Type Systems for Dummies

Andrea Asperti
Department of Computer Science,
University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
asperti@cs.unibo.it

Ferruccio Guidi
Department of Computer Science,
University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
fguidi@cs.unibo.it

Abstract

We extend Pure Type Systems with a function turning each term M of type A into a *dummy* $|M|$ of the same type ($|\cdot|$ *is not* an identity, in that $M \neq |M|$). Intuitively, a dummy represents an *unknown, canonical object* of the given type: dummies are *opaque* (cannot be internally inspected), and *irrelevant* in the sense that dummies of a same type are convertible to each other. This latter condition makes convertibility in PTS with dummies (DPTS) stronger than usual, hence raising not trivial consistency issues. DPTS offer an alternative approach to (proof) irrelevance, tagging irrelevant information at the level of terms and not of types, and avoiding the annoying syntactical duplication of products, abstractions and applications into an explicit and an implicit version, typical of systems like ICC^*.

Categories and Subject Descriptors F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic—Lambda calculus and related systems

General Terms Theory

Keywords Pure type system, proof irrelevance, canonical element

1. Introduction

Mechanized mathematical proofs are becoming a standard tool in research related to programming languages and software development methods (see e.g. [22, 23, 37, 44] for some major, recent achievements). In particular, proof assistants based on dependent type theory ([4, 13, 14]) seem to enjoy a growing popularity in this field, due to several attractive features of these formalisms:

1. type theories embed key computational constructs of functional programming languages: functions can be defined by (well-founded) recursion, and are live entities that can be tested and executed; data are typed terms identified modulo reduction to their normal form (conversion), allowing to distinguish between computation and reasoning, and to treat them differently (sometimes referred to as “Poincaré principle” [8]); the more powerful is the conversion rule, the more reasoning is reduced to mere computation, making the logical argument more concise and cogent (that is also the leading idea behind theorem proving modulo [16]);
2. proofs are an integrated part of the formalism, allowing, via the Curry Howard isomorphism [36], a smooth interplay between specification and reasoning: proofs are objects of the language, and can be treated as normal data, naturally leading to a programming style akin to proof-carrying-code [31], where chunks of software come equipped with proofs of (some of) their properties. Moreover, sharing a common syntax between the logical and the computational part of the theory reduces verification to type-checking, in such a way that only a small and well-identified software component (the so-called kernel [5]) is really critical for the reliability of the whole application (the so-called “de Bruijn principle”).

The relation between the logical and computational part is however far from being perfect: often, the logical part “gets in the way” at places where one would not expect it, making the logical reasoning more complex than expected. The typical, well known example is the case of a subset type $\{x : A \mid P\}$ of a type A, restricted by means of a given predicate P [1]. In type theories, elements of this set are pairs $\langle a, h\rangle$, composed of an element a of type A and a *proof* h that a satisfies the property P (i.e., a term h of type $P(a)$). The annoying part of the story is that two elements $\langle a, h'\rangle$ and $\langle a, h''\rangle$ may now be different just because we provided two *different proofs* h' and h'' of the property $P(a)$, while we probably considered this information as completely *irrelevant*.

The important issue of characterizing “irrelevant” information for the purposes of conversion, with the goal of making it more flexible and powerful, has received much attention in recent years [1, 9, 12, 25, 29, 30, 34, 35, 43] (in older literature, such as [10] or [3], proof irrelevant type theories are mostly investigated as a tool for proving properties of other systems).

All recent studies exploited the important relation between *irrelevance* and (program) *extraction*. Type theories are naturally constructive: every proof comes equipped with a computational content (a computable function) that can be automatically *extracted* from the proof by removing all the information that is *irrelevant* for the computation but is only used for type-checking purposes (see [24]). While the practical interest of code extraction is still questionable, this paradigm can be used as a leading idea to extend convertibility: instead of comparing the typed terms, we may just compare their extracted computational content. In Coq [41], the distinction between relevant and irrelevant information is based on the sort of types: terms of sort Prop (proofs) are removed during extraction, and proof-irrelevant versions of the Calculus of Inductive Constructions have been investigated by Werner [29, 43]. Miquel’s Implicit Calculus of Constructions - a Curry-style variant of the Calculus of Constructions - offers a more satisfactory alter-

[1] Subset types are also at the core of PVS [33]. However, in this formalism, objects of type $\{x : A|P\}$ are also of type A, making type checking undecidable.

native to the distinction between Prop and Set. In [9], Bernardo and Barras introduce a decidable variant of this calculus, called ICC^* obtained by explicitly decorating the irrelevant information inside the term. This technique, which is adopted by other authors as well (see for instance [1, 25, 30, 34, 35]), is quite invasive, since it requires a complete syntactical duplication of products, abstractions and applications into an explicit and an implicit variant. Moreover, while the technique is surely interesting for extraction, its logical interest is reduced by the incapability of the type system to exploit convertibility between terms with different types[2].

In this paper we propose a different approach, based on an explicit function allowing to tag each term M of type A as a dummy $|M|$ of the same type. The term $|M|$ becomes an opaque object, whose internal structure cannot be exploited; on the other hand, since we have no way to distinguish dummies of the same type from each other, we may consistently assume they are all convertible. The dummy function is hence similar to the Werner's ϵ term [43], but for the fact that we keep a copy of M for type checking purposes (mostly to ensure that the type A is inhabited in the given context).

For example, considering again the case of subset types, we may explicitly *declare* proofs as irrelevant, in such a way that the two terms $\langle a, |h'|\rangle$ and $\langle a, |h''|\rangle$ are convertible even if h' and h'' are not.

Dummy terms can also be used for partial specifications. For instance, when defining the predecessor function for natural numbers in type theory, we face the embarrassing choice of extending it on 0. The natural solution is to use a dummy value, defining $pred\ 0$ as, say, $|0|$ (any other choice of a natural number inside the dummy would be equivalent). The important benefit of using a dummy in the specification is to guarantee that, in our proofs, we will not make an improper, extremely fragile, (ab)use of a specific extension. For instance, if we define $pred\ 0 = 1$ we would be able to prove that $\forall x : nat.pred\ x \neq x$; if, later, we decide to change the definition letting $pred\ 0 = 0$ the previous theorem would become false. The robust way to state the result is to guard it in order to subsume undefined cases by absurdity: the statement $\forall x : nat.x > 0 \to pred\ x \neq x$ is still provable with the dummy extension, since under the assumption $0 > 0$ we can prove *anything*.

The issue of irrelevance, and its practical importance for improving usability of interactive provers based on type theory has been largely discussed in previous papers (see e.g. [9, 29, 43]), so we shall not insist on this point here. Dummies are not meant to improve on this respect, but to provide an alternative foundational view on irrelevance, and a more syntactical and operational approach to consistency. Dummies are a technical tool that can be used to recover more traditional functionality: for example, the process of systematically marking all subterms of a given sort (say, Prop) as dummies does not compromise the fact that a term is well typed, hence mimicking proof-irrelevance.

In this paper, we formally introduce and investigate the notion of dummy in the case of Pure Types Systems. We chose Pure Type System instead of more sophisticated Systems with Inductive Types for a double reason. First of all, Inductive Types introduce a heavy additional syntactical burden that makes the metatheory sensibly more entangled than for PTS; since we are introducing a new notion, it looks preferable to start from a more comfortable and standard setting. The second reason is technical: some of the techniques used in this paper do not extend to Inductive Types. In particular, if case analysis of dummies is opaque, normalization is stopped and we cannot derive consistency by a simple inspection on the shape of normal forms. This point can be bypassed in PTS by adding an "absorption" rule (see rule δ below) that is enough to normalize terms with dummies in a form suitable to entail consistency (note that this rule sensibly changes the intuitive meaning of dummies, turning them into something more akin to a bottom element). As we discuss in the conclusion, this technical approach cannot be extended to the case of inductive types with discrimination rules for constructors. So, while we strongly believe in the consistency of dummies (without δ) even in the case of Type Systems with Inductive Types, the proof of this claim (or its confutation) is an open, challenging issue.

The paper is structured in two main sections, dealing respectively with general PTS (Section 2) and with the dummy version (DCC) of the Calculus of Constructions (Section 3).

In particular, Section 2.1 and Section 2.2 provide syntax, reductions and typing rules of PTS with dummies (DPTS); Section 2.3 extends to DPTS most of the traditional meta-theory of PTS (substitution lemma 9, generation lemma 12, subject reduction property 14); Section 2.4 and Section 2.5 deal respectively with uniqueness of types and type inhabitance; finally Section 2.6 introduces the notion of Reducibility Candidates in the general framework of DPTS.

In Section 3 we prove the strong normalization of DCC. The section starts with the essential classification lemma (Section 3.1); then we provide an overview of the proof (Section 3.2), to conclude with the technical details in Section 3.3.

Section 4 contains our concluding remarks.

Our technical exposition intentionally adheres to standard introductory texts, such as [7] and [36]. In all proofs, we tried to emphasize the most interesting subcases and the new cases induced by dummies.

2. PTS with dummies

Pure Type Systems have been independently introduced by Berardi [11] and Terlouw [42] as a way of generalizing the presentation of logical systems in Barendregt's λ-cube [7]. They provide a unifying, neat and compact framework to express many different systems of typed λ-calculus *à la* Church, and are a basic backbone of the modern presentation of Type Theory (see e.g. [36]).

We extend the usual notion by adding a new rule that allows us to turn each term M of type A into a dummy $|M|_A$ of type A as well; convertibility of terms is the smallest congruence relation closed with respect to reduction *and* equalities of dummies: $|M|_A = |N|_A$. It is worth remarking that, working with well-typed terms, the type annotation A for the dummy $|M|_A$ will always be redundant, since it can be uniquely[3] inferred from the term M: it is just a technical artifice essentially devised to avoid to equate terms of a different nature (terms, types, kinds), that could open the way to paradoxes.

2.1 Raw terms, reduction and conversion

Let S be a set of *sorts* (types for types), ranged over by s, and let V be a set of variables, ranged over by x. We shall work with the following set of *raw terms*:

$$M ::= x \mid s \mid (M\ M) \mid \lambda x : M.M \mid \Pi x : M.M \mid |M|_M$$

The operators λ and Π are binders, and their scope is the term following the dot. The definition of the set $FV(M)$ of *free variables* of M, and that of the substitution operation $M[N/x]$ are the usual ones: in particular

- $FV(|M|_P) = FV(M) \cup FV(P)$
- $(|M|_P)[N/x] = |M[N/x]|_{P[N/x]}$

[2] even using the so-called John Major's Equality [27].

[3] Provided the DPTS is singly sorted, see Theorem 20.

A relation $\diamond$ over raw terms is said to be *compatible* if for all M, N such that $M \diamond N$, the following properties hold for any P

$$\begin{array}{ll}(M\ P) \diamond (N\ P) & \lambda x : M.P \diamond \lambda x : N.P \\ (P\ M) \diamond (P\ N) & \lambda x : P.M \diamond \lambda x : P.N \\ |M|_P \diamond |N|_P & \Pi x : M.P \diamond \Pi x : N.P \\ |P|_M \diamond |P|_N & \Pi x : P.M \diamond \Pi x : P.N\end{array}$$

We say that $\diamond$ is *preserved by substitution* if for all M, M', N, N' such that $M \diamond M'$ and $N \diamond N'$ we have $M[N/x] \diamond M'[N'/x]$.

The relation $\to_{\beta\delta}$ is the smallest compatible relation containing the following reduction rules:

$$\begin{array}{ll}(\beta) & (\lambda x : P.M)\ N \to M[N/x] \\ (\delta) & |M|_{\Pi x:A.B}\ N \to |MN|_{B[N/x]}\end{array}$$

(reduction inside dummies is allowed). The relation $\overset{*}{\to}_{\beta\delta}$ ($=_{\beta\delta}$) is the reflexive and transitive (and symmetric) closure of $\to_{\beta\delta}$. Rule (δ) states that dummies behave in a way similar to exceptions: they cannot be internally inspected, and whenever applied to an argument they just "absorbe" it. As we anticipated in the introduction, rule δ in not an essential property of dummies, but a technical artifice useful to recover, in the specific context of PTS, the subformula property on normal forms.

Using standard techniques, like e.g. parallel reduction [39], it is easy to prove the Church-Rosser property:

LEMMA 1. *$\overset{*}{\to}_{\beta\delta}$ is confluent.*

All dummies in a given type are equal to each other as stated by the following d-rule:

$$(d) \quad |M|_A =_d |N|_A$$

The smallest compatible equivalence relation containing the above rule will be denoted $=_d$ (*equality up to dummies*). Two terms are equal up to dummies if they are structurally identical up to dummy subterms of the same type.

The notion of *convertibility* we shall work with, denoted with $\cong$, is the smallest compatible equivalence relation containing the β-rule, the δ-rule and the d-rule. Obviously,

$$M =_d N \Rightarrow M \cong N \text{ and } M \overset{*}{\to}_{\beta\delta} N \Rightarrow M \cong N$$

Note that, if $\diamond$ is a reflexive, transitive, compatible relation containing the d-rule, then for all M, N

$$A \diamond B \Rightarrow |M|_A \diamond |N|_B$$

Hence, in particular,

$$A =_d B \Rightarrow |M|_A =_d |N|_B \text{ and } A \cong B \Rightarrow |M|_A \cong |N|_B$$

We shall now state a few more properties of the above relations, omitting the relatively simple and standard proofs.

LEMMA 2. *The relations $\overset{*}{\to}_{\beta\delta}, =_d$ and $\cong$ are all preserved by substitution.*

Equality up to dummies commutes with β-reduction:

LEMMA 3. *If $M =_d N$ and $M \to_{\beta\delta} M'$ then there exists N' such that $N \overset{*}{\to}_{\beta\delta} N'$ and $M' =_d N'$.*

Using the previous lemma it is easy to prove the following result, that generalizes the existence of a common reduct between two β-convertible terms:

LEMMA 4. *$M \cong N$ if and only if there exist two terms M' and N' such that $M \overset{*}{\to}_{\beta\delta} M' =_d N' \overset{*}{\leftarrow}_{\beta\delta} N$.*

DEFINITION 5.

1. *A term is in* normal form *if it does not contain any redex.*
2. *A term is in* weak head normal form *(*whnf*) if it belongs to the class of terms U generated by the following grammar, where M, N are arbitrary terms (U, D stands for Up and Down):*

   $$\begin{array}{rcl}U & := & \lambda x : M.N \mid |N|_M \mid D \\ D & := & x \mid s \mid \Pi x : M.U \mid (|N|_U\ M) \mid (D\ M)\end{array}$$

   where $(|N|_U M)$ is not a δ-redex.
3. *A term is in* head normal form *if it belongs to the class of terms U generated by the following grammar, where M is any term:*

   $$\begin{array}{rcl}U, U' & := & \lambda x : M.U \mid |U|_M \mid D \\ D & := & x \mid s \mid \Pi x : M.U \mid (|U|_{U'}\ M) \mid (D\ M)\end{array}$$

   where $(|U|_{U'} M)$ is not a δ-redex.

It is easy to see that any term in normal form must also be in (weak) head normal form. Luckily, as we shall see, the class of well typed (weak) head normal forms has a simpler syntactical structure. The notion of head normal form is useful for consistency issues (see Section 2.5).

A term is *strongly normalizing* if it does not originate infinite reduction sequences. Formally, it is defined as the accessible part of the (inverted) reduction relation.

2.2 Type rules

DEFINITION 6. *A PTS is specified by a triple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ where $\mathcal{S}$ is a set of* sorts, *$\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ is a set of* axioms, *$\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ is a set of* rules.

DEFINITION 7. *A* Type Judgment *is a triple of the form*

$$\Gamma \vdash A : B$$

expressing the fact that A has type B in the (raw) context Γ. A and B are raw terms, and Γ is an ordered list of items of the form $x : C$ assigning a type C (a raw term) to the variable x.

We denote with $\Gamma[N/x]$ the obvious extension of the notion of substitution to contexts (we assume that $x \notin \Gamma$); in particular: $\varnothing[N/x] = \varnothing$ and $(\Gamma, z : A)[N/x] = \Gamma[N/x], z : A[N/x]$.

DEFINITION 8. *The DPTS (PTS with dummies) determined by the specification $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ is the Type Judgment axiomatized by the following rules (in* start *and* weak *we assume $x \notin \Gamma$):*

$$(axiom) \quad \frac{(s_1, s_2) \in \mathcal{A}}{\vdash s_1 : s_2}$$

$$(start) \quad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$$

$$(weak) \quad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}$$

$$(prod) \quad \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash \Pi x : A.B : s_3}$$

$$(appl) \quad \frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B[N/x]}$$

$$(lambda) \quad \frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}$$

$$(dummy) \quad \frac{\Gamma \vdash A : B}{\Gamma \vdash |A|_B : B}$$

$$(conv) \quad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \qquad B \cong C}{\Gamma \vdash A : C}$$

2.3 Main Properties of DPTS

As usual, we shall use the notation $\Gamma \vdash A : B : C$ to mean $\Gamma \vdash A : B$ and $\Gamma \vdash B : C$. A raw context Γ is *legal* if there exist A, B such that $\Gamma \vdash A : B$.

The first major property of DPTS is the substitution lemma:

LEMMA 9. *Assume*

$$\Gamma, x : A, \Delta \vdash B : C \qquad\qquad \Gamma \vdash D : A$$

then

$$\Gamma, \Delta[D/x] \vdash B[D/x] : C[D/x]$$

Proof. The proof is by induction on the notion of derivation. We only treat the new cases of dummies and convertibility.

- (*dummy*) Suppose the last rule of the derivation is a dummy rule:
$$\frac{\Gamma, x : A, \Delta \vdash B : C}{\Gamma, x : A, \Delta \vdash |B|_C : C}$$
By induction hypothesis we know that
$$\Gamma, \Delta[D/x] \vdash B[D/x] : C[D/x]$$
and hence by an application of the dummy rules we get
$$\Gamma, \Delta[D/x] \vdash |B[D/x]|_{C[D/x]} : C[D/x]$$
Since, by definition, $|B[D/x]|_{C[D/x]} = |B|_C[D/x]$ the proof is complete.
- (*conv*) Suppose the last rule of the derivation is a conv rule:
$$\frac{\Gamma, x : A, \Delta \vdash B : C \qquad \Gamma, x : A, \Delta \vdash C' : s \qquad C \cong C'}{\Gamma, x : A, \Delta \vdash A : C'}$$
By induction hypothesis we know that
$$\Gamma, \Delta[D/x] \vdash B[D/x] : C[D/x]$$
and
$$\Gamma, \Delta[D/x] \vdash C'[D/x] : s[D/x] \equiv s$$
Using the fact that substitution preserves convertibility (Lemma 2), we have $C[D/x] \cong C'[D/x]$, hence we can apply the convertibility rule to close the goal. □

The following results generalize the rules *ax* and *weak*, allowing a more liberal treatment of the context.

LEMMA 10. *(Start lemma). Let Γ be a legal context, then:*

1. *if $(s_1, s_2) \in \mathcal{A}$ then $\Gamma \vdash s_1 : s_2$;*
2. *if $(x : A) \in \Gamma$ then $\Gamma \vdash x : A$.*

Proof. If Γ is legal, there exist A, B such that $\Gamma \vdash A : B$. The proof is a simple induction on this derivation. □

LEMMA 11. *(Weakening lemma). If $\Gamma, \Delta \vdash B : C$ and $\Gamma \vdash A : s$ then $\Gamma, x : A, \Delta \vdash B : C$, provided $x \notin \Gamma, \Delta$.*

Proof. The proof is by induction on the derivation of $\Gamma, \Delta \vdash B : C$. We only consider a couple of cases.

- (*axiom*) if the last rule is an axiom, we use the Start Lemma
- (*weak*) suppose the last rule of the derivation is a weakening rule:
$$\frac{\Gamma, \Delta \vdash B : C \qquad \Gamma\Delta \vdash A' : s'}{\Gamma, \Delta, x : A' \vdash B : C}$$
If Δ is empty A must be equal to A' and we have nothing to prove. Otherwise, by induction hypothesis we know that the two sequents $\Gamma, x : A, \Delta \vdash B : C$ and $\Gamma, x : A, \Delta \vdash A' : s'$ are provable, hence we close the goal with an application of the (*weak*) rule. □

The following result, known under the name of generation lemma, states the hypothesis required to derive a type assignment $\Gamma \vdash A : B$, according to the shape of A. The relevant cases, for the purposes of proving the subject reduction property, are functions, products and dummies; the other cases are however used in the uniqueness theorem in the next section.

LEMMA 12. *(Generation lemma) For all Γ, A, B, C*

1. *if $\Gamma \vdash s : C$ then there exists $(s_1, s_2) \in \mathcal{A}$ such that $C \cong s_2$*
2. *if $\Gamma \vdash x : C$ then there exist a sort s and a term B such that $\Gamma \vdash B : s$ and $(x : B) \in \Gamma$*
3. *if $\Gamma \vdash \Pi x : A.B : C$ then there exists $(s_1, s_2, s_3) \in \mathcal{R}$ such that $\Gamma \vdash A : s_1$ and $\Gamma, x : A \vdash B : s_2$ and $C \cong s_3$;*
4. *if $\Gamma \vdash (M\ N) : C$ then there exist A, B such that $\Gamma \vdash M : \Pi x : A.B$ and $\Gamma \vdash N : A$ and $C \cong B[N/x]$;*
5. *if $\Gamma \vdash \lambda x : A.M : C$ then there exist s, B such that $\Gamma \vdash \Pi x : A.B : s$ and $\Gamma, x : A \vdash M : B$ and $C \cong \Pi x : A.B$;*
6. *if $\Gamma \vdash |A|_B : C$ then $\Gamma \vdash A : B$ and $C \cong B$;*

Proof. Consider a derivation of $\Gamma \vdash A : C$ in the above cases. The rules of weakening and conversion do not change the term A [4] and we may assume A has been introduced by the last rule (we are implicitly using the transitivity of $\cong$). In the cases above, this must respectively be a (*prod*) (*lambda*) or a (*dummy*) rule, and the result follows by inspection of this rule. □

The following corollary states the important fact that the type of a type is always a sort.

LEMMA 13. *(Type validity) If $\Gamma \vdash A : B$ then there exists a sort s such that either $B \equiv s$ or $\Gamma \vdash B : s$.*

Proof. An easy induction on the derivation of $\Gamma \vdash A : B$. Let us just consider the interesting case of the application. Suppose the last rule is
$$\frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B[N/x]}$$
Since $\Pi x : A.B \not\equiv s$, then by induction hypothesis there exists a sort s such that $\Gamma \vdash \Pi x : A.B : s$. By the generation lemma for products, there exist s_1 and s_2 such that $(s_1, s_2, s) \in \mathcal{R}$, $\Gamma \vdash A : s_1$ and $\Gamma, x : A \vdash B : s_2$. Finally, by the substitution lemma, $\Gamma \vdash B[N/x] : s_2[N/x] \equiv s_2$. □

As a corollary of the previous result, we obtain that no term convertible with an abstraction can be inhabited. Indeed, by the generation lemma the type of an abstraction must be convertible with a product, which is not convertible with a sort. The inhabitation problem for dummy types is more interesting, since according to our rules a dummy term can have as type a sort. In section 2.5, we shall prove that dummy types cannot be inhabited by closed terms in normal form.

THEOREM 14. *(Subject reduction theorem for DPTS's)*

$$\Gamma \vdash A : B \wedge A \overset{*}{\rightarrow}_{\beta\delta} A' \Rightarrow \Gamma \vdash A' : B$$

In order to prove the previous theorem we need a stronger induction hypothesis, allowing reduction inside Γ; at the same time we shall generalize the statement to take equality up to dummies into account.

DEFINITION 15. *We denote with $\overset{?}{\rightarrow}$ the smallest* reflexive*, compatible relation containing the rules β, δ and d. The relation $\overset{?}{\rightarrow}$ is*

[4] when we work with De Bruijn notation, this is not entirely true for weakening, requiring some syntactical shuffling with the lifting function. This is also the case where we need the Weakening Lemma 11.

extended to contexts in the obvious (parallel) way: $\Gamma \overset{?}{\to} \Gamma'$ *if and only if* $\Gamma = x_1 : A_1, \dots, x_n : A_n$, $\Gamma' = x_1 : A'_1, \dots, x_n : A'_n$, *and* for all i, $A_i \overset{?}{\to} A'_i$.

Then, the Subject Reduction Theorem is an obvious corollary of the following result (this technique seems to be slightly simpler than the one used in [7]):

LEMMA 16.

$$\Gamma \vdash A : B \wedge \Gamma \overset{?}{\to} \Gamma' \wedge A \overset{?}{\to} A' \Rightarrow \Gamma' \vdash A' : B$$

Proof. The proof is by induction on the derivation of $\Gamma \vdash A : B$. Let us investigate some significant cases:

– (*appl*) The last rule is the application rule:

$$\frac{\Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B[N/x]}$$

As induction hypothesis, we know that,

$$\Gamma \overset{?}{\to} \Gamma' \wedge M \overset{?}{\to} M' \to \Gamma' \vdash M' : \Pi x : A.B$$

and

$$\Gamma \overset{?}{\to} \Gamma' \wedge N \overset{?}{\to} N' \to \Gamma' \vdash N' : A$$

If $(M\ N) \overset{?}{\to} P$, three cases are possible: (1) $P = (M'\ N')$ where $M \overset{?}{\to} M'$ and $N \overset{?}{\to} N'$; (2) $M = \lambda x : C.Q$ and $P = Q[N/x]$; (3) $M = |D|_{\Pi x:C.Q}$ and $P = |D\ N|_{Q[N/x]}$. We treat independently the three subcases:

1. supposing $\Gamma \overset{?}{\to} \Gamma'$, by induction hypothesis we have $\Gamma' \vdash M' : \Pi x : A.B$ and $\Gamma' \vdash N' : A$, and hence by the application rule

$$\Gamma' \vdash (M'\ N') : B[N'/x]$$

Since convertibility is preserved by substitution (note that $N \overset{?}{\to} N'$ implies $N \cong N'$), we conclude, as expected

$$\Gamma' \vdash (M'\ N') : B[N/x]$$

2. we know that $\Gamma \vdash \lambda x : C.Q : \Pi x : A.B$ and if $\Gamma \overset{?}{\to} \Gamma'$, the induction hypothesis tells us that

$$\Gamma' \vdash \lambda x : C.Q : \Pi x : A.B$$

We use the generation lemma to conclude that there exist s, B' such that $(i.)$ $\Gamma' \vdash \Pi x : C.B' : s$, $(ii.)$ $\Gamma', x : C \vdash Q : B'$ and $(iii.)$ $\Pi x : A.B \cong \Pi x : C.B'$. From (iii.), using Lemma 4, we obtain $A \cong C$ and $B \cong B'$.
Since $N \overset{?}{\to} N$, we know by induction hypothesis that $\Gamma' \vdash N : A$, and since $A \cong C$, by convertibility we also get $\Gamma' \vdash N : C$.
By $(ii.)$ and the substitution lemma we now conclude

$$\Gamma' \vdash Q[N/x] : B'[N/x]$$

and finally we use the fact that convertibility is preserved by substitution to conclude, as expected,

$$\Gamma' \vdash Q[N/x] : B[N/x]$$

3. we know that $\Gamma \vdash |D|_{\Pi x:C.Q} : \Pi x : A.B$ and if $\Gamma \overset{?}{\to} \Gamma'$, the induction hypothesis tells us that

$$\Gamma' \vdash |D|_{\Pi x:C.Q} : \Pi x : A.B$$

By the generation lemma we have $(i.)$ $\Gamma' \vdash D : \Pi x : C.Q$, and $(ii.)$ $\Pi x : C.Q \cong \Pi x : A.B$, that implies $C \cong A$ and $Q \cong B$.
By induction hypothesis $\Gamma' \vdash N : A$, and by congruence $\Gamma' \vdash N : C$. By application $\Gamma' \vdash (D\ N) : Q[N/x]$, and hence by the dummy rule

$$\Gamma' \vdash |D\ N|_{Q[N/x]} : Q[N/x]$$

Since convertibility is preserved by substitution, we conclude, as expected

$$\Gamma' \vdash |D\ N|_{Q[N/x]} : B[N/x]$$

– (*dummy*) the last rule is a dummy rule

$$\frac{\Gamma \vdash A : B}{\Gamma \vdash |A|_B : B}$$

If $|A|_B \overset{?}{\to} P$ then $P = |A'|'_B$ where $A \overset{?}{\to} A'$ and $B \overset{?}{\to} B'$ (hence $B \cong B'$). Provided $\Gamma \overset{?}{\to} \Gamma'$, the induction hypothesis tells us that $\Gamma' \vdash A' : B$. Since $B \cong B'$ we also have $\Gamma' \vdash A' : B'$, and by the dummy rule

$$\Gamma' \vdash |A'|_{B'} : B'$$

Using again the convertibility between B and B' we get the goal:

$$\Gamma' \vdash |A'|_{B'} : B$$

COROLLARY 17. *(Type reduction property)*

$$\Gamma \vdash A : B \wedge B \overset{*}{\to}_{\beta\delta} B' \Rightarrow \Gamma \vdash A : B'$$

Proof. If $\Gamma \vdash A : B$ then by Lemma 13, $B \equiv s$ or $\Gamma \vdash B : s$ for some sort s. In the first case $B' \equiv s$ and we are done. In the second case, by the Subject Reduction Theorem 14, $\Gamma \vdash B' : s$ and we may apply convertibility to get the goal.

Let us now extend the relation $=_d$ to contexts in the usual way: $\Gamma =_d \Gamma'$ if and only if $\Gamma = x_1 : A_1, \dots, x_n : A_n$, $\Gamma' = x_1 : A'_1, \dots, x_n : A'_n$, and *for all* i, $A_i =_d A'_i$. Then, as another corollary of Lemma 16, we obtain the following result.

THEOREM 18.

$$\Gamma \vdash A : B \wedge \Gamma =_d \Gamma' \wedge A =_d A' \Rightarrow \Gamma' \vdash A' : B$$

2.4 Uniqueness of types

Uniqueness of types may only hold under the assumption the sorts and products get unique types according to the specification in $\mathcal{A}$ and $\mathcal{R}$.

DEFINITION 19. *A DPTS* (S, A, R) *is* singly sorted *if for all sorts* s_1, s_2, s_3, s_4

1. $(s_1, s_2) \in \mathcal{A} \wedge (s_1, s_3) \in \mathcal{A} \Rightarrow s_2 \equiv s_3$
2. $(s_1, s_2, s_3) \in \mathcal{R} \wedge (s_1, s_2, s_4) \in \mathcal{R} \Rightarrow s_3 \equiv s_4$

THEOREM 20. *(Uniqueness of Types for singly sorted DPTS)*

$$\Gamma \vdash A : B \wedge \Gamma \vdash A : C \Rightarrow B \cong C$$

Proof. By induction on the structure of A, exploiting in each case the corresponding generation lemma. We treat a couple of cases:

– if $A \equiv s$, by the generation lemma there exist $(s, s_1), (s, s_2) \in \mathcal{A}$ such that $B \cong s_1$ and $C \cong s_2$. If the DPTS is singly sorted, then $s_1 \equiv s_2$ and by transitivity $B \cong C$.
– if $A \equiv |M|_N$, by the generation lemma $N \cong B$ and $N \cong C$, hence by transitivity $B \cong C$.

COROLLARY 21. *(Subject conversion for singly sorted DPTS) If* $\Gamma \vdash A_1 : B_1$ *and* $\Gamma \vdash A_2 : B_2$*, then* $A_1 \cong A_2$ *implies* $B_1 \cong B_2$.

Proof. The statement follows easily from Lemma 4, Theorem 14, Theorem 18, and Theorem 20. □

Even if the DPTS is not singly sorted, we can follow the proof of Theorem 20 to obtain a technical lemma needed in Section 2.5.

LEMMA 22. *If $\Gamma \vdash A : B$ and $\Gamma \vdash A : s_2$ for some sort s_2, then there exists a sort s_1 such that $B \cong s_1$.*

2.5 Inhabitation Properties

In this section we will show that some interesting types are inhabited just by terms having no normal form. It will follow that such types are empty in a normalizing DPTS (see Section 3).

We begin with a well-known technical result.

LEMMA 23. *(Free variable lemma)*
If $x_1 : C_1, \ldots, x_n : C_n \vdash A : B$ then $FV(A) \subseteq \{x_1, \ldots, x_n\}$.

Proof. By induction on the derivation of the type judgment. □

Then we use the generation lemma to restrict a bit the shape of head normal forms for well-typed terms.

LEMMA 24. *A well-typed term M in head normal form must belong to the class of terms U generated by the following grammar (where M is a generic term):*

$$U := \lambda x : M.U \mid |U|_M \mid s \mid \Pi x : M.U \mid D$$
$$D := x \mid (D\ M)$$

Proof. By the generation lemma it is easy to conclude that sorts, products, and non-functional dummies cannot be applied to any term. □

Two important corollaries follow almost immediately.

LEMMA 25. *if $\vdash A : B$ and the h.n.f. of B is not a product or a sort, then A is not normalizing.*

Proof. If A has a normal form, then we can suppose (Theorem 14) that A is in h.n.f. Now we proceed by induction on the structure of the spine of A taking Lemma 24 into account.

- (sort): $A \equiv s_1$ gives $B \cong s_2$ for some sort s_2 by Lemma 12(1), that is excluded;
- (prod): $A \equiv \Pi x : N.M$ gives $B \cong s$ for some sort s by Lemma 12(3), that is excluded;
- (lambda): $A \equiv \lambda x : N.M$ gives $B \cong \Pi x : N.C$ for some C by Lemma 12(5), that is excluded;
- (var): $A \equiv (x\ N_1\ \ldots\ N_n)$ is excluded because $x \in FV(A) = \emptyset$ by Lemma 23;
- (dummy): $A \equiv |M|_N$ gives $\vdash M : N$ and $B \cong N$ by Lemma 12(6). So the inductive hypothesis concludes. □

This lemma states that dummy types ($B \equiv |M|_N$) can be inhabited only by terms without a normal form.

LEMMA 26. *if $\vdash A : B$ and $B \cong \Pi x : s.x$ for a sort s, then A is not normalizing.*

Proof. Following the previous lemma, if A is normalizing we can assume $A = \lambda x : s.A'$ with A' in h.n.f. because the cases (sort), (prod), and (var) are excluded. Then Lemma 12(5) gives $x : s \vdash A' : B'$ with $B' \cong x$. Proceeding on the spine of A', the cases (sort), (prod), and (lambda) are excluded. So we are left with $A' \equiv (x\ N_1\ \ldots\ N_n)$ in which $n = 0$ since Lemma 10(2) yields $\vdash x : s$. Now Lemma 22 applied to $x : s \vdash x : B'$ and $x : s \vdash x : s$ gives $B' \cong s$, which is excluded by $B' \cong x$. □

This lemma states that the false proposition ($B \equiv \Pi a : \star.a$) of a logical DPTS is inhabited only by terms without a normal form.

2.6 Candidates of Reducibility

The Candidates of Reducibility [21] are subsets of λ-terms enjoying certain closure conditions used to establish some properties of various typed λ-calculi, such as the strong normalization and the confluence of reduction (see e.g. [17]).

In this section we recall such subsets in the context of a DPTS.

Firstly, we set some definitions. In particular the subset of strongly normalizing terms will hereafter be denoted by sn.

DEFINITION 27. *A term is* neutral *or* simple *if it is not of the form $\lambda x : N.M$ or $|M|_N$.*

The intuition underlying this definition is that whenever M is neutral and $(M\ N) \to_{\beta\delta} L$ then either $L = (M'\ N)$ with $M \to_{\beta\delta} M'$, or $L = (M\ N')$ with $N \to_{\beta\delta} N'$.

Our definition of a candidate of reducibility is inspired by Tait's saturation conditions ii and iii [38].

DEFINITION 28. *A subset $\mathcal{C}$ of terms (closed under α-conversion) is a* candidate of reducibility *(c.r.) if the following conditions hold:*

- *(S1) if $M \in \mathcal{C}$ then $M \in \mathsf{sn}$;*
- *(S2) if $L \in \mathsf{sn}$, $N \in \mathsf{sn}$, and $(M[N/x]\ N_1\ \ldots\ N_i\ \ldots\ N_n) \in \mathcal{C}$, then $((\lambda x : L.M)\ N\ N_1\ \ldots\ N_i\ \ldots\ N_n) \in \mathcal{C}$;*
- *(S3) if $M \in \mathsf{sn}$ is neutral and in weak head normal form, and if $N_1 \in \mathsf{sn}, \ldots N_i \in \mathsf{sn}, \ldots N_n \in \mathsf{sn}$, then $(M\ N_1\ \ldots\ N_i\ \ldots\ N_n) \in \mathcal{C}$;*
- *(S4) if $(M\ L_1\ \ldots\ L_i\ \ldots\ L_n) \in \mathcal{C}$ and $N \in \mathsf{sn}$, then $(|M|_N\ L_1\ \ldots\ L_i\ \ldots\ L_n) \in \mathcal{C}$.*

The class of candidates will be denoted by cr.

Our conditions S1, S2 and S3 easily follow from Girard's conditions CR1, CR2, CR3: in particular S1 is CR1, S2 is the typed version of Tait's condition ii, and S3 is a generalization of Tait's condition iii. For instance we can show that CR3 implies S3. To this aim we recall the following:

- (CR2) if $M \in \mathcal{C}$ and $M \to_{\beta\delta} M'$ then $M' \in \mathcal{C}$;
- (CR3) if M is neutral and if $M' \in \mathcal{C}$ for all M' such that $M \to_{\beta\delta} M'$, then $M \in \mathcal{C}$;

LEMMA 29. *The condition CR3 implies the condition S3.*

Proof. we denote with $\delta(M)$ the maximum number of steps in which $M \in \mathsf{sn}$ reduces to its normal form. Then we proceed by induction on $d \equiv \delta(M) + \delta(N_1) + \ldots + \delta(N_n)$. If $d = 0$ then $(M\ N_1\ \ldots\ N_n) \in \mathcal{C}$ being $(M\ N_1\ \ldots\ N_n)$ neutral and normal (CR3). If $d > 0$ and $(M\ N_1\ \ldots\ N_n) \to_{\beta\delta} L$, then we have two cases: $L = (M'\ N_1\ \ldots\ N_n)$ with $M \to_{\beta\delta} M'$, gives $M' \in \mathsf{sn}$ neutral and in weak head normal form with $\delta(M') < \delta(M)$. Therefore $L \in \mathcal{C}$ and CR3 concludes. $C = (M\ N_1\ \ldots\ N_i'\ \ldots\ N_n)$ with $N_i \to_{\beta\delta} N_i'$ for some i, gives $N_i' \in \mathsf{sn}$ with $\delta(N_i') < \delta(N_i)$. Therefore $L \in \mathcal{C}$ and CR3 concludes. □.

Next, we state some results on candidates:

LEMMA 30. *(properties of candidates)*

1. *cr contains sn;*
2. *cr is closed under arbitrary intersection;*
3. *if $\mathcal{C}_1 \in \mathsf{cr}$ and $\mathcal{C}_2 \in \mathsf{cr}$, then* $\mathcal{C}_1 \Rightarrow \mathcal{C}_2 \equiv \{M \mid \forall N \in \mathcal{C}_1.\ (M\ N) \in \mathcal{C}_2\} \in \mathsf{cr}$.

3. The Calculus of Constructions with Dummies

In this section we give some results concerning DCC: the DPTS based on λC [7]. In particular we prove the so-called Classification Lemma 35 and the strong normalization property (Theorem 48).

Thus, Lemma 25 and Lemma 26 yield crucial consequences:

THEOREM 31. *The system* DCC *satisfies:*

1. *consistency:* $\vdash A : \Pi a : \star . a$ *is excluded;*
2. *emptiness of dummy types:* $\vdash A : |B|_C$ *is excluded.*

We would like to stress that Theorem 31 holds even if the reduction δ is dropped from the system, since if $\Gamma \vdash A : B$ holds without δ, then it holds with δ as well.

Due to the Classification Lemma, the valid terms and contexts of DCC belong to the following stratified grammar:

$$\begin{array}{lrl}
Term & & ::= \Box \mid K \mid T \mid M \\
(kind) & H, K & ::= \star \mid \Pi a : H.K \mid \Pi x : U.K \mid |K|_\Box \\
(constructor) & T, U & ::= a \mid \Pi a : H.T \mid \Pi x : U.T \mid \lambda a : H.T \\
& & \mid \quad \lambda x : U.T \mid T\ U \mid T\ N \mid |T|_H \\
(object) & M, N & ::= x \mid \lambda a : H.M \mid \lambda x : U.M \mid M\ U \\
& & \mid \quad M\ N \mid |M|_U \\
Context & \Gamma & ::= \star \mid \Gamma, (a : H) \mid \Gamma, (x : U)
\end{array}$$

where we split the set V of variables into the subset $V^\Box$ of type variables, ranged over by a, and the subset $V^\star$ of object variables, ranged over by x. In this case, the set S of sorts is $\{\Box, \star\}$.

The notation $\Pi(\Box, \Box)$ will refer to the construction $\Pi a : H.K$ and to its type rule $prod(\Box, \Box)$. The notations $\lambda(\Box, \Box)$ and $@(\Box, \Box)$ will refer to the associated constructions $\lambda a : H.T$ and $(T\ U)$ respectively, and to their type rules as well. The notation $\beta(\Box, \Box)$ will refer to the β reduction rule involving $\lambda a : H.T$ and $(T\ U)$. The notation $\delta(\Box, \Box)$ will refer to the δ reduction rule involving $|T|_H$ and $(T\ U)$. These conventions will apply also to the other sort combinations available in DCC, which are: $(\Box, \star)$, $(\star, \Box)$, $(\star, \star)$. The notations $D(\Delta)$ [5], $D(\Box)$, $D(\star)$ will refer to the constructions $|K|_\Box$, $|T|_H$, $|M|_U$ respectively, and to their type rules as well.

It is convenient to state the type rules $\lambda(s_1, s_2)$, $@(s_1, s_2)$, $D(s)$, and $Conv(s)$ in the way we display below:

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, (x : A) \vdash C : B \qquad \Gamma \vdash \Pi x : A.B : s_2}{\Gamma \vdash \lambda x : A.C : \Pi x : A.B}$$

$$\frac{\Gamma \vdash N : A : s_1 \qquad \Gamma \vdash M : \Pi x : A.B : s_2}{\Gamma \vdash (M\ N) : B[N/x]}$$

$$\frac{\Gamma \vdash A : B : C}{\Gamma \vdash |A|_B : B}$$

$$\frac{\Gamma \vdash A : B : s \qquad \Gamma \vdash C : s \qquad B \cong C}{\Gamma \vdash A : C}$$

These rules are compatible with $(lambda)$, $(appl)$, $(dummy)$, and $(conv)$ respectively, since the additional premises are admissible by Lemma 10(2), the Type Validity lemma 13, the Subject Conversion lemma 21, and the Generation lemma 12(3).

3.1 Classification Lemma for DCC

In this section we prove the Classification Lemma for DCC using a *degree* assignment: a classical notion in typed λ-calculus [7].

DEFINITION 32. *The* degree *of a term A in the context Γ, denoted by $\#(A)_\Gamma$, is defined if Γ and A belong to the stratified grammar of* DCC *and is undefined otherwise. In particular:*

$$\#(\Box)_\Gamma \equiv 4; \quad \#(K)_\Gamma \equiv 3; \quad \#(T)_\Gamma \equiv 2; \quad \#(M)_\Gamma \equiv 1.$$

If $v \in V$, then $\#(v)_\Gamma$ is derived from the declaration of v in Γ.

LEMMA 33. *If $v \in FV(A)$ and $\#(B)_\Gamma = \#(v)_\Gamma$, then $\#(A[B/v])_\Gamma = \#(A)_\Gamma$, or these degrees are undefined.*

[5] Δ is intended as the third sort in a three-sorted logical PTS.

Proof. The degree of a term depends just on the degrees of its subterms, so it is preserved by replacement of equal subterms (w.r.t. the degree) as long as captures are avoided. □

Here is the main result of the section: a soundness theorem.

THEOREM 34. *If $\Gamma \vdash A : B$ in* DCC*, then $\#(A)_\Gamma = \#(B)_\Gamma$ - 1.*

Proof. We proceed by induction on $\Gamma \vdash A : B$.

- (axiom): $\Gamma \equiv \star$, and $A \equiv \star$, and $B \equiv \Box$, give $\Gamma \in Context$ and $\#(\star)_\Gamma = 3 = 4 - 1 = \#(\Box)_\Gamma$ - 1;
- (start): $\Gamma \equiv \Delta, (v : B)$, and $A \equiv v$, with the premise $\Delta \vdash B : s$, give $\Delta \overset{\text{IH}}{\in} Context$ and $\#(B)_\Delta \overset{\text{IH}}{=} \#(s)_\Delta - 1 \in \{3, 2\}$; so $\Gamma \in Context$ and $\#(v)_\Gamma = \#(B)_\Gamma - 1$ by definition;
- (weak): $\Gamma \equiv \Delta, (v : C)$, with the premises $\Delta \vdash C : s$, and $\Delta \vdash A : B$, gives $\Delta \overset{\text{IH}}{\in} Context$ and $\#(C)_\Delta \overset{\text{IH}}{=} \#(s)_\Delta - 1 \in \{3, 2\}$; so $\Gamma \in Context$ and $\#(A)_\Gamma \overset{\text{IH}}{=} \#(B)_\Gamma - 1$;
- (prod): $A \equiv \Pi v : C.E$, and $B \equiv s_2$, with the premises $\Gamma \vdash C : s_1$, and $\Gamma, (v : C) \vdash E : s_2$, give $\Gamma \overset{\text{IH}}{\in} Context$, and $\#(C)_\Gamma \overset{\text{IH}}{=} \#(s_1)_\Gamma - 1 \in \{3, 2\}$, and $\#(E)_{\Gamma,(v:C)} \overset{\text{IH}}{=} \#(s_2)_{\Gamma,(v:C)} - 1 = \#(s_2)_\Gamma - 1 \in \{3, 2\}$; so $\#(\Pi v : C.E)_\Gamma = \#(E)_{\Gamma,(v:C)} = \#(s_2)_\Gamma - 1$;
- (lambda): $A \equiv \lambda v : C.D$, and $B \equiv \Pi v : C.E$, with the premises $\Gamma \vdash C : s_1$, and $\Gamma, (v : C) \vdash D : E$, and $\Gamma \vdash \Pi v : C.E : s_2$, give $\Gamma \overset{\text{IH}}{\in} Context$, and $\#(C)_\Gamma \overset{\text{IH}}{=} \#(s_1)_\Gamma - 1 \in \{3, 2\}$, and $\#(D)_{\Gamma,(v:C)} \overset{\text{IH}}{=} \#(E)_{\Gamma,(v:C)} - 1 = \#(\Pi v : C.E)_\Gamma - 1 \overset{\text{IH}}{=} \#(s_2)_\Gamma - 2 \in \{2, 1\}$; so $\#(\lambda v : C.D)_\Gamma = \#(D)_{\Gamma,(v:C)} = \#(E)_{\Gamma,(v:C)} - 1 = \#(\Pi v : C.E)_\Gamma - 1$;
- (appl): $A \equiv (F\ D)$, and $B \equiv E[D/v]$, with the premises $\Gamma \vdash D : C : s_1$, and $\Gamma \vdash F : \Pi v : C.E : s_2$, give $\Gamma \overset{\text{IH}}{\in} Context$, and $\#(D)_\Gamma \overset{\text{IH}}{=} \#(C)_\Gamma - 1 \overset{\text{IH}}{=} \#(s_1)_\Gamma - 2 \in \{2, 1\}$, and $\#(F)_\Gamma \overset{\text{IH}}{=} \#(\Pi v : C.E)_\Gamma - 1 \overset{\text{IH}}{=} \#(s_2)_\Gamma - 2 \in \{2, 1\}$; so $\#(F\ D)_\Gamma = \#(F)_\Gamma = \#(\Pi v : C.E)_\Gamma - 1 = \#(E)_{\Gamma,(v:C)} - 1 = \#(E[D/v])_{\Gamma,(v:C)} - 1 = \#(E[D/v])_\Gamma - 1$ by Lemma 33 since $\#(v)_{\Gamma,(v:C)} = \#(C)_\Gamma - 1 = \#(D)_\Gamma = \#(D)_{\Gamma,(v:C)}$;
- (dummy) $A \equiv |D|_B$, with the premise $\Gamma \vdash D : B$, gives $\Gamma \in Context$, and $\#(D)_\Gamma \overset{\text{IH}}{=} \#(B)_\Gamma - 1$; so $\#(B)_\Gamma \in \{4, 3, 2\}$, and $\#(D)_\Gamma \in \{3, 2, 1\}$, therefore $\#(|D|_B)_\Gamma = \#(D)_\Gamma = \#(B)_\Gamma - 1$;
- (conv) the premises $\Gamma \vdash A : C : s$, and $\Gamma \vdash B : s$, with $C \cong B$, give $\Gamma \in Context$, and $\#(C)_\Gamma \overset{\text{IH}}{=} \#(s)_\Gamma - 1 \overset{\text{IH}}{=} \#(B)_\Gamma$; so $\#(A)_\Gamma \overset{\text{IH}}{=} \#(C)_\Gamma - 1 = \#(B)_\Gamma - 1$. □

The Classification Lemma is a simple corollary of soundness.

LEMMA 35. *(Classification) Any term and context valid in* DCC *belongs to the stratified grammar of* DCC.

Proof. The context Γ is valid when there exist A and B such that $\Gamma \vdash A : B$, so Theorem 34 states that $\#(A)_\Gamma$ and $\#(B)_\Gamma$ are defined, thus $\Gamma \in Context$. The term A is valid when there exist Γ and B such that either $\Gamma \vdash A : B$, or $\Gamma \vdash B : A$. In both cases Theorem 34 states that $\#(A)_\Gamma$ is defined, thus $A \in Term$. □.

3.2 Strong Normalization for DCC: an Overview

In the following sections we show that DCC enjoys the strong normalization property. The literature provides for two classes of proofs that CC (λC) is strongly normalizing (s.n.) and our aim is to extend one of these proofs. Some proofs have a multiple-step structure being based on the strong normalization of the system F_ω ($\lambda\omega$) [7, 18, 20, 36], and are rather involved. Other authors give single-step proofs [2, 15, 19, 26, 28, 32, 40], which are semantical in that s.n. terms are interpreted as elements of a suitable model. Of these, Geuvers [19] seems to give the simplest model to treat and extend for our purposes: the one of set-theoretic functions from c.r.'s to c.r.'s of any arity and order.

Intuitively, the proof runs as follows: our statement is that every valid term of DCC is strongly normalizing. Such a term is either $\Box$, for which we conclude immediately, or a term typed in a context. For each term A such that $\Gamma \vdash A : B$, we find a subset $[\![B]\!]$ of terms such that $A \in [\![B]\!] \in \mathsf{cr}$, and we conclude by S1.

Due to the shape of the *(appl)* type rule, this statement must be proved in a general form where substitution appears explicitly. So we state our claim as $(\!|A|\!)_\rho \in [\![B]\!]_\xi \in \mathsf{cr}$ where ρ and ξ are evaluation functions for the free variables of A and B respectively. In particular $(\!|A|\!)_\rho$ is the simultaneous substitution in A of every $v \in FV(A)$ for the term $\rho(v)$. That is to say that $\rho : V \to \mathit{Term}$.

We take the identity substitution as a canonical interpretation ρ.

As for the evaluation ξ, we apply it after the interpretation $[\![\cdot]\!]$ so the codomain of ξ is the codomain of $[\![\cdot]\!]$ rather than Term.

Moreover we see that $[\![\cdot]\!]_\xi$ is not applied to an object, so we can define it in such a way that ξ does not evaluate object variables. This happens because the constructors of DCC depend on objects in a uniform way, so these dependences can be safely forgotten. Similarly, kinds depend on constructors, and thus on objects, in a uniform way as well, so these dependences can be forgotten too. This erasing mechanism underlies the map from λC to $\lambda\omega$ used in [7] to connect the strong normalization of these systems.

Our system DCC features functional constructors (i.e. terms of the form $\lambda a : H.T$, or $\lambda(\Box, \Box)$), which are functions from constructors of a given type to constructors. Thus, it is straightforward to interpret these terms as functions of a given arity and order from c.r.'s to c.r.'s. This situation requires to define the class of the functions whose arity and order is derived from a kind H, especially because the definition of $[\![\Pi a : H.K]\!]_\xi$ and $[\![\Pi a : H.T]\!]_\xi$ involves a universal quantification over this class quite naturally.

We will denote such a class with $\langle\!\langle H \rangle\!\rangle$ and we will show that $\Gamma \vdash A : B$ implies $[\![A]\!]_\xi \in \langle\!\langle B \rangle\!\rangle$ when B is a kind or the term $\Box$. It should be noted that the definition of $\langle\!\langle B \rangle\!\rangle$ does not involve an evaluation function since B depends uniformly on its free variables.

It is also important to stress that for each H, a default function in the class $\langle\!\langle H \rangle\!\rangle$ exists, which will be used to define a canonical interpretation ξ based on the formerly mentioned context Γ.

As a concluding remark, we note that the codomain of $[\![\cdot]\!]$ must be the class of the functions from cr to cr of any arity and order. This class will be denoted hereafter by $\mathsf{cr}^{\to}$. Instead, the codomain of $\langle\!\langle \cdot \rangle\!\rangle$ must be the family of the classes of the functions from cr to cr of a given arity and order. This family will be denoted by $\mathsf{CR}^{\to}$.

3.3 Strong Normalization for DCC: the Proof

At this point we are ready to develop the detailed proof of strong normalization for the system DCC. We begin by giving a formal definition to the concepts we discussed in the previous section.

DEFINITION 36. *The family* $\mathsf{CR}^{\to}$ *is inductively defined by:*

$$\mathsf{cr} \in \mathsf{CR}^{\to}; \qquad \textit{if } \alpha \in \mathsf{CR}^{\to} \textit{ and } \beta \in \mathsf{CR}^{\to}, \textit{ then } \beta^\alpha \in \mathsf{CR}^{\to}$$

where β^α *is the class of the functions from* α *to* β.

For each class $\alpha \in \mathsf{CR}^{\to}$*, the* canonical element *of* α *is denoted by* $c\ \alpha$ *and is defined by cases on* α *as follows:*

$$c\ \mathsf{cr} \equiv \mathsf{sn}; \qquad c\ \beta^\alpha \equiv \_ : \alpha \mapsto c\ \beta.$$

The class of the functions from cr *to* cr *of any arity and order is:*

$$\mathsf{cr}^{\to} \equiv \bigcup_{\alpha \in \mathsf{CR}^{\to}} \alpha.$$

As expected, we easily have $c\ \alpha \in \alpha$ for each $\alpha \in \mathsf{CR}^{\to}$.

DEFINITION 37. *The interpretation* $\langle\!\langle \cdot \rangle\!\rangle$ *from Term to* $\mathsf{CR}^{\to}$ *is a partial function defined by cases on* $\Box$ *and on kinds as follows:*

$$\begin{aligned}
\langle\!\langle \Box \rangle\!\rangle &\equiv \mathsf{cr};\\
\langle\!\langle \star \rangle\!\rangle &\equiv \mathsf{cr};\\
\langle\!\langle \Pi a : H.K \rangle\!\rangle &\equiv \langle\!\langle K \rangle\!\rangle^{\langle\!\langle H \rangle\!\rangle};\\
\langle\!\langle \Pi x : U.K \rangle\!\rangle &\equiv \langle\!\langle K \rangle\!\rangle;\\
\langle\!\langle |K|_\Box \rangle\!\rangle &\equiv \mathsf{cr}.
\end{aligned}$$

DEFINITION 38. *Given an evaluation* ξ *from* $V^\Box$ *to* $\mathsf{cr}^{\to}$*, the interpretation* $[\![\cdot]\!]_\xi$ *from Term to* $\mathsf{cr}^{\to}$ *is a partial function defined by cases on* $\Box$*, on kinds, and on constructors as follows:*

$$\begin{aligned}
[\![\Box]\!]_\xi &\equiv \mathsf{sn};\\
[\![\star]\!]_\xi &\equiv \mathsf{sn};\\
[\![\Pi a : H.K]\!]_\xi &\equiv [\![H]\!]_\xi \Rightarrow \textstyle\bigcap_{f \in \langle\!\langle H \rangle\!\rangle} [\![K]\!]_{\xi(a \equiv f)};\\
[\![\Pi x : U.K]\!]_\xi &\equiv [\![U]\!]_\xi \Rightarrow [\![K]\!]_\xi;\\
[\![|K|_\Box]\!]_\xi &\equiv c\ \langle\!\langle \Box \rangle\!\rangle = \mathsf{sn};\\
[\![a]\!]_\xi &\equiv \xi(a);\\
[\![\Pi a : H.T]\!]_\xi &\equiv [\![H]\!]_\xi \Rightarrow \textstyle\bigcap_{f \in \langle\!\langle H \rangle\!\rangle} [\![T]\!]_{\xi(a \equiv f)};\\
[\![\Pi x : U.T]\!]_\xi &\equiv [\![U]\!]_\xi \Rightarrow [\![T]\!]_\xi;\\
[\![\lambda a : H.T]\!]_\xi &\equiv f : \langle\!\langle H \rangle\!\rangle \mapsto [\![T]\!]_{\xi(a \equiv f)};\\
[\![\lambda x : U.T]\!]_\xi &\equiv [\![T]\!]_\xi;\\
[\![T\ U]\!]_\xi &\equiv [\![T]\!]_\xi([\![U]\!]_\xi);\\
[\![T\ N]\!]_\xi &\equiv [\![T]\!]_\xi;\\
[\![|T|_H]\!]_\xi &\equiv c\ \langle\!\langle H \rangle\!\rangle.
\end{aligned}$$

We say that ξ agrees *with a context* Γ*, and we write* $\xi \vDash \Gamma$*, when* $\xi(a) \in \langle\!\langle H \rangle\!\rangle$ *for each type variable declaration* $(a : H) \in \Gamma$.

We say that ξ *is* canonical *for the context* Γ *when* $\xi(a) \equiv c\ \langle\!\langle H \rangle\!\rangle$ *for each type variable declaration* $(a : H) \in \Gamma$.

As expected, if ξ is canonical for Γ, then ξ agrees with Γ.

DEFINITION 39. *Given an evaluation* ρ *from* V *to Term, the interpretation* $(\!|A|\!)_\rho$ *of* $A \in \mathit{Term}$ *is the simultaneous substitution in* A *of* $\rho(v)$ *for each* $v \in FV(A)$.

We say that ξ *and* ρ agree *with a context* Γ*, and we write* $\xi, \rho \vDash \Gamma$*, when* $\xi \vDash \Gamma$ *and* $\rho(v) \in [\![B]\!]_\xi$ *for each variable declaration* $(v : B) \in \Gamma$.

We say that ρ *is* canonical *for the context* Γ *when* $\rho(v) \equiv v$ *for each variable declaration* $(v : B) \in \Gamma$.

Generally, we cannot say that ξ and ρ agree with Γ when ξ and ρ are canonical for Γ, since $[\![B]\!]_\xi \in \mathsf{cr}$ may be false. However, we shall see that this condition holds when Γ is valid in DCC.

We are now ready to prove the sequence of lemmas that will take us to the strong normalization for DCC.

Firstly, we prove a technical result.

LEMMA 40. *If* $B \in \mathcal{C} \in \mathsf{cr}$ *and* $(\!|A|\!)_{\rho(v \equiv C)} \in \mathcal{C}_2 \in \mathsf{cr}$ *for every* $C \in \mathcal{C}_1 \in \mathsf{cr}$*, then* $\lambda v : B.(\!|A|\!)_{\rho(v \equiv v)} \in \mathcal{C}_1 \Rightarrow \mathcal{C}_2$.

Proof. Take $C \in \mathcal{C}_1$ and rewrite $(\!|A|\!)_{\rho(v \equiv C)}$ as $(\!|A|\!)_{\rho(v \equiv v)}[C/v]$. Then, condition S2 of $\mathcal{C}_2$ gives $(\lambda v : B.(\!|A|\!)_{\rho(v \equiv v)})\ C \in \mathcal{C}_2$ and we conclude by the definition of the c.r. constructor $\Rightarrow$. $\Box$

Secondly, we prove three substitution lemmas.

LEMMA 41. $\langle\!\langle A[B/v] \rangle\!\rangle = \langle\!\langle A \rangle\!\rangle$*, or both members are undefined.*

Proof. If v is bound in A, then we conclude immediately. Otherwise, both the interpretation and the substitution distribute on the subterms of A. We proceed by induction on A.

- $\Pi(\Box, \Box)$: $A = \Pi a : H.K$ gives $\langle\!\langle (\Pi a : H.K)[B/v] \rangle\!\rangle = \langle\!\langle \Pi a : H[B/v].K[B/v] \rangle\!\rangle = \langle\!\langle K[B/v] \rangle\!\rangle^{\langle\!\langle H[B/v] \rangle\!\rangle} \overset{\text{IH}}{=} \langle\!\langle K \rangle\!\rangle^{\langle\!\langle H \rangle\!\rangle} = \langle\!\langle \Pi a : H.K \rangle\!\rangle$;

- $\Pi(\star,\Box)$: $A = \Pi x : U.K$ gives $\langle\!\langle(\Pi x : U.K)[B/v]\rangle\!\rangle = \langle\!\langle\Pi x : U[A/v].K[B/v]\rangle\!\rangle = \langle\!\langle K[B/v]\rangle\!\rangle \overset{\text{IH}}{=} \langle\!\langle K\rangle\!\rangle = \langle\!\langle\Pi x : U.K\rangle\!\rangle$;
- $D(\Delta)$: $A = |K|_\Box$ gives $\langle\!\langle|K|_\Box[B/v]\rangle\!\rangle = \langle\!\langle|K[B/v]|_\Box\rangle\!\rangle = \mathsf{cr} = \langle\!\langle|K|_\Box\rangle\!\rangle$. □

LEMMA 42. *$[\![A[U/a]]\!]_\xi = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$, or both members are undefined.*

Proof. If v is bound in A, then we have $[\![A[U/a]]\!]_\xi = [\![A]\!]_\xi = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$. Otherwise, $A = a$ or A is compound. In this case both the interpretation and the substitution distribute on the subterms of A. We proceed by induction on A.

- $\Pi(\Box,\Box)$: $A = \Pi b : H.K$ gives $[\![A[U/a]]\!]_\xi = [\![H[U/a]]\!]_\xi \Rightarrow \bigcap_{f\in\langle\!\langle H[U/a]\rangle\!\rangle} [\![K[U/a]]\!]_{\xi(b\equiv f)} = [\![H[U/a]]\!]_\xi \Rightarrow \bigcap_{f\in\langle\!\langle H\rangle\!\rangle} [\![K[U/a]]\!]_{\xi(b\equiv f)} \overset{\text{IH}}{=} [\![H]\!]_{\xi(a\equiv[\![U]\!]_\xi)} \Rightarrow \bigcap_{f\in\langle\!\langle H\rangle\!\rangle} [\![K]\!]_{\xi(a\equiv[\![U]\!]_\xi, b\equiv f)} = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$ by Lemma 41;
- $\Pi(\star,\Box)$: $A = \Pi x : V.K$ gives $[\![A[U/a]]\!]_\xi = [\![V[U/a]]\!]_\xi \Rightarrow [\![K[U/a]]\!]_\xi \overset{\text{IH}}{=} [\![V]\!]_{\xi(a\equiv[\![U]\!]_\xi)} \Rightarrow [\![H]\!]_{\xi(a\equiv[\![H]\!]_\xi)} = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$;
- $D(\Delta)$: $A = |K|_\Box$ gives $[\![A[U/a]]\!]_\xi = \mathsf{sn} = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$;
- $V^\Box$: $A = a$ gives $[\![A[U/a]]\!]_\xi = [\![U]\!]_\xi = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$;
- $\Pi(\Box,\star)$: like $\Pi(\Box,\Box)$;
- $\Pi(\star,\star)$: like $\Pi(\star,\Box)$;
- $\lambda(\Box,\Box)$: $B = \lambda b : H.T$ gives $[\![A[U/a]]\!]_\xi = f : \langle\!\langle H[U/a]\rangle\!\rangle \mapsto [\![T[U/a]]\!]_{\xi(b\equiv f)} = f : \langle\!\langle H\rangle\!\rangle \mapsto [\![T[U/a]]\!]_{\xi(b\equiv f)} \overset{\text{IH}}{=} f : \langle\!\langle H\rangle\!\rangle \mapsto [\![T]\!]_{\xi(a\equiv[\![U]\!]_\xi, b\equiv f)} = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$ by Lemma 41;
- $\lambda(\star,\Box)$: $A = \lambda x : V.T$ gives $[\![A[U/a]]\!]_\xi = [\![T[U/a]]\!]_\xi \overset{\text{IH}}{=} [\![T]\!]_{\xi(a\equiv[\![U]\!]_\xi)} = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$;
- $@(\Box,\Box)$: $A = T\ V$ gives $[\![A[U/a]]\!]_\xi = [\![T[U/a]]\!]_\xi([\![V[U/a]]\!]_\xi) \overset{\text{IH}}{=} [\![T]\!]_{\xi(a\equiv[\![U]\!]_\xi)}([\![V]\!]_{\xi(a\equiv[\![U]\!]_\xi)}) = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$;
- $@(\star,\Box)$: $A = T\ N$ gives $[\![A[U/a]]\!]_\xi = [\![T[U/a]]\!]_\xi \overset{\text{IH}}{=} [\![T]\!]_{\xi(a\equiv[\![U]\!]_\xi)} = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$;
- $D(\Box)$: $A = |T|_H$ gives $[\![A[U/a]]\!]_\xi = c\,\langle\!\langle H[U/a]\rangle\!\rangle = c\,\langle\!\langle H\rangle\!\rangle = [\![A]\!]_{\xi(a\equiv[\![U]\!]_\xi)}$ by Lemma 41. □

LEMMA 43. *$[\![A[N/x]]\!]_\xi = [\![A]\!]_\xi$, or both members are undefined.*

Proof. If x is bound in A, then we conclude immediately. Otherwise, both the interpretation and the substitution distribute on the subterms of A as for Lemma 42 without the case $V^\Box$. □

Thirdly, we prove two conversion lemmas.

LEMMA 44. *$A_1 \cong A_2$ implies $\langle\!\langle A_1\rangle\!\rangle = \langle\!\langle A_2\rangle\!\rangle$, or both members are undefined.*

Proof. We proceed by induction on $A_1 \cong A_2$. $\beta\delta$-reductions must not be treated because they involve constructors ($\beta\delta(\Box,\Box)$, $\beta\delta(\star,\Box)$) or objects ($\beta\delta(\Box,\star)$, $\beta\delta(\star,\star)$), which are erased by the interpretation $\langle\!\langle\cdot\rangle\!\rangle$. Reflexivity, symmetry, and transitivity are immediate. Compatibility is also straightforward because the interpretation distributes on the subterms of A_1 and A_2. d-equality must be treated explicitly.

- $\Pi(\Box,\Box)$: $A_1 = \Pi a_1 : H_1.K_1$ and $A_2 = \Pi a_2 : H_2.K_2$ give $\langle\!\langle A_1\rangle\!\rangle = \langle\!\langle K_1\rangle\!\rangle^{\langle\!\langle H_1\rangle\!\rangle} \overset{\text{IH}}{=} \langle\!\langle K_2\rangle\!\rangle^{\langle\!\langle H_2\rangle\!\rangle} = \langle\!\langle A_2\rangle\!\rangle$;
- $\Pi(\star,\Box)$: $A_1 = \Pi x_1 : U_1.K_1$ and $A_2 = \Pi x_2 : U_2.K_2$ give $\langle\!\langle A_1\rangle\!\rangle = \langle\!\langle K_1\rangle\!\rangle \overset{\text{IH}}{=} \langle\!\langle K_2\rangle\!\rangle = \langle\!\langle A_2\rangle\!\rangle$;
- $D(\Delta)$: $A_1 = |K_1|_\Box$ and $A_2 = |K_2|_\Box$ give $\langle\!\langle A_1\rangle\!\rangle = \mathsf{cr} = \langle\!\langle A_2\rangle\!\rangle$.

If $\langle\!\langle A_1\rangle\!\rangle$ and $\langle\!\langle A_2\rangle\!\rangle$ are defined, then A_1 and A_2 are in weak head normal form. Therefore they must start with the same construction in order to be convertible. □

LEMMA 45. *$A_1 \cong A_2$ implies $[\![A_1]\!]_\xi = [\![A_2]\!]_\xi$, or both members are undefined.*

Proof. We proceed by induction on $A_1 \cong A_2$. $\beta\delta$-reductions on objects must not be treated because objects are erased by the interpretation $[\![\cdot]\!]_\xi$. Reflexivity, symmetry, and transitivity are immediate. Compatibility is also straightforward because the interpretation distributes on the subterms of A_1 and A_2. d-equality and $\beta\delta$-reduction on constructors must be treated explicitly.

- $\Pi(\Box,\Box)$: $A_1 = \Pi a : H_1.K_1$ and $A_2 = \Pi a : H_2.K_2$ give $[\![A_1]\!]_\xi = [\![H_1]\!]_\xi \Rightarrow \bigcap_{f\in\langle\!\langle H_1\rangle\!\rangle} [\![K_1]\!]_{\xi(a\equiv f)} \overset{\text{IH}}{=} [\![H_2]\!]_\xi \Rightarrow \bigcap_{f\in\langle\!\langle H_1\rangle\!\rangle} [\![K_2]\!]_{\xi(a\equiv f)} = [\![H_2]\!]_\xi \Rightarrow \bigcap_{f\in\langle\!\langle H_2\rangle\!\rangle} [\![K_2]\!]_{\xi(a\equiv f)} = [\![A_2]\!]_\xi$ by Lemma 44;
- $\Pi(\star,\Box)$: $A_1 = \Pi x : U_1.K_1$ and $A_2 = \Pi x : U_2.K_2$ give $[\![A_1]\!]_\xi = [\![U_1]\!]_\xi \Rightarrow [\![K_1]\!]_\xi \overset{\text{IH}}{=} [\![U_2]\!]_\xi \Rightarrow [\![K_2]\!]_\xi = [\![A_2]\!]_\xi$;
- $D(\Delta)$: $A_1 \equiv |K_1|_\Box$ and $A_2 \equiv |K_2|_\Box$ give $[\![A_1]\!]_\xi = \mathsf{sn} = [\![A_2]\!]_\xi$;
- $\Pi(\Box,\star)$: like $\Pi(\Box,\Box)$;
- $\Pi(\star,\star)$: like $\Pi(\star,\Box)$;
- $\lambda(\Box,\Box)$: $A_1 = \lambda a : H_1.T_1$ and $A_2 = \lambda a : H_2.T_2$ give $[\![A_1]\!]_\xi = f : \langle\!\langle H_1\rangle\!\rangle \mapsto [\![T_1]\!]_{\xi(a\equiv f)} \overset{\text{IH}}{=} f : \langle\!\langle H_1\rangle\!\rangle \mapsto [\![T_2]\!]_{\xi(a\equiv f)} = f : \langle\!\langle H_2\rangle\!\rangle \mapsto [\![T_2]\!]_{\xi(a\equiv f)} = [\![A_2]\!]_\xi$ by Lemma 44;
- $\lambda(\star,\Box)$: $A_1 = \lambda x : U_1.T_1$ and $A_2 = \lambda x : U_2.T_2$ give $[\![A_1]\!]_\xi = [\![T_1]\!]_\xi \overset{\text{IH}}{=} [\![T_2]\!]_\xi = [\![A_2]\!]_\xi$;
- $@(\Box,\Box)$: $A_1 = T_1\ U_1$ and $A_2 = T_2\ U_2$ give $[\![A_1]\!]_\xi = [\![T_1]\!]_\xi([\![V_1]\!]_\xi) \overset{\text{IH}}{=} [\![T_2]\!]_\xi([\![V_2]\!]_\xi) = [\![B_2]\!]_\xi$; if $[\![A_1]\!]_\xi$ or $[\![A_2]\!]_\xi$ is defined, then the function application in $\mathsf{cr}^\rightarrow$ is well typed;
- $@(\star,\Box)$: $A_1 = T_1\ N_1$ and $A_2 = T_2\ N_2$ give $[\![A_1]\!]_\xi = [\![T_1]\!]_\xi \overset{\text{IH}}{=} [\![T_2]\!]_\xi = [\![B_2]\!]_\xi$;
- $D(\Box)$: $A_1 \equiv |T_1|_{H_1}$ and $A_2 \equiv |T_2|_{H_2}$ give $[\![A_1]\!]_\xi = c\,\langle\!\langle H_1\rangle\!\rangle = c\,\langle\!\langle H_2\rangle\!\rangle = [\![A_2]\!]_\xi$ by Lemma 44;
- $\beta(\Box,\Box)$: in this case we have the following $[\![(\lambda a : H.T)\ U]\!]_\xi = (f : \langle\!\langle H\rangle\!\rangle \mapsto [\![T]\!]_{\xi(a\equiv f)})([\![U]\!]_\xi) = [\![T]\!]_{\xi(a\equiv[\![U]\!]_\xi)} = [\![T[U/a]]\!]_\xi$ by Lemma 42; if $[\![A_1]\!]_\xi$ or $[\![A_2]\!]_\xi$ is defined, then the function application in $\mathsf{cr}^\rightarrow$ is well typed, which means that $[\![U]\!]_\xi \in \langle\!\langle H\rangle\!\rangle$;
- $\beta(\star,\Box)$: in this case we have the following $[\![(\lambda x : U.T)\ N]\!]_\xi = [\![T]\!]_\xi = [\![T[N/x]]\!]_\xi$ by Lemma 43;
- $\delta(\Box,\Box)$: in this case we have the following $[\![|T|_{\Pi a:H.K}\ U]\!]_\xi = c\,\langle\!\langle\Pi a : H.K\rangle\!\rangle([\![U]\!]_\xi) = c\,\langle\!\langle K\rangle\!\rangle = c\,\langle\!\langle K[U/a]\rangle\!\rangle = [\![|T\ U|]\!]_{K[U/a]}$ by Lemma 41; if $[\![A_1]\!]_\xi$ or $[\![A_2]\!]_\xi$ is defined, then the function application in $\mathsf{cr}^\rightarrow$ is well typed, which means that $[\![U]\!]_\xi \in \langle\!\langle H\rangle\!\rangle$;
- $\delta(\star,\Box)$: in this case we have the following $[\![|T|_{\Pi x:U.K}\ N]\!]_\xi = c\,\langle\!\langle\Pi x : U.K\rangle\!\rangle = c\,\langle\!\langle K\rangle\!\rangle = c\,\langle\!\langle K[N/x]\rangle\!\rangle = [\![|T\ N|]\!]_{K[N/x]}$ by Lemma 41. □

This soundness theorem is the main result of the section:

THEOREM 46. *If $\Gamma \vdash A : B$, then for any evaluation ξ and ρ*

1. *$\xi \vDash \Gamma$ implies $[\![A]\!]_\xi \in \langle\!\langle B\rangle\!\rangle$ when B is a kind or $\Box$;*
2. *$\xi, \rho \vDash \Gamma$ implies $(\!|A|\!)_\rho \in [\![B]\!]_\xi \in \mathsf{cr}$.*

Proof. We proceed by induction on $\Gamma \vdash A : B$.

- Axiom: $A \equiv \star$ and $B \equiv \Box$ give:
  1. $[\![\star]\!]_\xi = \mathsf{sn} \in \langle\!\langle\Box\rangle\!\rangle = \mathsf{cr}$;
  2. $(\!|\star|\!)_\rho = \star \in [\![\Box]\!]_\xi = \mathsf{sn} \in \mathsf{cr}$;

- Start($\Box$): $\Gamma \equiv \Delta, (a : H)$, and $A \equiv a$, and $B \equiv K$, with the premise $\Delta \vdash H : \Box$, give:
  1. $[\![a]\!]_\xi = \xi(a) \in \langle\!\langle H \rangle\!\rangle$;
  2. $(\!|a|\!)_\rho = \rho(a) \in [\![H]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$ since $\xi \vDash \Delta$;
- Start($\star$): $\Gamma \equiv \Delta, (x : U)$, and $A \equiv x$, and $B \equiv U$, with the premise $\Delta \vdash U : \star$, give:
  2. $(\!|x|\!)_\rho = \rho(x) \in [\![U]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \star \rangle\!\rangle = \mathsf{cr}$ since $\xi \vDash \Delta$;
- Weak($\Box$): $\Gamma \equiv \Delta, (a : H)$, with the premise $\Delta \vdash A : B$, gives:
  1. by inductive hypothesis since $\xi \vDash \Delta$;
  2. by inductive hypothesis since $\xi, \rho \vDash \Delta$;
- Weak($\star$): like Weak($\Box$);
- $\Pi(\Box, \Box)$: $A \equiv \Pi a : H.K$, and $B \equiv \Box$, with the premises $\Gamma \vdash H : \Box$, and $\Gamma, (a : H) \vdash K : \Box$, give $[\![H]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$, and $[\![K]\!]_{\xi(a \equiv f)} \overset{\mathrm{IH}}{\in} \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$ for every $f \in \langle\!\langle H \rangle\!\rangle$ since $\xi(a \equiv f) \vDash \Gamma, (a : H)$, so:
  1. $[\![\Pi a : H.K]\!]_\rho = [\![H]\!]_\xi \Rightarrow \bigcap_{f \in \langle\!\langle K1 \rangle\!\rangle} [\![K]\!]_{\xi(a \equiv f)} \in \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$;
  2. $(\!|\Pi a : H.K|\!)_\rho = \Pi a : (\!|H|\!)_\rho . (\!|K|\!)_{\rho(a \equiv a)} \overset{\mathrm{IH}}{\in} [\![\Box]\!]_\xi = \mathsf{sn} \in \mathsf{cr}$; the inductive hypotheses are applicable because $\xi(a \equiv c\, \langle\!\langle H \rangle\!\rangle), \rho(a \equiv a) \vDash \Gamma, (a : H)$ since $a \in [\![H]\!]_\xi$ follows from condition S3;
- $\Pi(\star, \Box)$: $A \equiv \Pi x : U.K$, and $B \equiv \Box$, with the premises $\Gamma \vdash U : \star$, and $\Gamma, (x : U) \vdash K : \Box$, give $[\![U]\!]_\xi \in \mathsf{cr}$, and $[\![K]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$ since $\xi \vDash \Gamma, (x : U)$, so:
  1. $[\![\Pi x : U.K]\!]_\rho = [\![U]\!]_\xi \Rightarrow [\![K]\!]_\xi \in \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$.
  2. $(\!|\Pi x : U.K|\!)_\rho = \Pi x : (\!|U|\!)_\rho . (\!|K|\!)_{\rho(x \equiv x)} \overset{\mathrm{IH}}{\in} [\![\Box]\!]_\xi = \mathsf{sn} \in \mathsf{cr}$; the inductive hypotheses are applicable because $\xi, \rho(x \equiv x) \vDash \Gamma, (x : U)$ since $x \in [\![U]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \star \rangle\!\rangle = \mathsf{cr}$ follows from condition S3;
- $\Pi(\Box, \star)$: like $\Pi(\Box, \Box)$ without statement 1;
- $\Pi(\star, \star)$: like $\Pi(\star, \Box)$ without statement 1;
- $\lambda(\Box, \Box)$: $A \equiv \lambda a : H.T$, and $B \equiv \Pi a : H.K$, with the premises $\Gamma \vdash H : \Box$, and $\Gamma, (a : H) \vdash T : K$, and $\Gamma \vdash B : \Box$, give $(\!|T|\!)_{\rho(a \equiv A)} \overset{\mathrm{IH}}{\in} \bigcap_{f \in \langle\!\langle H \rangle\!\rangle} [\![K]\!]_{\xi(a \equiv f)} \overset{\mathrm{IH}}{\in} \mathsf{cr}$ for every $A \in [\![H]\!]_\xi$ since $\xi(a \equiv f), \rho(a \equiv A) \vDash \Gamma, (a : H)$ for every $f \in \langle\!\langle H \rangle\!\rangle$ and $A \in [\![H]\!]_\xi$, so:
  1. $[\![\lambda a : H.T]\!]_\xi = f : \langle\!\langle H \rangle\!\rangle \mapsto [\![T]\!]_{\xi(a \equiv f)} \overset{\mathrm{IH}}{\in} \langle\!\langle B \rangle\!\rangle = \langle\!\langle K \rangle\!\rangle^{\langle\!\langle H \rangle\!\rangle}$; the inductive hypotheses are applicable because $\xi(a \equiv f) \vDash \Gamma, (a : H)$ for every $f \in \langle\!\langle H \rangle\!\rangle$;
  2. $(\!|\lambda a : H.T|\!)_\rho = \lambda a : (\!|H|\!)_\rho . (\!|T|\!)_{\rho(a \equiv a)} \overset{\mathrm{IH}}{\in} [\![B]\!]_\xi = [\![H]\!]_\xi \Rightarrow \bigcap_{f \in \langle\!\langle H \rangle\!\rangle} [\![K]\!]_{\xi(a \equiv f)} \overset{\mathrm{IH}}{\in} \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$ by Lemma 40 with the premise $(\!|H|\!)_\rho \in [\![\Box]\!]_\xi \in \mathsf{cr}$;
- $\lambda(\star, \Box)$: $A \equiv \lambda x : U.T$, and $B \equiv \Pi x : U.K$, with the premises $\Gamma \vdash U : \star$, and $\Gamma, (x : U) \vdash T : K$, and $\Gamma \vdash B : \Box$, give $(\!|T|\!)_{\rho(a \equiv A)} \in [\![K]\!]_\xi$ for every $A \in [\![U]\!]_\xi$ since $\xi, \rho(a \equiv A) \vDash \Gamma, (x : U)$ for every $A \in [\![U]\!]_\xi$, so:
  1. $[\![\lambda x : U.T]\!]_\xi = [\![T]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle B \rangle\!\rangle = \langle\!\langle K \rangle\!\rangle$; the inductive hypothesis is applicable because $\xi \vDash \Gamma, (x : U)$;
  2. $(\!|\lambda x : U.T|\!)_\rho = \lambda x : (\!|U|\!)_\rho . (\!|T|\!)_{\rho(x \equiv x)} \overset{\mathrm{IH}}{\in} [\![B]\!]_\xi = [\![U]\!]_\xi \Rightarrow [\![K]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$ by Lemma 40 with the premise $(\!|U|\!)_\rho \in [\![\star]\!]_\xi \in \mathsf{cr}$;
- $\lambda(\Box, \star)$: like $\lambda(\Box, \Box)$ without statement 1;
- $\lambda(\star, \star)$: like $\lambda(\star, \Box)$ without statement 1;
- $@(\Box, \Box)$: $A \equiv T\ U$, and $B \equiv K[U/a]$, with the premises $\Gamma \vdash U : H$, and $\Gamma \vdash T : \Pi a : H.K$, give:
  1. $[\![T\ U]\!]_\rho = [\![T]\!]_\xi([\![U]\!]_\xi) \overset{\mathrm{IH}}{\in} \langle\!\langle K \rangle\!\rangle = \langle\!\langle K[U/a] \rangle\!\rangle$ by Lemma 41;
  2. $(\!|T\ U|\!)_\rho = (\!|T|\!)_\rho\ (\!|U|\!)_\rho \overset{\mathrm{IH}}{\in} \bigcap_{f \in \langle\!\langle H \rangle\!\rangle} [\![K]\!]_{\xi(a \equiv f)} \overset{\mathrm{IH}}{\subseteq} [\![K]\!]_{\xi(a \equiv [\![U]\!]_\xi)} = [\![K[U/a]]\!]_\xi \in \mathsf{cr}$ by Lemma 42;
- $@(\star, \Box)$: $A \equiv T\ N$, and $B \equiv K[N/x]$, with the premises $\Gamma \vdash N : U$, and $\Gamma \vdash T : \Pi x : U.K$, give:
  1. $[\![T\ N]\!]_\rho = [\![T]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle \Pi x : U.K \rangle\!\rangle = \langle\!\langle K \rangle\!\rangle = \langle\!\langle K[N/x] \rangle\!\rangle$ by Lemma 41;
  2. $(\!|T\ N|\!)_\rho = (\!|T|\!)_\rho\ (\!|N|\!)_\rho \overset{\mathrm{IH}}{\in} [\![K]\!]_\xi = [\![K[N/x]]\!]_\xi \in \mathsf{cr}$ by Lemma 43;
- $@(\Box, \star)$: like $@(\Box, \Box)$ without statement 1;
- $@(\star, \star)$: like $@(\star, \Box)$ without statement 1;
- $D(\Delta)$: $A \equiv |K|_\Box$, and $B \equiv \Box$, with the premise $\Gamma \vdash K : \Box$, give:
  1. $[\![|K|_\Box]\!]_\xi = \mathsf{sn} \in \langle\!\langle \Box \rangle\!\rangle = \mathsf{cr}$.
  2. $(\!||K|_\Box|\!)_\rho = |(\!|K|\!)_\rho|_\Box \overset{\mathrm{IH}}{\in} [\![\Box]\!]_\xi = \mathsf{sn} \in \mathsf{cr}$; by condition S4;
- $D(\Box)$: $A \equiv |T|_H$, and $B \equiv H$, with the premises $\Gamma \vdash T : H$, and $\Gamma \vdash H : \Box$, give:
  1. $[\![|T|_H]\!]_\xi = \mathcal{C}\langle\!\langle H \rangle\!\rangle \in \langle\!\langle H \rangle\!\rangle$;
  2. $(\!||T|_H|\!)_\rho = |(\!|T|\!)_\rho|_{(\!|H|\!)_\rho} \overset{\mathrm{IH}}{\in} [\![H]\!]_\xi \in \mathsf{cr}$ by condition S4 since $(\!|H|\!)_\rho \overset{\mathrm{IH}}{\in} [\![\Box]\!]_\xi \in \mathsf{cr}$;
- $D(\star)$: like $D(\star)$ without statement 1;
- Conv($\Box$): $A \equiv T$, and $B \equiv K_2$, with the premises $\Gamma \vdash T : K_1$, and $\Gamma \vdash K_2 : \Box$, and $K_1 \cong K_2$, give $[\![K_1]\!]_\xi \overset{\mathrm{IH}}{\in} \mathsf{cr}$ and $[\![K_2]\!]_\xi \overset{\mathrm{IH}}{\in} \mathsf{cr} = \langle\!\langle \Box \rangle\!\rangle$, so:
  1. $[\![T]\!]_\xi \overset{\mathrm{IH}}{\in} \langle\!\langle K_1 \rangle\!\rangle = \langle\!\langle K_2 \rangle\!\rangle$ by Lemma 44;
  2. $(\!|T|\!)_\rho \overset{\mathrm{IH}}{\in} [\![K_1]\!]_\xi = [\![K_2]\!]_\xi \in \mathsf{cr}$ by Lemma 45;
- Conv($\star$): like Conv($\Box$) without statement 1. □

Finally, we give two easy corollaries of the soundness theorem, the second of which states strong normalization for DCC.

LEMMA 47. *If the evaluations ξ and ρ are canonical for the context Γ and Γ is valid in* DCC, *then $\xi, \rho \vDash \Gamma$.*

Proof. We already know that if ξ is canonical for Γ, then $\xi \vDash \Gamma$. What remains to prove is that $\rho(x) = x \in [\![U]\!]_\xi$ for any object variable declaration $(x : U) \in \Gamma$. We proceed by induction on the number of such declarations. If there is none, then we are done. If Γ is valid, then there exists $\Delta \subset \Gamma$ such that $\Delta \vdash U : \star$ by Lemma 10(2). So $\xi, \rho \vDash \Delta$ holds by the induction hypothesis and Theorem 46(1) gives $[\![U]\!]_\xi \in \langle\!\langle \star \rangle\!\rangle = \mathsf{cr}$. Therefore, we conclude $x \in [\![U]\!]_\xi$ by condition S3. □

THEOREM 48. *Every valid term in* DCC *is strongly normalizing.*

Proof. If $A \in \mathit{Term}$ is valid, then $A = \Box \in \mathsf{sn}$, or there exist Γ and B, both valid in DCC, such that $\Gamma \vdash A : B$. Let ξ and ρ be canonical evaluations for Γ (they exist by definition), then Lemma 47 gives $\xi, \rho \vDash \Gamma$, Theorem 46(2) gives $A = (\!|A|\!)_\rho \in [\![B]\!]_\xi \in \mathsf{cr}$, and condition S1 of $[\![B]\!]_\xi$ concludes $A \in \mathsf{sn}$. □

4. Conclusions

Dummy terms seem to provide a simple, yet powerful theoretical tool to investigate irrelevance in type systems. As a proof of concept, in this paper we studied them in the setting of Pure Type Systems, that are a well-known theoretical framework, with a particularly clear and well-assessed meta-theory. In particular, we proved that dummies integrate smoothly with the theory, preserving consistency when the Calculus of Constructions is considered.

Some of the techniques used in this paper only work in the case of PTS and cannot be easily extended to more complex systems, e.g. comprising Inductive types with strong elimination rules. This is the case of the δ-rule, which is not a constituent property of dummies, but a mere technical artifice added in the case of PTS to reduce terms with dummies in a form suitable to entail consistency via structural inspection.

This is a bit surprising, since the δ-rule gives rise to behaviors dangerously close to inconsistency. Consider e.g. the standard impredicative definition of the type of Booleans, namely $\mathcal{B} = \Pi X.X \to X \to X$; let $T = \lambda x, y : \mathcal{B}.x$, $F = \lambda x, y : \mathcal{B}.y$ and not $= \lambda b : \mathcal{B}.b\ \mathcal{B}\ F\ T$. Then, for any dummy boolean $c = |M|_{\mathcal{B}}$, we have

$$\text{not } c = |M|_{\mathcal{B}}\ \mathcal{B}\ F\ T = |M B\ F\ T|_{\mathcal{B}} = c$$

In systems like CC, this is not inconsistent, since we are not able to prove discrimination results, and in particular we cannot prove that $b \neq$ not b; however, this becomes provable when adding strong elimination of inductive types, hence preventing an extension of the δ-rule to case analysis (matching a dummy cannot turn the whole match into a dummy).

In spite of its dangerous nature, in the calculus of constructions, consistency follows more easily for the calculus with the δ-rule then without it. The point is that the δ-rule allows us to recover the subformula property: without this rule, normalization would stop too soon, leaving dummy terms of arbitrary types along the spine of the normal form. It is true that, by the generation lemma, a dummy term can always be removed *from the spine*, replacing it with its content; however, the term we obtain in this way is not any longer in normal form. Normalizing it, we could possibly recreate a configuration similar to the one we started from. The situation is depicted by the following (untyped) term

$$\Delta = |\lambda x.(|x|\ x)|\ \lambda x.(|x|\ x)$$

The term is is normal form; replacing (as a meta-step) $|\lambda x.(|x|\ x)|$ with its witness $\lambda x.(|x|\ x)$ we obtain

$$\Delta' = \lambda x.(|x|\ x)\ \lambda x.(|x|\ x)$$

that in one step reduces back to Δ.

For all he previous reasons, extending the theory of dummies beyond PTS is a challenging topic. We strongly believe in the consistency of dummies (without δ) for the whole Calculus of Inductive Constructions, but the proof of this conjecture (or its confutation) is an open issue.

To help clarifying the issue, we are also currently working at the implementation of dummies inside a prototype version of Matita [6] (which is based, as COQ, on the Calculus of Inductive Constructions). In order to be exploitable in practice, it looks convenient to propagate irrelevance at the level of function parameters. Say that a variable x is a *dummy parameter in* M if all occurrences of x in M are inside dummy terms. Then, we know that the function $F = \lambda x.M$ makes a dummy use of its argument[6]. This means that when converting $F\ M_1$ against $F\ M_2$ we may just assume the convertibility of M_1 and M_2 (even if they are not explicitly declared as dummies) since we know that by reduction they will eventually be embedded in some dummy term. We have a double possibility here: we may just use this information as an heuristic to speed up convertibility, with no logical content, or we may try to integrate this information inside the logical systems, which would definitely lead to a duplication of binders, in a way similar to ICC^*. The big difference is on types of higher order arguments: in the first case we have no logical way to distinguish between a function f that makes a dummy use of its argument from a function that doesn't, since the information is not in the type; on the other side, this is information that *cannot be autonomously inferred* by the system according to the use we make of f, so an explicit declaration is required, that is precisely the place where the duplication of binders becomes extremely painful for the user. So we have an interesting trade-off between precision and usability that must be carefully evaluated.

[6] This is very similar to the (I-LAM) rule of ICC^*, that introduces an *irrelevant binder* in case the variable x does not appear in the term extracted from M.

References

[1] A. Abel. Irrelevance in type theory with a heterogeneous equality judgement. In *Proceedings of Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Saarbrücken, Germany*, volume 6604 of *Lecture Notes in Computer Science*, pages 57–71. Springer, 2011.

[2] T. Altenkirch. *Constructions, Inductive types and Strong Normalization proof.* PhD thesis, University of Edinburgh, UK, 1993.

[3] T. Altenkirch. Extensional equality in intensional type theory. In *LICS*, pages 412–420, 1999.

[4] T. Altenkirch, C. McBride, and J. McKinna. Why dependent types matter. http://sneezy.cs.nott.ac.uk/epigram/, 2005.

[5] A. Asperti, W. Ricciotti, C. Sacerdoti Coen, and E. Tassi. A compact kernel for the Calculus of Inductive Constructions. *Sadhana*, 34(1): 71–144, 2009. doi: 10.1007/s12046-009-0003-3.

[6] A. Asperti, W. Ricciotti, C. S. Coen, and E. Tassi. The matita interactive theorem prover. In *Proceedings of the 23rd International Conference on Automated Deduction (CADE-2011), Wroclaw, Poland*, volume 6803 of *LNCS*, 2011.

[7] H. Barendregt. Lambda Calculi with Types. In Abramsky, Samson and others, editor, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1992.

[8] H. Barendregt. The impact of the lambda calculus in logic and computer science. *Bulletin of Symbolic Logic*, 3(2):181–215, 1997.

[9] B. Barras and B. Bernardo. The implicit calculus of constructions as a programming language with dependent types. In *Foundations of Software Science and Computational Structures, 11th International Conference, FoSSaCS 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 365–379. Springer, 2008.

[10] G. Barthe. The relevance of proof-irrelevance. In *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark*, volume 1443 of *Lecture Notes in Computer Science*, pages 755–768. Springer, 1998.

[11] S. Berardi. Towards a mathematical analysis of the coquand-huet calculus of constructions and the other systems in barendregt's cube. Technical report, Dept.of Computer Science, Carnegie Mellon Unviersity and Dipartimento di Matemtica, Università di Torino, 1988.

[12] B. Bernardo. Towards an implicit calculus of inductive constructions. In *Emerging Trends section of TPHOLs 2009*, To appear, 2009.

[13] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development*. Texts in Theoretical Computer Science. Springer Verlag, 2004. ISBN-3-540-20854-2.

[14] A. Chlipala. An introduction to programming and proving with dependent types in coq. *Journal of Formalized Reasoning*, 3(2):1–93, 2010.

[15] T. Coquand and J. Gallier. A proof of strong normalization for the theory of constructions using a kripke-like interpretation. In *Workshop on Logical Frameworks, Antibes, May 1990, Local Proceedings*, 1990.

[16] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *J. Autom. Reasoning*, 31(1):33–72, 2003. doi: 10.1007/3-540-46508-1_1.

[17] J. Gallier. On girard's "candidates de reductibilité". In P. Odifreddi, editor, *Logic and Computer Science*, pages 123–203. Academic Press, 1990.

[18] H. Geuvers. *Logics and Type Systems*. Ph.D. dissertation, Catholic University Nijmegen, 1993.

[19] H. Geuvers. A short and flexible proof of strong normalization for the calculus of constructions. In P. Dybjer, B. Nordström, and J. M. Smith, editors, *Types for Proofs and Programs, International Workshop TYPES'94, Båstad, Sweden, June 6-10, 1994, Selected Papers*, volume 996 of *Lecture Notes in Computer Science*, pages 14–38, Berlin, Germany, 1995. Springer.

[20] H. Geuvers and M.-J. Nederhof. Modular proof of strong normalization for the calculus of constructions. *J. Funct. Program.*, 1(2): 155–189, 1991.

[21] J.-Y. Girard. Une extension de l'interprétation de gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. E. Fenstad, editor, *Proceedings of the 2nd Scandinavian Logic Symposium, University of Oslo, 1970*, volume 63 of *Studies in logic and the foundations of mathematics*, pages 63–92, Amsterdam, The Netherlands, 1971. North-Holland.

[22] G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. sel4: formal verification of an operating-system kernel. *Commun. ACM*, 53(6):107–115, 2010.

[23] X. Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, 2009.

[24] P. Letouzey. A New Extraction for Coq. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs, Second International Workshop, TYPES 2002, Berg en Dal, The Netherlands, April 24-28, 2002*, volume 2646 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[25] W. Lovas and F. Pfenning. Refinement types for logical frameworks and their interpretation as proof irrelevance. *Logical Methods in Computer Science*, May 2010. To appear.

[26] Z. Luo. *An Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1990.

[27] C. McBride. Elimination with a motive. In *Types for Proofs and Programs, International Workshop, TYPES 2000, Durham, UK, December 8-12, 2000, Selected Papers*, volume 2277 of *Lecture Notes in Computer Science*, pages 197–216. Springer, 2000.

[28] P.-A. Melliès and B. Werner. A generic normalisation proof for pure type systems. In E. Giménez and C. Paulin-Mohring, editors, *Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers*, volume 1512 of *Lecture Notes in Computer Science*, pages 254–276, Berlin, Germany, 1998. Springer.

[29] A. Miquel and B. Werner. The not so simple proof-irrelevant model of CC. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs: International Workshop, TYPES 2002*, volume 2646 of *Lecture Notes in Computer Science*, pages 240–258. Springer-Verlag, 2003.

[30] N. Mishra-Linger and T. Sheard. Erasure and polymorphism in pure type systems. In *Proceedings od Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Budapest, Hungary*, volume 4962 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2008.

[31] G. C. Necula. Proof-Carrying Code. In P. Lee, F. Henglein, and N. Jones, editors, *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 106–119, New York, NY, USA, 1997. ACM.

[32] C.-H. L. Ong and E. Ritter. A generic strong normalization argument: Application to the calculus of constructions. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Computer Science Logic, 7th Workshop, CSL '93, Swansea, United Kingdom, September 13-17, 1993, Selected Papers*, volume 832 of *Lecture Notes in Computer Science*, pages 261–279, Berlin, Germany, 1994. Springer.

[33] S. Owre and N. Shankar. The formal semantics of pvs. Technical Report CSL-97-2R, SRI Technical Report, Revised March 1999.

[34] F. Pfenning. Intensionality, extensionality, and proof irrelevance in modal type theory. In J. Halpern, editor, *Proceedings of the 16th Annual Symposium on Logic in Computer Science (LICS'01)*, pages 221–230, Boston, Ma, USA, June 2001. IEEE Computer Society Press.

[35] J. Reed. Proof irrelevance and strict definitions in a logical framework. Technical Report Senior Thesis, CMU-CS-02-153, Carnegie Mellon Unviersity Technical Report, 2002.

[36] M. H. Sorensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism*, volume 149. Elsevier, 2006.

[37] J. Souyris, V. Wiels, D. Delmas, and H. Delseny. Formal verification of avionics software products. In *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands*, volume 5850 of *Lecture Notes in Computer Science*, pages 532–546. Springer, 2009.

[38] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium, Symposium on Logic Held at Boston, 1972-73*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251, Berlin, Germany, 1975. Springer.

[39] M. Takahashi. Parallel reductions in λ-calculus. *Information and Computation*, 118(1):120–127, 1995.

[40] J. Terlouw. Strong normalization in type systems: A model theoretical approach. *Annals of Pure and Applied Logic*, 73(1):53–78, 1995. A Tribute to Dirk van Dalen.

[41] The Coq Development Team. The Coq proof assistant reference manual.
`http://coq.inria.fr/refman/`, 2010.

[42] J. Torlouw. Een nadere bewijstheoretische analyse van GSTT's. Technical report, Dept.of Computer Science, University of Nijmegen, 1989.

[43] B. Werner. On the strength of proof-irrelevant type theories. *Logical Methods in Computer Science*, 4(3), 2008.

[44] J. Yang and C. Hawblitzel. Safe to the last instruction: automated verification of a type-safe operating system. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2010, Toronto, Ontario, Canada*, pages 99–110. ACM, 2010.