5- The starting point: dPA$^\omega$

Axiom of choice

The axiomatization of a theory, as we explained in Chapter 1, is to be understood as an intent to give a formal and truthful representation of a given world or structure. As long as this structure deals with finite objects that have a concrete representation in the physical world, it is easy to agree on what it "is" or "should be" (and thus on whether the axiomatization is truthful). However, as soon as the theory involves infinite objects, this question quickly turns out to be more the matter of one's personal "religion" than the empirical observation of a physical object. In particular, some undeniable properties of finite objects become much more questionable in the case of infinite sets. Consider for instance the following problem, as presented[1] by Russell [146, pp.125-127]:

> *[Imagine a] millionaire who bought a pair of socks whenever he bought a pair of shoes, and never at any other time, and who had such a passion for buying both that at last he had $\aleph_0$ pairs of shoes and $\aleph_0$ pairs of socks. The problem is: How many shoes had he, and how many socks?*

The cardinal $\aleph_0$ defines exactly the infinite quantity of natural numbers: an infinite set is of cardinality $\aleph_0$ if it can be enumerated by the natural numbers. In particular, since there is a bijection from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$, $\aleph_0$ is not increased by doubling.

> *One would naturally suppose that he had twice as many shoes and twice as many socks as he had pairs of each, and that therefore he had $\aleph_0$ of each [...].*

To prove this claim, it is thus necessary and sufficient to give an enumeration of the millionaire's shoes and socks. Yet, this is not possible *a priori*:

> *In our case it can be done with the shoes, but not with the socks, except by some very artificial device. The reason for the difference is this: Among shoes we can distinguish right and left, and therefore we can make a selection of one out of each pair, [...] but with socks no such principle of selection suggests itself [...].*
>
> *We may put the matter in another way. To prove that a class has $\aleph_0$ terms, it is necessary and sufficient to find some way of arranging its terms in a progression. There is no difficulty in doing this with the shoes. The pairs are given as forming an $\aleph_0$, and therefore as the field of a progression. Within each pair, take the left shoe first and the right second, keeping the order of the pair unchanged; in this way we obtain a progression of all the shoes. But with the socks we shall have to choose arbitrarily, with each pair, which to put first; and an infinite number of arbitrary choices is an impossibility. Unless we can find a rule for selecting, i.e. a relation which is a selector, we do not know that a selection is even theoretically possible. [...]*
> *The case of the socks, with a little goodwill on the part of the reader, may serve to show how a selection might be impossible.*

[1] Russell actually presented the story with boots. We replaced it with shoes in the quote, which we found to be more asymmetric. Russell might never had one of these ugly (and symmetric) plastic rain boots.

More generally, it is unclear if one should be able to pick an element of an infinite set, and from a theoretical point of view, this indeed requires an extra axiom, called the axiom of choice. This axiom, which was first introduced by Zermelo in the realm of set theory [163], is functionally expressed by:

$$AC \triangleq (\forall x \in A.\exists y \in B.P(x,y)) \rightarrow (\exists f \in B^A.\forall x \in A.P(x,f(x)))$$

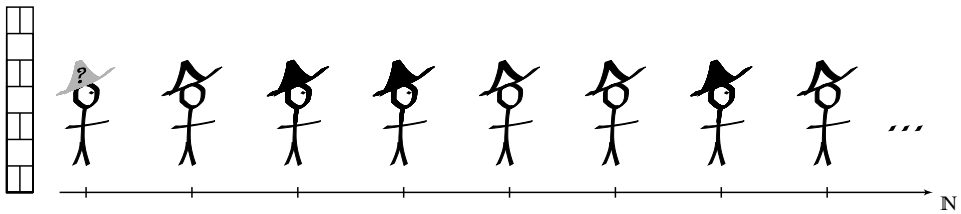
which stipulates the existence of a choice function[2]. This axiom was shown to be independent of Zermelo-Fraenkel set theory (ZF)[3]. Even if it is very tempting to consider natural the possibility of selecting one element within an infinite set (since it is for finite sets), such an axiom leads to very surprising consequences. The most striking example one is certainly the Banach-Tarski paradox [9], which shows that the unit ball

$$\mathcal{B} := \{(x,y,z) \in \mathcal{R}^3 : x^2 + y^2 + z^2 = 1\}$$

in three dimensions can be disassembled into a finite number of pieces, which can then be reassembled (after translating and rotating each of the pieces) to form two disjoint copies of the ball $\mathcal{B}$.

Another dazzling paradox is a variant of the famous riddle where a column of prisoners is facing a wall, each of them having a black or white hat on his head of which he ignores the color. Each prisoner (from the end of the line) has to guess in turns his hat color. They are eventually released if at most one prisoner is wrong. They are allowed to talk through a strategy in the beginning, and they indeed have a way to end up free in this situation. Now, let us turn the prisoners around and consider the following infinite version:

> *A countable infinite number of prisoners are placed on the natural numbers, facing in the positive direction (*i.e. *everyone can see an infinite number of prisoners). Hats will be placed and each prisoner will be asked what his hat color is.*



> *However, to complicate things, prisoners cannot hear previous guesses or whether they were correct. In this situation, what is the best strategy?*

Admitting the axiom choice[4], the answer is quite counter-intuitive: the prisoners have a (common) strategy to guess the color of their own hat, in such a way that only a finite number of them will make wrong guesses. Even more shocking, the strategy is so robust that we could consider any number of colors (even an uncountable one), the prisoners will still only make a finite number of wrong guesses... The solution is left to the sagacity of the reader[5] but the "problem" here is very similar to the Banach-Tarski paradox, where the pieces used in this decomposition are highly pathological in nature and cannot be constructed without the axiom of choice.

In short, the question of knowing whether the axiom of choice is wrong or not can not be given any mathematical answer. Indeed, the axiom of choice is independent from the axioms of set theory.

[2]If we define the predicate $P(x,y)$ as $y \in x$, it exactly says that if all the sets $x \in A$ are non-empty, there exists a choice function: $(\forall x \in A.x \neq \emptyset) \rightarrow \exists f \in \cup A^A.\forall x \in A.f(x) \in x$.

[3]Gödel proved that the theory ZF + AC is consistent, and Cohen proved the same for the theory ZF + ¬AC. Details on these proofs and much more about the axiom of choice can be found for instance in Jech's book on the topic [83].

[4]We also assume that each prisoner can see the ω prisoners in front of him, have infinite memory and so forth.

[5]Clue: the definition of clever equivalence classes and the use of AC to pick representatives can be helpful. The full answer is available here [125].

Intuitively, the axiom of choice does not reflect anything concrete in our living world. Adding it or not to a theory is thus a matter of one's belief, with its logical strength as benefits and its paradoxical consequences as withdraws.

Dependent and countable choices

In fact, a huge part of mathematics does not require the axiom of choice in full strength. For instance, most of analysis[6] can be done in a system of axioms containing a weaker form of choice, namely the axiom of dependent choice. This axiom expresses the possibility of constructing a sequence where each element has to be chosen in function of the anterior. Formally, it is defined by:

$$DC \triangleq (\forall x \in A.\exists y \in A.P(x,y)) \to \forall x_0 \in A.\exists f \in A^{\mathbb{N}}.(f(0) = x_0 \wedge \forall n \in \mathbb{N}.P(f(n), f(S(n))))$$

This axiom does not lead to the paradoxical consequences of the full axiom of choice, and is in practice expressive enough for most of the mathematics[7].

Another weaker form of choice, which is actually the one involved in Russell shoes-and-socks metaphor, is the axiom of countable choice. It is simply defined as the axiom of choice where universal variables are bound to the set of natural numbers $\mathbb{N}$:

$$AC_{\mathbb{N}} \triangleq (\forall x \in \mathbb{N}.\exists y \in B.P(x,y)) \to \exists f \in B^{\mathbb{N}}.\forall x \in \mathbb{N}.P(x, f(x))$$

It is quite easy to check that the full axiom of choice (AC) implies the axiom of dependent choice (DC), which itself implies the axiom of countable choice ($AC_{\mathbb{N}}$) (converse implications are false). Dependent and countable choices are the axiom that will be at the heart of this part of the monograph.

5.1 Computational content of the axiom of choice

5.1.1 Martin-Löf Type Theory

In the line of Curry-Howard isomorphism, it is natural to wonder what is the computational content of the axiom choice, that is, what would be a program whose type is (AC). In fact, through the Brouwer-Heyting-Kolmogoroff interpretation of intuitionistic logic (see Section 3.1.1), a proof of $\forall x.\exists y.P(x,y)$ is precisely a function which associates to any m a proof of $\exists y.P(m,y)$, which is itself a pair made of a certificate n and a proof of $P(m,n)$. Thus, there exists *de facto* a function f such that for any m, $P(m, f(m))$ holds. Otherwise said, through this interpretation, the axiom of choice should then be a trivial theorem.

This idea is the key of Martin-Löf's proof for the axiom choice in his constructive type theory [115]. One of the crucial differences with the different theories we presented until here, is that types (*i.e.* formulas) are now dependent on terms (*i.e.* on proofs). Just like first-order arithmetic includes a quantification $\forall x.A$ ranging over natural numbers and leading to formulas $A[n/x]$ for each possible instantiation $n \in \mathbb{N}$ of x, Martin-Löf type theory includes a dependent product type written $\Pi(x : A).B$ where the variable x ranges over the terms of type A. In particular, if t is a term of type $\Pi(x : A).B$ and u is a term of type A, the term tu is then of type $B[u/x]$:

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : \Pi(x : A).B}\ (\Pi_I) \qquad \frac{\Gamma \vdash t : \Pi(x : A).B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B[u/x]}\ (\Pi_E)$$

[6] Notably, Baire category theorem has been proved equivalent to the axiom of dependent choice. More generally, a large class of theorems whose proof are done by constructing a sequence by induction requires this axiom.

[7] More details on this (and more generally on the axiomatic strength required by theorems of mathematics) can be found in the introduction of Simpson's book [149].

It is worth noting that in the case where B does not refer to x, these rules exactly correspond to the usual rules $(\to_I)$ and $(\to_E)$.

The fact that formulas can now refer to terms allows us to strengthen the rules for existential quantification. They now reflect the BHK interpretation for existential proofs, which inhabits a dependent sum type written $\Sigma(x : A).B$: a proof term of type $\Sigma(x : A).B$ is a pair (t, u) such that t—the *witness*—is of type A, while u—the *proof*—is of type $B[t/x]$. Dually to this construction, there are now two elimination rules[8]: one with a destructor `wit` to extract the witness, the second one with a destructor `prf` to extract the proof:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash u : B[t/x]}{\Gamma \vdash (t,u) : \Sigma(x : A).B}\ (\Sigma_I) \qquad \frac{\Gamma \vdash t : \Sigma(x : A).B}{\Gamma \vdash \texttt{wit}\ t : A}\ (\texttt{wit}) \qquad \frac{\Gamma \vdash t : \Sigma(x : A).B}{\Gamma \vdash \texttt{prf}\ t : B[\texttt{wit}\ t/x]}\ (\texttt{prf})$$

Note that this extension of types with dependencies corresponds to the horizontal axis of the λ-cube (Section 2.4.2). In the sequel, we will present in more details a full dependent system with its type system and reduction rules. As for now, let us just mention that these terms reduce as follows:

$$(\lambda x.t)\, u \to t[u/x] \qquad \texttt{wit}\,(t,u) \to t \qquad \texttt{prf}\,(t,u) \to u$$

These reductions naturally induce a relation on types: we write $A \triangleright B$ if reducing some term occurring in A yields B. The reflexive-symmetric-transitive closure of this relation is written $A \equiv B$ and the type system includes a conversion rules according to this relation:

$$\frac{\Gamma \vdash t : A \quad A \equiv B}{\Gamma \vdash t : B}\ (\text{CONV})$$

Having said this, we dispose of enough structure to give a proof term for the axiom of choice, which is nothing more than an implementation of the intuition above: given a proof H of $\Pi(x : A).\Sigma(y : B).P(x, y)$, the choice function simply maps any x to the witness of Hx, while the proof that this function is sound w.r.t. P returns the corresponding witness. This term can indeed be given the type of the axiom of choice:

$$\vdash \lambda H.(\lambda x.\, \texttt{wit}\,(Hx), \lambda x.\, \texttt{prf}\,(Hx)) : AC$$

where AC is defined in terms of dependent product and sum:

$$AC \triangleq \Pi(x : A).\Sigma(y : B).P(x, y) \to \Sigma(f : \Pi(x : A).B).\Pi(x : A).P(x, f(x))$$

5.1.2 Incompatibility with classical logic

Unsurprisingly, this proof does not scale to classical logic (otherwise the axiom of choice would be a theorem of Zermelo-Fraenkel set theory, which is a classical theory). We give two explanations for this, first a metaphysical argument for this natural limitation in terms of computability, second a technical description of the incompatibility of classical logic and dependent types.

[8] Actually, the original presentation [115] only has one rule, called dependent elimination rule, given by:

$$\frac{\Gamma \vdash c : \Sigma(x : A).B \quad \Gamma, x : A, y : B[x] \vdash d : C[(x,y)]}{\Gamma \vdash E(c, \lambda xy.d) : C[c]}\ (\Sigma_E)$$

As for the reduction rule, it was defined by:

$$E((t,u), \lambda xy.d[(x,y)]) \to d[(t,u)]$$

It is easy to check that defining the primitives `wit` c and `prf` c respectively by $tE(c, \lambda xy.x)$ and $E(c, \lambda xy.y)$ allow to recover the corresponding typing and reduction rules, and vice-versa.

5.1.2.1 Computing the uncomputable

Imagine that we could dispose, in a type theoretic (or BHK interpretation, realizability) fashion, of a classical framework including a proof term t for the axiom of choice:

$$\vdash t : \forall x \in A.\exists y \in B.P(x,y) \to \exists f \in B^A.\forall x \in A.P(x,f(x))$$

Consider now any undecidable[9] predicate $U(x)$ over a domain X. Since we are in a classical framework, using the middle-excluded, the formula $U(x) \vee \neg U(x)$ is true for any $x \in X$. This can be strengthened into the formula:

$$\forall x \in X.\exists y \in \{0,1\}.(U(x) \wedge y = 1) \vee (\neg U(x) \wedge y = 0)$$

which is provable as well and thus should have a proof term u. Now, this has the shape of the hypothesis of the axiom of choice, so that by application of t to u, we should obtain a term:

$$\vdash t\,u : \exists f \in \{0,1\}^X.\forall x \in X.(U(x) \wedge f(x) = 1) \vee (\neg U(x) \wedge f(x) = 0)$$

In particular, the term $\mathtt{wit}\ (t\,u)$ would be a function which, for any $x \in X$, outputs 1 if $U(x)$ is true, and 0 otherwise. This is absurd, since U is undecidable.

This handwavy explanation gives us a metamathematical argument on the impossibility of having a proof system which is classical as a logic, entails the axiom of choice and where proofs fully compute. Since the existence of consistent classical theories with the axiom of choice (like set theory) has been proven, the incompatibility is to be found with the constructive character of Martin-Löf type theory. Actually, the compatibility of AC with constructive theories is very sensitive to the definition of "constructive" and is already discussed[10]. In the next sections, we will present an intent to give a proof of the axiom of dependent choice that is constructive and yet compatible with classical logic.

5.1.2.2 Inconsistency

Technically, another reason why Martin-Löf type theory cannot scale to classical logic is that the simultaneous presence of control operators and dependent types leads to inconsistencies. This was observed by Herbelin [69] in a weaker setting, which we recap hereafter.

Let us adopt here a stratified presentation of dependent types, by syntactically distinguishing *terms*—that represent mathematical objects—from *proof terms*—that represent mathematical proofs. In other words, we syntactically separate the categories corresponding to witnesses and proofs in dependent sum types. Consider a minimal logic of Σ-types and equality, whose formulas, terms (only representing natural number) and proofs are defined as follows:

Formulas	A,B	::=	$t = u \mid \exists x^{\mathbb{N}}.A$
Terms	t,u	::=	$n \in \mathbb{N} \mid \mathtt{wit}\ p$
Proofs	p,q	::=	$\mathtt{refl} \mid \mathtt{subst}\ p\,q \mid (t,p) \mid \mathtt{prf}\ p$

Let us explain the different proof terms by presenting their typing rules. First of all, the pair (t,p) is a proof for an existential formula $\exists x^{\mathbb{N}}.A$ (or $\Sigma(x : \mathbb{N}).A$) where t is a witness for x and p is a certificate for $A[t/x]$. This implies that both formulas and proofs are dependent on terms, which is usual in mathematics. What is less usual in mathematics is that, as in Martin-Löf type theory, dependent types also allow for terms (and thus for formulas) to be dependent on proofs, by means of the constructors $\mathtt{wit}\ p$

[9] That is to say that $U(x)$ is a predicate such that there exists no program p which, given any input $x \in X$, computes whether $U(x)$ is true or not.

[10] There is plenty of literature on constructive choiceless mathematics. The reader can for instance read this very interesting argument of Andrej Bauer rejecting AC in a constructive (in the sense of computable) setting: https://mathoverflow.net/a/23043.

and $\mathtt{prf}\ p$. The typing rules are the same as in the previous section for Σ-types, except that there are separated typing judgments for terms, which can only be of type $\mathbb{N}$.:

$$\frac{\Gamma \vdash p : A(t) \quad \Gamma \vdash t : \mathbb{N}}{\Gamma \vdash (t,p) : \exists x^{\mathbb{N}} A}\ (\exists_I) \qquad \frac{\Gamma \vdash (t,p) : \exists x^{\mathbb{N}}.A}{\Gamma \vdash \mathtt{prf}\ p : A[\mathtt{wit}\ p/x]}\ (\mathtt{prf}) \qquad \frac{\Gamma \vdash t : \exists x^{\mathbb{N}}.A}{\Gamma \vdash \mathtt{wit}\ t : \mathbb{N}}\ (\mathtt{wit}) \qquad \frac{n \in \mathbb{N}}{\Gamma \vdash n : \mathbb{N}}$$

Then, $\mathtt{refl}$ is a proof term for equality, and $\mathtt{subst}\ p\, q$ allows to use a proof of an equality $t = u$ to convert a formula $A(t)$ into $A(u)$:

$$\frac{t \to u}{\Gamma \vdash \mathtt{refl} : t = u}\ (\mathtt{refl}) \qquad \frac{\Gamma \vdash p : t = u \quad \Gamma \vdash q : B[t]}{\Gamma \vdash \mathtt{subst}\ p\, q : B[u]}\ (\mathtt{subst})$$

The reduction rules for this language, which are safe with respect to typing, are then:

$$\mathtt{wit}\,(t,p) \to t \qquad \mathtt{prf}\,(t,p) \to p \qquad \mathtt{subst}\ \mathtt{refl}\ p \to p$$

Starting from this (sound) minimal language, Herbelin showed that its classical extension with the control operators $\mathtt{call/cc}_k$ and $\mathtt{throw}\ k$ permits to derive a proof of $0 = 1$ [69]. The $\mathtt{call/cc}_k$ operator, which is a binder for the variable k, is intended to catch its surrounding evaluation context. On the contrary, $\mathtt{throw}\ k$ (in which k is bound) discards the current context and restores the context captured by $\mathtt{call/cc}_k$. The addition to the type system of the typing rules for these operators (that are similar to the different control operators presented in the prelude):

$$\frac{\Gamma, k : \neg A \vdash p : A}{\Gamma \vdash \mathtt{call/cc}_k\, p : A} \qquad \frac{\Gamma, k : \neg A \vdash p : A}{\Gamma, k : \neg A \vdash \mathtt{throw}\ k\, p : B}$$

allows the definition of the following proof:

$$p_0 \triangleq \mathtt{call/cc}_k\,(0, \mathtt{throw}\ k\,(1, \mathtt{refl})) : \exists x^{\mathbb{N}}.x = 1$$

Intuitively such a proof catches the context, give 0 as witness (which is incorrect), and a certificate that will backtrack and give 1 as witness (which is correct) with a proof of the equality.

If besides, the following reduction rules[11] are added:

$$\begin{array}{rcll} \mathtt{wit}\,(\mathtt{call/cc}_k\, p) & \to & \mathtt{call/cc}_k(\mathtt{wit}\,(p[k(\mathtt{wit}\,\{\ \})/k])) & \\ \mathtt{call/cc}_k\, t & \to & t & (k \notin FV(t)) \end{array}$$

then we can formally derive obtain a proof of $1 = 0$. Indeed, the seek of a witness by the term $\mathtt{wit}\ p_0$ will reduce to $\mathtt{call/cc}_k\ 0$, which itself reduces to 0. The proof term $\mathtt{refl}$ is thus a proof of $\mathtt{wit}\ p_0 = 0$, and we obtain indeed a proof of $1 = 0$:

$$\dfrac{\dfrac{\vdash p_0 : \exists x^{\mathbb{N}}.x = 1}{\vdash \mathtt{prf}\ p_0 : \mathtt{wit}\ p_0 = 1}\ (\mathtt{prf}) \quad \dfrac{\mathtt{wit}\ p_0 \to 0}{\vdash \mathtt{refl} : \mathtt{wit}\ p_0 = 0}\ (\mathtt{refl})}{\vdash \mathtt{subst}\,(\mathtt{prf}\ p_0)\ \mathtt{refl} : 1 = 0}\ (\mathtt{subst})$$

The bottom line of this example is that the same proof p_0 is behaving differently in different contexts thanks to control operators, causing inconsistencies between the witness and its certificate. The easiest and usual approach to prevent this is to impose a restriction to values (which are already reduced) for proofs appearing inside dependent types and within the operators $\mathtt{wit}$ and $\mathtt{prf}$, together with a call-by-value discipline. In particular, in the present example this would prevent us from writing $\mathtt{wit}\ p_0$ and $\mathtt{prf}\ p_0$.

[11] Technically this requires to extend the language to authorize the construction of terms $\mathtt{call/cc}_k\, t$ and of proofs $\mathtt{throw}\ t$. The first rule expresses that $\mathtt{call/cc}_k$ captures the context $\mathtt{wit}\,\{\,\}$ and replaces every occurrence of $\mathtt{throw}\ k\, t$ with $\mathtt{throw}\ k\,(\mathtt{wit}\ t)$. The second one just expresses the fact that $\mathtt{call/cc}_k$ can be dropped when applied to a term t which does not contain the variable k.

5.2 A constructive proof of dependent choice compatible with classical logic

We shall now present dPA$^\omega$, a proof system that was introduced by Herbelin [70] as a mean to give a computational content to the axiom of choice in a classical setting. The calculus is a fine adaptation of Martin-Löf proof which circumvents the different difficulties caused by classical logic. Rather than restating dPA$^\omega$ in full details, for which we refer the reader to [70], let us describe informally the rationale guiding its definition and the properties that it verifies. We shall then present the missing bit of his calculus which led us to this work, namely the normalization, and our approach to prove it.

5.2.1 Realizing countable and dependent choices in presence of classical logic

As we saw in Section 5.1.1, the dependent sum type of Martin-Löf's type theory provides a strong existential elimination, which allows us to prove the full axiom of choice. The proof is simple and constructive:

$$\begin{array}{rcl} AC_A & := & \lambda H.(\lambda x.\,\mathtt{wit}\,(Hx), \lambda x.\,\mathtt{prf}\,(Hx)) \\ & : & \forall x^A.\exists y^B.P(x,y) \to \exists f^{A\to B}.\forall x^A.P(x,f(x)) \end{array}$$

To scale up this proof to classical logic, the first idea in Herbelin's work [70] is to restrict the dependent sum type to a fragment of his system which is called *negative-elimination-free* (NEF). This fragment contains slightly more proofs than just values, but is still computationally compatible with classical logic.

The second idea is to represent a countable universal quantification as an infinite conjunction. This allows us to internalize into a formal system the realizability approach of [15, 40] as a direct proofs-as-programs interpretation. Informally, let us imagine that given a proof $H : \forall x^A.\exists y^B.P(x,y)$, we could create the sequence $H_\infty = (H0, H1, \ldots, Hn, \ldots)$ and select its n^{th}-element with some function $\mathtt{nth}$. Then one might wish that

$$\lambda H.(\lambda n.\,\mathtt{wit}\,(\mathtt{nth}\ n\ H_\infty), \lambda n.\,\mathtt{prf}\,(\mathtt{nth}\ n\ H_\infty))$$

could stand for a proof for $AC_{\mathbb{N}}$. However, even if we were effectively able to build such a term, H_∞ might still contain some classical proof. Therefore two copies of $H\,n$ might end up being different according to the contexts in which they are executed, and then return two different witnesses. This problem could be fixed by using a shared version of H_∞, say

$$\lambda H.\,\mathtt{let}\ a = H_\infty\ \mathtt{in}\ \ (\lambda n.\,\mathtt{wit}\,(\mathtt{nth}\ n\ a), \lambda n.\,\mathtt{prf}\,(\mathtt{nth}\ n\ a))\,.$$

It only remains to formalize the intuition of H_∞. This is done by means of a coinductive fixpoint operator. We write $\mathtt{cofix}^t_{bx}[p]$ for the co fixpoint operator binding the variables b and x, where p is a proof and t a term. Intuitively, such an operator is intended to reduce according to the rule:

$$\mathtt{cofix}^t_{bx}[p] \ \to\ p[t/x][\lambda y.\mathtt{cofix}^t_{bx}[p]/b]$$

This is to be compared with the usual inductive fixpoint operator which we write $\mathtt{ind}^t_{bx}[p_0 \mid p_S]$ (which binds the variables b and x) and which reduces as follows:

$$\mathtt{ind}^0_{bx}[p_0 \mid p_S] \ \to\ p_0 \qquad\qquad \mathtt{ind}^{S(t)}_{bx}[p_0 \mid p_S] \ \to\ p_S[t/x][\mathtt{ind}^t_{bx}[p_0 \mid p_S]/b]$$

The presence of coinductive fixpoints allows us to consider the proof term $\mathtt{cofix}^0_{bn}[(Hn, b(S(n)))]$, which implements a stream eventually producing the (informal) infinite sequence H_∞. Indeed, this proof term reduces as follows:

$$\mathtt{cofix}^0_{bn}[(Hn, b(S(n)))] \to (H\,0, \mathtt{cofix}^1_{bn}[(Hn, b(S(n)))]) \to (H\,0, (H\,1, \mathtt{cofix}^2_{bn}[(Hn, b(S(n))))]) \to \ldots$$

This allows for the following definition of a proof term for the axiom of countable choice:

$$AC_{\mathbb{N}} := \lambda H.\, \mathtt{let}\, a = \mathtt{cofix}^0_{bn}[(Hn, b(S(n)))]\, \mathtt{in}\, (\lambda n.\, \mathtt{wit}\, (\mathtt{nth}\, n\, a), \lambda n.\, \mathtt{prf}\, (\mathtt{nth}\, n\, a))\,.$$

Whereas $\mathtt{let}\, a = \ldots \mathtt{in} \ldots$ suggests a call-by-value discipline, we cannot afford to pre-evaluate each component of the stream. In turn, this imposes a *lazy* call-by-value evaluation discipline for coinductive objects. However, this still might be responsible for some non-terminating reductions, all the more as classical proofs may contain backtrack.

If we analyze what this construction does at the level of types[12], at first approximation it turns a proof (H) of the formula $\forall x^{\mathbb{N}}.A(x)$ (with $A(x) = \exists y.P(x,y)$ in that case) into a proof (the stream H_∞) of the (informal) infinite conjunction $A(0) \wedge A(1) \wedge A(2) \wedge \ldots$. Formally, a proof $\mathtt{cofix}^t_{bx}[p]$ is an inhabitant of a coinductive formula, written $\nu^t_{Xx}A$ (where t is a terms and which binds the variables X and n). The typing rule is given by:

$$\frac{\Gamma \vdash t : T \quad \Gamma, x : T, b : \forall y^T.Xy \vdash p : A}{\Gamma \vdash \mathtt{cofix}^t_{bx}[p] : \nu^t_{Xx}A}\ (\mathtt{cofix})$$

with the side condition that X can only occurs in positive position in A. Coinductive formulas are defined with a reduction rules which is very similar to the rule for the co-fixpoint:

$$\nu^t_{Xx}A \quad \triangleright \quad A[t/x][\nu^t_{Xy}A/Xy]$$

In particular, the term $\mathtt{cofix}^0_{bn}[(Hn, b(S(n))]$ is thus an inhabitant of a coinductively defined (infinite) conjunction, written $\nu^0_{Xn}(A(n) \wedge X(S(n)))$. This formula indeed reduces accordingly to the reduction of the stream:

$$\nu^0_{Xn}(A(n) \wedge X(S(n))) \triangleright A(0) \wedge [\nu^1_{Xn}(A(n) \wedge X(S(n)))] \triangleright A(0) \wedge A(1) \wedge [\nu^0_{Xn}(A(n) \wedge X(S(n)))] \triangleright \ldots$$

More generally, at the level of formulas, the key was to identify the formula $A(x)$ and a suitable law $g : \mathbb{N} \to T$ to turn a proof of $\forall x^T.A(x)$ into the conjunction $A(g(0)) \wedge A(g(1)) \wedge A(g(2)) \wedge \ldots$. In the case of the axiom of countable choice, this law was simply this identity. In the case of the axiom of dependent choice, the law g we are looking for is precisely the choice function. We can thus use the same trick to define a proof term for DC. The stream we actually construct corresponds to the coinductive formula $\nu^{x_0}_{Xn}[\exists y^{\mathbb{N}}.(P(x,y) \wedge X(y))]$, which ultimately unfolds into:

$$\nu^{x_0}_{Xn}[\exists y.(P(x,y) \wedge X(y))] \quad \triangleright \quad \cdots \quad \triangleright \quad \exists x_1^{\mathbb{N}}.(P(x_0,x_1) \wedge \exists x_2^{\mathbb{N}}.(P(x_1,x_2) \wedge \exists x_3^{\mathbb{N}}.(P(x_2,x_3) \wedge \ldots)))$$

Given a proof $H : \forall x.\exists y.P(x,y)$ and a term x_0 , we can define a stream corresponding to this coinductive formula by $\boldsymbol{str}\, x_0 := \mathtt{cofix}^{x_0}_{bn}[(\mathtt{dest}\ H\, n\ \mathtt{as}\ ((y,c))\ \mathtt{in}\, (y,(c,(b\, y))))]$. This term reduces as expected:

$$(x_0, \boldsymbol{str}\, x_0) \ \to\ (x_0,(x_1,(p_1, \boldsymbol{str}\, x_1))) \ \to\ (x_0,(x_1,(p_1,(x_2,(p_2, \boldsymbol{str}\, x_2))))) \ \to\ \ldots$$

where $p_i : P(x_{i-1}, x_i)$. From there, it is almost direct to extract the choice function f (which maps any $n \in \mathbb{N}$ to x_n) and the corresponding certificate that $(f(0) = x_0 \wedge \forall n \in \mathbb{N}.P(f(n), f(S(n))))$. In practice, it essentially amounts to define the adequate $\mathtt{nth}$ function. We will give a complete definition of the proof term for the axiom of dependent choice in Chapter 8.

[12]We delay the formal introduction of a type system and the given of the typing derivation for $AC_{\mathbb{N}}$ to Chapter 8.

5.2.2 An overview of dPA$^\omega$

Formally, the calculus dPA$^\omega$ is a proof system for the language of classical arithmetic in finites types (abbreviated PA$^\omega$), where the 'd' stands for "dependent". Its stratified presentation allows us to separate terms (the arithmetical objects) from proofs. Finite types and formulas are thus separated as well, corresponding to the following syntax:

$$\begin{array}{lrcl} \textbf{Types} & T,U & ::= & \mathbb{N} \mid T \to U \\ \textbf{Formulas} & A,B & ::= & \top \mid \bot \mid t = u \mid A \wedge B \mid A \vee B \mid \Pi a : A.B \mid \forall x^T.A \mid \exists x^T.A \mid \nu^t_{x,f} A \end{array}$$

Terms, denoted by $t, u, \ldots$ are meant to represent arithmetical objects, their syntax thus includes:

- a term 0 and a successor S;
- an operator $\mathtt{rec}^t_{xy}[t_0 \mid t_S]$ for recursion, which binds the variables x and y: where t is the term on which the recursion is performed, t_0 is the term for the case $t = 0$ ans t_S is the term for case $t = S(t')$;
- λ-abstraction $\lambda x.t$ to define functions;
- terms application $t\,u$;
- a `wit` constructor to extract the witness of a dependent sum.

As for proofs, denoted by $p, q, \ldots$, they contain:

- pairs (p, q) to prove logical conjunctions;
- destructors of pairs $\mathtt{split}\ p\ \mathtt{as}\ (a_1, a_2)\ \mathtt{in}\ q$ which binds the variables a_1 and a_2 in q;
- injections $\iota_i(p)$ for the logical disjunction;
- pattern-matching $\mathtt{case}\ p\ \mathtt{of}\ [a_1.p_1 \mid a_2.p_2]$ which binds the variables a_1 in p_1 and a_2 in p_2;
- a proof term `refl` which is the proof of atomic equalities $t = t$;
- $\mathtt{subst}\ p\ q$ which eliminates an equality proof $p : t = u$ to get a proof of $B[u]$ from a proof $q : B[t]$;
- pairs (t, p) where t is a term and p a proof for the dependent sum type;
- $\mathtt{prf}\ p$ which allows us to extract the certificate of a dependent pair;
- non-dependent destructors $\mathtt{dest}\ p\ \mathtt{as}\ (x, a)\ \mathtt{in}\ q$ which binds the variables x and a in q;
- abstractions over terms $\lambda x.p$ and applications $p\,t$;
- (possibly) dependent abstractions over proofs $\lambda a.p$ and applications $p\,q$;
- a construction $\mathtt{let}\ a = p\ \mathtt{in}\ q$, which binds the variable a in q and which allows for sharing;
- operators $\mathtt{ind}^t_{ax}[p_0 \mid p_S]$ and $\mathtt{cofix}^t_{bx}[p]$ that we already described for inductive and coinductive reasoning;
- control operators $\mathtt{catch}_\alpha\ p$ (which binds the variable α in p) and $\mathtt{throw}\ \alpha\ p$ (where α is a variable and p a proof)
- $\mathtt{exfalso}\ p$ where p is intended to be a proof of false.

This results in the following syntax:

$$\begin{array}{lrcl} \textbf{Terms} & t,u & ::= & x \mid 0 \mid S(t) \mid \mathtt{rec}^t_{xy}[t_0 \mid t_S] \mid \lambda x.t \mid t\ u \mid \mathtt{wit}\ p \\ \textbf{Proofs} & p,q & ::= & a \mid \iota_i(p) \mid \mathtt{case}\ p\ \mathtt{of}\ [a_1.p_1 \mid a_2.p_2] \mid (p,q) \mid \mathtt{split}\ p\ \mathtt{as}\ (a_1,a_2)\ \mathtt{in}\ q \\ & & \mid & (t,p) \mid \mathtt{prf}\ p \mid \mathtt{dest}\ p\ \mathtt{as}\ (x,a)\ \mathtt{in}\ q \mid \lambda x.p \mid p\,t \\ & & \mid & \lambda a.p \mid p\,q \mid \mathtt{let}\ a = p\ \mathtt{in}\ q \mid \mathtt{refl} \mid \mathtt{subst}\ p\,q \\ & & \mid & \mathtt{ind}^t_{ax}[p_0 \mid p_S] \mid \mathtt{cofix}^t_{bx}[p] \mid \mathtt{exfalso}\ p \mid \mathtt{catch}_\alpha\ p \mid \mathtt{throw}\ \alpha\ p \end{array}$$

The problem of degeneracy caused by the conjoint presence of classical proofs and dependent types is solved by enforcing a compartmentalization between them. Dependent types are restricted to the set of *negative-elimination-free* proofs (NEF), which are a generalization of values preventing from backtracking evaluations by excluding expressions of the form $p\,q$, $p\,t$, $\mathtt{exfalso}\ p$, $\mathtt{catch}_\alpha\ p$ or $\mathtt{throw}\ \alpha p$ which are outside the body of a λx or λa. Syntactically, they are defined by:

$$\begin{array}{lll}
\textbf{Values} & V_1, V_2 ::= & a \mid \iota_i(V) \mid (V_1, V_2) \mid (t, V) \mid \lambda x.p \mid \lambda a.p \mid \mathtt{refl} \\
\textbf{NEF} & N_1, N_2 ::= & a \mid \iota_i(N) \mid \mathtt{case}\ p\ \mathtt{of}\ [a_1.N_1 \mid a_2.N_2] \mid (N_1, N_2) \mid \mathtt{split}\ N_1\ \mathtt{as}\ (a_1, a_2)\ \mathtt{in}\ N_2 \\
& & \mid\ (t, N) \mid \mathtt{prf}\ N \mid \mathtt{dest}\ N_1\ \mathtt{as}\ (x, a)\ \mathtt{in}\ N_2 \mid \lambda x.p \\
& & \mid\ \lambda a.p \mid \mathtt{let}\ a = N_1\ \mathtt{in}\ N_2 \mid \mathtt{refl} \mid \mathtt{subst}\ N_1\ N_2 \\
& & \mid\ \mathtt{ind}^t_{ax}[N_0 \mid N_S] \mid \mathtt{cofix}^t_{bx}[N]
\end{array}$$

This allows to restrict typing rules involving dependencies, notably the rules for $\mathtt{prf}$ or $\mathtt{let} = \mathtt{in}$:

$$\frac{\Gamma \vdash p : \exists x^T.A(x) \quad p \in \text{NEF}}{\Gamma \vdash \mathtt{prf}\ p : A(\mathtt{wit}\ p)}\ (\mathtt{prf}) \qquad \frac{\Gamma \vdash p : A \quad \Gamma, a : A \vdash q : B \quad a \notin FV(B) \text{ if } p \notin \text{NEF}}{\Gamma \vdash \mathtt{let}\ a = p\ \mathtt{in}\ q : B[p/a]}\ (\text{Cut})$$

About reductions, let us simply highlight the fact that they globally follow a call-by-value discipline, for instance in this sample:

$$\begin{array}{lcl}
(\lambda a.p)\,q & \to & \mathtt{let}\ a = q\ \mathtt{in}\ p \\
\mathtt{let}\ a = (p_1, p_2)\ \mathtt{in}\ p & \to & \mathtt{let}\ a_1 = p_1\ \mathtt{in}\ \mathtt{let}\ a_2 = p_2\ \mathtt{in}\ p[(a_1, a_2)/a] \\
\mathtt{let}\ a = V\ \mathtt{in}\ p & \to & p[V/a]
\end{array}$$

except for co-fixpoints which are lazily evaluated:

$$\begin{array}{lcl}
F[\mathtt{let}\ a = \mathtt{cofix}^t_{bx}[q]\ \mathtt{in}\ p] & \to & \mathtt{let}\ a = \mathtt{cofix}^t_{bx}[q]\ \mathtt{in}\ F[p] \\
\mathtt{let}\ a = \mathtt{cofix}^t_{bx}[q]\ \mathtt{in}\ D[a] & \to & \mathtt{let}\ a = q[\lambda y.\mathtt{cofix}^y_{bx}[q]/b][t/x]\ \mathtt{in}\ D[a]
\end{array}$$

In the previous rules, the first one expresses the fact that evaluation of co-fixpoint under contexts $F[\,]$ are momentarily delayed. The second rules precisely corresponds to a context where the co-fixpoint is linked to a variable a whose value is needed, a step of unfolding is then performed.

The full type system, as well as the complete set of reduction rules, are given in [70], and will be restated with a different presentation in Chapter 8. In the same paper, some important properties of the calculus are given. In particular, dPA$^\omega$ verifies the property of subject reduction, and provided it is normalizing, there is no proof of false.

Theorem 5.1 (Subject reduction)**.** *If $\Gamma \vdash p : A$ and $p \to q$, then $\Gamma \vdash q : A$.*

Proof (sketch). By induction on the derivation of $p \to q$, see [70]. □

Theorem 5.2 (Conservativity)**.** *Provided dPA$^\omega$ is normalizing, if A is $\to$-ν-$\mathtt{wit}$-$\forall$-free, and $\vdash_{dPA^\omega} p : A$, there is a value V such that $\vdash_{HA^\omega} V : A$.*

Proof (sketch). Considering a closed proof p of A, p can be reduced. By analysis of the different possible cases, it can be found a closed value of type A. Then using the fact that A is a $\to$-ν-$\mathtt{wit}$-$\forall$-free formula, V does not contain any subexpression of the form $\lambda x.p$ or $\lambda a.p$, by extension it does not contain either any occurrence of $\mathtt{exfalso}\ p$, $\mathtt{catch}_\alpha\ p$ or $\mathtt{throw}\ \alpha p$ and is thus a proof of A already in HA^ω. □

Theorem 5.3 (Consistency)**.** *Provided dPA$^\omega$ is normalizing, it is consistent, that is: $\nvdash_{dPA^\omega} p : \bot$.*

Proof. The formula $\bot$ is a particular case of $\to$-ν-$\mathtt{wit}$-$\forall$-free formula, thus the existence of a proof of false in dPA$^\omega$ would imply the existence of a contradiction already in HA^ω, which is absurd. □

The last two results rely on the property of normalization. Unfortunately, the proof sketch that is given in [70] to support the claim that dPA$^\omega$ normalizes turns out to be hard to formalize properly. Since, moreover, dPA$^\omega$ contains both control operators (allowing for backtrack) and co-fixpoints (allowing infinite objects, like streams), which can be combined and interleaved, we should be very suspicious *a priori* about this property. Anyhow, the proof sketch from [70] to use metamathematical arguments, which are more distant from a computational analysis through a proof by realizability or by means of a continuation-passing style translation. Such proofs are of interest in themselves already for what they taught us about the fine behavior of a calculus.

5.3 Toward a proof of normalization for dPA$^\omega$

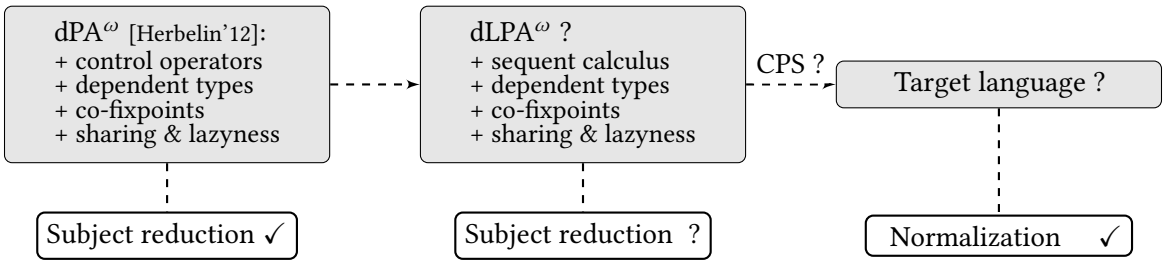
5.3.1 The big picture

An important part of this thesis has been devoted to the search for a proof of normalization for dPA$^\omega$ by means of a realizability interpretation or by a continuation-passing-style translation. Aside from the very result of normalization, this approach is of interest for different reasons which are deeply related to the difficulties of obtaining such a proof. Indeed, a direct continuation-passing style is very harsh to obtain for dPA$^\omega$ as such. In addition to the difficulties caused by control operators and co-fixpoints, the reduction system is defined in a natural-deduction style with contextual rules (as in the rule to reduce proofs of the shape $\mathtt{let}\, a = \mathtt{cofix}^t_{bx}[p]\, \mathtt{in}\, D[a]$) where the contexts involved can be of arbitrary depth. This kind of rules are, in general and especially in this case, very difficult to translate faithfully through a continuation-passing style translation.

All in all, there are several difficulties in getting a direct proof by CPS or realizability. Hence, we shall study them separately, hopefully solving them independently will lead us to a solution to the main problem. Roughly, our strategy consists of two steps:

1. reduce dPA$^\omega$ to an equivalent presentation in a sequent calculus fashion,
2. use the methodology of semantic artifacts to define a CPS or a realizability interpretation.

Indeed, a sequent calculus presentation of a calculus is usually a good intermediate step for compilation or for CPS translations [39]. This presentation should of course verify at least the property of subject reduction and its reduction system should mimic the one of dPA$^\omega$. Schematically, this corresponds to the following roadmap where question marks indicate what is to be done:



To be fair, this approach is idealistic. In particular, we will not formally define an embedding for the first arrow, since we are not interested in dPA$^\omega$ for itself, but rather in the computational content of the proofs for countable and dependent choice. Hence, we will content ourself with a sequent calculus presentation of dPA$^\omega$ which allows for similar proof terms, which we call dLPA$^\omega$, without bothering to prove that the reduction systems are equivalent. As for the second arrow, as advocated in the previous section, the search for a continuation-passing style translation or a realizability interpretation can coincide for a large part. We shall thus apply the methodology of semantic artifacts and in the end, choose the easiest possibility.

From this roadmap actually arises two different subproblems that are already of interest in themselves. Forgetting about the general context of dPA$^{\omega}$, we shall first wonder whether these easier questions have an answer:

1. Is it possible to define a (classical) sequent calculus with a form of dependent types? If so, would it be compatible with a typed continuation-passing style translation?
2. Can we prove the normalization of a call-by-need calculus with control operators? Can we define a Krivine realizability interpretation of such a calculus?

5.3.2 Realizability interpretation and CPS translation of classical call-by-need

Fortunately, there were already some work in the direction proposed by the second item. In two consecutive articles, Ariola *et al.* studied the question of defining sequent-calculus style versions of call-by-need, leading to a natural extension of call-by-need with control operators [6, 4]. Such a calculus can be expressed in the framework of the $\lambda\mu\tilde{\mu}$-calculus (Chapter 4), and by applying the same methodology of semantic artifacts, the authors showed how to derive (an untyped) CPS translation to the pure λ-calculus. This translation is in fact an environment-and-continuation-passing style translation, so that there is no direct way of inferring a type translation from the computational one. The question thus becomes: can we type this translation to prove the normalization of a call-by-need calculus with control operators? Does this translation lead to a realizability interpretation as it usually does with the call-by-name and call-by-value $\lambda\mu\tilde{\mu}$-calculi?

We shall see in Chapter 6 that the methodology of semantics artifacts can be pushed one step further to obtain a realizability interpretation for the $\overline{\lambda}_{[lv\tau\star]}$-calculus, a call-by-need calculus with control operators and explicit stores. Aside to prove the normalization of the calculus, this also open the door to the interpretation of stores, memory cells in Krivine realizability. Besides, this interpretation a type system, which is an extension of system F and that we call F_{Υ}. This allows us to type the CPS translation from [4]. Interestingly, we will see that through the translation, the preservation of typing for the store (which is extensible) is obtained by means of a Kripke-style forcing. As far as we know, all these results constitute new contributions.

5.3.3 A sequent calculus with dependent types

The first question, that is to develop a (classical) sequent calculus with dependent types and to ensure the compatibility with a CPS translation, is harder. Indeed, while sequent calculi smoothly supports abstract machine and continuation-passing style interpretations, there is no such presentation of a language with dependent types. Besides, viewed the other way round—can we add control operators to a language with dependent types?—, the question has to do with the more general problem of including side-effects in (dependent) type theory. This issue is one of the hot topic from the past few years in theoretical computer science, in that it aims at filling the gap between type theories and mainstream languages. If there have been proposals for different classes of side-effects, mainly through monads, control operators and classical logic usually do not fit in the picture.

In Chapter 7, we shall start from the call-by-value $\lambda\mu\tilde{\mu}$-calculus and see how to design a minimal language with a value restriction and a type system that includes a list of explicit dependencies to maintain type safety. We will then show how to relax the value restriction and introduce delimited continuations to directly prove the consistency by means of a continuation-passing-style translation. The translation will faithfully embody the dependencies and preserve the normalization. Finally, we will relate our calculus to a similar system by Lepigre [108], whose consistency is proved by means of a realizability interpretation. We present a methodology to transfer properties from his system to our calculus, in particular we can infer proofs of normalization and soundness for our calculus.