On the Degeneracy of Σ-Types in Presence of Computational Classical Logic

Hugo Herbelin

LIX - INRIA-Futurs - PCRI
École Polytechnique
F-91128 Palaiseau Cedex

Abstract. We show that a minimal dependent type theory based on Σ-types and equality is degenerated in presence of computational classical logic. By computational classical logic is meant a classical logic derived from a control operator equipped with reduction rules similar to the ones of Felleisen's $\mathcal{C}$ or Parigot's μ operators. As a consequence, formalisms such as Martin-Löf's type theory or the (`Set`-predicative variant of the) Calculus of Inductive Constructions are inconsistent in presence of computational classical logic. Besides, an analysis of the role of the η-rule for control operators through a set-theoretic model of computational classical logic is given.

1 Introduction

1.1 Computational Classical Logic

The `call-with-current-continuation` operator is a construct that has been introduced in Scheme a few decades ago. Numerous variants of the original `call-with-current-continuation` have been considered. Felleisen introduced the operators $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{A}$ and studied calculi based on these operators [4]. The SML language introduced the `callcc` and `throw` operators, all equipped with comparable reduction rules.

Griffin [5] showed that Felleisen's $\mathcal{C}$ operator was typable under some conditions of type $\neg\neg A \rightarrow A$ in a simply typed framework, thus extending the Curry-Howard correspondence to classical logic.

Parigot [7] introduced a distinction between the (ordinary) variables and the continuation variables, together with operators μ and brackets, leading to the elegant $\lambda\mu$-calculus. A variant of $\lambda\mu$-calculus based on SML `callcc` (there renamed `catch`) and `throw` has been given in Crolard [3].

Basically, computational classical calculus comes with commutation rules (called structural rules or ζ rules in the context of $\lambda\mu$-calculus), an elimination rule (also called simplification or η_μ rule in the context of $\lambda\mu$-calculus), and an idempotency rule (also called renaming or β_μ rule) .

As an introduction to computational classical logic, we here describe $\lambda\mu$-calculus:

$$\begin{array}{lll} t, u ::= \lambda x.t \mid tu \mid x \mid \mu\alpha.c & & \textit{terms} \\ c \quad ::= [\alpha]t & & \textit{commands} \end{array}$$

To express the reduction rules, we need to define the notion of substitution of a continuation variable α by an evaluation context C for commands (i.e. a command with a placeholder $\{\,\}$):

$$\begin{array}{ll}
x[C/\alpha] & = x \\
(\lambda x.t)[C/\alpha] & = \lambda x.(t[C/\alpha]) \\
(tu)[C/\alpha] & = t[C/\alpha]u[C/\alpha] \\
(\mu\beta.c)[C/\alpha] & = \mu\beta.(c[C/\alpha]) \\
([\alpha]t)[C/\alpha] & = C\{t[C/\alpha]\} \\
([\beta]t)[C/\alpha] & = [\beta](t[C/\alpha]) \qquad \alpha \neq \beta
\end{array}$$

where x in the second rule and β in the fourth rule are chosen such that no capture of free variables in C happens.

In a call-by-name setting, the reduction rules express as:

$$\begin{array}{ll}
(\lambda x.t)u \to t[u/x] & \beta \\
(\mu\alpha.c)t \to \mu\alpha.(c[[\alpha](\{\,\}t)/\alpha]) & \zeta_{app} \\
[\beta]\mu\alpha.c \to c[[\beta]\{\,\}/\alpha] & \beta_\mu \\
\mu\alpha.[\alpha]t \to t \qquad (\alpha \text{ not free in } t) & \eta_\mu
\end{array}$$

Thanks to computational classical logic, classical proofs of simple formulae such as Σ^0_1 formulae eventually normalise to proofs which are (essentially) intuitionistic.

Actually, it is worth to notice that reduction rules close to the rules above were already present in Prawitz' proof of normalisation of the classical extension of natural deduction [9]. What was apparently missing, even after the emergence of the intuitionistic part of the proof-as-program paradigm, was the conviction that they were computationally meaningful in practise.

1.2 Computational vs Platonistic Classical Logic

We oppose computational classical logic to Platonistic classical logic. In Platonistic classical logic, computationally undecidable properties are transcendentally decided by an oracle which transgresses the infinity of time. A Platonistic interpretation of classical logic is given in Sect. 2.5.

1.3 Classical Logic, Axiom of Choice and Definite Description

The axiom of choice (in its functional form) strongly interacts with classical logic. Coquand [2] showed that their conjunction forces propositions to be embeddable in the booleans, thus forbidding non trivial realisability models of the propositional world. Especially, predicative logics with quantification over functions become impredicative in presence of the axiom of choice in its functional form and classical logic (observation attributed to Spector [11]). Also, logics with non degenerated impredicative sets such as the Calculus of Inductive Constructions are inconsistent in presence of the axiom of choice in its functional form and classical logic.

More precisely, what strongly modifies the semantics of a classical logic is not strictly speaking the (functional form of the) axiom of choice but its underlying principle of definite description (also called axiom of unique choice or function construction principle), as shown by Pottinger [8]. Indeed, the functional form of the axiom of choice in type theory

$$(\forall x : A, \exists y : B, R(x, y)) \to \exists f : A \to B, \forall x : A, R(x, f(x))$$

can be shown equivalent in impredicative type theory to the conjunction of its relational form

$$(\forall x : A, \exists y : B, R(x, y)) \to \exists R' \subset R, \forall x : A, \exists ! y : B, R'(x, y)$$

and of the principle of definite description

$$(\forall x : A, \exists ! y : B, R(x, y)) \to \exists f : A \to B, \forall x : A, R(x, f(x)) \ .$$

In presence of classical logic, the principle of definite description alone is enough to force propositions to be embedded in the booleans.

1.4 Computational Classical Logic and Strong Existential Quantification

Classical logic inherits a computational interpretation through the reduction rules assigned to control operators. The functional form of the axiom of choice also inherits a computational interpretation through the reduction rules of strong existential quantification, which is existential quantification equipped with its first and second projections, the type of the second projection being dependent on the first projection.

It is a natural question to study their interaction at a computational level, knowing that they imply at the logical level the existence of a retraction from propositions to the booleans [2].

The computational analysis of the proof of embedding of the propositions within the booleans shows a failure of subject reduction which is due to the dependency of the type of the second projection of the strong existential in the first projection, making untypable the commutation rule of the control operator used for the interpretation with the second projection of the strong existential.

Subject reduction can be restored to the price of assuming proof-irrelevance. Concurrently, it can be shown that the commutation rule of the control operator used for the interpretation with the first projection itself leads to proof-irrelevance, or, more generally, to the degeneracy of the quantification domain. This is the purpose of the current paper.

2 The Degeneracy of Computationally Classical Type Theory with *Σ*-Types

In this section, we use the terminology *Σ*-types to denote indifferently strong existential quantification or usual *Σ*-types with both projections (also referred to as strong sums).

$$\frac{\vdash \pi : A(t)}{\vdash (t, \pi) : \Sigma x.A(x)} \qquad \frac{\vdash \pi : \Sigma x.A(x)}{\vdash \mathsf{prf}\ \pi : A(\mathsf{wit}\ \pi)}$$

$$\frac{t \to u}{\vdash \mathsf{refl} : t = u} \qquad \frac{\vdash \pi_1 : t = u \qquad \vdash \pi_2 : A(t)}{\vdash \mathsf{subst}\ \pi_1\ \pi_2 : A(u)}$$

Fig. 1. Inference rules of TT_Σ

2.1 A Minimal Logic of Σ-Types and Equality

We consider a type theory TT_Σ based on strong existential quantification (i.e. Σ-types) over a unique domain. We use the variable names x, y, ... to range over the elements of the domain. The syntax of proofs and terms is mutually given by

$$\begin{array}{ll} t, u & ::= x \mid \mathsf{wit}\ \pi \\ \pi & ::= (t, \pi) \mid \mathsf{prf}\ \pi \mid \mathsf{refl} \end{array}$$

The syntax of formulae is given by

$$A, B ::= t = u \mid \Sigma x.A$$

The set $FV(A)$ of free variables of A is defined as usual.

This theory is equipped with a single reduction rule on the language of terms.

$$\mathsf{wit}(t, \pi) \to t \qquad\qquad (\iota_{\mathsf{wit}})$$

The inference rules are on Fig. 1.

Proposition 1. *TT_Σ is not degenerated, i.e., for distinct variables x and y, $\nvdash x = y$.*

This is direct by interpreting Σ-types on a domain $\mathcal{D}$ with (at least) two distinct elements $a \neq b$. For distinct elements in $\mathcal{D}$, equality is interpreted as the empty set, otherwise as a singleton set with the unique element interpreting the reflexivity proof. The construction (t, π) is interpreted as pairing and wit and prf as the first and second projections so that $\mathsf{wit}\ (t, \pi)$ and t are identical through the interpretation and the reflexivity rule is valid.

If our only reduction rule is ι_{wit}, it is because it is enough to infer the results shown in the next subsections. We would have got a better-behaved reduction system by adding the rules $\mathsf{prf}(t, \pi) \to \pi$ and $\mathsf{subst}\ \pi_1\ (t, \pi_2) \to (t, \mathsf{subst}\ \pi_1\ \pi_2))$. Moreover, with the premise of the reflexivity rule generalised to the congruent reflexive-symmetric-transitive closure of $\to$, and the extra rule $\mathsf{subst}\ \mathsf{refl}\ \mathsf{refl} \to \mathsf{refl}$ added, we would have got normalisability of the proofs, and, as a consequence, the subformula property and a syntactic evidence of the non-derivability of the degeneracy of the domain.

$$\frac{\Gamma \vdash \pi : A(t)}{\Gamma \vdash (t,\pi) : \Sigma x.A(x)} \qquad \frac{\Gamma \vdash \pi : \Sigma x.A(x)}{\Gamma \vdash \mathsf{prf}\ \pi : A(\mathsf{wit}\ \pi)}$$

$$\frac{t \to u}{\Gamma \vdash \mathsf{refl} : t = u} \qquad \frac{\Gamma \vdash \pi_1 : t = u \qquad \Gamma \vdash \pi_2 : A(t)}{\Gamma \vdash \mathsf{subst}\ \pi_1\ \pi_2 : A(u)}$$

$$\frac{\Gamma, k : \neg A \vdash \pi : A}{\Gamma \vdash \mathtt{cc}_k\ \pi : A} \qquad \frac{\Gamma, k : \neg A \vdash \pi : A}{\Gamma, k : \neg A \vdash \mathtt{th}\ k\ \pi : B}$$

Fig. 2. Inference rules of TT_Σ^{cc}

2.2 ... and its Computationally Classical Extension TT_Σ^{cc}

We now extend the type theory with classical logic. To allow reasoning by contradiction on a formula A, we add the operator $\mathtt{cc}_k\ \pi$ that tries to prove A under the assumption $k : \neg A$. A contradiction is derived at any point of the derivation by applying the new operator $\mathtt{th}\ k\ \pi$ to any proof π of A in the context $k : \neg A$. We thus extend the syntax of proofs with

$$\pi ::= \ldots \mid \mathtt{cc}_k\ \pi \mid \mathtt{th}\ k\ \pi$$

where k ranges over a set of continuation variables. The operators $\mathtt{cc}$ and $\mathtt{th}$ are similar to the $\mathtt{catch}$ and $\mathtt{throw}$ operators studied in Crolard [3]. In terms of the $\lambda\mu$-calculus, $\mathtt{cc}_k\ \pi$ and $\mathtt{th}\ k\ \pi$ are essentially the same as $\mu k.[k]\pi$ and $\mu\_.[k]\pi$ where $\_$ denotes a fresh continuation variables that do not occur in π.

The associated inference rules involved contexts of negated formulae. The rules for $\mathtt{cc}$ and $\mathtt{th}$ are reminiscent of Peirce's law and negation elimination. The full resulting set of inference rules is given on Fig. 2.

Since proofs occur in terms and that we want the classical extension to be computational, we also extend the syntax of terms. This extension requires to extend also the syntax of proofs with a construction which actually occurs only as argument of wit in terms.

$$\begin{aligned} t &::= \ldots \mid \mathtt{cc}_k\ t \\ \pi &::= \ldots \mid \mathtt{th}\ k\ t \end{aligned}$$

We want this classical extension to be computational. We add a subset of the standard computation rules for $\mathtt{cc}$ and $\mathtt{th}$ [3], but adapted to Σ-types. It is just enough to be able to derive the degeneracy of the domain.

$$\begin{array}{lll} \mathsf{wit}(\mathtt{cc}_k\ \pi) \to \mathtt{cc}_k\ \mathsf{wit}(\pi[k(\mathsf{wit}\ \{\ \})/k]) & & (\zeta_{\mathsf{wit}}) \\ \mathtt{cc}_k\ t \qquad\quad \to t & k \text{ not free in } t & (\eta_{\mathsf{cc}}) \end{array}$$

where $[k(\mathsf{wit}\ \{\ \})/k]$ denotes the capture-free substitution which replaces every occurrence of $\mathtt{th}\ k\ t$ with $\mathtt{th}\ k\ (\mathsf{wit}\ t)$. Notice that rule η_{cc} is identical to η_μ along the interpretation of $\mathtt{cc}_k\ t$ as $\mu k.[k]t$.

The terms and proofs of $TT_{\Sigma}^{\mathtt{cc}}$ contain a context binder (the operator $\mathtt{cc}$) but do not include any term or proof binder. Hence, the reduction rules of $TT_{\Sigma}^{\mathtt{cc}}$ do not commit to a call-by-name or call-by-value discipline of reduction. This is in contrast with $\lambda\mu$-calculus where the presence of a term binder (the λ-abstraction) introduces a critical pair (observable on the redex $(\lambda x.t)(\mu\alpha.c)$) that can be resolved by committing the reduction system either to a call-by-name or a call-by-value discipline.

2.3 Deriving the Collapse of the Quantification Domain

The domain of terms in $TT_{\Sigma}^{\mathtt{cc}}$ is degenerated. Indeed, we have

Proposition 2. *For any two variables x and y, $x = y$ is derivable in $TT_{\Sigma}^{\mathtt{cc}}$.*

The proof proceeds as follows.

- First prove $\Sigma z.z = x$ using the artificially classical proof
$$\pi_0 \triangleq \mathtt{cc}_k\ (x, \mathtt{th}\ k\ (x, \mathsf{refl}))\ .$$
- Deduce $\mathsf{wit}\ \pi_0 = x$ whose proof is $\pi_1 \triangleq \mathsf{prf}\ \pi_0$.
- Observe that
$$\begin{array}{rll} \mathsf{wit}\ \pi_0 & \rightarrow \mathtt{cc}_k(\mathsf{wit}(x, \mathtt{th}\ k\ \mathsf{wit}(x, \mathsf{refl}))) & (\zeta_{\mathsf{wit}}) \\ & \rightarrow \mathtt{cc}_k\ x & (\iota_{\mathsf{wit}}) \end{array}$$
so that
$$\pi_2 \triangleq \mathsf{subst\ refl}\ \pi_1$$
is a proof of $\mathtt{cc}_k\ x = x$.
- Show then $\Sigma z.z = y$ using the artificially classical proof
$$\pi_3 \triangleq \mathtt{cc}_k\ (x, \mathtt{th}\ k\ (y, \mathsf{refl}))\ .$$
- Observe also that
$$\begin{array}{rll} \mathsf{wit}\ \pi_3 & \rightarrow \mathtt{cc}_k\ (\mathsf{wit}(x, \mathtt{th}\ k\ \mathsf{wit}(y, \mathsf{refl}))) & (\zeta_{\mathsf{wit}}) \\ & \rightarrow \mathtt{cc}_k\ x & (\iota_{\mathsf{wit}}) \end{array}$$
to conclude that
$$\pi_4 \triangleq \mathsf{subst\ refl}\ (\mathsf{prf}\ \pi_3)$$
is a proof of $\mathtt{cc}_k\ x = y$.
- Conclude that
$$\mathsf{subst}\ \pi_3\ \pi_4$$
is a proof of $x = y$.

Notice that we only used the ζ_{wit} and ι_{wit} rules. The next section shows that for typed control operators, one can exhibit a set-theoretic model of the system.

2.4 Explicit Typing of cc and th: System $TT_{\Sigma}^{\mathsf{cc}_T}$

We now consider explicitly typed cc and th. The new syntax of terms is

$$\begin{array}{ll} t, u & ::= x \mid \mathsf{wit}\ \pi \mid \mathsf{cc}_k^{x.A} t \\ \pi & ::= (t, \pi) \mid \mathsf{prf}\ \pi \mid \mathsf{refl} \mid \mathtt{cc}_{k:\neg A}\ \pi \mid \mathtt{th}_B\ k\ \pi \mid \mathtt{th}_B\ k\ t \end{array}$$

The typing rules are similar: just add the constraint for typing $\mathtt{cc}_{k:\neg A}\ \pi$ that $\neg A$ is the type of k in the context and add the constraint that the type of $\mathtt{th}_B\ k\ \pi$ is B. The new reduction rules now take care of types.

$$\begin{array}{llll} \mathsf{wit}(t, \pi) & \rightarrow t & & (\iota_{\mathsf{wit}}) \\ \mathsf{wit}(\mathtt{cc}_{k:\neg \Sigma x.A}\ \pi) & \rightarrow \mathsf{cc}_k^{x.A} \mathsf{wit}(\pi[k(\mathsf{wit}\ \{\ \})/k]) & & (\zeta_{\mathsf{wit}}) \\ \mathsf{cc}_k^{x.A} t & \rightarrow t & k \text{ not free in } t & (\eta_{\mathsf{cc}}) \end{array}$$

Thanks to the explicit typing, the previous proof of degeneracy do not work any longer. Indeed, the two occurrences of $\mathtt{cc}_k\ x$ now appear as $\mathsf{cc}_k^{z.z=x} x$ and $\mathsf{cc}_k^{z.z=y} x$ so that they are not convertible any more. The next section shows that ι_{wit} and ζ_{wit} together with explicitly typed cc and th do not allow to derive the degeneracy of the quantification domain.

2.5 A Set-Theoretic Model of $TT_{\Sigma}^{\mathsf{cc}_T}$ without η_{cc}

Let $\mathcal{D}$ be a non empty domain and d_0 be an element of $\mathcal{D}$. To distinguish the different roles we give to $\emptyset$, we use the abbreviation $\bullet$ to denote $\emptyset$ when seen as an element rather than as a set. We interpret the formulae of $TT_{\Sigma}^{\mathsf{cc}_T}$ by sets in $\mathcal{T}$ where $\mathcal{T}$ is defined by

$$\begin{array}{ll} \mathcal{T}_0 & = \{\emptyset, \{\bullet\}\} \\ \mathcal{T}_{n+1} & = \{\Sigma_{a \in \mathcal{D}} T_a | (T_a)_{a \in \mathcal{D}} \in \mathcal{T}_n^{\mathcal{D}}\} \\ \mathcal{T} & = \bigcup_n \mathcal{T}_n \end{array}$$

where $\Sigma_{a \in \mathcal{D}} T_a = \{(a, p) | p \in T_a\}$.

For each inhabited $\Sigma_a T_a$, we let $d_{\Sigma_a T_a}$ be a canonical witness of the set, i.e. a constant in $\mathcal{D}$ such that $T_{d_{\Sigma_a T_a}}$ is inhabited (we need the axiom of choice if the domain is not countable). For empty $\Sigma_a T_a$, we let $d_{\Sigma_a T_a}$ be d_0. To each $T \in \mathcal{T}$, we associate a canonical witness $\epsilon(T)$ (which is the same for all empty T). It is defined by

$$\begin{array}{ll} \epsilon(\emptyset) = \epsilon(\{\bullet\}) & = \bullet \\ \epsilon(\Sigma_a T_a) & = (d_{\Sigma_a T_a}, \epsilon(T_{d_{\Sigma_a T_a}})) \end{array}$$

and it satisfies $\epsilon(T) \in T$ for non empty T. We use the letter ρ to denote substitutions from the set of variables of the logic to $\mathcal{D}$. The notation $\rho, (x \leftarrow a)$ denotes the substitution which binds (or rebinds) x to a. For a given substitution ρ, we define the interpretations of terms, proofs and formulae as follows:

$$\begin{array}{lll}
[\![x]\!]_\rho & = \rho(x) & \\
[\![\mathtt{cc}_k^{x.A}\ t]\!]_\rho & = d_{[\![\Sigma x.A]\!]_\rho} & \\
[\![\mathsf{wit}\ \pi]\!]_\rho & = \mathit{fst}([\![\pi]\!]_\rho) & \\
 & & \\
[\![\mathsf{refl}]\!]_\rho & = \bullet & \\
[\![\mathsf{subst}\ \pi_1\ \pi_2]\!]_\rho & = [\![\pi_2]\!]_\rho & \\
[\![(t,\pi)]\!]_\rho & = ([\![t]\!]_\rho, [\![\pi]\!]_\rho) & \\
[\![\mathsf{prf}\ \pi]\!]_\rho & = \mathit{snd}([\![\pi]\!]_\rho) & \\
[\![\mathtt{cc}_k^{A}\ \pi]\!]_\rho & = \epsilon([\![A]\!]_\rho) & \\
[\![\mathtt{th}_B\ k\ \pi]\!]_\rho & = \epsilon([\![B]\!]_\rho) & \\
 & & \\
[\![t = u]\!]_\rho & = \{\bullet\} & \text{if } [\![t]\!]_\rho = [\![u]\!]_\rho \\
[\![t = u]\!]_\rho & = \emptyset & \text{otherwise} \\
[\![\Sigma x.A(x)]\!]_\rho & = \Sigma_{a\in\mathcal{D}}[\![A(x)]\!]_{\rho,(x\leftarrow a)} &
\end{array}$$

Notice that we don't need to define $[\![\mathtt{th}\ k\ t]\!]_\rho$ since this pattern occurs only within proofs π occurring in terms of the form $\mathtt{cc}_k\ (\mathsf{wit}\ (t',\pi))$.

Lemma 3. *The interpretation validates the reduction rules ι_{wit} and ζ_{cc}.*

$$\begin{array}{ll}
[\![\mathsf{wit}(t,\pi)]\!]_\rho & = [\![t]\!]_\rho \\
[\![\mathsf{wit}(\mathsf{cc}_{k:\Sigma x.A(x)}\ \pi)]\!]_\rho & = [\![\mathsf{cc}_k^{x.A(x)}\mathsf{wit}(\pi[k(\mathsf{wit}\ \{\ \})/k])]\!]_\rho
\end{array}$$

Moreover,

$$[\![A(x)]\!]_{\rho,(x\leftarrow[\![t]\!]_\rho)} \ = \ [\![A(t)]\!]_\rho$$

Proposition 4 (Soundness). *If $\Gamma \vdash \pi : A$ then, forall $\rho \in FV(A) \to \mathcal{D}$, if forall $k_i : \neg A_i$ in Γ, $[\![A_i]\!]_\rho$ is empty, then $[\![\pi]\!]_\rho \in [\![A]\!]_\rho$*

Proof. The proof is by induction.

– If $\Gamma \vdash \mathsf{refl} : t = u$ with $t \to u$ then by validity of the reduction rules, $[\![t]\!]_\rho = [\![u]\!]_\rho$ and $[\![t = u]\!]_\rho = \{\bullet\}$.
– If $\Gamma \vdash (t,\pi) : \Sigma x.A(x)$ with $\Gamma \vdash \pi : A(t)$ then, by induction $[\![\pi]\!]_\rho \in [\![A(t)]\!]_\rho = [\![A(x)]\!]_{\rho,(x\leftarrow[\![t]\!]_\rho)}$, hence $[\![(t,\pi)]\!]_\rho = ([\![t]\!]_\rho, [\![\pi]\!]_\rho) \in [\![\Sigma x.A(x)]\!]_\rho$.
– If $\Gamma \vdash \mathsf{prf}\ \pi : A(\mathsf{wit}\ \pi)$ with $\Gamma \vdash \pi : \Sigma x.A(x)$ then, by induction, we get $[\![\pi]\!]_\rho \in [\![\Sigma x.A(x)]\!]_\rho$, so that there exists c and p such $[\![\pi]\!]_\rho = (c,p)$ and $p \in [\![A(x)]\!]_{\rho,(x\leftarrow c)}$. Since $c = \mathit{fst}([\![\pi]\!]_\rho) = [\![\mathsf{wit}\ \pi]\!]_\rho$, we have $[\![\mathsf{prf}\ \pi]\!]_\rho = \mathit{snd}((c,p)) = p \in [\![A(x)]\!]_{\rho,(x\leftarrow c)} = [\![A(\mathsf{wit}\ \pi)]\!]_\rho$.
– If $\Gamma \vdash \mathsf{subst}\ \pi_1\ \pi_2 : A(u)$ with $\Gamma \vdash \pi_1 : t = u$ and $\Gamma \vdash \pi_2 : A(t)$ then, by induction, $[\![\pi_1]\!]_\rho \in [\![t = u]\!]_\rho$, so that $[\![t = u]\!]_\rho$ is not empty and $[\![t]\!]_\rho = [\![u]\!]_\rho$. Also, $[\![\pi_2]\!]_\rho \in [\![A(t)]\!]_\rho$ so that we have $[\![\mathsf{subst}\ \pi_1\ \pi_2]\!]_\rho = [\![\pi_2]\!]_\rho \in [\![A(t)]\!]_\rho = [\![A(x)]\!]_{\rho,(x\leftarrow[\![t]\!]_\rho)} = [\![A(x)]\!]_{\rho,(x\leftarrow[\![u]\!]_\rho)} = [\![A(u)]\!]_\rho$.

- If $\Gamma \vdash \mathtt{cc}_k\ \pi : A$ with $\Gamma, k : \neg A \vdash \pi : A$ then, $[\![A]\!]_\rho$ is either empty or inhabited. If it is inhabited, then $\epsilon_\rho(A) \in [\![A]\!]_\rho$, hence $[\![\mathtt{cc}_k\ \pi]\!]_\rho \in [\![A]\!]_\rho$. Otherwise, we can apply the induction hypothesis and get $[\![\pi]\!]_\rho \in [\![A]\!]_\rho$, which is contradictory with the assumption that $[\![A]\!]_\rho$ is empty.
- If $\Gamma, k : \neg A \vdash k\pi : B$ with $\Gamma, k : \neg A \vdash \pi : A$ then, by induction, we get $[\![\pi]\!]_\rho \in [\![A]\!]_\rho$ which contradicts the assumption that $[\![A]\!]_\rho$ is empty.

Taking for $\mathcal{D}$ a domain with at least two elements, we get the following corollary.

Corollary 5. *$TT_\Sigma^{\mathtt{cc}_T}$ without the $\eta_{\mathtt{cc}}$ rule is not degenerated, i.e. for distinct variables x and y, we have $\nvdash x = y$.*

2.6 Deriving the Collapse of the Quantification Domain with Explicitly Typed Control Operators

Though $TT_\Sigma^{\mathtt{cc}_T}$ without the $\eta_{\mathtt{cc}}$ rule is not degenerated, it gets degenerated by considering the $\eta_{\mathtt{cc}}$ rule. Indeed, we have again

Proposition 6. *For any two variables x and y, $x = y$ is derivable in $TT_\Sigma^{\mathtt{cc}_T}$.*

The new proof proceeds as follows.

- First prove $\Sigma z.x = z$ using the artificially classical proof
$$\pi_0 \triangleq \mathtt{cc}_{k:\Sigma z.x=z}\ (y, \mathtt{th}\ k\ (x, \mathsf{refl}))\ .$$
- Then observe that
$$\begin{aligned} \mathsf{wit}\ \pi_0 &\rightarrow \mathtt{cc}_{k:\Sigma z.x=z}\ (\mathsf{wit}(y, \mathtt{th}\ k\ \mathsf{wit}(x, \mathsf{refl}))) && (\zeta_{\mathsf{wit}}) \\ &\rightarrow \mathtt{cc}_k^{z.x=z} y && (\iota_{\mathsf{wit}}) \\ &\rightarrow y && (\eta_{\mathtt{cc}}) \end{aligned}$$
- Conclude that
$$\mathsf{subst}\ \mathsf{refl}\ (\mathsf{prf}\ \pi_0)$$
is a proof of $x = y$.

2.7 Inconsistency of Martin-Löf's Type Theory Extended with Computational Classical Logic

Since Martin-Löf's type theory [6] has Σ-types in `Set`, its extension with computational classical logic is inconsistent. We first extend the syntax of terms:

$$t ::= \ldots \mid \mathtt{cc}_k\ t$$

Then, we let $\neg A \triangleq A \rightarrow N_0$ and we add the following inference rules:

$$\frac{\begin{array}{c}(k \in \neg A)\\ t \in A\end{array}}{\mathtt{cc}_k\ t \in A} \qquad \frac{\begin{array}{c}(k \in \neg A)\\ t = u \in A\end{array}}{\mathtt{cc}_k\ t = \mathtt{cc}_k\ u \in A}$$

For equality, we restrict the commutation of cc with the elimination operator E for Σ-types to the *non dependent* case, i.e. to the case where x and y do not occur free in C:

$$\frac{\begin{array}{cc} (k \in \neg(\Sigma x \in A)B(x)) & (x \in A, y \in B(x)) \\ t \in (\Sigma x \in A)B(x) & u \in C \end{array}}{\mathtt{E}(\mathtt{cc}_k\ t, (x,y)u) = \mathtt{cc}_k\ (\mathtt{E}(t[k(\mathtt{E}(\{\,\}, (x,y)u))/k], (x,y)u)) \in C}$$

$$\frac{t \in A \qquad k \text{ not free in } t}{\mathtt{cc}_k\ t = t \in A}$$

Without universes, one can only show that the theory is proof-irrelevant, as enforced by Smith's result on the independence of Peano's fourth axiom in the theory without universes [10]. With one universe, $\neg 0 = 1$ is provable and the computational classical theory is inconsistent.

2.8 Inconsistency of the Set-predicative Calculus of Inductive Constructions Extended with Computational Classical Logic

Since the Calculus of Inductive Constructions [1] has Σ-types in `Set` and non degenerated datatypes, its extension with computational classical logic in `Set`, even in its `Set`-predicative version that Coq version 8 implements, is inconsistent.

We let $\bot \triangleq \forall C : Set.C$ and $\neg A \triangleq A \to \bot$. To get computational classical logic, the syntax of terms is extended with the following construction

$$t ::= \ldots \mid \mathtt{cc}_{k:\neg A}\ t$$

The new inference rule is

$$\frac{\Gamma, k : \neg A \vdash t : A}{\Gamma \vdash \mathtt{cc}_k\ t : A}$$

And the new set of reduction rules, at least, contains $\eta_{\mathtt{cc}}$ and a commutation rule for *non dependent* case analysis.

$$\begin{array}{l} \mathtt{case}_P\ (\mathtt{cc}_{k:\neg A}\ t)\ \mathtt{of}\ t_1 \ldots t_n \\ \quad \to \quad \mathtt{cc}_{k:\neg P}\ (\mathtt{case}_P\ (t[k(\mathtt{case}_P\ \{\,\}\ \mathtt{of}\ t_1 \ldots t_n)/k])\ \mathtt{of}\ t_1 \ldots t_n) \\ \mathtt{cc}_{k:\neg A}\ t \quad \to \quad t \qquad\qquad\qquad k \text{ not free in } t \end{array}$$

Of course, one would expect of a fully-fledged computationally classical Calculus of Inductive Constructions commutation rules of cc for all kinds of constructors (application, inductive types, constructors of inductive types, ...) and only for a given reduction semantics (call-by-name of call-by-value), so as to preserve confluence. But to get an inconsistency, commutation of cc with case analysis (which is the construction needed for defining wit and prf) is enough.

3 Remarks

3.1 Commutation of cc with respect to prf

We did not consider the commutation rule of cc with respect to prf though it would be needed for completion of the reduction system. The reason was that it was not necessary in order to derive the degeneracy of the quantification domain of the logic. In fact, the naive formulation of this rule

$$\mathsf{prf}(\mathsf{cc}_k\ \pi) \to \mathsf{cc}_{k'}\ \mathsf{prf}(\pi[k'(\mathsf{prf}\ \{\ \})/k]) \qquad (\zeta_{\mathsf{prf}})$$

is problematic since it does not satisfy subject reduction. Indeed, if k has type $\Sigma x.A(x)$ on the left-hand-side, then, on the right-hand-side, it maps to a continuation variable k' (we chose a different name to emphasise the difference of types) that cannot be typed consistently in the general case. The binding occurrence of k' is intended to have type $A(\mathsf{wit}(\mathsf{cc}_k\ \pi)$ while each place of the form $\mathtt{th}\ k'\ (\mathsf{prf}\ \pi')$ where it occurs bound expects it to be of type $A(\mathsf{wit}(\mathsf{cc}_k\ \pi')$ for π' a strict subproof of π. There is no reason that each of the $\mathsf{wit}(\mathsf{cc}_k\ \pi')$ (that reduce to $\mathsf{cc}_k^{x.A}\mathsf{wit}(\pi'[k(\mathsf{wit}\ \{\ \})/k]))$, and also $\mathsf{wit}(\mathsf{cc}_k\ \pi)$ (that reduces to $\mathsf{cc}_k^{x.A}\mathsf{wit}(\pi[k(\mathsf{wit}\ \{\ \})/k]))$, all are convertible (and there are effectively not convertible for the degeneracy proofs given in Sect. 2.3 and 2.6).

The mismatch can be solved by inserting a coercion that derives $A(\mathsf{cc}_k^{x.A(x)}\ \pi)$ from $A(t)$ for any A, t and π. It may be worth to notice that along the interpretation in section 2.5 that throws away the argument of each typed cc and see it as an Hilbert-style ϵ operator, the coercion simply corresponds to the characteristic axiom of this ϵ operator.

3.2 Intuitionistic Uses of cc

The degeneracy proof needs that some calls to the continuation variables are done to inhabit a priori non inhabited types, such as $x = y$ for distinct variables x and y.

Since cc can still be interesting from a computational point of view even in an non essentially classical framework (typically to reason intuitionistically on algorithms that "backtracks" thanks to cc), it can be interesting to restrict the call to continuation variables set up by cc only on inhabited types.

By this way, any derivation using cc can trivially be translated into an intuitionistic one: just replace each occurrence of $\mathtt{th}\ k\ t$ of type B by b_0 where b_0 is an inhabitant of B. Hence the logic is non degenerated. The soundness of this replacement remains to be investigated in presence of the (expected) commutation rule ζ_{prf}.

3.3 Axiom of Choice *vs* Principle of Definite Description

For simplicity (since definite existential quantification is heavier to deal with than indefinite existential quantification), we only considered ordinary (indefinite) existential quantification.

However, we believe that the results still hold with $\Sigma!$ instead of Σ. Especially, all witnesses occurring in the proofs we considered were unique witnesses.

Acknowledgements

I thank Freek Wiedijk for helpful discussions on the set-theoretic interpretation of the functional space $\neg\neg A \to A$.

References

1. The Coq development team: The Coq Proof Assistant Reference Manual, Version 8.0. (2004). Available at http://coq.inria.fr/doc.
2. Coquand, T.: Metamathematical investigations of a calculus of constructions. In Odifreddi, P., ed.: Logic and Computer Science. Apic Series 31. Academic Press (1990) 91–122. Also INRIA Research Report number 1088, sept 1989.
3. Crolard, T.: A confluent lambda-calculus with a catch/throw mechanism. Journal of Functional Programming **9(6)** (1999) 625–647
4. Felleisen, M., Friedman, D.P., Kohlbecker, E., Duba, B.F.: Reasoning with continuations. In: First Symposium on Logic and Computer Science. (1986) 131–141
5. Griffin, T.G.: The formulae-as-types notion of control. In: Conf. Record 17th Annual ACM Symp. on Principles of Programming Languages, POPL '90, San Francisco, CA, USA, 17-19 Jan 1990, ACM Press, New York (1990) 47–57
6. Martin-Löf, P.: Intuitionistic type theory. Bibliopolis (1984)
7. Parigot, M.: Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In: Logic Programming and Automated Reasoning: International Conference LPAR '92 Proceedings, St. Petersburg, Russia, Springer-Verlag (1992) 190–201
8. Pottinger, G.: Definite descriptions and excluded middle in the theory of constructions (1989). Communication to the TYPES electronic mailing list.
9. Prawitz, D.: Natural Deduction - A proof-theoretical study. Almquist and Wiksell, Stockholm (1965)
10. Smith, J.M.: The independence of Peano's fourth axiom from Martin-Löf's type theory without universes. Journal of Symbolic Logic **53** (1988) 840–845
11. Spector, C.: Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In: Recursive Function Theory: Proc. Symposia in Pure Mathematics. Volume 5., American Mathematical Society (1962) 1–27