{T} \exists S \in \mathcal{S}[\Gamma \vdash A : B$ and $\Gamma \vdash B : S]$. We have two categories of elements: constructors and objects. We take $\Gamma^{\vdash}$-constructors $= \{A \in \mathcal{T} \mid \exists B \in \mathcal{T}[\Gamma \vdash A : B : \square]\}$ and $\Gamma^{\vdash}$-objects $= \{A \in \mathcal{T} \mid \exists B \in \mathcal{T}[\Gamma \vdash A : B : *]\}$.*
6. *$A \in \mathcal{T}$ is called $\vdash$-legal if $\exists \Gamma[A \in \Gamma^{\vdash}$-terms$]$. Moreover, A is a $\vdash$-X, if $\exists \Gamma[A \in \Gamma^{\vdash}$-$Xs]$ for $X \in \{$type, term, kind, object, constructor$\}$.*

Definition 2.25 *Define a map $\# : \mathcal{T} \longrightarrow \{0, 1, 2, 3\}$ by $\#(\square) = 3$, $\#(*) = 2$, $\#(x^\square) = 1$, $\#(x^*) = 0$, $\#(A) = \#(\mathtt{endvar}(A))$. For $A \in \mathcal{T}$, $\#(A)$ is called the* **degree** *of A.*

We shall use $\#$ to prove that the classes of kinds, constructors and objects are mutually exclusive. First we collect some basic facts about $\square$ and $*$ in the type systems:

Lemma 2.26

1. *If $\Gamma \vdash A : B$ then $A \not\equiv \Box$.*

2. *If Γ is a legal context, then $\Box \notin \Gamma$.*

3. *If A is a legal term, then $A \equiv \Box$ or $\Box \notin A$.*

4. *Suppose $\Gamma \vdash A : B$, then $\texttt{endvar}(A) \equiv * \Longleftrightarrow B \equiv \Box$.*

5. *If $(A\delta)$ is an item in a legal context then $\texttt{endvar}(A) \not\equiv *$.*

6. *If $(A\delta)$ is an item in a legal term then $\texttt{endvar}(A) \not\equiv *$.*

Proof:

1. *induction on the derivation rules.*

2. *simultaneous induction with 3. on the derivation rules using 1.*

4. *induction on the derivation rules; for $\Rightarrow$ use 1. and 3. We treat the case in which $\Gamma \vdash A : B'$ is a consequence of $\Gamma \vdash A : B$, $\Gamma \vdash B' : S$ and $\Gamma \vdash B = B'$. From the induction hypothesis it follows that $B \equiv \Box$. Then substituting and reducing introduce no $\Box$ in B' as by 1. $\Box \notin \Gamma$, so $\Box \in B'$. But then by 3.: $B' \equiv \Box$.*

5. *induction on the derivation rules; use 4. and 2.*

6. *induction on the derivation rules; use 5., 4. and 3.*

$\Box$

Now we can prove that whenever $\Gamma \vdash A : B$ then $\#(A) + 1 = \#(B)$.

Lemma 2.27 *Call a statement $A : B$ OK iff $\#(A) + 1 = \#(B)$, call a definition d OK iff $\#(\texttt{def}(d)) = \#(\texttt{subj}(d)) = \#(\texttt{pred}(d)) - 1$, and call a judgement $\Gamma \vdash A : B$ OK iff $A : B$ is OK, all definitions $d \in \Gamma\texttt{-def}$ are OK and for all items $(C\mathcal{O}_x) \in \Gamma, A, B$ ($\mathcal{O} \in \{\lambda, \Pi\}$): $x : C$ is OK.*

Then for all contexts Γ and terms A, B: if $\Gamma \vdash A : B$ then $\Gamma \vdash A : B$ is OK.

Proof: *We use induction on the derivation rules, we treat three cases.*

- *$\Gamma \vdash (a\delta)F : B[x := a]$ as a consequence of $\Gamma \vdash F : (A\Pi_x)B$, $\Gamma \vdash a : A$, then by the induction hypothesis $\#(x) = \#(A) - 1 = \#(a)$ and it can easily be seen that $\#(x) = \#(a) \Rightarrow \#(B[x := a]) = \#(B)$.*

- *$\Gamma \vdash dC : [D]_d$ out of $\Gamma d \vdash C : D$, then by the induction hypothesis: for all subdefinitions d' of d, $\#(\texttt{def}(d')) = \#(\texttt{subj}(d'))$ so by repeatedly applying $\#(x) = \#(a) \Rightarrow \#(B[x := a]) = \#(B)$ we get $\#([D]_d) = \#(D)$.*

- *$\Gamma \vdash A : B'$ out of $\Gamma \vdash A : B$, $\Gamma \vdash B' : S'$, $\Gamma \vdash B = B'$, then by the generation corollary 3.12 $B \equiv \Box$ or $\Gamma \vdash B : S$ for some sort S.*

  If $B \equiv \Box$ then $\Gamma \vdash B = B'$ implies $B' \equiv \Box$ as in the proof of lemma 2.26.

  If $B \not\equiv \Box$ then $S \not\equiv \Box \wedge B' \not\equiv \Box$ implies $S \equiv S'$; suppose now $S \equiv \Box$, then $\Gamma \vdash B : \Box$ so by lemma 2.26 $\texttt{endvar}(B) \equiv *$ so again by lemma 2.26 also $\texttt{endvar}(B') \equiv *$, hence $\#(B') = \#(B) = 2$. If $S' \equiv \Box$ then similar $\#(B) = \#(B') = 2$.

Corollary 2.28 *If Γ is a legal context, then*

1. $\Gamma^{\vdash}$*-kinds* $\cap\, \Gamma^{\vdash}$*-constructors* $= \emptyset$,
   $\Gamma^{\vdash}$*-kinds* $\cap\, \Gamma^{\vdash}$*-objects* $= \emptyset$,
   $\Gamma^{\vdash}$*-constructors* $\cap\, \Gamma^{\vdash}$*-objects* $= \emptyset$,
   $\Box \notin \Gamma^{\vdash}$*-kinds* $\cup\, \Gamma^{\vdash}$*-constructors* $\cup\, \Gamma^{\vdash}$*-objects.*
2. *If $(A\Pi_x)B$ is a $\Gamma^{\vdash}$-term then A and B are both a $\Gamma^{\vdash}$-kind or a $\Gamma^{\vdash}$-type.*
3. *If $(A\lambda_x)B$ is a $\Gamma^{\vdash}$-term then A is a $\Gamma^{\vdash}$-kind or a $\Gamma^{\vdash}$-type and B is a $\Gamma^{\vdash}$-constructor or a $\Gamma^{\vdash}$-object.*
4. *If $(A\delta)B$ is a $\Gamma^{\vdash}$-term then A and B are both a $\Gamma^{\vdash}$-constructor or a $\Gamma^{\vdash}$-object.*

Proof: *1. is a direct consequence of lemma 2.27.*
2., 3. and 4. are an easy corollary of the relevant Generation Lemma and Generation Corollary. $\Box$

2.3 Machinery for Strong Normalisation

In [BKN 9x], we used the technique of [Barendregt 92] to show Strong Normalisation for $\lambda_{\rightarrow}$ with extended reduction. However, here we use the proof of [Geuvers 94] due to its flexibility and the possibility of its generalisation to systems beyond the Cube, which we may be investigating in the future. Here is the terminology that will be needed. Let $\twoheadrightarrow$ be a reduction relation containing $\twoheadrightarrow_{\beta}$, which is Church Rosser and where the least equivalence relation closed under it, denoted $=_{\rightarrow}$ is the same as $=_{\beta}$, and let $\vdash$ be a typing relation for which the sets of objects, constructors and kinds are pairwise disjoint.

Lemma 2.29 *(Soundness of $\twoheadrightarrow$) If $A, B \in \mathcal{T}$ are legal terms such that $A =_{\rightarrow} B$ then there is a path of one-step reductions and expansions via legal terms between A and B.*
Proof: *By Church-Rosser there exists a term C such that $A \twoheadrightarrow_{\beta} C$ and $B \twoheadrightarrow_{\beta} C$. By Subject Reduction for ordinary β-reduction all terms on the path $A \cdots C \cdots B$ are legal.* $\Box$

Definition 2.30

- *Define the set of untyped λ-terms by*

$$\Lambda = V \mid C \mid (\Lambda\delta)\Lambda \mid (\lambda_V)\Lambda$$

- *We say that a term $M \in \Lambda$ is* **strongly normalising** *with respect to $\twoheadrightarrow$ iff every $\twoheadrightarrow$-reduction path starting at M, terminates.*
- *We define $SN_{\rightarrow} = \{M \in \Lambda : M$ is strongly normalising with respect to $\twoheadrightarrow\}$.*
- *For $A, B \subseteq \Lambda$ we define $A \longrightarrow B = \{M \in \Lambda \mid \forall N \in A[(N\delta)M \in B]\}$.*

Definition 2.31 *Define the* **key redex** *of a term M as follows:*

1. *$(A\delta)(B\lambda_x)C$ has key redex $(A\delta)(B\lambda_x)C$.*

2. If M has key redex N, then $(P\delta)M$ has key redex N.

Define $\mathbf{red_k(M)}$ *to be the term obtained from M by contracting its key redex. Note that not all terms have a key redex and that if a term has a key redex then it is unique.*

Definition 2.32

- *Define the set of base terms $\mathcal{B}_{\rightarrow} \subseteq \Lambda$ by*
  1. *$V \subseteq \mathcal{B}_{\rightarrow}$.*
  2. *If $M \in \mathcal{B}_{\rightarrow}, N \in SN_{\rightarrow}$ then also $(N\delta)M \in \mathcal{B}_{\rightarrow}$.*
- *We call $X \subseteq \Lambda$* $\mathbf{saturated}_{\twoheadrightarrow}$ *iff:*
  1. *$X \subseteq SN_{\rightarrow}$.*
  2. *$\mathcal{B}_{\rightarrow} \subseteq X$.*
  3. *For all $M \in \Lambda$: if $M \in SN_{\rightarrow}$ and $red_k(M) \in X$ then also $M \in X$.*
- *We define $SAT_{\rightarrow} = \{X \subseteq \Lambda : X \text{ is saturated}_{\rightarrow}\}$*

Lemma 2.33

1. *$SN_{\rightarrow} \in SAT_{\rightarrow}$.*
2. *$\forall X \in SAT_{\rightarrow} : X \neq \emptyset$.*
3. *If $N \in SN_{\rightarrow}, M \in X \in SAT_{\rightarrow}$ and $x \notin FV(M)$ then $(N\delta)(\lambda_x)M \in X$. (Note here that [Geuvers 94] takes $(N\delta)(M\delta)(\lambda_y)(\lambda_x)y$ instead of $(N\delta)(\lambda_x)M$. The first however, will not fit our purposes as is explained in Remark 5.12)*
4. *$A, B \in SAT_{\rightarrow} \Rightarrow A \longrightarrow B \in SAT_{\rightarrow}$.*
5. *If I is a set and $X_i \in SAT_{\rightarrow}$ for all $i \in I$, then $\bigcap_{i \in I} X_i \in SAT_{\rightarrow}$.*

Proof:

1. *$SN_{\rightarrow} \subseteq SN_{\rightarrow}$, $\mathcal{B}_{\rightarrow} \subseteq SN_{\rightarrow}$. Furthermore, if $M \in SN_{\rightarrow}$ and $red_k(M) \in SN_{\rightarrow}$ then also $M \in SN_{\rightarrow}$ as $\twoheadrightarrow_\beta \subseteq \twoheadrightarrow$.*
2. *By 2. of the definition of saturated sets.*
3. *By 3. of the definition of saturated sets.*
4. *Suppose $A, B \in SAT_{\rightarrow}$.*
   - *As $v \in A$ for all $v \in V$, we see: $t \in A \longrightarrow B \Rightarrow (v\delta)t \in B \Rightarrow (v\delta)t \in SN_{\rightarrow} \Rightarrow t \in SN_{\rightarrow}$. So $A \longrightarrow B \subseteq SN_{\rightarrow}$.*
   - *If $x \in V$, $N \in A$ then $(N\delta)x \in B$ as $\mathcal{B}_{\rightarrow} \subseteq B$, so $V \subseteq A \longrightarrow B$. Also, if $M \in \mathcal{B}_{\rightarrow} \cap A \longrightarrow B$, $N \in SN_{\rightarrow}$ then for all $N' \in A : N' \in SN_{\rightarrow}$ so $(N'\delta)(N\delta)M \in \mathcal{B}_{\rightarrow} \subseteq B$ so $(N\delta)M \in A \longrightarrow B$. Hence $\mathcal{B}_{\rightarrow} \subseteq A \longrightarrow B$.*
   - *If $M \in SN_{\rightarrow}$, $red_k(M) \in A \longrightarrow B$ then for all $N \in A$: $(N\delta)red_k(M) \in B$ hence $(N\delta)M \in B$, hence also $M \in A \longrightarrow B$.*

5. *Easy.* $\square$

We define three maps, first $CP^{\vdash}_{\rightarrow}$ of $\Gamma^{\vdash}$-kinds to the function space of $SAT_{\rightarrow}$, then $[\![\ ]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ of $\Gamma^{\vdash}$-terms$\backslash\Gamma^{\vdash}$-objects to elements of the function space of $SAT_{\rightarrow}$, and third $(\!|\ |\!)_{\rho^{\vdash}_{\twoheadrightarrow}}$ of $\Gamma^{\vdash}$-terms to Λ, such that when certain conditions are met we have:
$\Gamma \vdash A : B : \square \Rightarrow [\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \in CP^{\vdash}_{\rightarrow}(B), [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \in SAT_{\rightarrow}$ and $\Gamma \vdash A : B \Rightarrow (\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$.

Definition 2.34 *Define for all kinds A the set of computability predicates for A in the following way:*

- $CP^{\vdash}_{\rightarrow}(*) = SAT_{\rightarrow}$,
- $CP^{\vdash}_{\rightarrow}((A\Pi_{x^{\square}})B) = CP^{\vdash}_{\rightarrow}(A) \rightarrow CP^{\vdash}_{\rightarrow}(B)$
- $CP^{\vdash}_{\rightarrow}((A\Pi_{x^{*}})B) = CP^{\vdash}_{\rightarrow}(B)$
- $CP^{\vdash}_{\rightarrow}(dA) = CP^{\vdash}_{\rightarrow}(A) \qquad$ *if d a definition*

(with $CP^{\vdash}_{\rightarrow}(A) \rightarrow CP^{\vdash}_{\rightarrow}(B)$ is meant the function space of $CP^{\vdash}_{\rightarrow}(A)$ to $CP^{\vdash}_{\rightarrow}(B)$).
Now define $\mathbf{CP}^{\vdash}_{\twoheadrightarrow} = \bigcup\{CP^{\vdash}_{\rightarrow}(A) \mid A$ is a $\vdash$-kind $\}$.

Lemma 2.35

1. *If A is a legal kind, B a legal constructor and C is a legal object, then $CP^{\vdash}_{\rightarrow}(A) = CP^{\vdash}_{\rightarrow}(A[x^{\square} := B])$ and $CP^{\vdash}_{\rightarrow}(A) = CP^{\vdash}_{\rightarrow}(A[x^{*} := C])$.*
2. *If dA is a legal kind (remember Remark* **??***) where d is a definition, then $CP^{\vdash}_{\rightarrow}(dA) = CP^{\vdash}_{\rightarrow}(A)$.*

Proof: *1. is by induction on the structure of A, noting that A cannot contain bachelor δ- or λ-items, 2. is by 1. noting that all definienda in a definition are either constructors or objects.* $\square$

Definition 2.36 *Let Γ be a $\vdash$-legal context.*

- *A* **Γ-constructor valuation***, notation $\xi^{\vdash}_{\rightarrow} \models^{\square} \Gamma$, is a map $\xi^{\vdash}_{\rightarrow} : V^{\square} \longrightarrow \mathbf{CP}^{\vdash}_{\twoheadrightarrow}$ such that for all $(A\lambda_x) \in' \Gamma$ with A a Γ-kind (i.e. $x \in V^{\square}$): $\xi^{\vdash}_{\rightarrow}(x) \in \mathbf{CP}^{\vdash}_{\twoheadrightarrow}(A)$.*
- *If $\xi^{\vdash}_{\rightarrow}$ is a constructor valuation, then $[\![\ ]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} : \Gamma^{\vdash}$-terms$\backslash\Gamma^{\vdash}$-objects $\rightarrow \mathbf{CP}^{\vdash}_{\twoheadrightarrow}$ is defined inductively as follows:*

$$
\begin{array}{lcl}
[\![\square]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & := & SN_{\rightarrow} \\
[\![*]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & := & SN_{\rightarrow} \\
[\![x^{\square}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & := & \xi^{\vdash}_{\rightarrow}(x^{\square}) \\
[\![(A\delta)B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & := & \begin{cases} [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}[\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & \text{if } A \in \Gamma^{\vdash}\text{-constructors} \\ [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & \text{if } A \in \Gamma^{\vdash}\text{-objects} \end{cases} \\
[\![(A\lambda_x)B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & := & \begin{cases} \boldsymbol{\lambda} f \in CP^{\vdash}_{\rightarrow}(A).[\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)} & \text{if } A \in \Gamma^{\vdash}\text{-kinds} \\ [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & \text{if } A \in \Gamma^{\vdash}\text{-types} \end{cases} \\
[\![(A\Pi_x)B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & := & \begin{cases} [\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \rightarrow \bigcap_{f \in CP^{\vdash}_{\twoheadrightarrow}(A)} [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)} & \text{if } A \in \Gamma^{\vdash}\text{-kinds}, x \in V^{\square} \\ [\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \rightarrow [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} & \text{if } A \in \Gamma^{\vdash}\text{-types}, x \in V^{*} \end{cases}
\end{array}
$$

where $\xi^{\vdash}_{\hookrightarrow}(x := N)$ is the valuation that assigns $\xi^{\vdash}_{\hookrightarrow}(y)$ to $y \not\equiv x$ and N to x. Furthermore, with $[\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}[\![B]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ we mean application of the function $[\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ onto its argument $[\![B]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ and by $\boldsymbol{\lambda}$ we mean function-abstraction.

Now we have to verify that $[\![\ ]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ is a well defined mapping, but first we need some helpful facts about $[\![\ ]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$.

Lemma 2.37 *Let $A, A' \in \Gamma^{\vdash}$-terms$\backslash\Gamma^{\vdash}$-objects$, B \in \Gamma^{\vdash}$-constructors$, C \in \Gamma^{\vdash}$-objects$, x^{\Box}$ a constructor variable and x^* an object variable. Then*

1. $[\![A[x^{\Box} := B]]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} = [\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}(x^{\Box} := [\![B]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}})}$
2. $[\![A[x^* := C]]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} = [\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$
3. $A =_{\beta} A' \Rightarrow [\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} = [\![A']\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$

Proof: *1. and 2. are by induction on the structure of A.*
3. is by induction on the generation of $=_{\beta}$. □

Remark 2.38 Note that we use $=_{\beta}$ and not $=_{\rightarrow}$, because the equality relations generated by $\twoheadrightarrow_{\beta}$ and $\hookrightarrow\!\!\!\rightarrow_{\beta}$ are both $=_{\beta}$.

Lemma 2.39 *(Soundness of $[\![\ ]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$)*

If $\Gamma \vdash A : B : \Box$ then for all $\xi^{\vdash}_{\hookrightarrow}$ such that $\xi^{\vdash}_{\hookrightarrow} \models^{\Box} \Gamma$, we have: $[\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ and $[\![B]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ are well-defined and $[\![A]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in \mathbf{CP}^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}(B)$, $[\![B]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in SAT_{\hookrightarrow}$.

Proof: *By induction on the derivation rules. We treat two cases:*

- *$\Gamma \vdash (a\delta)F : B[x := A]$ as a consequence of $\Gamma \vdash F : (A\Pi_x)B$ and $\Gamma \vdash a : A$. It is not difficult to see that $[\![B[x := A]]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in SAT_{\hookrightarrow}$ if $[\![B[x := A]]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ is a kind, because by Lemma 2.27, then also B is a kind. Furthermore, by the induction hypothesis $[\![F]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}((A\Pi_x)B)$ and if A is a kind then also $[\![a]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}(A)$.*

  If A is not a kind, then $[\![(a\delta)F]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} = [\![F]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}((A\Pi_x)B) = CP^{\vdash}_{\hookrightarrow}(B)$. If A is a kind, then $[\![F]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}((A\Pi_x)B) = CP^{\vdash}_{\hookrightarrow}(A) \rightarrow CP^{\vdash}_{\hookrightarrow}(B)$ and hence $[\![(a\delta)F]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} = [\![F]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}[\![a]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}(B) \overset{\text{Lemma 2.35}}{=} CP^{\vdash}_{\hookrightarrow}(B[x := a])$.

- *$\Gamma \vdash dC : [D]_d$ as a consequence of $\Gamma d \vdash C : D$. Then by the induction hypothesis $[\![C]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}(D)$ for all $\xi^{\vdash}_{\hookrightarrow} \models^{\Box} \Gamma d$ and if D is a kind, then $[\![D]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in SAT_{\hookrightarrow}$. Now let $\xi^{\vdash}_{\hookrightarrow} \models^{\Box} \Gamma$, then $[\![dC]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \overset{\text{Lemma 2.37.3}}{=} [\![[C]_d]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} = [\![C]\!]_{\xi'^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ where $\xi'^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}(x^{\Box}) = \xi^{\vdash}_{\hookrightarrow}(x^{\Box})$ if $x^{\Box}$ is not the subject of a subdefinition in d, and $\xi'^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}(x^{\Box}) = [\![def(d')]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$ if $x^{\Box}$ is the subject of a d' a subdefinition of d.*

  But $\xi'^{\vdash}_{\hookrightarrow\!\!\!\rightarrow} \models^{\Box} \Gamma d$, so $[\![C]\!]_{\xi'^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}} \in CP^{\vdash}_{\hookrightarrow}(D) \overset{\text{Lemma 2.35}}{=} CP^{\vdash}_{\hookrightarrow}([D]_d)$.

□

Definition 2.40 *If $\xi^{\vdash}_{\hookrightarrow} \models^{\Box} \Gamma$, then we call $\xi^{\vdash}_{\hookrightarrow}$ cute with respect to Γ if for all $d \in \Gamma$-$\mathtt{def}$ such that $\mathtt{subj}(d) \in V^{\Box}$, $\xi^{\vdash}_{\hookrightarrow}(\mathtt{subj}(d)) = [\![\mathtt{def}(d)]\!]_{\xi^{\vdash}_{\hookrightarrow\!\!\!\rightarrow}}$.*

Lemma 2.41

1. *If* $\xi^{\vdash}_{\rightarrow} \models^{\Box} \Gamma$ *and* A *is* Γ*-legal, then* $[\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *depends only on the values of* $\xi^{\vdash}_{\rightarrow}$ *on the free constructor variables of* A*.*

2. *If* $\xi^{\vdash}_{\rightarrow} \models^{\Box} \Gamma$ *then there is a cute* ${\xi'}_{\twoheadrightarrow}^{\vdash}$ *such that* ${\xi'}_{\twoheadrightarrow}^{\vdash} \models^{\Box} \Gamma$ *and* ${\xi'}_{\twoheadrightarrow}^{\vdash} = \xi^{\vdash}_{\rightarrow}$ *on the non-definitional constructor variables of* $dom(\Gamma)$*.*

3. *If* $\xi^{\vdash}_{\rightarrow} \models^{\Box} \Gamma$ *and* $\xi^{\vdash}_{\rightarrow}$ *is cute with respect to* Γ *then* $\Gamma \vdash A =_{\texttt{def}} B \Longrightarrow [\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$*.*

Proof: *1. is easy, 2. is a consequence of 1. and 3. is proved by induction on the generation of* $=_{\texttt{def}}$ *using Lemma 2.37.*

Definition 2.42

- *Let* $\xi^{\vdash}_{\rightarrow} \models^{\Box} \Gamma$ *such that* $\xi^{\vdash}_{\rightarrow}$ *is cute with respect to* Γ*. An object valuation of* Γ *with respect to* $\xi^{\vdash}_{\rightarrow}$*, notation* $\rho^{\vdash}_{\rightarrow}, \xi^{\vdash}_{\rightarrow} \models \Gamma$*, is a map* $\rho^{\vdash}_{\rightarrow} : V \rightarrow \Lambda$ *such that for all* $(A\lambda_x) \in' \Gamma$*:* $\rho^{\vdash}_{\rightarrow}(x) \in [\![A]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *(regardless of whether* $A \in \Gamma^{\vdash}$*-kinds or* $A \in \Gamma^{\vdash}$*-types).*

- *For* $\rho^{\vdash}_{\rightarrow}, \xi^{\vdash}_{\rightarrow} \models \Gamma$ *we define a map* $(\!|\ |\!)_{\rho^{\vdash}_{\twoheadrightarrow}} : \Gamma^{\vdash}$*-terms* $\longrightarrow \Lambda$ *as follows:*

$$\begin{array}{rcll}
(\!|x|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & := & \rho^{\vdash}_{\rightarrow}(x) & \\
(\!|*|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & := & * & \\
(\!|\Box|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & := & \Box & \\
(\!|(N\delta)M|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & := & ((\!|N|\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\!|M|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & \\
(\!|(A\lambda_x)B|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & := & ((\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\lambda_y)(\lambda_x)(\!|B|\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)} & \quad (\textit{where } y \notin FV(B)) \\
(\!|(A\Pi_x)B|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} & := & ((\lambda_x)(\!|B|\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)}\delta)((\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)x &
\end{array}$$

- *We define another map* $\lceil\ \rceil : \Gamma^{\vdash}$*-terms* $\longrightarrow \Lambda$ *by*

$$\begin{array}{rcll}
\lceil x \rceil & := & x & \\
\lceil * \rceil & := & * & \\
\lceil \Box \rceil & := & \Box & \\
\lceil (N\delta)M \rceil & := & (\lceil N \rceil\delta)\lceil M \rceil & \\
\lceil (A\lambda_x)B \rceil & := & (\lceil A \rceil\delta)(\lambda_y)(\lambda_x)\lceil B \rceil & \quad (\textit{where } y \notin FV(B)) \\
\lceil (A\Pi_x)B \rceil & := & ((\lambda_x)\lceil B \rceil\delta)(\lceil A \rceil\delta)x &
\end{array}$$

Definition 2.43 *Let* Γ *be a context,* $A, B \in \Gamma^{\vdash}$*-terms. We say that* Γ *satisfies that* A *is of type* B *with respect to* $\vdash$ *and* $\twoheadrightarrow$*, notation* $\Gamma \models^{\vdash}_{\rightarrow} A : B$*, iff*
$\forall \xi^{\vdash}_{\rightarrow}, \rho^{\vdash}_{\rightarrow}[\rho^{\vdash}_{\rightarrow}, \xi^{\vdash}_{\rightarrow} \models \Gamma \Rightarrow (\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}]$.

Lemma 2.44

1. *If* $\Gamma(A\delta)\underline{d}(B\lambda_x)\Delta$ *is a legal context and* $\rho^{\vdash}_{\rightarrow}, \xi^{\vdash}_{\rightarrow} \models \Gamma(A\delta)\underline{d}(B\lambda_x)\Delta$ *then* $(\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *and* $(\!|B|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in SAT_{\rightarrow}$*.*

2. $\Gamma d \models A : B \Longrightarrow \Gamma \models dA : [B]_d$

Proof:

1. *Induction on the derivation rules of* $\vdash$.

2. *Induction on* $\mathtt{weight}(d)$. *If* $d \equiv \emptyset$ *then nothing to prove, suppose now* $d \equiv (C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2$. *Then by the induction hypothesis* $\Gamma(C\delta)\overline{s}_1(D\lambda_x) \models \overline{s}_2 A : [B]_{\overline{s}_2}$.

   - *Suppose* $x \in V^*$. *Let* $\rho^{\vdash}_{\to}, \xi^{\vdash}_{\to} \models \Gamma\overline{s}_1$. *Then for all* $E \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *we have* $\rho^{\vdash}_{\to}(x := E), \xi^{\vdash}_{\to} \models \Gamma(C\delta)\overline{s}_1(D\lambda_x)$. *Hence* $(\![\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=E)} \in [\![[B]_{\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$, *hence* $(\lambda_x)(\![\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)} \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \to [\![[B]_{\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *and also* $((\![D]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\lambda_y)(\lambda_x)(\![\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)} \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \to [\![[B]_{\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *(by 1.* $(\![D]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in SAT_{\to}$, *use Lemma 2.33).*
     This means $\Gamma\overline{s}_1 \models (D\lambda_x)\overline{s}_2 A : (D\Pi_x)[B]_{\overline{s}_2}$, *so by the induction hypothesis* $\Gamma \models \overline{s}_1(D\lambda_x)\overline{s}_2 A : ([D]_{\overline{s}_1}\Pi_x)[B]_{\overline{s}_1\overline{s}_2}$. *If* $\rho^{\vdash}_{\to}, \xi^{\vdash}_{\to} \models \Gamma$ *then by 1.* $(\![C]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *and* $(\![\overline{s}_1(D\lambda_x)\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![[D]_{\overline{s}_1}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \to [\![[B]_{\overline{s}_1\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$, *hence* $((\![C]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\![\overline{s}1(D\lambda_x)\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![[B]_{\overline{s}_1\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = [\![[B]_{(C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$, *so* $\Gamma \models (C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2 A : [B]_{(C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2}$.
   - *Suppose* $x \in V^{\Box}$. *Let* $\rho^{\vdash}_{\to}, \xi^{\vdash}_{\to} \models \Gamma\overline{s}_1$. *Then* $\rho^{\vdash}_{\to}(x := E), \xi^{\vdash}_{\to}(x := f) \models \Gamma(C\delta)\overline{s}_1(D\lambda_x)$ *for all* $f \in \mathbf{CP}^{\vdash}_{\twoheadrightarrow}(D)$ *and* $E \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$, *so* $(\![\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=E)} \in [\![[B]_{\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}\xi^{\vdash}_{\to}(x := f)$, *hence* $(\lambda_x)(\![\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)} \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \to \bigcap_{f\in\mathbf{CP}^{\vdash}_{\twoheadrightarrow}(D)}[\![[B]_{\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)}$. *But then also (use 1. and Lemma 2.33)* $((\![D]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\lambda_y)(\lambda_x)(\![\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)} \in [\![D]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \to \bigcap_{f\in\mathbf{CP}^{\vdash}_{\twoheadrightarrow}(D)}[\![[B]_{\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)}$.
     Hence we see: $\Gamma\overline{s}_1 \models (D\lambda_x)\overline{s}_2 A : (D\Pi_x)[B]_{\overline{s}_2}$, *so by the induction hypothesis* $\Gamma \models \overline{s}_1(D\lambda_x)\overline{s}_2 A : ([D]_{\overline{s}_1}\Pi_x)[B]_{\overline{s}_1\overline{s}_2}$.
     Now let $\rho^{\vdash}_{\to}, \xi^{\vdash}_{\to} \models \Gamma$. *Then* $(\![\overline{s}_1(D\lambda_x)\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![([D]_{\overline{s}_1}\Pi_x)[B]_{\overline{s}_1\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *and* $(\![C]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![[D]_{\overline{s}_1}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ *by 1., so* $((\![C]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\rho^{\vdash}_{\to}\delta)((\![\overline{s}_1(D\lambda_x)\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\rho^{\vdash}_{\to}) \in \bigcap_{f\in\mathbf{CP}^{\vdash}_{\twoheadrightarrow}(D)}[\![[B]_{\overline{s}_1\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)}$.
     This means $(\![(C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2 A]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\rho^{\vdash}_{\to} \in [\![[B]_{\overline{s}_1\overline{s}_2}]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=[\![C]\!]_{\xi^{\vdash}_{\rightsquigarrow_\beta}})} = [\![[B]_{\overline{s}_1\overline{s}_2}[x := C]]\!]_{\xi^{\vdash}_{\rightsquigarrow_\beta}} \overset{VC}{=} [\![[B]_{(C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2}]\!]_{\xi^{\vdash}_{\rightsquigarrow_\beta}}$, *hence* $\Gamma \models (C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2 A : [B]_{(C\delta)\overline{s}_1(D\lambda_x)\overline{s}_2}$.

$\Box$

Lemma 2.45 *(*$(\![\ ]\!)_{\rho^{\vdash}_{\twoheadrightarrow}}$ *versus* $\lceil\ \rceil$*)*

1. *For all* $M \in \Gamma^{\vdash}$*-terms, for all* $\rho^{\vdash}_{\to}$*:* $(\![M]\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \equiv \lceil M\rceil[\vec{x} := \overrightarrow{\rho^{\vdash}_{\to}(x)}]$ *where* $\vec{x}$ *are the free variables of* M.

2. *If* $\overline{s}$ *is a well-balanced segment then* $\lceil\overline{s}A\rceil \equiv \lceil\overline{s}\rceil\lceil A\rceil$ *and* $\lceil\overline{s}\rceil$ *is also well-balanced. Moreover,* $FV(\lceil A\rceil) = FV(A)$.

3. *For all* $M \in \Gamma^{\vdash}$*-terms:* $\lceil M\rceil$ *is strongly normalising* $\Rightarrow M$ *is strongly normalising.*

Proof: *The first statement is easy to verify. The second statement is also easy. The third statement can be proved as follows: we prove by induction on the structure of* M, *that whenever* $M \twoheadrightarrow N$, *then* $\lceil M\rceil \twoheadrightarrow \lceil N\rceil$. *We show the only non-trivial case (note that when* $\twoheadrightarrow$ *is* $\twoheadrightarrow_\beta$, *then* $\overline{s} \equiv \emptyset$*).*

If $M \equiv (A\delta)\overline{s}(B\lambda_x)C \twoheadrightarrow \overline{s}(C[x := A]) \equiv N$,
then $\lceil M\rceil \equiv (\lceil A\rceil\delta)\lceil\overline{s}(B\lambda_x)C\rceil \equiv (\lceil A\rceil\delta)\lceil\overline{s}\rceil(\lceil B\rceil\delta)(\lambda_y)(\lambda_x)\lceil C\rceil$
$\twoheadrightarrow (\lceil A\rceil\delta)\lceil\overline{s}\rceil(\lambda_x)\lceil C\rceil$ *(note that* $y \notin FV((\lambda_x)\lceil C\rceil)$*)*
$\twoheadrightarrow \lceil\overline{s}\rceil\lceil C\rceil[x := \lceil A\rceil] \equiv \lceil\overline{s}\rceil\lceil C[x := A]\rceil \equiv \lceil\overline{s}(C[x := A])\rceil \equiv \lceil N\rceil$.

$\Box$

Lemma 2.46 $\Gamma \vdash A : B \Rightarrow \Gamma \models A : B$

Proof: *Use induction on the structure of A to prove that if $\rho^{\vdash}_{\rightarrow}, \xi^{\vdash}_{\rightarrow} \models \Gamma$ then $(\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$:*

- *$A \equiv x$. Then by the generation lemma for some B': $\Gamma \vdash B' =_{\texttt{def}} B$ and $(B'\lambda_x) \in \Gamma$-`decl`, so by $\rho^{\vdash}_{\rightarrow}, \xi^{\vdash}_{\rightarrow} \models \Gamma, (B'\lambda_x) \in \Gamma$-`decl`, and Lemma 2.37, we get $(\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} = \rho^{\vdash}_{\rightarrow}(x) \in [\![B']\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$.*

- *$A \equiv (P\lambda_x)Q$, with $P \in \Gamma^{\vdash}$-kinds.*

  Then by the generation lemma for some R, $\Gamma(P\lambda_x) \vdash Q : R$ with $\Gamma \vdash (P\Pi_x)R =_{\texttt{def}} B$, $\Gamma \vdash P : \Box$. By IH we find that $(\!|Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=p)} \in [\![R]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)}$ for all $p \in [\![P]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}, f \in CP^{\vdash}_{\rightarrow}(P)$, so $(\!|Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=p)} \in \bigcap_{f \in CP^{\vdash}_{\twoheadrightarrow}(P)} [\![R]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)}$. By IH also $(\!|P|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![\Box]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = SN_{\rightarrow}$ so by Lemma 2.33 $(\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} = (\!|(P\lambda_x)Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} = ((\!|P|\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\lambda_y)(\lambda_x)(\!|Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}(x:=x)} \in [\![P]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \rightarrow \bigcap_{f \in CP^{\vdash}_{\twoheadrightarrow}(P)} [\![R]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)} = [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$.

- *$A \equiv (P\lambda_x)Q$ with $P \in \Gamma^{\vdash}$-types. Then similar to the previous case.*

- *If $\vdash$ is ordinary typing and $A \equiv (P\delta)Q$ with $P \in \Gamma^{\vdash}$-objects. Then $\Gamma \vdash Q : (R\Pi_x)T$, $\Gamma \vdash P : R$ for some R, T with $\Gamma \vdash T[x := P] =_{\texttt{def}} B$ (again generation lemma). Now by IH and lemma 2.33 we see that $(\!|Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![R]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \longrightarrow [\![T]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$ and $(\!|P|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![R]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$,*

  so $(\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} = (\!|(P\delta)Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} = ((\!|P|\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)(\!|Q|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![T]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = [\![T[x := P]]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$.

- *$A \equiv dP$ where d is a definition. Then by the Generation Lemma $\Gamma d \vdash^{e} P : C$, $\Gamma d \vdash^{e} C =_{\texttt{def}} B$. By the induction hypothesis we then know that $(\!|P|\!)_{\rho^{\vdash sh}_{\twoheadrightarrow}} \in [\![C]\!]_{\xi^{\vdash sh}_{\twoheadrightarrow}}$. Now by Lemma 2.44 we get that also $(\!|dP|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![C]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$.*

- *$A \equiv (P\delta)Q$ with $P \in \Gamma^{\vdash}$-constructors where $(P\delta)$ is bachelor in $(P\delta)Q$ then also similar.*

- *$A \equiv (P\Pi_x)Q$. Then by generation $\Gamma \vdash P : S_1$, $\Gamma(P\lambda_x) \vdash Q : S_2$, $S_2 =_{\beta} B$.*

  If $P \in \Gamma^{\vdash}$-kinds, then IH says $(\!|P|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} \in [\![\Box]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$, $(\!|Q|\!)_{\rho(x:=p)} \in [\![S_2]\!]_{\xi^{\vdash}_{\twoheadrightarrow}(x:=f)}$ for all $p \in [\![P]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}, f \in CP(P)$, hence $[\![P]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} \in SN, (\lambda_x)(\!|Q|\!)_{\rho(x:=x)} \in SN$.

  But this means $(\!|A|\!)_{\rho^{\vdash}_{\twoheadrightarrow}} = ((\lambda_x)(\!|Q|\!)_{\rho(x:=x)}\delta)((\!|P|\!)_{\rho^{\vdash}_{\twoheadrightarrow}}\delta)x \in SN = [\![S_2]\!]_{\xi^{\vdash}_{\twoheadrightarrow}} = [\![B]\!]_{\xi^{\vdash}_{\twoheadrightarrow}}$.

  If $P \in \Gamma^{\vdash}$-types, then similar.

$\Box$

3 The ordinary typing relation and its properties

In the Cube as presented in [Barendregt 92], the only declarations allowed are of the form $(A\lambda_x)$. Hence there are no definitions. Therefore, $\Gamma \vdash_{\smile} d$ is of the form $\Gamma \vdash_{\smile} (A\lambda_x)$ and means that $\Gamma \vdash A : S$ for some S and that x is fresh in Γ, A. Moreover, for any $d \equiv (A\lambda_x)$, $\underline{d} \equiv \emptyset$, $\texttt{subj}(d) \equiv x$ and $\texttt{pred}(d) \equiv A$.

3.1 The typing relation

As the Cube is a generalisation of eight systems, the rules of the Cube are divided in two:

1. The general axioms and rules valid for all systems of the Cube.
2. The specific rules, which are specific to the various systems of the Cube. All these specific rules are parameterised Π-introduction rules.

Now the general rules are as follows:

Definition 3.1 *(General axioms and rules of the Cube)*

(axiom)	$<>\vdash * : \Box$
(start rule)	$\dfrac{\Gamma \vdash_{\smile} d}{\Gamma d \vdash \mathtt{subj}(d) : \mathtt{pred}(d)}$
(weakening rule)	$\dfrac{\Gamma \vdash_{\smile} d \qquad \Gamma \underline{d} \vdash D : E}{\Gamma d \vdash D : E}$
(application rule)	$\dfrac{\Gamma \vdash F : (A\Pi_x)B \qquad \Gamma \vdash a : A}{\Gamma \vdash (a\delta)F : B[x := a]}$
(abstraction rule)	$\dfrac{\Gamma(A\lambda_x) \vdash b : B \qquad \Gamma \vdash (A\Pi_x)B : S}{\Gamma \vdash (A\lambda_x)b : (A\Pi_x)B}$
(conversion rule)	$\dfrac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : S \qquad \Gamma \vdash B =_{\mathtt{def}} B'}{\Gamma \vdash A : B'}$

The specific rules are given by (S_1, S_2) rules which we sometimes refer to as formation rules:

Definition 3.2 *(The specific rules of the Cube)*

(S_1, S_2) *rule*	$\dfrac{\Gamma \vdash A : S_1 \qquad \Gamma(A\lambda_x) \vdash B : S_2}{\Gamma \vdash (A\Pi_x)B : S_2}$

The systems of the Cube are defined by taking the general rules plus a specific subset of the set $\{(*,*),(*,\Box),(\Box,*),(\Box,\Box)\}$ as (S_1, S_2) rules. In this Cube, there are eight systems of typed lambda calculus. They differ in whether $*$ and/or $\Box$ may be taken for S_1 and S_2 respectively in the above (S_1, S_2) rule. 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). The system with only $(*,*)$ for (S_1, S_2) is the system λ-Church or $\lambda_{\rightarrow}$ (this is essentially the Automath-system AUT-68). The addition of $(*,\Box)$ gives λP, which is a system that is rather close to another variant of the Automath-family, AUT-QE (see [de Bruijn 80]). The addition of $(\Box,*)$ to $\lambda_{\rightarrow}$ gives the second order typed lambda calculus, also called $\lambda 2$. Adding $(\Box,\Box)$ to $\lambda_{\rightarrow}$, we obtain $\lambda\underline{\omega}$. There are three systems that are defined by adding a combination of two of the three last-mentioned possibilities to $\lambda_{\rightarrow}$. When all mentioned (S_1, S_2)-combinations are permitted, we have a version of the Calculus of Constructions (λC) (see [CH 88]). Here is the table which presents the eight systems of the Cube:

System	*Set of specific rules*			
$\lambda_{\to}$	$(*,*)$			
$\lambda 2$	$(*,*)$	$(\Box,*)$		
λP	$(*,*)$		$(*,\Box)$	
$\lambda P2$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$
$\lambda P\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
$\lambda P\omega = \lambda C$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

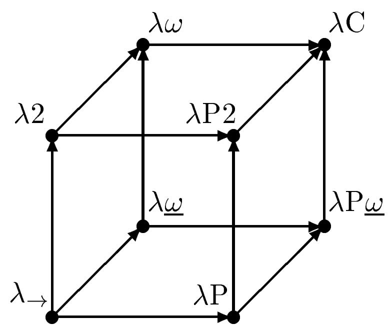

Figure 3: The Cube

Here are examples of typable terms in some systems of the Cube that we will use further on.

Example 3.3
1. $\vdash_{\lambda 2} (*\Pi_\alpha)(\alpha\Pi_y)\alpha : *$ as we have the rule $(\Box,*)$, but $\not\vdash_{\mathcal{L}} (*\Pi_\alpha)(\alpha\Pi_y)\alpha : \tau$ for any τ where $\mathcal{L} \in \{\lambda_{\to}, \lambda\underline{\omega}, \lambda P, \lambda P\underline{\omega}\}$.
2. $(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \beta$ can be seen by using the following

derivation steps and filling in the extra conditions:

$$\begin{array}{ll}
\vdash * : \Box & \\
(*\lambda_\beta) \vdash_{\lambda 2} \beta : * : \Box & \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} y' : \beta : * : \Box & \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha) \vdash_{\lambda 2} \alpha : * & \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha)(\alpha\lambda_y) \vdash_{\lambda 2} y : \alpha : * & \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha)(\alpha\lambda_y)(\alpha\lambda_x) \vdash_{\lambda 2} x : \alpha : * & \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha)(\alpha\lambda_y) \vdash_{\lambda 2} (\alpha\Pi_x)\alpha : * & (*,*) \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha)(\alpha\lambda_y) \vdash_{\lambda 2} (\alpha\lambda_x)x : (\alpha\Pi_x)\alpha : * & \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha)(\alpha\lambda_y) \vdash_{\lambda 2} (y\delta)(\alpha\lambda_x)x : \alpha & \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha) \vdash_{\lambda 2} (\alpha\Pi_y)\alpha : * & (*,*) \\
(*\lambda_\beta)(\beta\lambda_{y'})(*\lambda_\alpha) \vdash_{\lambda 2} (\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : (\alpha\Pi_y)\alpha : * & \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} (*\Pi_\alpha)(\alpha\Pi_y)\alpha : * & (\Box,*) \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} (*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : (*\Pi_\alpha)(\alpha\Pi_y)\alpha & \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : (\beta\Pi_y)\beta & \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2} (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \beta &
\end{array}$$

But If $\mathcal{L} \in \{\lambda_\rightarrow, \lambda\underline{\omega}\}$, then $(*\lambda_\beta)(\beta\lambda_{y'}) \not\vdash_{\mathcal{L}} (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \beta$. The reason is that the term of part 1 of this example is not typable in these systems. Note that when we introduce definitions in the Cube, the last 9 of the above steps will be replaced by a single one. See Example 6.4.
3. $(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N) \vdash_{\lambda P} (N\delta)(t\delta)(\sigma\lambda_x)((x\delta)Q\lambda_y)(y\delta)((x\delta)Q\lambda_Z)Z : (t\delta)Q$ but this derivation could not be obtained in $\lambda_\rightarrow$, $\lambda\underline{\omega}$ or $\lambda 2$ as we need the $(*,\Box)$ rule in order to derive that $(\sigma\Pi_q)* : \Box$ and hence that $((\sigma\Pi_q) * \lambda_Q)$ is allowed in the context.

3.2 Properties of the ordinary typing relation

Here we list the properties of the Cube without proofs. The reader can refer to [Barendregt 92] for details. These properties will be established for the Cube extended with term reshuffling, shuffle reduction and definition mechanisms. Now, here are the properties of the Cube that we will concentrate on.

Theorem 3.4 *(The Church Rosser Theorem for $\twoheadrightarrow_\beta$)*
If $A \twoheadrightarrow_\beta B$ and $A \twoheadrightarrow_\beta C$ then there exists D such that $B \twoheadrightarrow_\beta D$ and $C \twoheadrightarrow_\beta D$. □

Lemma 3.5 *(Free variable lemma for $\vdash$)*
Let Γ be a legal context such that $\Gamma \vdash B : C$. Then the following holds:

1. *If d and d' are two different elements of Γ-`decl`, then $\mathtt{subj}(d) \not\equiv \mathtt{subj}(d')$.*
2. *$FV(B), FV(C) \subseteq dom(\Gamma)$.*
3. *For s_1 a main item of Γ, $FV(s_1) \subseteq \{\mathtt{subj}(d) \mid d \in \Gamma\text{-}\mathtt{decl}, d \text{ is to the left of } s_1 \text{ in } \Gamma\}$.*

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

The following lemmas show that legal contexts behave as expected.

Lemma 3.6 *(Start Lemma for $\vdash$)*
Let Γ be a legal context. Then $\Gamma \vdash * : \Box$ and $\forall d \in' \Gamma[\Gamma \vdash d]$.

Proof: *As Γ is legal, then $\exists A, B \in \mathcal{T}$ such that $\Gamma \vdash A : B$. Now use induction on the derivation $\Gamma \vdash A : B$.* $\Box$

Lemma 3.7 *(Invitation Lemma for $\vdash$)*
If Γd is legal then $\Gamma \vdash_{\smile} d$.

Proof: *By induction on the derivation $\Gamma d \vdash A : B$.* $\Box$

Lemma 3.8 *(Transitivity Lemma for $\vdash$)*
Let Γ and Δ be legal contexts. Then: $[\Gamma \vdash \Delta \wedge \Delta \vdash A : B] \Rightarrow \Gamma \vdash A : B$.

Proof: *Induction on the derivation rules.* $\Box$

Lemma 3.9 *(Substitution Lemma for $\vdash$)*
Assume $\Gamma(A\lambda_x)\Delta \vdash B : C$ and $\Gamma \vdash D : A$ then $\Gamma(\Delta[x := D]) \vdash B[x := D] : C[x := D]$.

Proof: *By induction on the derivation rules.* $\Box$

Lemma 3.10 *(Thinning Lemma for $\vdash$)*
Let Γ and Δ be legal contexts such that $\Gamma \subseteq' \Delta$. Then $\Gamma \vdash A : B \Rightarrow \Delta \vdash A : B$

Proof: *By induction on the length of the derivation of $\Gamma \vdash A : B$.* $\Box$

Lemma 3.11 *(Generation Lemma for $\vdash$)*

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: *By induction on the derivation rules, using thinning lemma.* $\Box$

Corollary 3.12 *(Generation Corollary for $\vdash$)*

1. $\Gamma \vdash A : B \Rightarrow \exists S[B \equiv S$ *or* $\Gamma \vdash B : S]$

2. $\Gamma \vdash A : (B_1\Pi_x)B_2 \Rightarrow \exists S[\Gamma \vdash (B_1\Pi_x)B_2 : S]$

3. *If A is a $\Gamma^{\vdash}$-term, then A is $\Box$, a $\Gamma^{\vdash}$-kind or a $\Gamma^{\vdash}$-element.*

4. *If A is legal and B is a subexpression of A then B is legal.* $\Box$

Theorem 3.13 *(Subject Reduction for $\vdash$ and $\twoheadrightarrow_\beta$)*
$\Gamma \vdash A : B \wedge A \twoheadrightarrow_\beta A' \Rightarrow \Gamma \vdash A' : B$

Proof: $\Gamma \vdash A : B \wedge A \rightarrow_\beta A' \Rightarrow \Gamma \vdash A' : B$ *and* $\Gamma \vdash A : B \wedge \Gamma \rightarrow_\beta \Gamma' \Rightarrow \Gamma' \vdash A : B$, *where* $\Gamma \rightarrow_\beta \Gamma'$ *means* $\Gamma \equiv \Gamma_1(A\lambda_x)\Gamma_2$, $\Gamma' \equiv \Gamma_1(A'\lambda_x)\Gamma_2$ *and* $A \rightarrow_\beta A'$, *are proved simultaneously by induction on the derivation rules.* $\Box$

Corollary 3.14 *(SR Corollary for $\vdash$ and $\twoheadrightarrow_\beta$)*

1. *If $\Gamma \vdash A : B$ and $B \twoheadrightarrow_\beta B'$ then $\Gamma \vdash A : B'$.*
2. *If A is a $\Gamma^{\vdash}$-term and $A \twoheadrightarrow_\beta A'$ then A' is a $\Gamma^{\vdash}$-term.* □

Lemma 3.15 *(Unicity of Types for $\vdash$ and $\twoheadrightarrow_\beta$)*

1. $\Gamma \vdash A : B_1 \wedge \Gamma \vdash A : B_2 \Rightarrow B_1 =_\beta B_2$
2. $\Gamma \vdash A : B \wedge \Gamma \vdash A' : B' \wedge A =_\beta A' \Rightarrow B =_\beta B'$
3. $\Gamma \vdash B : S, B =_\beta B', \Gamma \vdash A' : B'$ *then* $\Gamma \vdash B' : S$.

Proof: *1. by induction on the structure of A, 2. by Church Rosser, Subject Reduction and 1, and 3. by Corollary 3.12, Subject Reduction and 1.* □

Theorem 3.16 *(Strong Normalisation with respect to $\vdash$ and $\twoheadrightarrow_\beta$)*
For all $\vdash$-legal terms M, M is strongly normalising with respect to $\twoheadrightarrow_\beta$. □

4 Term reshuffling

In this section we shall rewrite terms so that all the newly visible redexes (obtained as a result of our item notation), can be subject to the ordinary classical β-reduction $\twoheadrightarrow_\beta$. We shall show in this section that this term rewriting is correct in the sense that A is semantically equivalent to B in that $A =_\beta TS(A)$. Moreover, A and $TS(A)$ are procedurally equivalent.

Let us go back to the definition of $\delta\mathcal{O}$-couples. Recall that if $\overline{s} \equiv s_1 \cdots s_m$ for $m > 1$ where $s_1 s_m$ is a $\delta\mathcal{O}$-couple then $s_2 \cdots s_{m-1}$ is a well-balanced segment, s_1 is the δ-item of the $\delta\mathcal{O}$-couple and s_m is its $\mathcal{O}$-item. Now, we can move s_1 in $\overline{s}$ so that it occurs adjacently to s_m. That is, we may rewrite $\overline{s}$ as $s_2 \cdots s_{m-1} s_1 s_m$.

Example 4.1 The term $A \equiv (u\delta)(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(R\lambda_z)(z\delta)(y\delta)x$ is reshuffled to $TS(A) \equiv (w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(u\delta)(R\lambda_z)(z\delta)(y\delta)x$ by moving the item $(u\delta)$ to the right. Hence, we can rewrite (or *reshuffle*) a term so that all δ-items stand next to their matching $\mathcal{O}$-items. This means that we can contract redexes in any order. Such an action of reshuffling is not easy to describe in the classical notation. That is, it is difficult to describe how $((\lambda_{x:P}.(\lambda_{y:Q}.\lambda_{z:R}.xyz)u)w)u$ is reshuffled to $(\lambda_{x:P}.(\lambda_{y:Q}.(\lambda_{z:R}.xyz)u)v)w$.

Note furthermore that the shuffling is not problematic because we use the Barendregt Convention which means that no free variable will become unnecessarily bound after reshuffling due to the fact that names of bound and free variables are distinct.

Lemma 4.2 *If x° is a free occurrence of x in $s\,\overline{s_1}\,A$, then x° is free in $\overline{s_1}\,s\,A$.*

Proof: *By BC as λ_x does not occur in $s\,\overline{s_1}A$.* □

Example 4.3 Note that in Example 4.1, reshuffling does not affect the "meaning" of the term. In fact, in $A \equiv (u\delta)(w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(R\lambda_z)(z\delta)(y\delta)x$, none of the free variable of u can be captured by λ_x or λ_y. Moreover, A is equivalent, semantically and procedurally, to $TS(A) \equiv (w\delta)(P\lambda_x)(v\delta)(Q\lambda_y)(u\delta)(R\lambda_z)(z\delta)(y\delta)x$.

We call this process of moving δ-items of $\delta\mathcal{O}$-couples in a term to occupy positions adjacent to their $\mathcal{O}$-partners, *term reshuffling*. This term reshuffling should be such that *all* the δ-items of well-balanced segments in a term are shifted to the right until they meet their $\mathcal{O}$-partners. To do this however, we must study the classes of partnered and bachelor items in a term.

4.1 Partitioning the term into bachelor and well-balanced segments

With Definition 2.19, we may categorize the main items of a term A into different classes:

1. The "partnered" items (i.e. the δ- and $\mathcal{O}$-items which are partners, hence "coupled" to a matching one).

2. The "bachelors" (i.e. the bachelor $\mathcal{O}$-items and bachelor δ-items).

Lemma 4.4 *Let $\overline{s}$ be the body (NIET GEDEFINIEERD???) of a term A. Then the following holds:*

1. *Each bachelor main $\mathcal{O}$-item in $\overline{s}$ precedes each bachelor main δ-item in $\overline{s}$.*

2. *The removal from $\overline{s}$ of all bachelor main items, leaves behind a* well-balanced *segment.*

3. *The removal from $\overline{s}$ of all main $\delta\mathcal{O}$-couples, leaves behind a $\underbrace{\mathcal{O}\cdots\mathcal{O}}_{n}\underbrace{\delta\cdots\delta}_{m}$-segment, consisting of all bachelor main $\mathcal{O}$- and δ-items.*

Proof: *1 is by induction on* $\texttt{weight}(\overline{s'})$ *for* $\overline{s} \equiv \overline{s'}(B\mathcal{O}_x)\overline{s''}$ *and* $(B\mathcal{O}_x)$ *bachelor in* $\overline{s}$*. 2 and 3 are by induction on* $\texttt{weight}(\overline{s})$*.* $\square$

Note that we have assumed $\emptyset$ well-balanced. We assume it moreover non-bachelor.

Corollary 4.5 *For each non-empty segment $\overline{s}$, there is a unique* **partitioning** *in segments $\overline{s_0}, \overline{s_1}, \ldots, \overline{s_n}$, such that*

1. $\overline{s} \equiv \overline{s_0}\,\overline{s_1}\cdots\overline{s_n}$,

2. *For all $0 \le i \le n$, $\overline{s_i}$ is well-balanced for even i and $\overline{s_i}$ is bachelor in $\overline{s}$ for odd i.*

3. *For all $0 \le i, j \le n$: if $\overline{s_i}$ contains bachelor $\mathcal{O}$-items and $\overline{s_j}$ contains bachelor δ-items then $i \le j$.*

4. $\overline{s_{2n}} \not\equiv \emptyset$ *for* $n > 0$. $\square$

Example 4.6 $\overline{s} \equiv (A\lambda_x)(B\lambda_y)(C\delta)(D\Pi_z)(E\lambda_u)(F\delta)(a\delta)(b\delta)(c\lambda_v)(d\lambda_w)(e\delta)$ has the following partitioning:

- well-balanced segment $\overline{s_0} \equiv \emptyset$,
- bachelor segment $\overline{s_1} \equiv (A\lambda_x)(B\lambda_y)$,
- well-balanced segment $\overline{s_2} \equiv (C\delta)(D\Pi_z)$,
- bachelor segment $\overline{s_3} \equiv (E\lambda_u)(F\delta)$,
- well-balanced segment $\overline{s_4} \equiv (a\delta)(b\delta)(c\lambda_v)(d\lambda_w)$,
- bachelor segment $\overline{s_5} \equiv (e\delta)$.

4.2 A reshuffling procedure and its properties

In what follows, we use $\omega_1, \omega_2, \ldots$ to range over $\{\delta\} \cup \{\lambda_x; x \in V\} \cup \{\Pi_x; x \in V\}$.

Definition 4.7 *TS is defined recursively such that:*

$TS(\overline{s}x)$	$=_{df}$	$TS(\overline{s})x$	
$TS((A_1\omega_1)\cdots(A_n\omega_n))$	$=_{df}$	$(TS(A_1)\omega_1)\cdots(TS(A_n)\omega_n)$	*if $(A_1\omega_1)\cdots(A_n\omega_n)$ is bachelor*
$TS(\overline{s_0}\cdots\overline{s_n})$	$=_{df}$	$TS(\overline{s_0})\cdots TS(\overline{s_n})$	*If $\overline{s_0},\ldots,\overline{s_n}$ is the unique partitioning of Corollary 4.5*
$TS(\overline{s_1}\ldots\overline{s_n})$	$=_{df}$	$TS(\overline{s_1})\ldots TS(\overline{s_n})$	*if $\overline{s_i}$ is well-balanced*
$TS((A\delta)\overline{s}(B\lambda_x))$	$=_{df}$	$TS(\overline{s})(TS(A)\delta)(TS(B)\lambda_x)$	*if $\overline{s}$ is well-balanced*
$TS((A\delta)\overline{s}(B\Pi_x))$	$=_{df}$	$(TS(A)\delta)TS(\overline{s})(TS(B)\Pi_x)$	*if $\overline{s}$ is well-balanced*

Note that in this definition, we use $\overline{s}$ bachelor to mean $\overline{s}$ bachelor in $\overline{s}$.

The following lemma will be needed in the proofs:

Lemma 4.8

1. *If $\overline{s}$ contains no items which are partnered in A then $TS(\overline{s}A) \equiv TS(\overline{s})TS(A)$.*
2. *If $\overline{s}$ is bachelor in $\overline{s}A$ or is well-balanced, then $TS(\overline{s}A) \equiv TS(\overline{s})TS(A)$.*

Proof: *1: let $A \equiv \overline{s_0}\cdots\overline{s_n}v$ and $\overline{s} \equiv \overline{s'_0}\cdots\overline{s'_m}$ be partitionings. Use cases on $\overline{s_0}$ being empty or not and on $\overline{s'_m}$ being bachelor or well-balanced. 2: This is a corollary of 1.* □

The following lemma shows that $TS(A)$ changes all $\delta\lambda$-couples of A to $\delta\lambda$-segments.

Lemma 4.9 *For every term M, the following holds:*

1. *$TS(M)$ is well-defined.*
2. *If $\overline{s} \equiv (B\delta)\overline{s'}(C\lambda_x)$ is a segment occurring in M where $\overline{s'}$ is well-balanced, then $TS(\overline{s}) \equiv TS(\overline{s'})(TS(B)\delta)((TS(C)\lambda_x)$.*
3. *If $\overline{s} \equiv (A_1\omega_1)\cdots(A_n\omega_n)$ is a bachelor segment in M, then $TS(\overline{s}) \equiv (TS(A_1)\omega_1)\cdots(TS(A_n)\omega_n)$ is a bachelor subterm of $TS(A)$.*
4. *If $\overline{s}$ is a subsegment occurring in M which is well-balanced, then $TS(\overline{s})$ is well-balanced.*

Proof: *By induction on the structure of M.*

- *Case $M \equiv x$ then all $1\cdots 4$ hold.*
- *Case $M \equiv (B\omega)C$ where $(B\omega)$ bachelor in M. Then $M \equiv \overline{s}C'$ where $\overline{s}$ is of maximal weight and bachelor in M, and $TS(M) \equiv TS(\overline{s})TS(C)$ by Lemma 4.8(2) and $1\cdots 4$ hold by IH on $\overline{s}$ and C.*
- *Case $M \equiv (B\delta)\overline{s_1}(D\lambda_x)\overline{s_2}E$ where $\overline{s_1}$, $\overline{s_2}$ well-balanced, E a variable or starting with a bachelor item. Then by using Lemma 4.8, we see:*

$$\begin{aligned} TS(M) \equiv\ & TS((B\delta)\overline{s_1}(D\lambda_x)\overline{s_2}E) \\ \equiv\ & TS(\overline{s_1})(TS(B)\delta)(TS(D)\lambda_x)TS(\overline{s_2})TS(E) \end{aligned}$$

and again $1\cdots 4$ hold by IH.

- *Case* $M \equiv (B\delta)\overline{s_1}(D\Pi_x)\overline{s_2}E$ *where* $\overline{s_1}$, $\overline{s_2}$ *well-balanced,* E *a variable or starting with a bachelor item. Then by using Lemma 4.8, we see:*

$$\begin{aligned} TS(M) \equiv\ & TS((B\delta)\overline{s_1}(D\Pi_x)\overline{s_2}E) \\ \equiv\ & (TS(B)\delta)TS(\overline{s_1})(TS(D)\lambda_x)TS(\overline{s_2})TS(E) \end{aligned}$$

  and again $1 \cdots 4$ *hold by IH.* □

Note that it is not the case in general that $TS(A[x := B]) \equiv TS(A)[x := TS(B)]$. This can be seen by the following counterexample: let $A \equiv (y\delta)(y\delta)x$ and $B \equiv (z\lambda_u)(z\lambda_v)v$. Then $TS(A[x := B]) \equiv TS((y\delta)(y\delta)(z\lambda_u)(z\lambda_v)v) \equiv (y\delta)(z\lambda_u)(y\delta)(z\lambda_v)v$, whereas $TS(A)[x := TS(B)] \equiv (y\delta)(y\delta)(z\lambda_u)(z\lambda_v)v$.

Lemma 4.10 *For all variables* x *and terms* A, B *we have:*

1. $TS(A) \equiv TS(TS(A))$
2. $TS(A[x := B]) \equiv TS(TS(A)[x := TS(B)])$.

Proof:

1. *Induction on the structure of* A *like in the proof of Lemma 4.9.*
2. *Induction on the structure of* A*, use 1. in case* $A \equiv (C\mathcal{O}_y)D$ *where* $y \equiv x$.

□

Lemma 4.11 *For all pseudoterms* A *without* $\delta\Pi$*-couples:* $A =_\beta TS(A)$.

Proof: *by induction on the number of symbols in* A*:*

- $A \equiv x$*, nothing to prove.*
- $A \equiv (B\omega)C$*, where* $(B\omega)$ *is bachelor in* A*, then* $A \equiv (B\omega)C \stackrel{IH}{=}_\beta (TS(B)\omega)TS(C) \equiv TS((B\omega)C) \equiv TS(A)$.
- $A \equiv (B\delta)\overline{s}(C\lambda_x)D$, $\overline{s}$ *well-balanced, then* $\overline{s}$ *contains no* Π*-items and* $TS(A) \equiv^{Lemmas\ 4.8,\ 4.9}$ $TS(\overline{s})(TS(B)\delta)(TS(C)\lambda_x)TS(D) =_\beta^{IH} \overline{s}(B\delta)(C\lambda_x)D =_\beta^{Lemmas\ 2.14} (B\delta)\overline{s}(C\lambda_x)D \equiv A$, *as no binding variables of* $\overline{s}$ *are free in* B *by* VC □

Corollary 4.12 *For all pseudoterms* A, B *which do not contain partnered* Π*-items, we have:* $TS(A) =_\beta TS(B)$ *iff* $A =_\beta B$.

Proof: $A =_\beta TS(A) =_\beta TS(B) =_\beta B$. □

Corollary 4.13 *Let* B *contain no partnered* Π*-item. For all* $A \in [B]$*,* A *and* B *are semantically equivalent.*

Proof: $A \in [B]$ *implies* A *contains no partnered* Π*-items. As* $TS(A) \equiv TS(B)$*, then by Corollary 4.12,* $A =_\beta B$.

4.3 Another reshuffling procedure and its properties

The reshuffling that we introduced in Section 4.2, takes a term $\overline{s}x \equiv \overline{s_0}\,\overline{s_1}\ldots\overline{s_n}x$ according to the partitioning of Corollary 4.5 and reshuffles it to $\overline{s'_0}\,\overline{s'_1}\ldots\overline{s'_n}x$ where: $\overline{s'_1} \equiv TS(\overline{s_i})$ and if $\overline{s_i} \equiv (A_1\omega_1)\ldots(A_m\omega_m)$, is bachelor, then $\overline{s'_i} \equiv (TS(A_1)\omega_1)\ldots(TS(A_m)\omega_m)$, is bachelor and if $\overline{s_i}$ is well-balanced, then all the $\delta\lambda$-couples in $\overline{s_i}$ become $\delta\lambda$-segments. This means that the structure $\overline{s_0}\,\overline{s_1}\ldots\overline{s_n}$ does not change as the partitioning $\overline{s'_0}\,\overline{s'_1}\ldots\overline{s'_n}$ corresponds to $\overline{s_0}\,\overline{s_1}\ldots\overline{s_n}$.

Example 4.14 Let $\overline{s} \equiv (u\delta)(x\delta)(x\lambda_y)(y\lambda_z)$ and $\overline{s'} \equiv (x\delta)(x\lambda_y)(u\delta)(y\lambda_z)$. Now, $(u\lambda_x)(x\delta)\overline{s}(z\delta)z$ is reshuffled to $(u\lambda_x)(x\delta)\overline{s'}(z\delta)z$

It can be claimed however that $A \equiv (u\lambda_x)\overline{s}(x\delta)(z\delta)z$ and $B \equiv (u\lambda_x)(x\delta)\overline{s}(z\delta)z$ have the same meaning semantically and procedurally and that if $TS(A)$ is to create the class of all terms which are equivalent semantically and procedurally, then $TS(A)$ must be the same as $TS(B)$. For this, we refine TS as follows:

Definition 4.15 *TS is defined recursively such that:*

$TS(\overline{s}x)$	$=_{df}$	$TS(\overline{s})x$	
$TS((A\mathcal{O}_x)\overline{s})$	$=_{df}$	$(TS(A)\mathcal{O}_x)TS(\overline{s})$	*if $(A\mathcal{O}_x)$ is bachelor in $\overline{s}$*
$TS((A_1\delta)\ldots(A_n\delta)x)$	$=_{df}$	$(TS(A_1)\delta)\ldots(TS(A_n)\delta)$	
$TS((A_1\delta)\ldots(A_n\delta)\overline{s}B)$	$=_{df}$	$TS(\overline{s})TS((A_1\delta)\ldots(A_n\delta)B)$	*if $\overline{s}$ is well-balanced, $(A_i\delta)$ bachelor in B*
$TS((A\delta)\overline{s}(B\lambda_x))$	$=_{df}$	$TS(\overline{s})(TS(A)\delta)(TS(B)\lambda_x)$	*if $\overline{s}$ is well-balanced*
$TS((A\delta)\overline{s}(B\Pi_x))$	$=_{df}$	$(TS(A)\delta)TS(\overline{s})(TS(B)\Pi_x)$	*if $\overline{s}$ is well-balanced*

Now, $TS((u\lambda_x)(x\delta)\overline{s_2}(z\delta)z) \equiv (u\lambda_x)TS(\overline{s_2})(x\delta)(z\delta)z$. This means that $[A] = \{B \mid TS(A) \equiv TS(B)\}$ according to the reshuffling of this section contains more elements than according to the TS of Section 4.2. These extra terms should themselves belong to $[A]$.

Now we come to the point of whether we can also increase the class of those terms which are equivalent procedurally and semantically. With term reshuffling, well-balanced segments must only be rewritten so that $\delta\lambda$-couples become $\delta\lambda$-segments. Moreover, all bachelor main δ-items, are moved to the right of all well-balanced segements. Hence, for any A, $TS(A)$ becomes $\overline{s_0}\,\overline{s_1}x$ where $\overline{s_1}$ consists of all the bachelor main δ-items of A. $\overline{s_0}$ is of the form $\overline{s_2}\,\overline{s_3}\ldots\overline{s_n}$ where $\overline{s_i}$ is either a $\delta\lambda$-segment or a $\delta\Pi$-couple where all $\delta\lambda$-couples are $\delta\lambda$-segments or a bachelor main $\mathcal{O}$-item. Now, look for example at $P \equiv (A\lambda_z)(B\delta)(C\lambda_x)(D\lambda_y)(F\delta)(G\lambda_u)(H\delta)(I\delta)x$. The question one poses now is whether all the bachelor main $\mathcal{O}$-items can be moved to the left of all nonempty well-balanced segments. I.e. can we rewrite P above (assuming $A\ldots I$ are already reshuffled) as $P' \equiv (A\lambda_z)(D\lambda_y)(B\delta)(C\lambda_x)(F\delta)(G\lambda_u)(H\delta)(I\delta)x$? The answer is no as D may contain variables bound by the λ_x. So, we can't move main bachelor $\mathcal{O}$-items to the right, but can we move them to the left? The answer is again no. In P above, B and C may contain variables bound by λ_z so λ_z cannot move to the right of $(B\delta)(C\lambda_x)$. Hence, in a term A, all main bachelor $\mathcal{O}$-items will occur in the same position as in $TS(A)$.

In this paper we shall stick to the term reshuffling of this section as in Definition 4.15. Now, let us show the properties of this new TS.

Lemma 4.16

1. *For all pseudoterms M, $TS(M)$ is well defined.*
2. *If $\overline{s}$ is well-balanced, then $TS(\overline{s}A) \equiv TS(\overline{s})TS(A)$.*

Proof:

1. *Every time a rule $TS(M)$ is used, weights of the resulting terms become shorter or TS disappears.*

2. *By induction on $\overline{s}$.*

□

Lemma 4.17
If $\overline{s}$ is a well-balanced segment, then $TS(\overline{s})$ is well-balanced.
Proof: *By induction on* $\mathtt{weight}(\overline{s})$. □

Lemma 4.18 *For a term A, $TS(A) \equiv \overline{s_0}\,\overline{s_1}x$ where $x \equiv \mathtt{endvar}(A)$, $\overline{s_1}$ consists of the term reshufflings of all bachelor main δ-items of A and $\overline{s_0}$ is a sequence of term reshufflings of main $\delta\lambda$-segments and bachelor main $\mathcal{O}$-items.*
Proof: *Induction on* $\mathtt{weight}(A)$.

- $A \equiv x$, *then nothing to prove.*

- $A \equiv (B\mathcal{O}_x)C$ *or* $A \equiv \overline{s}C$ *where $\overline{s}$ well-balanced, use lemma 4.16 and IH on C.*

- $A \equiv (B_1\delta)\cdots(B_n\delta)\overline{s}C$ *where $\overline{s}$ well-balanced and ($\overline{s} \not\equiv \emptyset$ or ($C \equiv x$ and $n > 0$)). $TS(A) \equiv TS(\overline{s})TS((B_1\delta)\cdots(B_n\delta)C)$. By lemma 4.17, $\overline{s}$ is well-balanced and hence by IH on $\overline{s}$, $TS(\overline{s})$ is a sequence of $\delta\mathcal{O}$-segments, if $\overline{s} \not\equiv \emptyset$ then by IH on $(B_1\delta)\cdots(B_n\delta)C$ we are done, else $C \equiv x$ and by IH $TS((B_1\delta)\cdots(B_n\delta)C) \equiv (TS(B_1)\delta)\cdots(TS(B_n)\delta)x$ which also has the required format.*

□

Lemma 4.19 *For all pseudoterms A, B and variable x:*

1. $TS(A) \equiv TS(TS(A))$

2. $TS(A[x := B]) \equiv TS(TS(A)[x := TS(B)])$

Proof:

1. *induction on the structure of A.*
   - $A \equiv x$, *then* $A \equiv TS(A)$.
   - $A \equiv (B\mathcal{O}_x)C$, *use IH.*
   - $A \equiv (B_1\delta)\cdots(B_n\delta)\overline{s}C$ *where $\overline{s}$ well-balanced and ($\overline{s} \not\equiv \emptyset$ or ($C \equiv x$ and $n > 0$)), use lemma 4.16.2 and IH.*

2. *induction on the structure of A, use 1.*

□

Lemma 4.20 *For all pseudoterms A not containing partnered Π-items: $A =_\beta TS(A)$.*
Proof: *by induction on the number of symbols in A:*

- $A \equiv x$ *obvious.*
- $A \equiv (B_1\delta)\cdots(B_n\delta)x$, *then* $TS(A) \equiv (TS(B_1)\delta)\cdots(TS(B_n)\delta)x \stackrel{IH}{=}_\beta (B_1\delta)\cdots(B_n\delta)x \equiv A$.
- $A \equiv (B\mathcal{O}_x)C$, *then* $TS(A) \equiv (TS(B)\mathcal{O}_x)TS(C) \stackrel{IH}{=}_\beta (B\mathcal{O}_x)C \equiv A$.
- $A \equiv (B_1\delta)\cdots(B_n\delta)\overline{s}E$, *where* $n \geq 1$, $\overline{s} \not\equiv \emptyset$ *and* $\overline{s}$ *well-balanced. Then* $TS(A) \equiv TS((B_1\delta)\cdots(B_n\delta)\overline{s}E) \equiv TS(\overline{s})TS((B_1\delta)\cdots(B_n\delta)E) \stackrel{IH}{=}_\beta \overline{s}(B_1\delta)\cdots(B_n\delta)E$ ${}^{n \text{ times}} \stackrel{\text{Lemma 2.14}}{=}_\beta (B_1\delta)\cdots(B_n\delta)\overline{s}E \equiv A$ *(use VC).*
- $A \equiv (B\delta)\overline{s}(C\lambda_x)D$, *where* $\overline{s}$ *well-balanced, then* $TS(A) \stackrel{\text{Lemmas 4.16, 4.17}}{\equiv} TS(\overline{s})(TS(B)\delta)(TS(C)\lambda_x)TS(D) \stackrel{IH}{=}_\beta \overline{s}(B\delta)(C\lambda_x)D \stackrel{\text{Lemma 2.14}}{=}_\beta (B\delta)\overline{s}(C\lambda_x)D \equiv A$.

□

Corollary 4.21 *For all pseudoterms A, B not containing partnered Π-terms: if $TS(A) =_\beta TS(B)$ then $A =_\beta B$.* □

Lemma 4.22 *If A contains no partnered Π-items then for all $B \in [A]$, B contains no partnered Π-items.*

Lemma 4.23 *Let B contain no partnered Π-item. For all $A \in [B]$, A and B are semantically and procedurally equivalent.*

Proof: *$A \in [B]$ implies A contains no partnered Π-items. As $TS(A) \equiv TS(B)$, then by Corollary 4.12, $A =_\beta B$.*

Lemma 4.24 *for all $A \in [B]$, A and B are semantically and procedurally equivalent.*

Proof: *By induction on the number of symbols in A:*

- $A \equiv x$ *nothing to prove.*
- $A \equiv (C\omega)D$ *where* $\omega \equiv \mathcal{O}_x$ *or* $\omega \equiv \delta$ *and* $(C\delta)$ *is bachelor, then* $B \equiv (C'\omega)D'$ *where* $C \in [C']$ *and* $D \in [D']$.
  - *Case r is in C then by IH,* $\exists r' \in C'$ *such that* $r_C =_\beta r'_{C'}$. *Hence* $r_A \equiv r_C =_\beta r'_{C'} \equiv r'_B$.
  - *Case r is in D then by IH* $\exists r' \in D'$ *such that* $r_D = |ber'_{D'}$. *Hence* $r_A \equiv (C\omega)r_D =_\beta^{Corollary\ 4.12} (C'\omega)r'_{D'} \equiv r_B$.
- *If* $A \equiv (C\delta)\overline{S}(D\mathcal{O}_x)E$ *then* $B \equiv \overline{s_1}(C'\delta)\overline{s_2}(D'\mathcal{O}_x)E'$ *where* $TS((C\delta)\overline{s}(D\mathcal{O}_x)) \equiv TS(\overline{s_1}(C'\delta)\overline{s_2}(D'\mathcal{O}_x))$. *The only case worth considering is if* $r \equiv (C\delta)(D\mathcal{O}_x)$. *The other cases have been dealt with above. Take* $r' \equiv (C'\delta)(D'\mathcal{O}_x)$. *Now,* $r_A \equiv A$ *and* $r_B \equiv B$ *and by Corollary 4.12,* $A =_\beta B$. *Hence, we are done.*

□

5 Shuffle reduction

Let us recall that to reduce A, we reshuffle it and then use ordinary β-reduction. As we see, $A \leadsto_\beta A'$ for any $A' \in \{B; TS(A) \to_\beta B\}$.

Definition 5.1 *(Extended redexes and β-reduction in item notation)*
In the item notation of the λ-calculus, an extended redex is of the form $(C\delta)\overline{s}(B\lambda_x)A$ where $\overline{s}$ is well-balanced not containing partnered Π-items. The shuffle class of a term A is $[A] = \{A' \mid TS(A) \equiv TS(A')\}$. General one-step β-reduction $\leadsto_\beta$ is the least compatible relation generated out of the following axiom:

$$(C\delta)\overline{s}(B\lambda_x)A \leadsto_\beta TS(\overline{s})(TS(A)[x := TS(C)])$$

In other words,

$$(C\delta)\overline{s}(B\lambda_x)A \leadsto_\beta B \text{ iff } TS((C\delta)\overline{s}(B\lambda_x)A) \to_\beta B$$

Note that $\leadsto_\beta$ is compatible and transitive because $\to_\beta$ is. General $\twoheadrightarrow\!\!\!\!\leadsto_\beta$ is the reflexive and transitive closure of $\leadsto_\beta$ and $\approx_\beta$ is the least equivalence relation generated by $\twoheadrightarrow\!\!\!\!\leadsto_\beta$.

Remark 5.2 Now it is not in general true that $A \twoheadrightarrow\!\!\!\!\leadsto_\beta B \Rightarrow \exists A' \in [A] \exists B' \in [B][A' \twoheadrightarrow_\beta B']$. This can be seen by the following counterexample:

Let $A \equiv ((\alpha\lambda_u)(\alpha\lambda_v)v\delta)((\alpha\Pi_u)(\alpha\Pi_v)\alpha\lambda_x)(w\delta)(w\delta)x$ and $B \equiv (w\delta)(\alpha\lambda_u)w$. Now $A \leadsto_\beta (w\delta)(w\delta)(\alpha\lambda_u)(\alpha\lambda_v)v \leadsto_\beta B$,

but $[A] = \{A, (w\delta)((\alpha\lambda_u)(\alpha\lambda_v)v\delta)((\alpha\Pi_u)(\alpha\Pi_v)\alpha\lambda_x)(w\delta)x, (w\delta)(w\delta)((\alpha\lambda_u)(\alpha\lambda_v)v\delta)((\alpha\Pi_u)(\alpha\Pi_v)\alpha\lambda_x)x\}$, $[B] = \{B\}$ and if $A' \in [A]$ then the only $\to_\beta$ reduct of A' is $(w\delta)(w\delta)(\alpha\lambda_u)(\alpha\lambda_v)v$, which doesn't $\to_\beta$-reduce to B.

Lemma 5.3 *Let A be a pseudoterm which does not contain partnered Π-items. If $A \to_\beta B$ then for all $A' \in [A]$ $A' \leadsto_\beta B$.*

Proof: *It is sufficient to prove $(A\delta)(B\lambda_x)C \to_\beta C[x := B]$ and $TS(A') \equiv (A\delta)(B\lambda_x)C$ then $A \leadsto_\beta C[x := B]$. The compatibility cases are easy.* □

Lemma 5.4 *If C is an extended redex in A, then C is a classical redex in $TS(A)$, and if $(B\delta)(C\lambda_x)$ is a classical redex in $TS(A)$ then there exist terms B',C' such that $TS(B') \equiv B$, $TS(C') \equiv C$ and $(B'\delta)(C'\lambda_x)$ is an extended redex in A.*

Proof: *by induction on the number of symbols in A:*

- *$A \equiv (B_1\delta)\cdots(B_n\delta)x$, then extended redexes of A are extended redexes of B_i for some i, use IH on B_i. As $TS(A) \equiv (TS(B_1)\delta)\cdots(TS(B_n)\delta)x$, classical redexes in $TS(A)$ are in one of $TS(B_i)$, use IH on B_i.*
- *$A \equiv (B_1\mathcal{O}_x)B_2$, then similar to the previous case.*
- *$A \equiv (D\delta)(E\lambda_x)F$. then the extended redex $(D\delta)(E\lambda_x)$ in A corresponds to the classical redex $(TS(D)\delta)(TS(E)\lambda_x)$ in $TS(A)$, for extended redexes in D,E or F use IH, for classical redexes in $TS(D)$, $TS(E)$ or $TS(F)$, use IH.*
- *$A \equiv (D_1\delta)\cdots(D_n\delta)(D\delta)(E\lambda_x)F$, then $TS(A) \equiv (TS(D)\delta)(TS(E)\lambda_x)TS((D_1\delta)\cdots(D_n\delta)F)$. Now extended redexes of A are either in $(D_1\delta)\cdots(D_n\delta)F$ or in $(D\delta)(E\lambda_x)$, use IH on these terms. Classical redexes in $TS(A)$ are either in $(TS(D)\delta)(TS(E)\lambda_x)x$ or in $TS((D_1\delta)\cdots(D_n\delta)F)$, so use IH on these terms, noting that an extended redex in $(D_1\delta)\cdots(D_n\delta)F$ is also an extended redex in $(D_1\delta)\cdots(D_n\delta)(D\delta)(E\lambda_x)F$.*

$\square$

Lemma 5.5 *Let $A, B \in \mathcal{T}$. If $A \to_\beta B$ in the sense of Definition 2.11, then $A \rightsquigarrow_\beta B$ in the sense of Definition 5.1. Moreover, if $A \rightsquigarrow_\beta B$ comes from contracting a $\delta\lambda$-segment then $A \to_\beta B$.*

Proof: *easy induction on the structure of A.* $\square$

Lemma 5.6 *If $A' \in [A]$ then $A' =_\beta A$.*

Proof: *See Corollary 4.12* $\square$

Lemma 5.7 *Let A, B have no partnered Π-items. If $A \rightsquigarrow_\beta B$ then $A =_\beta B$.*

Proof: *If $A \rightsquigarrow_\beta B$ then $TS(A) \to_\beta B$. But by Lemma 4.20, $A =_\beta TS(A) =_\beta B =_\beta TS(B)$.* $\square$

Corollary 5.8

1. *If $A \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta B$ then $A =_\beta B$.*
2. *$A \approx_\beta B$ iff $A =_\beta B$.*

$\square$

This Corollary is important. It shows the typing relation of Section 3 does not change as a result of the conversion rule.

Theorem 5.9 *(The general Church Rosser theorem for $\rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta$)*
If $A \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta B$ and $A \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta C$, then there exists D such that $B \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta D$ and $C \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta D$.

Proof: *As $A \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta B$ and $A \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta C$ then by Corollary 5.8, $A =_\beta B$ and $A =_\beta C$. Hence, $B =_\beta C$ and by the Church Rosser property for the classical lambda calculus, there exists D such that $B \twoheadrightarrow_\beta D$ and $C \twoheadrightarrow_\beta D$. But, $A \twoheadrightarrow_\beta B$ implies $A \rightsquigarrow\!\!\!\!\!\twoheadrightarrow_\beta B$. Hence the Church-Rosser theorem holds for the general β-reduction.* $\square$

Note that our example in subsection 4.2, can be easily adapted to an example showing the following: if $A \to_\beta B$ and if all the $\delta\lambda$-couples in A are $\delta\lambda$-segments, then it is not necessary that all the $\delta\lambda$-couples of B are $\delta\lambda$-segments. In other words, we can have $TS(C) \to_\beta D$ where $D \not\equiv TS(D)$. Consider for example the terms $C \equiv ((z\lambda_u)(z\lambda_v)v\delta)(w\lambda_x)(y\delta)(y\delta)x$ and $D \equiv (y\delta)(y\delta)(z\lambda_u)(z\lambda_v)v$. Then $TS(A) \equiv C \to_\beta D$ whereas $TS(D) \equiv (y\delta)(z\lambda_u)(y\delta)(z\lambda_v)v$. But we still can show that in a certain sense, term reshuffling preserves β-reduction.

Lemma 5.10 *If $A, B \in \mathcal{T}$ and $A \rightsquigarrow_\beta B$ then $(\exists B' \in [B])[TS(A) \to_\beta B']$. In other words, the following diagram commutes:*

$$\begin{tikzcd} A \arrow[r, rightsquigarrow, "\beta"] \arrow[d, "TS"'] & B \arrow[d] \\ TS(A) \arrow[r, "\beta"] & B' \in [B] \end{tikzcd}$$

Proof: *we prove with induction to the structure of A' that if $A' \to_\beta B' \in [B]$, then for some B'', $TS(A') \to_\beta B'' \in [B]$.*

- *$A' \equiv x$, then nothing to prove.*
- *$A' \equiv (C\omega)D$, where $(C\omega)$ bachelor in A'. Assume $B' \equiv (C'\omega)D$, the case $B' \equiv (C\omega)D'$ is similar. Then by IH there is C'' such that $TS(C) \to_\beta C'' \in [C']$.*

  Now $TS(A') \equiv (TS(C)\omega)TS(D) \to_\beta (C''\omega)TS(D)$, and $TS((C''\omega)TS(D)) \equiv (TS(C'')\omega)TS(TS(D)) \overset{\text{Lemma 4.16}}{\equiv} (TS(C'')\omega)TS(D) \equiv TS((C'\omega)D) \in [B]$.
- *$A' \equiv (C\delta)\overline{s}(D\mathcal{O}_x)E$, where $\overline{s}$ well-balanced. If $B' \equiv (C'\delta)\overline{s}(D'\mathcal{O}_x)E'$ then similar to the previous case, assume now $B' \equiv \overline{s}E[x := C]$. Then $TS(A') \equiv TS(\overline{s})(TS(C)\delta)(TS(D)\mathcal{O}_x)TS(E)$ $\to_\beta TS(\overline{s})(TS(E)[x := TS(C)])$, and $TS(TS(\overline{s})(TS(E)[x := TS(C)])) \equiv TS(TS(\overline{s}))TS(TS(E)[x := TS(C)]) \overset{\text{Lemma 4.16}}{\equiv} TS(\overline{s})TS(E[x := C]) \equiv TS(B') \in [B]$.*

$\square$

Corollary 5.11 *If $A \rightsquigarrow\!\!\!\!\rightarrow_\beta B$ then there exist $A_0, A_1, \dots, A_n$ such that* $[(A \equiv A_0) \wedge (TS(A_0) \to_\beta A_1) \wedge (TS(A_1) \to_\beta A_2) \wedge \cdots \wedge (TS(A_{n-1}) \to_\beta A_n) \wedge (TS(A_n) \equiv TS(B))]$

Proof: $\square$

5.1 Properties of ordinary typing with generalised reduction

If we look at Section 3.2 and because $=_\beta$ and $\approx_\beta$ are equivalent according to Lemma 5.8, we see that the only lemmas/theorems affected by our extension of reductions are those which have $\twoheadrightarrow_\beta$ in their heading. Hence, the only (very important) properties that get affected by $\rightsquigarrow\!\!\!\!\rightarrow_\beta$ are: Church Rosser (Theorem 3.4), Subject Reduction (Theorem 3.13) and its Corollary 3.14, Unicity of Types (Lemma 3.15) and Strong Normalisation (Theorem 3.16). In this section, we shall show that Church Rosser and Strong Normalisation hold for the Cube with generalised reduction. We shall moreover show that Subject Reduction holds for $\lambda\underline{\omega}$ and $\lambda_\to$ but not for any of the other six systems. Unicity of typing depends on SR and on the fact that $=_\beta$ is the same as $\rightsquigarrow\!\!\!\!\rightarrow_\beta$. Hence, we ignore it here as once we prove SR, its proof will be exactly that of Lemma 3.15.

Now we come to the proof of Strong Normalisation for the Cube with extended reduction. Those familiar with the proof of Strong Normalisation of the Cube, will notice that we have accommodated $\rightsquigarrow\!\!\!\!\rightarrow_\beta$ in the definition of $SN_{\rightsquigarrow_\beta}$ (recall Section 2.3).

Remark 5.12 With Definition **??**, it becomes clear why we depart from [Geuvers 94] by using $\lceil (A\lambda_x)B \rceil$ to be $(\lceil A\rceil\delta)(\lambda_y)(\lambda_x)\lceil B\rceil$ instead of $(\lceil A\rceil\delta)((\lambda_x)\lceil B\rceil\delta)(\lambda_u)(\lambda_v)u$.

Consider for example $P \equiv (A\delta)(B\delta)(C\lambda_x)(D\lambda_y)E$ and $Q \equiv (B\delta)(C\lambda_x)E[y := A]$. It is obvious that $P \rightsquigarrow_\beta Q$ and that $\lceil P\rceil \equiv (\lceil A\rceil\delta)(\lceil B\rceil\delta)(\lceil C\rceil\delta)(\lambda_p)(\lambda_x)(\lceil D\rceil)\delta)(\lambda_q)(\lambda_y)\lceil E\rceil \rightsquigarrow\!\!\!\!\rightarrow_\beta \lceil Q\rceil \equiv (\lceil B\rceil\delta)(\lceil C\rceil\delta)(\lambda_p)(\lambda_x)\lceil E\rceil[y := \lceil A\rceil]$. Yet, if we use the translation of [Geuvers 94], then we get $\lceil P\rceil \equiv (\lceil A\rceil\delta)(\lceil B\rceil\delta)(\lceil C\rceil\delta)((\lambda_x)\lceil (D\lambda_y)E\rceil\delta)(\lambda_u)(\lambda_v)u$ $\not\rightsquigarrow\!\!\!\!\rightarrow_\beta \lceil Q\rceil \equiv (\lceil B\rceil\delta)(\lceil C\rceil\delta)((\lambda_x)\lceil E\rceil[y := \lceil A\rceil]\delta)(\lambda_s)(\lambda_t)s$.

Theorem 5.13 *(Strong Normalisation with respect to $\vdash$ and $\rightsquigarrow\!\!\!\!\rightarrow_\beta$)*
For all $\vdash$-legal terms M, M is strongly normalising with respect to $\rightsquigarrow\!\!\!\!\rightarrow_\beta$.

Proof: *Let M be a legal term. Then either $M \equiv \Box$ or for some context Γ and term N, $\Gamma \vdash M : N$.*

In the first case, clearly M is strongly normalising.

In the second case, define canonical elements $c^A \in CP^{\vdash}_{\rightsquigarrow_\beta}(A)$ for all $A \in \Gamma^{\vdash}$-kinds as follows:

$$\begin{array}{lll} c^* & := & SN_{\rightsquigarrow_\beta} \\ c^{(A\Pi_x)B} & := & \boldsymbol{\lambda} f \in CP^{\vdash}_{\rightsquigarrow_\beta}(A).c^B \quad \text{if } A \in \Gamma^{\vdash}\text{-kinds} \\ c^{(A\Pi_x)B} & := & c^B \qquad\qquad\qquad\qquad\; \text{if } A \in \Gamma^{\vdash}\text{-types} \end{array}$$

Take $\xi^{\vdash}_{\rightsquigarrow_\beta}$ such that $\xi^{\vdash}_{\rightsquigarrow_\beta}(x) = c^A$ whenever $(A\lambda_x) \in' \Gamma$ and take $\rho^{\vdash}_{\rightsquigarrow_\beta} = id$.

Then $\rho^{\vdash}_{\rightsquigarrow_\beta}, \xi^{\vdash}_{\rightsquigarrow_\beta} \models \Gamma$, hence $(\!|M|\!)_{\rho^{\vdash}_{\rightsquigarrow_\beta}} \in [\![N]\!]_{\xi^{\vdash}_{\rightsquigarrow_\beta}}$, where $(\!|M|\!)_{\rho^{\vdash}_{\rightsquigarrow_\beta}} = \lceil M \rceil$ as mentioned in lemma 2.45. Hence $\lceil M \rceil \in [\![N]\!]_{\xi^{\vdash}_{\rightsquigarrow_\beta}} \subseteq SN_{\rightsquigarrow_\beta}$. By lemma 2.45 now also $M \in SN_{\rightsquigarrow}$. $\Box$

Hence, up to now, almost all the properties of the Cube hold when reduction is generalised. The only exception is Subject Reduction. Here we show that it holds for $\lambda\underline{\omega}$ and $\lambda_{\rightarrow}$, yet fails for $\lambda 2$. In the following, $\mathcal{L}$ stands for one of the systems $\lambda\underline{\omega}$, $\lambda_{\rightarrow}$.

Lemma 5.14 *If $\Gamma \vdash_{\mathcal{L}} A : \Box$ then $A \in \{*, (*\Pi_x)*, (*\Pi_x)(*\Pi_y)*, ((*\Pi_x) * \Pi_y)*, \ldots\}$.*

Proof: *By induction on the derivation rules.* $\Box$

Lemma 5.15 *If B is a legal $\mathcal{L}$-term, B' is a $\mathcal{L}$-kind and $B =_\beta B'$ then B is a kind.*

Proof: *First show by induction on the derivations: If $*$ is a subterm of A and A is legal then A is a kind or $*$ is type-information in A (as in $(*\lambda_x)y$). Now, as B' is a kind, B' is in normal form, hence $B \rightarrow_\beta B'$ and it can easily be seen using the former result that B must be a kind too.* $\Box$

Lemma 5.16 *If $\Gamma \vdash_{\mathcal{L}} (A\Pi_x)B : S$, then $\Gamma \vdash_{\mathcal{L}} A : S$, $\Gamma(A\lambda_x) \vdash_{\mathcal{L}} B : S$ and $x \notin FV(B)$.*

Proof: *Show by induction on the derivation of $\Gamma \vdash_{\mathcal{L}} A : B$ that if B a kind, then for all $(C\lambda_{x^*}) \in \Gamma$-`decl`, $x^* \notin FV(A)$.*

- *application rule: $\Gamma \vdash_{\mathcal{L}} (a\delta)F : B[x := a]$ out of $\Gamma \vdash_{\mathcal{L}} F : (A\Pi_x)B$ and $\Gamma \vdash_{\mathcal{L}} a : A$. Suppose $B[x := a]$ is a kind and $(C\lambda_y) \in' \Gamma$, $\Gamma \vdash_{\mathcal{L}} C : *$. If $x \notin FV(B)$ then B is a kind, so A and $(A\Pi_x)B$ are kinds too, hence $y \notin FV(a), FV(F)$ by the induction hypothesis.*

  If $x \in FV(B)$ then a is a kind (as $B[x := a]$ is a kind) and hence $A \equiv \Box$ which is impossible as $\Gamma \vdash_{\mathcal{L}} F : (A\Pi_x)B$.

- *conversion rule: $\Gamma \vdash_{\mathcal{L}} A : B'$ out of $\Gamma \vdash_{\mathcal{L}} A : B$, $\Gamma \vdash_{\mathcal{L}} B' : S$, $B =_\beta B'$. Suppose B' is a kind, then by lemma 5.15: B is a kind, hence by induction hypothesis we are done.*

- *the other cases are easy.* $\Box$

Lemma 5.17

1. *$\Gamma \vdash (A\delta)B : C \Rightarrow \Gamma \vdash C : S$ for some sort S.*

2. *If $\Gamma \vdash_{\mathcal{L}} A : S_1, \Gamma \vdash_{\mathcal{L}} B : S_2$ and $A =_\beta B$ then $S_1 \equiv S_2$.*

Proof:

1. *Generation Lemma gives*

$$\begin{aligned} \Gamma &\vdash A : D \\ \Gamma &\vdash B : (D\Pi_x)E \\ E[x := A] &=_\beta C \\ \textit{if } E[x := A] \not\equiv C \textit{ then } \Gamma &\vdash C : S \end{aligned}$$

So suppose $E[x := A] \equiv C$, *then* $\Gamma \vdash B : (D\Pi_x)E$ *implies by lemma 5.16 that* $\Gamma \vdash E : S, x \notin FV(E)$ *hence* $C \equiv E$ *and we are done.*

2. *Note that* $S_1 \equiv \Box$ *or* $S_2 \equiv \Box$, *hence by Lemma 5.15,* $S_1 \equiv S_2$.

$\Box$

The crucial step in the proof of Subject Reduction in $\lambda\underline{\omega}$ and $\lambda_\to$ will be proved in the following 'shuffle'-lemma:

Lemma 5.18 *(Shuffle Lemma for* $\lambda\underline{\omega}$ *and* $\lambda_\to$*)*
$\Gamma \vdash_{\mathcal{L}} \overline{s}_1(A\delta)\overline{s}_2 B : C \iff \Gamma \vdash_{\mathcal{L}} \overline{s}_1\overline{s}_2(A\delta)B : C$ *where* $\overline{s}_2$ *is well-balanced and the binding variables in* $\overline{s}_2$ *are not free in* A.

Proof: *By induction on* $\texttt{weight}(\overline{s}_2)$.

- *Case* $\texttt{weight}(\overline{s}_2) = 0$ *then nothing to prove.*
- *Case* $\texttt{weight}(\overline{s}_2) = 2$, *say* $\overline{s}_2 \equiv (D\delta)(E\lambda_x)$. *We use induction on* $\texttt{weight}(\overline{s}_1)$. *Suppose first,* $\texttt{weight}(\overline{s}_1) = 0$.

  $\Rightarrow$) *suppose* $\Gamma \vdash_{\mathcal{L}} (A\delta)(D\delta)(E\lambda_x)B : C$

  Using the Generation Lemma three times, we obtain:

$$\begin{aligned} \Gamma &\vdash_{\mathcal{L}} A : F && (1) \\ \Gamma &\vdash_{\mathcal{L}} (D\delta)(E\lambda_x)B : (F\Pi_y)G && (2) \\ G \equiv G[y := A] &=_\beta C \quad (Lemma\ 5.16, Corollary\ 3.12) && (3) \\ \Gamma &\vdash_{\mathcal{L}} D : H && (4) \\ \Gamma &\vdash_{\mathcal{L}} (E\lambda_x)B : (H\Pi_z)I && \\ I \equiv I[z := D] &=_\beta (F\Pi_y)G \quad (Lemma\ 5.16, Corollary\ 3.12) && (5) \\ \Gamma &\vdash_{\mathcal{L}} (E\Pi_x)J : S_1 && \\ \Gamma(E\lambda_x) &\vdash_{\mathcal{L}} B : J && (6) \\ (H\Pi_z)I &=_\beta (E\Pi_x)J && (7) \end{aligned}$$

  Out of (7) and Lemma 5.16 we see that $x \equiv z$, $H =_\beta E$, $I =_\beta J$, $y \notin FV(G), x \notin FV(I) \cup FV(J)$, $\quad \Gamma \vdash_{\mathcal{L}} F, G, H, I, E : S_1$ *(8)*

  and out of (7) and (5): $J =_\beta (F\Pi_y)G$. *Hence* *(9)*

$$\Gamma(E\lambda_x) \vdash_{\mathcal{L}} B : (F\Pi_y)G \qquad \textit{(conversion, (6), (9), (8) implies by the generation and thinning} \qquad (10)$$

$$
\begin{array}{rcr}
 & \textit{lemmas: } \Gamma(E\lambda_x) \vdash_{\mathcal{L}} (F\Pi_y)G : S_1\textit{)} & \\
\Gamma(E\lambda_x) \vdash_{\mathcal{L}} A : F & \textit{(thinning lemma, (1))} & (11) \\
\Gamma(E\lambda_x) \vdash_{\mathcal{L}} (A\delta)B : G & \textit{((10), (11), application, } G[y := A] \equiv G\textit{)} & (12) \\
\Gamma \vdash_{\mathcal{L}} (H\Pi_x)G, (E\Pi_x)G : S_1 & \textit{(formation, thinning, } \Gamma \vdash_{\mathcal{L}} H, G, E : S_1\textit{)} & (13) \\
\Gamma \vdash_{\mathcal{L}} (E\lambda_x)(A\delta)B : (H\Pi_x)G & \textit{((12), (13), abstraction, conversion,} & (14) \\
 & (8) \Rightarrow (E\Pi_x)G =_\beta (H\Pi_x)G) & \\
\Gamma \vdash_{\mathcal{L}} (D\delta)(E\lambda_x)(A\delta)B : G & \textit{((14), application, (4), } G[x := D] \equiv G\textit{)} & (15) \\
\Gamma \vdash_{\mathcal{L}} C : S & \textit{(Lemma 5.17, hypothesis)} & (16) \\
\Gamma \vdash_{\mathcal{L}} (D\delta)(E\lambda_x)(A\delta)B : C & \textit{(conversion, (15), (16), (3))} & (17) \\
 & & (18)
\end{array}
$$

$\Leftarrow$) *Suppose* $\Gamma \vdash_{\mathcal{L}} (D\delta)(E\lambda_x)(A\delta)B : C$

$$
\begin{array}{llll}
\textit{Then} & \Gamma \vdash_{\mathcal{L}} C : S_1 & \textit{(Lemma 5.17)} & (19) \\
\textit{and by} & \textit{generation three times we get:} & & \\
 & \Gamma \vdash_{\mathcal{L}} D : F & & (20) \\
 & \Gamma \vdash_{\mathcal{L}} (E\lambda_x)(A\delta)B : (F\Pi_y)G & & \\
 & G \equiv G[y := D] =_\beta C & (Lemma\ 5.16, Corollary\ 3.12) & (21) \\
 & \Gamma \vdash_{\mathcal{L}} (E\Pi_x)H : S_2 & & (22) \\
 & \Gamma(E\lambda_x) \vdash_{\mathcal{L}} (A\delta)B : H & & \\
 & (E\Pi_x)H =_\beta (F\Pi_y)G & & (23) \\
 & \Gamma(E\lambda_x) \vdash_{\mathcal{L}} A : I & & (24) \\
 & \Gamma(E\lambda_x) \vdash_{\mathcal{L}} B : (I\Pi_z)J & & (25) \\
 & J \equiv J[z := A] =_\beta H & (Lemma\ 5.16, Corollary\ 3.12) & (26)
\end{array}
$$

Now (25) and Corollary 3.12 imply that for some S_3, $\Gamma(E\lambda_x) \vdash_{\mathcal{L}} (I\Pi_z)J : S_3$.

Hence, by Lemma 5.16, $z \notin FV(J), \Gamma(E\lambda_x) \vdash_{\mathcal{L}} J : S_3$.

Also, by Lemma 5.16, we get out of (22) that $\Gamma \vdash_{\mathcal{L}} E : S_2, \Gamma(E\lambda_x) \vdash_{\mathcal{L}} H : S_2$ *and* $x \notin FV(H)$.

Now, $J =_\beta H$ *from (26), hence* $x \notin FV(J)$.

Moreover, by Lemma 5.17, we see $S_2 \equiv S_3$. *Hence,*

$$
\begin{array}{rcr}
\Gamma \vdash_{\mathcal{L}} (E\Pi_x)(I\Pi_z)J : S_2 & \textit{formation} & (27) \\
\Gamma \vdash_{\mathcal{L}} (E\lambda_x)B : (E\Pi_x)(I\Pi_z)J & \textit{((27), (25), abstraction)} & (28) \\
\Gamma \vdash_{\mathcal{L}} (D\delta)(E\lambda_x)B : (I\Pi_z)J & \textit{(application, (28), } x \notin FV(I, J)\textit{)} & \\
 & \Gamma \vdash_{\mathcal{L}} D : E \textit{ because (23)} & (29) \\
 & \textit{implies } E =_\beta F & \\
 & \textit{and we use conversion, (20), } \Gamma \vdash_{\mathcal{L}} E : S_2\textit{)} & \\
\Gamma \vdash_{\mathcal{L}} (A\delta)(D\delta)(E\lambda_x)B : J & \textit{(out of } \Gamma(E\lambda_x) \vdash_{\mathcal{L}} A : I \textit{ and } \Gamma \vdash D : E & (30) \\
 & \textit{we find by substitution (} x \notin FV(A, I)\textit{),} &
\end{array}
$$

$$\Gamma \vdash_{\mathcal{L}} (A\delta)(D\delta)(E\lambda_x)B : C$$

$\Gamma \vdash_{\mathcal{L}} A : I$. Now, use application) ((30), (conversion; $C =_\beta J$ follows from (26), (23) and (21))

Now suppose $\texttt{weight}(\overline{s}_1) = n + 1$.

Using the generation lemma we obtain $\Gamma' \vdash_{\mathcal{L}} \overline{s}'_1(A\delta)\overline{s}_2B : C'$, *where* $\texttt{weight}(\overline{s}'_1) = n$, *hence the induction hypothesis says* $\Gamma' \vdash_{\mathcal{L}} \overline{s}'_1\overline{s}_2(A\delta)B : C'$ *and by applying the appropriate derivation rule we obtain* $\Gamma \vdash_{\mathcal{L}} \overline{s}_1\overline{s}_2(A\delta)B : C$.

- *case* $\texttt{weight}(\overline{s}_2) = 2(n+1), n \geq 1$.

  Then $\overline{s}_2 \equiv (D\delta)\overline{s}_3(E\lambda_x)\overline{s}_4$ *for some terms* C, D, *variable* x *and well-balanced segments* $\overline{s}_3, \overline{s}_4$. *Then,* $\texttt{weight}(\overline{s}_3), \texttt{weight}(\overline{s}_4) \leq 2n$ *and we see:*

  $\Gamma \vdash_{\mathcal{L}} \overline{s}_1(A\delta)(D\delta)\overline{s}_3(E\lambda_x)\overline{s}_4B : C \overset{I.H.}{\Longleftrightarrow}$

  $\Gamma \vdash_{\mathcal{L}} \overline{s}_1(A\delta)\overline{s}_3(D\delta)(E\lambda_x)\overline{s}_4B : C \overset{I.H.}{\Longleftrightarrow}$

  $\Gamma \vdash_{\mathcal{L}} \overline{s}_1\overline{s}_3(A\delta)(D\delta)(E\lambda_x)\overline{s}_4B : C \overset{I.H.}{\Longleftrightarrow}$

  $\Gamma \vdash_{\mathcal{L}} \overline{s}_1\overline{s}_3(D\delta)(E\lambda_x)(A\delta)\overline{s}_4B : C \overset{I.H.}{\Longleftrightarrow}$

  $\Gamma \vdash_{\mathcal{L}} \overline{s}_1(D\delta)\overline{s}_3(E\lambda_x)(A\delta)\overline{s}_4B : C \overset{I.H.}{\Longleftrightarrow}$

  $\Gamma \vdash_{\mathcal{L}} \overline{s}_1(D\delta)\overline{s}_3(E\lambda_x)\overline{s}_4(A\delta)B : C$ □

Now we can prove Subject Reduction for generalised β-reduction.

Theorem 5.19 (Generalised Subject Reduction for $\lambda\underline{\omega}$ and $\lambda_\to$ for $\vdash$ and $\rightsquigarrow_\beta$) *If* $\Gamma \vdash_{\mathcal{L}} A : B$ *and* $A \rightsquigarrow_\beta A'$ *then* $\Gamma \vdash_{\mathcal{L}} A' : B$.

Proof: *We prove by simultaneous induction on the generation of* $\Gamma \vdash_{\mathcal{L}} A : B$ *that*

$$\begin{array}{lcll} \Gamma \vdash_{\mathcal{L}} A : B \wedge A \rightsquigarrow_\beta A' & \Rightarrow & \Gamma \vdash_{\mathcal{L}} A' : B & (i) \\ \Gamma \vdash_{\mathcal{L}} A : B \wedge \Gamma \rightsquigarrow_\beta \Gamma' & \Rightarrow & \Gamma' \vdash_{\mathcal{L}} A : B & (ii) \end{array}$$

where $\Gamma \rightsquigarrow_\beta \Gamma'$ *means* $\Gamma \equiv \Gamma_1(A\lambda_x)\Gamma_2, \Gamma' \equiv \Gamma_1(A'\lambda_x)\Gamma_2$ *and* $A \rightsquigarrow_\beta A'$ *for some* $\Gamma_1, \Gamma_2, A, A', x$. *The cases in which the last rule applied is axiom, start, weakening or conversion are easy (for start: use conversion). We treat the three other cases.*

- *The last rule applied is the formation rule:* $\Gamma \vdash_{\mathcal{L}} (A_1\Pi_x)B_1 : S_1$ *is a direct consequence of* $\Gamma \vdash_{\mathcal{L}} A_1 : S_1$ *and* $\Gamma(A_1\lambda_x) \vdash_{\mathcal{L}} B_1 : S_1$. *Now (i) follows from IH(i) and IH(ii); (ii) follows from IH(ii).*

- *The last rule applied is the abstraction rule: similar to the previous case.*

- *The last rule applied is the application rule:* $\Gamma \vdash_{\mathcal{L}} (a\delta)F : B_1[x := a]$ *is a direct consequence of* $\Gamma \vdash_{\mathcal{L}} F : (A_1\Pi_x)B_1$ *and* $\Gamma \vdash_{\mathcal{L}} a : A_1$. *Now (ii) follows from IH(ii). We consider various cases:*

  - *Subcase 1:* $(a\delta)F \rightsquigarrow_\beta (a\delta)F'$ *because* $F \rightsquigarrow_\beta F'$. *Then (i) follows from IH(i).*

– *Subcase 2:* $(a\delta)F \rightsquigarrow_\beta (a'\delta)F$ *because* $a \rightsquigarrow_\beta a'$. *Then from IH(i) and application, it follows that* $\Gamma \vdash (a'\delta)F : B_1[x := a']$. *Moreover, from Corollary 3.12, it follows that for some sort* S_1: $\Gamma \vdash_{\mathcal{L}} (A_1\Pi_x)B_1 : S_1$ *and hence by the generation lemma:* $\Gamma(A\lambda_x) \vdash_{\mathcal{L}} B_1 : S_1$ *and thus by the substitution lemma* $\Gamma \vdash_{\mathcal{L}} B_1[x := a] : S_1$. *Now conversion gives* $\Gamma \vdash_{\mathcal{L}} (a'\delta)F : B_1[x := a]$ *which proves (i).*

– *Subcase 3:* $F \equiv \overline{s}(A'\lambda_y)F'$, $\overline{s}$ *well-balanced and* $(a\delta)F \rightsquigarrow_\beta \overline{s}F'[y := a]$. *Now, by lemma 5.18 we have* $\Gamma \vdash_{\mathcal{L}} \overline{s}(a\delta)(A'\lambda_y)F' : B_1[x := a]$ *and* $\overline{s}(a\delta)(A'\lambda_y)F' \to_\beta \overline{s}F'[y := a]$ *so by subject reduction for ordinary* β*-reduction we have:*
$\Gamma \vdash_{\mathcal{L}} \overline{s}F'[y := a] : B_1[x := a]$ *which proves (i).* □

Hence SR is valid for $\lambda_\to$ and $\lambda\underline{\omega}$. It is not however valid for the remaining six systems of the Cube as the following examples show:

Example 5.20 *(SR does not hold in* $\lambda 2$ *using* $\rightsquigarrow\!\!\!\!\rightarrow_\beta$*)*
$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash_{\lambda 2}$ ($\not\vdash_{\mathcal{L}}$ for $\mathcal{L} \in \{\lambda_\to, \lambda\underline{\omega}\}$) $(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \beta$ (see Example 3.3).
Moreover, $(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x \rightsquigarrow_\beta (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x$.
Yet, $(*\lambda_\beta)(\beta\lambda_{y'}) \not\vdash_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \beta$.
Even, $(*\lambda_\beta)(\beta\lambda_{y'}) \not\vdash_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \tau$ for any τ.
The reason why this really fails is that $(\alpha\lambda_x)x : (\alpha\Pi_x)\alpha$ and $y : \beta$ yet α and β are unrelated and hence we fail in firing the application rule to find the type of $(y'\delta)(\alpha\lambda_x)x$. If one looks 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. We will redefine the rules of the Cube so that such information can be taken into account. Finally note that failure of SR in $\lambda 2$, means its failure in $\lambda P2, \lambda\omega$ and λC

Example 5.21 *(SR does not hold in* λP *using* $\rightsquigarrow\!\!\!\!\rightarrow_\beta$*)*
$(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q)*\lambda_Q)((t\delta)Q\lambda_N) \vdash_{\lambda P} (N\delta)(t\delta)(\sigma\lambda_x)((x\delta)Q\lambda_y)(y\delta)((x\delta)Q\lambda_Z)Z : (t\delta)Q$. Note here that this cannot be derived in $\lambda_\to$, $\lambda 2$ or $\lambda\underline{\omega}$ (see Example 3.3).
And $(N\delta)(t\delta)(\sigma\lambda_x)((x\delta)Q\lambda_y)(y\delta)((x\delta)Q\lambda_Z)Z \rightsquigarrow_\beta (t\delta)(\sigma\lambda_x)(N\delta)((x\delta)Q\lambda_Z)Z$
Now, $N : (t\delta)Q, t : \sigma, y : (x\delta)Q, x : \sigma, (t\delta)Q \neq (x\delta)Q$.
$(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N) \not\vdash_{\lambda P} (t\delta)(\sigma\lambda_x)(N\delta)((x\delta)Q\lambda_Z)Z : \tau$ for any τ.
Here again the reason of failure is similar to the above example. At one stage, we need to match $(x\delta)Q$ with $(t\delta)Q$ 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

6 Extending the Cube with definition mechanisms

As a first step in the direction of including extended reduction in the systems of the Cube, we now investigate adding definitions to the Cube. We already defined what definitions are like in contexts, now we shall extend the derivation rules so that we can use definitions in the context. The rules remain unchanged except for the addition of one rule, the (*def rule*), and that the use of $\Gamma \vdash B =_{\mathtt{def}} B'$ in the conversion rule really has an effect now, rather than simply postulating $B =_\beta B'$.

6.1 The definition mechanisms and extended typing

Definition 6.1 *(General axioms and rules of the Cube extended with definitions)*

(axiom) $$<> \vdash^{sh} * : \Box$$

(start rule) $$\frac{\Gamma \vdash^{sh}_{\smile} d}{\Gamma d \vdash^{sh} \mathtt{subj}(d) : \mathtt{pred}(d)}$$

(weakening rule) $$\frac{\Gamma \vdash^{sh}_{\smile} d \qquad \Gamma \underline{d} \vdash^{sh} D : E}{\Gamma d \vdash^{sh} D : E}$$

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

(abstraction rule) $$\frac{\Gamma(A\lambda_x) \vdash^{sh} b : B \qquad \Gamma \vdash^{sh} (A\Pi_x)B : S}{\Gamma \vdash^{sh} (A\lambda_x)b : (A\Pi_x)B}$$

(def rule) $$\frac{\Gamma d \vdash^{sh} C : D}{\Gamma \vdash^{sh} dC : [D]_d} \text{ if } d \text{ is a definition}$$

(conversion rule) $$\frac{\Gamma \vdash^{sh} A : B \qquad \Gamma \vdash^{sh} B' : S \qquad \Gamma \vdash^{sh} B =_{\mathtt{def}} B'}{\Gamma \vdash^{sh} A : B'}$$

Definition 6.2 *(The specific rules of the Cube)*

(S_1, S_2) *rule* $$\frac{\Gamma \vdash^{sh} A : S_1 \qquad \Gamma(A\lambda_x) \vdash^{sh} B : S_2}{\Gamma \vdash^{sh} (A\Pi_x)B : S_2}$$

Remark 6.3 Note that in the abstraction rule, it follows that $(A\lambda_x)$ is bachelor in $\Gamma(A\lambda_x)$. The reason is that we can show that if Γ is legal then Γ contains no bachelor main δ-items. Hence as $\Gamma \vdash^{\mathrm{sh}} (A\Pi_x)B : S$, Γ has no bachelor δ-items and so $(A\lambda_x)$ cannot be matched in Γ.

The (*def rule*) says that if $C : D$ can be deduced from a concatenation of definitions d, then dC will be of type D where all the sub-definitions in d have been unfolded in D. Note that the *(def rule)* does global substitution in the predicate of all the occurrences of subjects in d. The reason is that d no longer remains in the context. In the conversion rule however, substitution is local as Γ keeps all its information (see Definition 2.22). The following examples show how this works:

Example 6.4 With this definition, let us show how the term in Example 3.3 is typed in $\lambda 2$ and how its $\leadsto_\beta$-contractum of Example 5.20 is given the same type too.
$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathrm{sh}}_{\lambda 2} (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \beta$ can be seen by using the following deriva-

tion steps and filling in the needed conditions:

$$\begin{array}{l}
\vdash^{\mathrm{sh}}_{\lambda 2} * : \Box \\
(*\lambda_\beta) \vdash^{\mathrm{sh}}_{\lambda 2} \beta : * : \Box \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathrm{sh}}_{\lambda 2} y' : \beta : * : \Box \\
(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathrm{sh}}_{\lambda 2} y' : \beta : * : \Box, \alpha : * \\
(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathrm{sh}}_{\lambda 2} \alpha =_{\mathtt{def}} \beta \\
(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathrm{sh}}_{\lambda 2} y' : \alpha : * \\
(*\lambda_\beta)(\beta\lambda_{y'})(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y) \vdash^{\mathrm{sh}}_{\lambda 2} y : \alpha : * \\
(*\lambda_\beta)(\beta\lambda_{y'})(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x) \vdash^{\mathrm{sh}}_{\lambda 2} x : \alpha \\
(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathrm{sh}}_{\lambda 2} (y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x : \alpha[x := y][y := y'][\alpha := \beta] \equiv \beta
\end{array}$$

Note how much quicker we can type terms here once we have a context. Note also that the other derivation given in Example 3.3 of this term is also valid here. Yet it is more clear and efficient to use the definitional segments $(y\delta)(\alpha\lambda_x)$ and $(y'\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_y)$, and furthermore we see that this derivation is even valid in the system $\lambda_\to$, because we don't need the term $(*\lambda_\alpha)(\alpha\lambda_y)(y\delta)(\alpha\lambda_x)x$ to have a type due to the (*def rule*).

Now, also $(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathrm{sh}}_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \beta$ as follows (needed derivation steps, including $(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha) \vdash^{\mathrm{sh}}_{\lambda_2} y' : \alpha$ by *(conversion)* , are left to the reader):

$(*\lambda_\beta)(\beta\lambda_{y'})(\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x) \vdash^{\mathrm{sh}}_{\lambda 2} x : \alpha$ so by (*def rule*):
$(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\mathrm{sh}}_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \alpha[x := y'][\alpha := \beta] \equiv \beta$

Example 6.5 Also the term of Example 5.20 can be easily and quickly typed in λP (note that this term cannot be typed in $\lambda_\to$ as the term Q can't):

$$(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N)(N\delta)(t\delta)(\sigma\lambda_x)((x\delta)Q\lambda_y)(y\delta)((x\delta)Q\lambda_Z) \vdash^{\mathrm{sh}}_{\lambda P} Z : (x\delta)Q$$
$$(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N) \vdash^{\mathrm{sh}}_{\lambda P} (N\delta)(t\delta)(\sigma\lambda_x)((x\delta)Q\lambda_y)(y\delta)((x\delta)Q\lambda_Z)Z : (t\delta)Q$$

Its $\rightsquigarrow_\beta$-contractum gets the same type as follows:

$$(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N)(t\delta)(\sigma\lambda_x)(N\delta)((x\delta)Q\lambda_Z) \vdash^{\mathrm{sh}}_{\lambda P} Z : (x\delta)Q$$
$$(*\lambda_\sigma)(\sigma\lambda_t)((\sigma\Pi_q) * \lambda_Q)((t\delta)Q\lambda_N) \vdash^{\mathrm{sh}}_{\lambda P} (t\delta)(\sigma\lambda_x)(N\delta)((x\delta)Q\lambda_Z)Z : (t\delta)Q$$

Remark 6.6 It might be asked why we need $\Gamma \vdash^{\mathrm{sh}} A =_{\mathtt{def}} B$ instead of $A =_\beta B$ in the conversion rule? The reason is that we want from $(*\lambda_A)(A\delta)(*\lambda_x) \vdash^{\mathrm{sh}} A : *$ and y is fresh to derive not only $(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} y : A$
but also $(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} y : x$.
This is not possible if conversion is left with $B =_\beta B'$:
how can we ever derive $(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} y : x$ as $x \neq_\beta A$?
If we change to the conversion rule using $=_{\mathtt{def}}$, then we are fine:
$(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} y : A$
$(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} x : *$
$(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} x =_{\mathtt{def}} A$ and so with conversion,
$(*\lambda_A)(A\delta)(*\lambda_x)(A\lambda_y) \vdash^{\mathrm{sh}} y : x$

6.2 Properties of the Cube with definitions

If we look at Section 3.2 and because we have changed $\vdash$ to $\vdash^{\mathrm{sh}}$ but left $\twoheadrightarrow_\beta$ unchanged, we see that all the lemmas and theorems which had $\vdash$ in their heading get affected. In this section, we will list these lemmas and theorems for $\vdash^{\mathrm{sh}}$ and give their proofs.

Lemma 6.7 *(Free variable lemma for $\vdash^{sh}$)*
Let Γ be a legal context such that $\Gamma \vdash^{sh} B : C$. Then the following holds:

1. *If d and d' are two different elements of Γ-$\mathtt{decl}$, then $\mathtt{subj}(d) \not\equiv \mathtt{subj}(d')$.*
2. *$FV(B), FV(C) \subseteq dom(\Gamma)$.*
3. *For s_1 a main item of Γ, $FV(s_1) \subseteq \{\mathtt{subj}(d) \mid d \in \Gamma\text{-}\mathtt{decl}, d \text{ is to the left of } s_1 \text{ in } \Gamma\}$.*

Proof: *All by induction on the derivation of $\Gamma \vdash^{sh} B : C$.* $\square$

The following lemmas show that legal contexts behave as expected.

Lemma 6.8 *(Start Lemma for $\vdash^{sh}$)*
Let Γ be a legal context. Then $\Gamma \vdash^{sh} * : \square$ and $\forall d \in' \Gamma[\Gamma \vdash^{sh} d]$.

Proof: *As Γ is legal, then $\exists B, C \in \mathcal{T}$ such that $\Gamma \vdash^{sh} B : C$. Now use induction on the derivation $\Gamma \vdash^{sh} B : C$.* $\square$

Lemma 6.9 *(Invitation Lemma for $\vdash^{sh}$)*
If Γd is legal then $\Gamma \vdash^{sh}_{\smile} d$.

Proof: *By induction on the derivation $\Gamma d \vdash^{sh} A : B$.* $\square$

Lemma 6.10 *(Transitivity Lemma for $\vdash^{sh}$)*
Let Γ and Δ be legal contexts. Then: $[\Gamma \vdash^{sh} \Delta \wedge \Delta \vdash^{sh} A : B] \Rightarrow \Gamma \vdash^{sh} A : B$.

Proof: *Induction on the derivation $\Delta \vdash^{sh} A : B$.* $\square$

Lemma 6.11 *(Definition-shuffling for $\vdash^{sh}$)*

1. *If $\Gamma d \Delta \vdash^{sh} C =_{\mathtt{def}} D$ then $\Gamma \underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \vdash^{sh} C =_{\mathtt{def}} D$ for d a definition.*
2. *If $\Gamma d \Delta \vdash^{sh} C : D$ then $\Gamma \underline{d}(\mathtt{def}(d)\delta)(\mathtt{pred}(d)\lambda_{\mathtt{subj}(d)})\Delta \vdash^{sh} C : D$ for d a definition.*

Proof: *1. is by induction on the generation of $\Gamma(A\delta)\overline{s}(B\lambda_x)\Delta \vdash^{sh} C =_{\mathtt{def}} D$. 2. is by induction on the proof of $\Gamma(A\delta)\overline{s}(B\lambda_x)\Delta \vdash^{sh} C : D$ using 1. for conversion.* $\square$

Lemma 6.12 *(Thinning for $\vdash^{sh}$)*

1. *If $\Gamma_1\Gamma_2 \vdash^{sh} A =_{\mathtt{def}} B$, $\Gamma_1\Delta\Gamma_2$ is a legal context, then $\Gamma_1\Delta\Gamma_2 \vdash^{sh} A =_{\mathtt{def}} B$.*
2. *If Γ and Δ are legal contexts such that $\Gamma \subseteq' \Delta$ and if $\Gamma \vdash^{sh} A : B$, then $\Delta \vdash^{sh} A : B$.*

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

- *If* $\Gamma\Delta \vdash^{sh} A : B$, $\Gamma \vdash^{sh} C : S$, *x is fresh, and no λ-item in Δ is bound by a δ-item in Γ, then also* $\Gamma(C\lambda_x)\Delta \vdash^{sh} A : B$. *We show this by induction on the derivation* $\Gamma\Delta \vdash^{sh} A : B$ *using 1. for conversion.*

- *If* $\Gamma\overline{s}\Delta \vdash^{sh} A : B$, $\Gamma\overline{s} \vdash^{sh} C : D : S$, $[C]_{\overline{s}} \equiv C$, *$x$ is fresh, $\overline{s}$ is well-balanced, then also* $\Gamma(C\delta)\overline{s}(D\lambda_x)\Delta \vdash^{sh} A : B$. *We show this by induction on the derivation* $\Gamma\overline{s}\Delta \vdash^{sh} A : B$. *In the case of (start) where* $\Gamma(A\delta)\overline{s}(B\lambda_x) \vdash^{sh} x : A$ *comes from* $\Gamma\overline{s} \vdash^{sh} A : B : S$, $[A]_{\overline{s}} \equiv A$, *$x$ fresh, then* $[A]_{(C\delta)\overline{s}(D\lambda_x)} \equiv A$ *because x fresh and* $\Gamma(C\delta)\overline{s}(D\lambda_x) \vdash^{sh} A : B : S$ *by IH.*

- *If* $\Gamma\overline{s}(A\lambda_x)\Delta \vdash^{sh} B : C, (A\lambda_x)$ *bachelor, $\overline{s}$ well-balanced,* $\Gamma\overline{s} \vdash^{sh} D : A, [D]_{\overline{s}} \equiv D$, *then* $\Gamma(D\delta)\overline{s}(A\lambda_x)\Delta \vdash^{sh} B : C$. *We show this by induction on the derivation* $\Gamma\overline{s}(A\lambda_x)\Delta \vdash^{sh} B : C$ *(for conversion, use 1.).* $\square$

Lemma 6.13 *(Substitution lemma for* $\vdash^{sh}$*)*

1. *If* $\Gamma d\Delta \vdash^{sh} A =_{\mathtt{def}} B$, *$d$ a definition, A and B are $\Gamma d\Delta$-legal terms, then* $\Gamma[\Delta]_d \vdash^{sh} [A]_d =_{\mathtt{def}} [B]_d$
2. *If B is a Γd-legal term, d a definition, then* $\Gamma d \vdash^{sh} B =_{\mathtt{def}} [B]_d$
3. *If* $\Gamma(A\delta)(B\lambda_x)\Delta \vdash^{sh} C : D$ *then* $\Gamma\Delta[x := A] \vdash^{sh} C[x := A] : D[x := A]$
4. *If* $\Gamma(B\lambda_x)\Delta \vdash^{sh} C : D$, $\Gamma \vdash^{sh} A : B$, $(B\lambda_x)$ *bachelor in Γ, then* $\Gamma\Delta[x := A] \vdash^{sh} C[x := A] : D[x := A]$
5. *If* $\Gamma d\Delta \vdash^{sh} C : D$, *$d$ a definition, then* $\Gamma[\Delta]_d \vdash^{sh} [C]_d : [D]_d$

Proof:

1. *Induction to the derivation rules of* $=_{\mathtt{def}}$.

   Case $\Gamma d\Delta \vdash^{sh} d_1 C =_{\mathtt{def}} \underline{d_1}(C[\mathtt{subj}(d_1) := \mathtt{pred}(d_1)])$.

   Then $[d_1 C]_d \equiv ([\mathtt{def}(d_1)]_d\delta)[\underline{d_1}]_d([\mathtt{pred}(d_1)]_d\lambda_{\mathtt{subj}(d_1)})$

   ($d_1 C$ *is* $\Gamma d\Delta$*-legal* $\Rightarrow \mathtt{subj}(d_1) \notin dom(d)$*)*

   and $[\underline{d1}(C[\mathtt{subj}(d_1) := \mathtt{pred}(d_1)])]_d \equiv [\underline{d_1}]_d([C]_d[\mathtt{subj}(d_1) := [\mathtt{pred}(d_1)]_d])$*,*

   hence $\Gamma[\Delta]_d \vdash^{sh} [d_1 C]_d =_{\mathtt{def}} [\underline{d_1}(C[\mathtt{subj}(d_1) := \mathtt{pred}(d_1)])]_d$
2. *Induction on the structure of B.*

   Case $B \equiv x \in dom(d)$*: use (*$=_{\mathtt{def}}$ *def).*

   Case $B \equiv x \notin dom(d)$*: use (*$=_{\mathtt{def}}$ *refl).*

   Case $B \equiv (C\delta)D$*: use (*$=_{\mathtt{def}}$ *comp1).*

   Case $B \equiv (C\mathcal{O}_x)D$ $(\mathcal{O} \in \{\lambda, \Pi\})$*: use (*$=_{\mathtt{def}}$ *comp2).*
3. *Induction to the derivation rules, use 1., 2. and the thinning lemma.*
4. *Idem.*

5. *Corollary of 3.* $\square$

Lemma 6.14 *(Generation Lemma for $\vdash^{sh}$)*

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

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

3. *If* $\Gamma \vdash^{sh} (A\Pi_x)B : C$ *then for some sorts* S_1, S_2*:* $\Gamma \vdash^{sh} A : S_1$, $\Gamma \vdash^{sh} B : S_2$, $(S_1, S_2) \in \mathcal{R}$, $\Gamma \vdash^{sh} C =_{\mathtt{def}} S_2$ *and if* $S_2 \not\equiv C$ *then* $\Gamma \vdash^{sh} C : S$ *for some sort* S.

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

5. *If* $\Gamma \vdash^{sh} \overline{s}A : B$*, then* $\Gamma\overline{s} \vdash^{sh} A : B$

Proof: *1., 2., 3. and 4. follow by a tedious but straightforward induction on the derivations (use the thinning lemma).*

As to 5., we use induction on $\mathtt{weight}(\overline{s})$*:*

- $\mathtt{weight}(\overline{s}) = 0$*: nothing to prove.*

- *If we have proven the hypothesis for all segments* $\overline{s}$ *that obey* $\mathtt{weight}(\overline{s}) \leq 2n$ *and* $\mathtt{weight}(\overline{s}) = 2n + 2$, $\overline{s} \equiv \overline{s_1 s_2}$ *(neither* $\overline{s_1} \equiv \emptyset$ *nor* $\overline{s_2} \equiv \emptyset$*) then by the induction hypothesis:*

  $\Gamma\overline{s_1} \vdash^{sh} \overline{s_2}A : B$*, again applying the induction hypothesis gives* $\Gamma\overline{s_1 s_2} \vdash^{sh} A : B$.

- *If we have proven the hypothesis for all segments* $\overline{s}$ *for which* $\mathtt{weight}(\overline{s}) \leq 2n$ *and* $\mathtt{weight}(\overline{s}) = 2n + 2, \overline{s} \equiv (D\delta)\overline{s_1}(E\lambda_x)$ *where* $\mathtt{weight}(\overline{s_1}) = 2n$ *then an easy induction to the derivation rules shows that one of the following two cases is applicable:*

  - $\Gamma\overline{s} \vdash^{sh} A : B'$, $\Gamma \vdash^{sh} [B']_{\overline{s}} =_{\mathtt{def}} B$ *and if* $[B']_{\overline{s}} \not\equiv B$ *then* $\Gamma \vdash^{sh} B : S$ *for some sort* S.
  - $\Gamma \vdash^{sh} D : F$, $\Gamma \vdash^{sh} \overline{s_1}(E\lambda_x)A : (F\Pi_y)G$, $\Gamma \vdash^{sh} B =_{\mathtt{def}} G[y := D]$ *and if* $G[y := D] \not\equiv B$ *then* $\Gamma \vdash^{sh} B : S$ *for some sort* S.

  In the first case we note that $FV(B) \cap dom(\overline{s}) = \emptyset$ *and that by thinning* $\Gamma\overline{s} \vdash^{sh} [B']_{\overline{s}} =_{\mathtt{def}} B$*, by substitution* $\Gamma\overline{s} \vdash^{sh} [B']_{\overline{s}} =_{\mathtt{def}} B'$ *so* $\Gamma\overline{s} \vdash^{sh} B' =_{\mathtt{def}} B$ *and by conversion we get* $\Gamma\overline{s} \vdash^{sh} A : B$.

  In the second case we know by the induction hypothesis that $\Gamma\overline{s_1} \vdash^{sh} (E\lambda_x)A : (F\Pi_y)G$,

  Now 2. tells us $\Gamma\overline{s_1}(E\lambda_x) \vdash^{sh} A : L$, $\Gamma\overline{s_1} \vdash^{sh} (E\Pi_x)L =_{\mathtt{def}} (F\Pi_y)G$ *and if* $(E\Pi_x)L \not\equiv (F\Pi_y)G$ *then* $\Gamma\overline{s_1} \vdash^{sh} (F\Pi_y)G : S_1$ *for some sort* S_1.

This means that $x \equiv y$, $\Gamma\overline{s_1} \vdash^{sh} E =_{\mathtt{def}} F$, $\Gamma\overline{s_1} \vdash^{sh} L =_{\mathtt{def}} G$. *Out of* $\Gamma\overline{s_1} \vdash^{sh} (E\Pi_x)L : S$ *we get by 3. that* $\Gamma\overline{s_1} \vdash^{sh} E : S_2$ *for some sort* S_2*, thinning gives* $\Gamma\overline{s_1} \vdash^{sh} D : F$ *so by conversion and thinning* $\Gamma\overline{s} \vdash^{sh} A : L$.

Out of $\Gamma \vdash^{sh} B =_{\mathtt{def}} G[x := D]$ *we get (thinning and substitution)* $\Gamma\overline{s} \vdash^{sh} B =_{\mathtt{def}} G$*, out of* $\Gamma\overline{s_1} \vdash^{sh} L =_{\mathtt{def}} G$ *we get* $\Gamma\overline{s} \vdash^{sh} L =_{\mathtt{def}} G$*, hence* $\Gamma\overline{s} \vdash^{sh} B =_{\mathtt{def}} L$.

Now if $G[y := D] \not\equiv B$ *then* $\Gamma \vdash^{sh} B : S$ *for some sort* S*, and if* $G[y := D] \equiv B$ *then we get out of* $\Gamma\overline{s_1} \vdash^{sh} (E\lambda_x)A : (F\Pi_y)G$ *that* $\Gamma\overline{s_1} \vdash^{sh} G : S'$ *for some sort* S'*, by thinning and substitution we get that* $\Gamma\overline{s} \vdash^{sh} G[y := D] : S'$*. In any case, we get* $\Gamma\overline{s} \vdash^{sh} B : S$ *for some sort* S *and by conversion we may conlude* $\Gamma\overline{s} \vdash^{sh} A : B$.

$\square$

Theorem 6.15 *(Subject Reduction for* $\vdash^{sh}$ *and* $\twoheadrightarrow_\beta$*)*
$\Gamma \vdash^{sh} A : B$ *and* $A \twoheadrightarrow A'$ *then* $\Gamma \vdash^{sh} A' : B$.

Proof: *We only need to consider* $A \rightarrow_\beta A'$.

Basic case: suppose $\Gamma \vdash^{sh} (A\delta)(B\lambda_x)C : D$.

Then by the generation lemma: $\Gamma(A\delta)(B\lambda_x) \vdash^{sh} C : D$*, and by the substitution lemma we get* $\Gamma \vdash^{sh} C[x := A] : D[x := A]$*, but as* $x \notin FV(D)$*,* $D[x := A] \equiv D$.

The compatibility cases are easy. $\square$

Now here is the proof of Strong Normalisation for the Cube extended with definitions.

Theorem 6.16 *(Strong Normalisation for the Cube with respect to* $\vdash^{sh}$ *and* $\twoheadrightarrow_\beta$*)*
For all $\vdash^{sh}$*-legal terms* M*,* M *is strongly normalising with respect to* $\twoheadrightarrow_\beta$.

Proof: *Let* M *be a* $\vdash^{sh}$*-legal term. Then either* $M \equiv \square$ *or for some context* Γ *and term* N*,* $\Gamma \vdash^{sh} M : N$.

In the first case, clearly M *is strongly normalising.*

In the second case, define canonical elements $c^A \in CP^{\vdash^{sh}}_{\rightarrow_\beta}(A)$ *for all* $A \in \Gamma^{\vdash^{sh}}$*-kinds as follows:*

$$\begin{array}{lll} c^* & := & SN_{\rightarrow_\beta} \\ c^{(A\Pi_x)B} & := & \boldsymbol{\lambda} f \in CP^{\vdash^{sh}}_{\rightarrow_\beta}(A).c^B \quad \text{if } A \in \Gamma^{\vdash^{sh}}\text{-kinds} \\ c^{(A\Pi_x)B} & := & c^B \quad\quad\quad\quad\quad\quad\quad\;\; \text{if } A \in \Gamma^{\vdash^{sh}}\text{-types} \end{array}$$

Take $\xi^{\vdash^{sh}}_{\rightarrow}$ *such that* $\xi^{\vdash^{sh}}_{\rightarrow}(x) = c^A$ *whenever* $(A\lambda_x) \in' \Gamma$ *and* $\xi^{\vdash^{sh}}_{\rightarrow}(\mathtt{subj}(d)) = [\![\mathtt{def}(d)]\!]_{\xi^{\vdash^{sh}}_{\twoheadrightarrow}}$ *whenever* $d \in' \Gamma$*-*$\mathtt{def}$ *and take* $\rho^{\vdash^{sh}}_{\rightarrow_\beta}$ *such that* $\rho^{\vdash^{sh}}_{\rightarrow_\beta}(\mathtt{subj}(d)) = (\!|\mathtt{def}(d)|\!)_{\rho^{\vdash^{sh}}_{\twoheadrightarrow}}$ *for all subdefinitions* d *of* Γ *and* $\rho^{\vdash^{sh}}_{\rightarrow_\beta}(x) = x$ *otherwise.*

Then $\rho^{\vdash^{sh}}_{\rightarrow_\beta}, \xi^{\vdash^{sh}}_{\rightarrow} \models \Gamma$*, hence* $(\!|M|\!)_{\rho^{\vdash^{sh}}_{\twoheadrightarrow}} \in [\![N]\!]_{\xi^{\vdash^{sh}}_{\twoheadrightarrow}}$*, where* $(\!|M|\!)_{\rho^{\vdash^{sh}}_{\twoheadrightarrow}} = \lceil M \rceil$ *as mentioned in lemma 2.45. Hence* $\lceil M \rceil \in [\![N]\!]_{\xi^{\vdash^{sh}}_{\twoheadrightarrow}} \subseteq SN_{\rightarrow_\beta}$*. By lemma 2.45 now also* $M \in SN_{\rightarrow_\beta}$. $\square$

7 The Cube with definitions and shuffle-reduction

Now we extend the type system of section 6 by changing the reduction $\twoheadrightarrow_\beta$ into $\leadsto\!\!\!\!\twoheadrightarrow_\beta$. As was the case in section **??** the derivation rules stay the same as those with classical β-reduction, hence almost all lemmas that have been proved for the system in section 6 are still valid. The only properties that have to be investigated are Church-Rosser, Subject Reduction and Strong Normalisation. We will show now that all these properties too are still valid.

Theorem 7.1 *(The general Church Rosser theorem for $\rightsquigarrow\!\!\!\!\rightarrow_\beta$)*
If $A \rightsquigarrow\!\!\!\!\rightarrow_\beta B$ and $A \rightsquigarrow\!\!\!\!\rightarrow_\beta C$, then there exists D such that $B \rightsquigarrow\!\!\!\!\rightarrow_\beta D$ and $C \rightsquigarrow\!\!\!\!\rightarrow_\beta D$.
Proof: *see theorem 5.9.* □

Theorem 7.2 *(Subject Reduction for $\vdash^{sh}$ and $\rightsquigarrow\!\!\!\!\rightarrow_\beta$)*
If $\Gamma \vdash^{sh} A : B$ and $A \rightsquigarrow\!\!\!\!\rightarrow_\beta A'$ then $\Gamma \vdash^{sh} A' : B$.
Proof: *We only need to consider $A \rightsquigarrow_\beta A'$.*
Basic case: suppose $\Gamma \vdash^{sh} dC : D$.
Then by the generation lemma: $\Gamma d \vdash^{sh} C : D$. Hence by definition-shuffling (6.11, say $A \equiv \mathtt{def}(d)$, $B \equiv \mathtt{pred}(d)$ and $x \equiv \mathtt{subj}(d)$): $\Gamma \underline{d}(A\delta)(B\lambda_x) \vdash^{sh} C : D$, hence by substitution $\Gamma \underline{d} \vdash^{sh} C[x := A] : D[x := A]$, and by (def rule) *$\Gamma \vdash^{sh} \underline{d}(C[x := A]) : [D[x := A]]_{\underline{d}}$, which is $\Gamma \vdash^{sh} \underline{d}(C[x := A]) : [D]_d$.*
Now by the variable convention $[D]_d \equiv D$ so we are done.
The compatibility cases are easy. □

Theorem 7.3 *(Strong Normalisation for the Cube with respect to $\vdash^{sh}$ and $\rightsquigarrow\!\!\!\!\rightarrow_\beta$)*
For all $\vdash^{sh}$-legal terms M, M is strongly normalising with respect to $\rightsquigarrow\!\!\!\!\rightarrow_\beta$.
Proof: *This is exactly as the proof of Theorem 6.16 where every occurrence of $\twoheadrightarrow_\beta$ is replaced by $\rightsquigarrow\!\!\!\!\rightarrow_\beta$.* □

8 Comparing the type system with definitions to the original type system

In this section we will compare the type system generated by $\vdash^{sh}$ with the one generated by $\vdash$, from two different points of view. The first is the conservativity, where we show that in a certain sense, definitions are harmless. That is, even though we can type more terms using $\vdash^{sh}$ than using $\vdash$, whenever a judgement is derivable in a theory $\mathcal{L}$ using definitions and $\vdash^{sh}$, it is also derivable in the theory $\mathcal{L}$ without definitions, using only $\vdash$ and where all the definitions are unfolded. The second viewpoint is about the effectiveness of derivations. More work has to be done yet but it is certain that there is a gain in using definitions.

8.1 Conservativity

As we already saw in example 6.4, in the type systems with definitions there are more legal terms. Therefore, it has to be investigated to what extent the set of legal terms has changed. Note first that all derivable judgements in a type system of the λ-cube are derivable in the same type system extended with definitions as we only extended, not changed, the derivation rules.

A second remark concerns the bypassing of the *formation rule* by using the *weakening* and *definition rule* instead: In $\lambda 2$ without definitions we can derive the following judgement by using the formation rules $(*, *)$ and $(\square, *)$:

$\Gamma \vdash^{\mathrm{sh}} (y\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_x)x : \beta$ where $\Gamma \equiv (*\lambda_\beta)(\beta\lambda_y)$, namely:

$\Gamma \vdash_{\lambda 2} y : \beta : * : \Box$	
$\Gamma(*\lambda_\alpha) \vdash_{\lambda 2} \alpha : *$	(start)
$\Gamma(*\lambda_\alpha)(\alpha\lambda_x) \vdash_{\lambda 2} x : \alpha : *$	(start resp weakening)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda 2} (\alpha\Pi_x)\alpha : *$	(formation rule $(*,*)$)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda 2} (\alpha\lambda_x)x : (\alpha\Pi_x)\alpha$	(abstraction)
$\Gamma \vdash_{\lambda 2} (*\Pi_\alpha)(\alpha\Pi_x)\alpha : *$	(fromation rule $(\Box,*)$)
$\Gamma \vdash_{\lambda 2} (*\lambda_\alpha)(\alpha\lambda_x)x : (*\Pi_\alpha)(\alpha\Pi_x)\alpha$	(abstraction)
$\Gamma \vdash_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(\alpha\lambda_x)x : (\beta\Pi_x)\beta$	(application, we already knew $\Gamma \vdash_{\lambda 2} \beta : *$)
$\Gamma \vdash_{\lambda 2} (y\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_x)x : \beta$	(application, we already knew $\Gamma \vdash_{\lambda 2} y : \beta$)

It is not possible to derive this judgement in $\lambda_\to$ as the *formation rule* $(\Box,*)$ is needed. Now we observe that the term $(y\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_x)x$ can be seen as x with two definitions added, and using this observation we can derive the judgement in a type system with definition without having to use the formation rules $(*,*)$ and $(\Box,*)$:

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

This example shows that in $\lambda_{\to\mathtt{def}}$ we have more legal judgements than in $\lambda_\to$. Now we take a look at the judgement $\Gamma \vdash (\beta\delta)(*\lambda_\alpha)(M\lambda_x)x : (M\Pi_x)M$ where $M \equiv (y\delta)(\beta\lambda_z)(\beta\delta)(*\lambda_\gamma)\gamma$ and $\Gamma \equiv (*\lambda_\beta)(\beta\lambda_y)$. This judgement can be derived in λC using the formation rules $(\Box,\Box)$, $(\Box,*)$, $(*,\Box)$ and $(*,*)$ in the following way:

$\Gamma \vdash_{\lambda C} \beta : * : \Box$	
$\Gamma(*\lambda_\alpha) \vdash_{\lambda C} \beta : * : \Box$	(weakening)
$\Gamma(*\lambda_\alpha)(\beta\lambda_z) \vdash_{\lambda C} z : \beta : * : \Box$	(start resp. weakening)
$\Gamma(*\lambda_\alpha)(\beta\lambda_z)(*\lambda_\gamma) \vdash_{\lambda C} \gamma : * : \Box$	(start resp. weakening)
$\Gamma(*\lambda_\alpha)(\beta\lambda_z) \vdash_{\lambda C} (*\Pi_\gamma)* : \Box$	(formation rule $(\Box,\Box)$)
$\Gamma(*\lambda_\alpha)(\beta\lambda_z) \vdash_{\lambda C} (*\lambda_\gamma)\gamma : (*\Pi_\gamma)*$	(abstraction)
$\Gamma(*\lambda_\alpha)(\beta\lambda_z) \vdash_{\lambda C} (\beta\delta)(*\lambda_\gamma)\gamma : *$	(application)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda C} (\beta\Pi_z)* : \Box$	(formation rule $(*,\Box)$)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda C} (\beta\lambda_z)(\beta\delta)(*\lambda_\gamma)\gamma : (\beta\Pi_z)*$	(abstraction)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda C} M : *$	(application, $M \equiv (y\delta)(\beta\lambda_z)(\beta\delta)(*\lambda_\gamma)\gamma$)
$\Gamma(*\lambda_\alpha)(M\lambda_x) \vdash_{\lambda C} x : M : *$	(start resp. weakening)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda C} (M\Pi_x)M : *$	(formation rule $(*,*)$)
$\Gamma(*\lambda_\alpha) \vdash_{\lambda C} (M\lambda_x)x : (M\Pi_x)M$	(abstraction)
$\Gamma \vdash_{\lambda C} (*\Pi_\alpha)(M\Pi_x)M : *$	(formation rule $(\Box,*)$)
$\Gamma \vdash_{\lambda C} (*\lambda_\alpha)(M\lambda_x)x : (*\Pi_\alpha)(M\Pi_x)M$	(abstraction)
$\Gamma \vdash_{\lambda C} (\beta\delta)(*\lambda_\alpha)(M\lambda_x)x : (M\Pi_x)M$	(application)

Note that it is impossible to derive this judgement in any other system of the cube than

λC as all four formation rules are needed. Analogous to the previous example we can also derive this judgement in $\lambda_{\rightarrow\mathtt{def}}$:

$\Gamma \vdash^{\text{sh}}_{\lambda_\rightarrow} \beta : * : \square$	
$\Gamma(\beta\delta)(*\lambda_\alpha) \vdash^{\text{sh}}_{\lambda_\rightarrow} \beta : * : \square$	(weakening)
$\Gamma(\beta\delta)(*\lambda_\alpha)(y\delta)(\beta\lambda_z) \vdash^{\text{sh}}_{\lambda_\rightarrow} \beta : * : \square$	(weakening)
$\Gamma(\beta\delta)(*\lambda_\alpha)(y\delta)(\beta\lambda_z)(\beta\delta)(*\lambda_\gamma) \vdash^{\text{sh}}_{\lambda_\rightarrow} \gamma : *$	(weakening)
$\Gamma(\beta\delta)(*\lambda_\alpha) \vdash^{\text{sh}}_{\lambda_\rightarrow} (y\delta)(\beta\lambda_z)(\beta\delta)(*\lambda_\gamma)\gamma : *[\gamma := \beta][z := y]$ i.e. $M : *$	(definition rule)
$\Gamma(\beta\delta)(*\lambda_\alpha)(M\lambda_x) \vdash^{\text{sh}}_{\lambda_\rightarrow} x : M : *$	(start resp. weakening)
$\Gamma(\beta\delta)(*\lambda_\alpha) \vdash^{\text{sh}}_{\lambda_\rightarrow} (M\Pi_x)M : *$	(formation rule $(*,*)$)
$\Gamma(\beta\delta)(*\lambda_\alpha) \vdash^{\text{sh}}_{\lambda_\rightarrow} (M\lambda_x)x : (M\Pi_x)M$	(abstraction)
$\Gamma \vdash^{\text{sh}}_{\lambda_\rightarrow} (\beta\delta)(*\lambda_\alpha)(M\lambda_x)x : (M\Pi_x)M[\alpha := \beta] \equiv (M\Pi_x)M$	(definition rule)

This example shows that in every system of the λ-cube (except λC), adding definitions gives more derivable judgements. As was shown in example 6.4, also the judgement $(*\lambda_\beta)(\beta\lambda_{y'}) \vdash^{\text{sh}}_{\lambda 2} (\beta\delta)(*\lambda_\alpha)(y'\delta)(\alpha\lambda_x)x : \beta$ is derivable in $\lambda 2_{\mathtt{def}}$ and hence is also derivable in $\lambda C_{\mathtt{def}}$, but this judgement cannot be derived in λC as the term y is of type β and not of type α.

At first sight this might cause the reader to suspect type systems with definitions of having too much derivable judgements. However, we have a conservativity result stating that a judgement that can be derived in $\mathcal{L}_{\mathtt{def}}$ can be derived in $\mathcal{L}$ when all definitions in the whole judgement have been unfolded.

Definition 8.1 *For $\Gamma \vdash^{sh} A : B$ a judgement we define the unfolding of $\Gamma \vdash^{sh} A : B$ to be the judgement obtained from $\Gamma \vdash^{sh} A : B$ in the following way:*

- *first, mark all visible $\delta\lambda$-couples in Γ, A and B,*
- *second, contract in Γ, A and B all these marked $\delta\lambda$-couples.*

It is meant here when $\Gamma \equiv \cdots (C\delta)\overline{s}(D\lambda_x)\cdots$, then contracting $(C\delta)(D\lambda_x)$ amounts to substituting all free occurrences of x in the scope of λ_x by C; these free occurrences may also be in one of the terms A and B. The result is independent of the order in which the redexes are contracted, as one can see this unfolding as a complete development (see [Barendregt 84]) in a certain sense.

Example 8.2 The unfolding of
$(*\lambda_\beta)(\beta\lambda_y)(y\delta)(\beta\delta)(*\lambda_\alpha)(\alpha\lambda_x)(\alpha\lambda_z) \vdash^{\text{sh}} ((\alpha\lambda_u)u\delta)((\alpha\Pi_u)\beta\lambda_v)(x\delta)v : \alpha$ is the judgement
$(*\lambda_\beta)(\beta\lambda_y)((\alpha\lambda_z)[x := y][\alpha := \beta]) \vdash^{\text{sh}} (((x\delta)v)[v := (\alpha\lambda_u)u])[x := y][\alpha := \beta] : \alpha[x := y][\alpha := \beta]$,
which is $(*\lambda_\beta)(\beta\lambda_y)(\beta\lambda_z) \vdash^{\text{sh}} (y\delta)(\beta\lambda_u)u : \beta$.

Note that the resulting context contains only λ-items and that the resulting subject and predicate need not be in normal form.

Theorem 8.3 *Let $\mathcal{L}$ be one of the systems of the λ-cube, Γ a context with definitions and A, B pseudoterms.*

If $\Gamma \vdash^{sh}_{\mathcal{L}} A : B$ then $\Gamma' \vdash_{\mathcal{L}} A' : B'$, where $\Gamma' \vdash_{\mathcal{L}} A' : B'$ is the unfolding of $\Gamma \vdash^{sh}_{\mathcal{L}} A : B$ according to definition 8.1

Proof: *use induction on the derivation of* $\Gamma \vdash^{sh}_{\mathcal{L}} A : B$. axiom, abstraction *and* formation rules *are easy, we treat the other cases.*

- *The last rule applied is the* start *rule. Then* $\Gamma d \vdash^{sh}_{\mathcal{L}} \mathtt{subj}(d) : \mathtt{pred}(d)$ *as a consequence of* $\Gamma \vdash^{sh}_{\mathcal{L}} d$. *Now if* $d \equiv (A\lambda_x)$ *then by the induction hypothesis* $\Gamma' \vdash_{\mathcal{L}} A' : S$ *(S a sort, x fresh) so by the* start rule $\Gamma'(A\lambda_x) \vdash_{\mathcal{L}} x : A'$.

  On the other hand, if d is a definition, say $d \equiv (A\delta)\underline{d}(B\lambda_x)$, *then by the induction hypothesis* $(\Gamma\underline{d})' \vdash_{\mathcal{L}} A' : B' : S$ *(S a sort), which is* $\Gamma' \vdash_{\mathcal{L}} A' : B' : S$ *as d will be fully unfolded, and the unfolding of* $\Gamma d \vdash^{sh}_{\mathcal{L}} \mathtt{subj}(d) : \mathtt{pred}(d)$ *is* $\Gamma' \vdash_{\mathcal{L}} \mathtt{def}(d)' : \mathtt{pred}(d)'$ *which is* $\Gamma' \vdash_{\mathcal{L}} A' : B'$ *so we are done.*

- *The last rule applied is the weakening rule, say* $\Gamma \vdash^{sh}_{\mathcal{L}}$ *as a consequence of* $\Gamma \vdash^{sh}_{\mathcal{L}}$ *and* $\Gamma \vdash^{sh}_{\mathcal{L}} d$. *Because* $\mathtt{subj}(d)$ *is fresh we have that* $(\Gamma d)' \vdash_{\mathcal{L}} D' : E'$ *is the same as* $(\Gamma\underline{d})' \vdash_{\mathcal{L}} D' : E'$ *so by the induction hypothesis we are done.*

- *The last rule applied is the* application rule. *Then* $\Gamma \vdash^{sh}_{\mathcal{L}} (a\delta)F : B[x := a]$ *as a consequence of* $\Gamma \vdash^{sh}_{\mathcal{L}} F : (A\Pi_x)B$ *and* $\Gamma \vdash^{sh}_{\mathcal{L}} a : A$. *By the induction hypothesis and the* application rule *we get* $\Gamma' \vdash_{\mathcal{L}} (a'\delta)F' : B'[x := a']$. *Now by subject reduction also* $\Gamma' \vdash_{\mathcal{L}} ((a'\delta)F')' : B'[x := a']$. *If* $B'[x := a'] \equiv (B'[x := a'])'$ *then we are done, otherwise, by the Generation Corollary* $\Gamma' \vdash_{\mathcal{L}} B'[x := a'] : S$ *for some sort S, so by subject reduction* $\Gamma' \vdash_{\mathcal{L}} (B'[x := a'])' : S$ *and as* $B'[x := a'] =_\beta (B'[x := a'])'$ *by* conversion *we are done.*

- *The last rule applied is the* conversion rule . *Then* $\Gamma \vdash^{sh}_{\mathcal{L}} A : B_2$ *as a consequence of* $\Gamma \vdash^{sh}_{\mathcal{L}} A : B_1$, $\Gamma \vdash^{sh}_{\mathcal{L}} B_2 : S$ *and* $\Gamma \vdash^{sh}_{\mathcal{L}} B_1 =_{\mathtt{def}} B_2$. *Now* $\Gamma \vdash^{sh}_{\mathcal{L}} B_1 =_{\mathtt{def}} B_2$ *implies* $B_1' =_\beta B_2'$ *because if C results from D by locally unfolding a definition of* Γ *then* $C' \equiv D'$, *so the result follows by the induction hypothesis.*

Remark 8.4 It is not sufficient in theorem 8.3 to unfold all the definitions in the context only, because a redex in the subject may have been used to change the type when it was still in the context, this is illustrated by the judgement $(*\lambda_\beta)(\beta\lambda_y) \vdash^{\mathrm{sh}}_{\lambda_\to} (\beta\delta)(*\lambda_\alpha)(y\delta)(\alpha\lambda_x)x : \beta$ which cannot be derived using $\vdash_{\lambda_\to}$. It is the case however that this judgement where all the definitions are unfolded in context, subject and predicate, is derivable using $\vdash$. That is, $(*\lambda_\beta)(\beta\lambda_y) \vdash_{\lambda_\to} y : \beta$.

8.2 Shorter derivations

As we already noted, derivations using the definition mechanism tend to need considerably less derivation steps to derive a judgement that can also be derived without definitions. Without making precise too much details about the specifc way in which a term is being typed, we can still make some remarks on this subject.

The idea is that there exists an algorithm that determines for any given term M whether M is well typed and if so, it gives a derivation of a type of this term M. Now for every $\delta\lambda$-segment in M this typing algorithm has to do all of the following steps (say the $\delta\lambda$-segment is $(A\delta)(B\lambda_x)$ followed by the term C, and A, B and C have been type checked already, the type of C being D):

- is the type of A β-equal to B?
- add $(B\lambda_x)$ to the context

- use the *formation rule* to form $(B\Pi_x)D$
- use the *abstraction rule* to derive $(B\lambda_x)C : (B\Pi_x)D$
- use the *application rule* to derive $(A\delta)(B\lambda_x)C : D[x := A]$.

Now by using definitions all these steps can be replaced by

- is the type of A β-equal to B?
- add $(A\delta)(B\lambda_x)$ to the context
- use the *definition rule* to derive $(A\delta)(B\lambda_x)C : D[x := A]$.

Hence we need two steps less for any $\delta\lambda$-segment in the term M. For the well-balanced segments in M, the number of steps decreases even more as we only need to use the *definition rule* once for an entire segment of definitions.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy, "Explicit Substitutions", *Functional Programming 1 (4)* (1991) 375-416.

[Barendregt 84] Barendregt, H., *Lambda Calculus: its Syntax and Semantics,* North-Holland, 1984.

[Barendregt 92] Barendregt, H., Lambda calculi with types, *Handbook of Logic in Computer Science*, volume II, ed. Abramsky S., Gabbay D.M., Maibaum T.S.E., Oxford University Press, 1992.

[BKKS 87] Barendregt, H.P., Kennaway, J.R., Klop, J.W., and Sleep M.R., Needed reduction and spine strategies for the λ-calculus, *Information and Computation 75 (3)*, 1191-231, 1987.

[Jutting 90] van Benthem Jutting, L.S., Typing in Pure Type Systems, ms., University of Nijmegen, Department of Computing Science, 1990.

[BKN 9x] Bloo, R., Kamareddine, F., Nederpelt, R., Beyond β-reduction in Church's $\lambda_\rightarrow$, Computing Science Note 94/20, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1994.

[BKN 9x] Bloo, R., Kamareddine, F., Nederpelt, R., The Barendregt Cube with definitions and generalised reduction, Computing Science Note 94/34, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1994.

[de Bruijn 93] Bruijn, N.G. de, Algorithmic definition of lambda-typed lambda calculus, in Huet, G. and Plotkin, G. eds. *Logical Environments*, 131-146, Cambridge University Press, 1993.

[de Bruijn 80] Bruijn, N.G. de, A survey of the project AUTOMATH, in *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, ed. Hindley, J.R., and Seldin, J.P., Academic Press, 29-61, 1980.

[Church 40] Church, A., A formulation of the simple theory of types, *Journal of Symbolic Logic 5*, 56-68, 1940.

[CH 88] Coquand T., and Huet G., The calculus of constructions, *Information and Computation 76*, 95-120, 1988.

[Dow 91] Dowek, G. et al. The Coq proof assistant version 5.6, users guide, rapport de recherche 134, INRIA, 1991.

[Gardena 94] Gardena, P., Discovering Needed Reductions Using Type Theory, to appear in TACS, 1994.

[Geuvers 94] Geuvers, H., A short and flexible proof of Strong Normalisation for the Calculus of Constructions, notes of a talk given at the BRA types workshop, Bastad, Sweden, 1994.

[KN 93] Kamareddine, F., and Nederpelt, R.P., On stepwise explicit substitution, *International Journal of Foundations of Computer Science 4 (3)*, 197-240, 1993.

[KN 9z] Kamareddine, F., and Nederpelt, R.P., *The beauty of the λ-calculus*, in preparation.

[KN 9y] Kamareddine, F., and Nederpelt, R.P., Canonical Typing and Π–conversion, submitted for publication.

[KN94b] Kamareddine, F., and Nederpelt, R.P., Refining reduction in the λ-calculus, Computing Science Note 94/18, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1994.

[Launchbury 93] Launchbury, J., A natural semantics of lazy evaluation, *ACM POPL 93*, 144-154, 1993.

[Lévy 80] Lévy, J.-J. Optimal reductions in the lambda calculus, in *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. Seldin and R. Hindley eds, Academic Press, 1980.

[LP 92] Luo Z., and Pollack, R., LEGO proof development system: User's manual, Technical report ECS-LFCS-92-211, LFCS, University of Edinburgh, 1992.

[GM 93] Gordon M.J.C. and Melham, T.F. (eds), *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*, Cambridge University Press, 1993.

[Nederpelt 73] Nederpelt, R.P., *Strong normalisation in a typed lambda calculus with lambda structured types*, Ph.D. thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science, 1973. ALso in Nederpelt, R.P., Geuvers, J.H. and de Vrijer, R.C., eds., *Selected Papaers on Automath*, North Holland, 1994.

[Nederpelt 73] Nederpelt, R.P., Geuvers, J.H. and de Vrijer, R.C., eds., *Selected Papaers on Automath*, North Holland, 1994.

[NK 94] Nederpelt, R.P., and Kamareddine, F., A unified approach to type theory through a refined λ-calculus, Proceedings of the 1992 conference on *Mathematical Foundations of Programming Semantics*, ed. M. Mislove et. al., 1994.

[SP 93] Severi, P., and Poll, E., Pure Type Systems with Definitions, Computing Science Note 93/24, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1993.