A Computational Cantor-Bernstein and Myhill's Isomorphism Theorem in Constructive Type Theory (Proof Pearl)

Yannick Forster
Inria, Gallinette Project-Team
Nantes, France
Saarland University
Saarbrücken, Germany
yannick.forster@inria.fr

Felix Jahn
Saarland University
Saarbrücken, Germany
s8fejahn@stud.uni-saarland.de

Gert Smolka
Saarland University
Saarbrücken, Germany
smock@cs.uni-saarland.de

Abstract

The Cantor-Bernstein theorem (CB) from set theory, stating that two sets which can be injectively embedded into each other are in bijection, is inherently classical in its full generality, i.e. implies the law of excluded middle, a result due to Pradic and Brown. Recently, Escardó has provided a proof of CB in univalent type theory, assuming the law of excluded middle. It is a natural question to ask which restrictions of CB can be proved without axiomatic assumptions. We give a partial answer to this question contributing an assumption-free proof of CB restricted to enumerable discrete types, i.e. types which can be computationally treated.

In fact, we construct several bijections from injections:

The first is by translating a proof of the Myhill isomorphism theorem from computability theory – stating that 1-equivalent predicates are recursively isomorphic – to constructive type theory, where the bijection is constructed in stages and an algorithm with an intricate termination argument is used to extend the bijection in every step.

The second is also constructed in stages, but with a simpler extension algorithm sufficient for CB.

The third is constructed directly in such a way that it only relies on the given enumerations of the types, not on the given injections.

We aim at keeping the explanations simple, accessible, and concise in the style of a "proof pearl". All proofs are machine-checked in Coq but should transport to other foundations – they do not rely on impredicativity, on choice principles, or on large eliminations.

CCS Concepts: • **Theory of computation** → **Computability**; **Type theory**.

Keywords: type theory, computability theory, constructive mathematics, constructive logic, Coq

ACM Reference Format:
Yannick Forster, Felix Jahn, and Gert Smolka. 2023. A Computational Cantor-Bernstein and Myhill's Isomorphism Theorem in Constructive Type Theory (Proof Pearl). In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '23), January 16–17, 2023, Boston, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3573105.3575690

1 Introduction

It is well-known that theorems in constructive type theory can be interpreted as theorems about computable functions through realizability interpretations. That the converse direction can be fruitful as well is not surprising, but rarely exploited: any construction from computability theory that does not rely on a universal function (as provided by e.g. a "universal" Turing machine) gives rise to a theorem in constructive type theory. We use this inverse approach to prove a fully constructive version of the Cantor-Bernstein theorem (CB) in constructive type theory.

In set theory, the Cantor-Bernstein theorem states that whenever there are injections $f : X \to Y$ and $g : Y \to X$ one can construct a bijection. The argument usually presented in modern textbooks is by a back-and-forth construction in stages due to Kőnig [12], which makes use of the law of excluded middle. The theorem was discovered by Cantor [2] and Bernstein published the first correct proof not relying on the axiom of choice [1].[1]

CPP '23, January 16–17, 2023, Boston, MA, USA

ACM ISBN 979-8-4007-0026-2/23/01…$15.00
https://doi.org/10.1145/3573105.3575690

[1] The theorem is often also called "Cantor-Schröder-Bernstein", "Schröder-Bernstein", or "Cantor-Bernstein-Schröder" theorem. Ernst Schröder published a proof of CB in the same year as Bernstein [16], but it turned out to be wrong as observed by Korselt [11]. Schröder acknowledged this fact to Korselt, stating "Daß ich Herrn F. Bernstein die Ehre, den G. Cantorschen Satz bewiesen zu haben, allein überlasse, hatte ich einstweilen einem Freunde desselben, Herrn Dr. Max Dehn (jetzt in Münster) schon vorigen

More than 120 years later, Pradic and Brown show that in fact Cantor-Bernstein implies the law of excluded middle in IZF [14], i.e. in Zermelo-Fraenkel set theory without the law of excluded middle. Given a constructive type theory with the axiom of unique choice such as homotopy type theory, the proof can be directly translated. Recently, Escardó has given a proof of the Cantor-Bernstein theorem in constructive type theory, concretely in homotopy type theory as described by the HoTT book [19], necessarily relying on the law of excluded middle [4]. In this paper, we consider the question:

For which types is CB provable without any assumptions?

The question has been explicitly posed as a challenge by Escardó [5], who asks "how much beyond finite sets with decidable equality can [CB] apply to in constructive mathematics?" Our partial answer to this question is that if X and Y are enumerable discrete types – or equivalently retracts of the natural numbers – then the bijection can be computably constructed and no assumptions are necessary.

From a cardinality perspective available in classical logic, the theorem is relatively easy to prove. Types which are retracts of the natural numbers are discrete and either finite or in bijection to the natural numbers. Then the following four cases are possible:

- X and Y are both finite, then the existence of injections enforces that they are of same cardinality, and finite types with same cardinality are in bijection.
- only exactly one of X and Y is finite, then one can obtain a contradiction using the injections.
- X and Y are both in bijection to the natural numbers and thus in bijection.

Thus, the challenge in proving this CB for enumerable discrete types constructively lies in finding a *uniform* construction covering all cases, without a global case distinction on finiteness which is not allowed in constructive logic. Such a uniform construction is not necessary from the perspective of classical logic, but as usual an aesthetic argument can be made for uniform proofs without global case distinctions.

Herbst resp. Sommer – natürlich zum Weitergeben – gesagt, desgleichen über diese Angelegenheit (unspezifiziert, als durch meinerseits zu erledigende Gewissenssache) Herrn Cantor einen Brief in Aussicht gestellt und Sept. 01 angefangen, jedoch leider noch nicht zu Ende gebracht..." ("That I leave the honour of having proved G. Cantor's theorem to Mr. F. Bernstein alone, I had already told a friend of his, Dr. Max Dehn (now in Münster), last autumn or summer – to pass on, of course – and likewise about this matter (unspecified as a matter of conscience on my part) I had also held out the prospect of a letter to Mr. Cantor and had begun it on Sept. 01, but unfortunately had not yet finished it.") Cantor published a proof before Bernstein, but it relied, unbeknown to Cantor, on the axiom of choice. Dedekind independently proved the theorem twice, but published it later than Bernstein. We thus adapt the terminology of Pradic and Brown and call the theorem "Cantor-Bernstein" theorem: after the person who discovered it and the person who gave the first proof. See the Wikipedia article for a historic overview [20].

We give several algorithmically different proofs of the computational form of the Cantor-Bernstein theorem for enumerable discrete types:

The first observes that CB for enumerable discrete types is a consequence of the Myhill isomorphism theorem from computability theory, which does not rely on a universal machine and can thus be proved in constructive type theory. The proof proceeds in stages which are constructed on after another.

The second analyses the argument used in the Myhill isomorphism theorem and simplifies it to a construction based on a computational pigeonhole principle, which is sufficient for CB but too naïve for the Myhill isomorphism theorem. This proof still constructs the bijection in stages.

The third changes the construction further to not rely on stages. Furthermore, this proof yields a bijection which is computationally independent of the provided injections.

We consider the third proof to be the simplest proof and put emphasis on providing a standalone proof script for it. We describe all proofs in informal natural language, but provide Coq proofs for all results, which can be found at

https://github.com/uds-psl/coq-synthetic-computability/tree/cantor_myhill.

The Lemmas, Theorems, and Corollaries of this paper can be clicked to see the corresponding html version of the Coq files.

We do not rely on advanced features of Coq's type theory (the Calculus of Inductive Constructions, CIC) such as impredicativity, and we discuss for all constructions explicitly how they can be carried out in versions of Martin-Löf type theory (MLTT), where no universe of propositions is explicit, and of homotopy type theory (HoTT), where being a proposition is a semantic rather than syntactic notion.

As a running example for the whole paper, we will consider the (finite) types $\mathbb{8}$ and $\mathbb{A}$ with injections $f : \mathbb{8} \to \mathbb{A}$ and $g : \mathbb{A} \to \mathbb{8}$ between them as follows:

$$\mathbb{8} := 1|2|3|4|5|6|7|8$$

$$\mathbb{A} := A|B|C|D|E|F|G|H$$

	A	B	C	D	E	F	G	H
f	↓	↓	↓	↓	↓	↓	↓	↓
	4	3	6	1	5	8	2	7

	1	2	3	4	5	6	7	8
g	↓	↓	↓	↓	↓	↓	↓	↓
	C	E	A	B	D	H	F	G

Of course it is trivial for the example to give a bijection, but we explain how the different uniform constructions of a bijection $F : \mathbb{8} \to \mathbb{A}$ and $G : \mathbb{A} \to \mathbb{8}$ act on the example, yielding three different bijections.

Central for all results are then notions of enumerability and discreteness for types, which we introduce in section 2, alongside with a discussion of existence and the so-called Δ^0_1-choice principle in different type theories. In section 3 we

give a proof of the Myhill isomorphism theorem and deduce CB for enumerable discrete types in section 4. In section 5 we give a proof of CB based on a computable pigeonhole principle. In section 6 we give a direct proof of CB, before concluding in section 7.

2 Enumerable Discrete Types

We write $\mathbb{P}$ for the type of propositions we use. Different type theories take different stances on propositions. In CIC, this is exactly the impredicative universe of propositions. In MLTT, there is no single universe $\mathbb{P}$ and one has to read $\mathbb{P}$ as a (suitable) type universe. In HoTT, $\mathbb{P}$ has to be read as the universe of homotopy propositions `hProp`, i.e. of propositions with unique proofs. Independent of the concrete incarnation of HoTT, there is always an operation to turn a type X into the proposition $\|X\|$ stating that X is inhabited.

Depending on the system, types in $\mathbb{P}$ can be computationally treated as input types to functions or nto. In MLTT, this is possible without restrictions, since propositions are just types, e.g one can write non-constant functions of type $\top \vee \top \to \mathbb{B}$. In CIC, so-called large eliminations are only allowed on inductive types of a certain syntactic form enforcing the absence of computational content, and one cannot write a non-constant function of type $\top \vee \top \to \mathbb{B}$ (but also not prove the absence of such non-constant functions). In all incarnations of HoTT, elimination of propositions $\|X\|$ is allowed if the target type is an `hProp`, i.e. where the result of the function cannot depend on the proof. Note that one can write a non-constant function of type $\top \vee \top \to \mathbb{B}$, but $\top \vee \top$ is no proposition since it has two proofs.

We will only need that proofs of propositions of the form

$$\forall x : \mathbb{N}.\exists y : \mathbb{N}.Fxy = \mathsf{true}$$

for $F : \mathbb{N} \to \mathbb{N} \to \mathbb{B}$ can be turned into functions, we will discuss such a choice-like principle later in this section.

First, we introduce the relevant terminology regarding types. We call a type X discrete if its equality relation is decidable, i.e. if

$$\exists f : X \to X \to \mathbb{B}.\ \forall x_1 x_2 : X.\ x_1 = x_2 \leftrightarrow f x_1 x_2 = \mathsf{true}.$$

Here, as usual, the type $\mathbb{B}$ is the inductive type with constructors true and false.

We call a type X enumerable if there is a function enumerating all its elements – possibly with repetitions:

$$\exists f : \mathbb{N} \to \mathbb{O}X.\ \forall x : X.\ \exists n : \mathbb{N}.\ fn = \mathsf{Some}\, x.$$

Here, Some is the constructor of the option type of type $X \to \mathbb{O}X$, and None is the (only) other element of this type. Sometimes, enumerable types are also called "countable", but we use the computational terminology to emphasise the computational content of the definition. Note that for enumerability, if X is non-empty one can equivalently ask for a function $\mathbb{N} \to X$. All of these definitions are standard and frequently used in type theory.

A type X is a retract of a type Y if there are functions $I : X \to Y$ and $R : Y \to \mathbb{O}X$ such that $\forall x : X.\ R(Ix) = \mathsf{Some}\, x$.

Lemma 2.1. *If X is a retract of Y via $I : X \to Y$ and $R : Y \to \mathbb{O}X$, then I is injective, i.e. $\forall x_1 x_2.\ Ix_1 = Ix_2 \to x_1 = x_2$.*

We use this definition of injectivity throughout the paper. Note that while every retraction gives rise to an injection, the converse does not hold constructively.

Lemma 2.2. *Let X be a retract of Y.*

1. *If Y is discrete, then X is discrete.*
2. *If Y is enumerable, then X is enumerable.*

Corollary 2.3. *If X is a retract of $\mathbb{N}$, then X is enumerable and discrete.*

To be able to show the inverse direction we need to briefly discuss how to obtain functions from statements of the form $\forall x.\exists y$. In general, such a principle is called the type-theoretic axiom of choice. For the above use case, it suffices to consider the restricted case $\forall x.\exists y : \mathbb{N}.\ Rxy$ for R being a decidable relation.

In constructive mathematics, this is known as Δ^0_1-choice, in Coq's standard library it is called constructive indefinite ground description and is a consequence of large elimination on the accessibility predicate, in MLTT it is trivially provable since the meaning of $\forall x.\exists y$ is already a function, and in HoTT it is a consequence of unique choice (since one can equivalently ask for the least y such that Rxy), which in turn is a consequence of the universal property of propositional truncation.

Lemma 2.4. *If $F : X \to Y \to \mathbb{B}$ and Y is enumerable and discrete, then whenever $\forall x.\exists y.\ Fxy = \mathsf{true}$ one can define a function $I : X \to Y$ with $\forall x.\ Fx(Ix) = \mathsf{true}$.*

Note that if one defines $\exists y.\ py$ in a weaker way, e.g. as $\forall A.(\forall y.\ py \to A) \to (A \to \bot) \to \bot$ (or, equivalently, $\neg\neg(\Sigma y.\ py)$) in MLTT, then the choice principle is only provable under the assumption of Markov's principle.

Lemma 2.5. *Any enumerable discrete type X is a retract of $\mathbb{N}$.*

Proof. Let d decide equality on X and e be an enumerator. We can take $R := e$. We know that $\forall x.\exists n.\ Rn = \mathsf{Some}\, x$ then, but we need to be able to compute n.

The choice principle for decidable relations yields such a function I, since $Rn = \mathsf{Some}\, x$ is decidable using d. □

We will use the terminology "constructive" in this paper to denote provability in CIC without referring to any other large eliminations than the ones explained in this section.

3 Myhill Isomorphism Theorem

We write $p \preceq_1 q$ if a predicate $p : X \to \mathbb{P}$ is one-one reducible to $q : Y \to \mathbb{P}$ and define

$$p \preceq_1 q := \exists f : X \to Y.\ (\forall x.\ px \leftrightarrow q(fx)) \wedge f \text{ is injective.}$$

We call the function f a one-one reduction from p to q. We prove that when two predicates are one-one reducible to each other, with no relation between the reductions functions, one can construct one-one reductions which are inverses of each other, i.e. a type-theoretic version of Myhill's isomorphism theorem from computability theory [13]. We follow the proof by Rogers [15, §7.4 Th. VI], where the isomorphism is constructed in stages, formed by *correspondence sequences* between predicates p and q, which are finitary bijections represented as lists $C : \mathbb{L}(X \times Y)$ (we use $\mathbb{L}$ for the type of lists) such that for all $(x, y) \in C$

1. $px \leftrightarrow qy$
2. $\forall y'.\ (x, y') \in C \to y = y'$
3. $\forall x'.\ (x', y) \in C \to x = x'$

We write $x \in_1 C$ ($x \in_2 C$) if x is an element of the first (second) projection of C.

The crux of the theorem is that for any correspondence sequence C with $p \preceq_1 q$ and $x_0 \notin_1 C$ one can *compute* y_0 such that $(x_0, y_0) :: C$ is a correspondence sequence again, with no condition on p and q.

The termination argument is only given informally in Rogers' textbook but in fact not trivial to formally reconstruct: It requires reasoning about cardinality of the projections of the constructed correspondence sequence, with several case distinctions. We give a direct construction of an algorithm which does not require an intricate termination argument, considerably simplifying the overall proof.

Lemma 3.1. *Let f be a one-one reduction from $p : X \to \mathbb{P}$ to $q : Y \to \mathbb{P}$. There is a function* $\mathsf{find} : \mathbb{L}(X \times Y) \to X \to Y$ *such that if C is a correspondence sequence for p and q and $x_0 \notin_1 C$, then* $\mathsf{find}\ C\ x_0 \notin_2 C$ *and $px_0 \leftrightarrow q(\mathsf{find}\ C\ x_0)$.*

Proof. We first define a function $\gamma : \mathbb{L}(X \times Y) \to X \to X$ recursive in $|C|$:

$$\gamma C x := x \text{ if } fx \notin_2 C$$
$$\gamma C x := \gamma\ (\mathsf{filter}(\lambda t.\, t \neq_{\mathbb{B}} (x', fx))\ C)\ x' \text{ if } (x', fx) \in C$$

For a correspondence sequence C between p and q and $x \notin_1 C$ we have (1) $px \leftrightarrow p(\gamma C x)$, (2) $\gamma C x = x$ or $\gamma C x \in_1 C$, and (3) $f(\gamma C x) \notin_2 C$. The proof is by induction on the length of C, exploiting the injectivity of f. Now $\mathsf{find}\ C\ x_0 := f(\gamma C x_0)$ is the wanted function. □

The intuition is that $\gamma C x_0$ returns an x such that $fx \notin_2 C$. It starts by x_0. If $fx_0 \notin_2 C$, the search has already succeeded. If however $(x, fx_0) \in C$, then γ proceeds recursively with x and C with the pair (x, fx_0) removed. The result will still have the right properties, because if (x, fx_0) is in C, then $px_0 \leftrightarrow px$ by the properties of f and since C is a correspondence sequence.

The algorithm terminates, because C is shorter in every recursive call – and once C does not contain any x with $px \leftrightarrow px_0$ the check $fx_0 \notin_2 C$ will be true. Implicitly, the termination argument hinges on the fact that there has to be some x in the set $A := \{x_0\} \cup \{x' \in_1 C \mid px' \leftrightarrow px_0\}$ such that fx is not in the second projection of C, because $|A| > |\{y' \in_2 C \mid (x', y') \in C \wedge px' \leftrightarrow px_0\}|$. This fact can however not be derived constructively. Classically, one would use a pigeonhole principle. However, pigeonhole principles are only constructive for finite sets and set A can not be proved to be finite constructively. In the next section, we will use a computational pigeonhole principle explicitly because we can work with finite sets there.

For the rest of this section we fix enumerable discrete types X and Y such that (I_X, R_X) and (I_Y, R_Y) are retractions from X and Y respectively to $\mathbb{N}$. We construct the isomorphism via a cumulative correspondence sequence C_n with $I_X x < n \to x \in_1 C_n$ and $I_Y y < n \to y \in_2 C_n$.

$$C'_n := \begin{cases} (x, \mathsf{find}\ C_n\ x) :: C_n & \text{if } R_X n = \mathsf{Some}\ x \wedge x \notin_1 C_n \\ C_n & \text{otherwise} \end{cases}$$

$$C_{n+1} := \begin{cases} (\mathsf{find}\ \overleftrightarrow{C'_n}\ y, y) :: C'_n & \text{if } R_Y n = \mathsf{Some}\ y \wedge y \notin_2 C'_n \\ C'_n & \text{otherwise} \end{cases}$$

where $C_0 := []$ and $\overleftrightarrow{C} := \mathsf{map}\ (\lambda(x, y).(y, x))\ C$.

Lemma 3.2. *C_n is a correspondence sequence for p and q with*

1. $n \leq m \to C_n \subseteq C_m$
2. $I_X x < n \to x \in_1 C_n$
3. $I_Y y < n \to y \in_2 C_n$

Theorem 3.3 (Myhill)**.** *Let X and Y be enumerable discrete types, $p : X \to \mathbb{P}$, and $q : Y \to \mathbb{P}$. If $p \preceq_1 q$ and $q \preceq_1 p$, then there exist $f : X \to Y$ and $g : Y \to X$ such that for all $x : X$ and $y : Y$:*

$$px \leftrightarrow q(fx), \quad qy \leftrightarrow p(gy), \quad g(fx) = x, \quad f(gy) = y$$

Proof. fx is defined as the unique y for which $(x, y) \in C_{I_X x+1}$ (which exists by Lemma 3.2 (2) and is unique because $C_{I_X x+1}$ is a correspondence sequence). Note that since $C_{I_X x+1}$ is a (finite) list over discrete type we do *not* need to use any kind of choice principle here.

Vice versa, gy is symmetrically defined as the unique x for which $(x, y) \in C_{I_Y y+1}$. (1) and (2) are immediate since C_n is a correspondence sequence. (3) and (4) are by case analysis whether $I_X x \leq I_Y y$ or vice versa. □

Proofs of the theorem have appeared in different forms in the Bachelor's thesis of the second author [9], the PhD thesis of the first [6], and in an unpublished pre-print [8], which contains both the Myhill isomorphism theorem and a discussion of general constructive synthetic reducibility theory as published in [7].

4 Cantor-Bernstein from Myhill

The Cantor-Bernstein theorem for enumerable discrete types can be directly deduced from the Myhill isomorphism theorem.

Theorem 4.1 (Computational Cantor-Bernstein). *For enumerable discrete types X, Y with injections $X \to Y$ and $Y \to X$ there are functions $X \to Y$ and $Y \to X$ inverting each other.*

Proof. By applying the Myhill isomorphism theorem with $px := \top$ and $qy := \top$. □

For the running example, the constructed bijection depends heavily on f and g. Recall the definition of f and g:

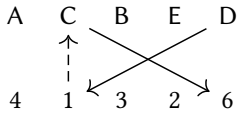
A	B	C	D	E	F	G	H
↓	↓	↓	↓	↓	↓	↓	↓
4	3	6	1	5	8	2	7

1	2	3	4	5	6	7	8
↓	↓	↓	↓	↓	↓	↓	↓
C	E	A	B	D	H	F	G

Applying CB as resulting from the Myhill isomorphism theorem yields the following bijection. We depict the bijection in stages from left to right. If in a stage no elements are added because they are already present we write x.

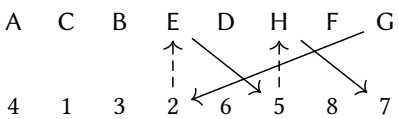
AC	BE	xx	Dx	xH	Fx	Gx	xx
↕	↕		↕	↕	↕	↕	
41	32	xx	6x	x5	8x	7x	xx

Building the first and second stage works regularly by just using f and g. At the third stage, both C and 3 are already part of the finite bijection and thus do not have to be added. At the fourth stage, adding (D, fD) is not possible because $fD = 1$ and $(C, 1)$ is already in the sequence. Thus, $(D, 6)$ is added because $fC = 6$. We depict this search process as follows for later comparison:



Vice versa, 4 is again already present and thus does not have to be added.

At the fifth stage, E is already part of the sequence. Adding $(g5, 5)$ is not possible, because $g5 = D$ and $(D, 6)$ is already there. Thus, $(H, 5)$ is added, because $g6 = H$. At the sixth stage, F is regularly added, and 6 is already present. At the seventh stage, (G, fG) cannot be added because $fG = 2$ and $(E, 2)$ is already present. Then, (G, fE) can also not be added because $fE = 5$ and $(H, 5)$ is already present. Thus, $(G, 7)$ is added because $fH = 7$. We depict this search as follows for later comparison:



5 Cantor-Bernstein Using a Computational Pigeonhole Principle

The implementation of the find function is more involved than necessary to just derive the Cantor-Bernstein theorem, because it additionally maintains a property regarding the predicates p and q. In fact it suffices to be able to compute given a duplicate-free list l_1 and a list l_2 with $|l_1| > |l_2|$ an element of l_1 which is not in l_2. This is a computational form of a pigeonhole principle which was before not directly applicable because the list l_1 is not computable if considering p and q.

Intuitively, for the example of f and g, the function F will again be built in levels using lists of pairs. The notion of correspondence sequence becomes irrelevant if no predicates are involved, and we instead maintain the invariant that both projections of the list are duplicate-free. At level n, we work with a list of pairs $[(x_1, y_1), \dots, (x_k, y_k)]$. We need to check whether x_n (the n-the element of x according to the enumeration) is already part of the first projection. If this is the case, we proceed symmetrically with the second projection. If this is not the case, then we can obtain the duplicate-free list $[fx_n, fx_1, \dots, fx_k]$ of $k+1$ many elements of Y, i.e. one more element than $[y_1, \dots, y_n]$ allowing us to compute an element y of $[fx_n, fx_1, \dots, fx_n]$ which we can add as pair (x_n, y), and proceed symmetrically with the second projection.

We formally just describe the lemma corresponding to Lemma 3.1, the subsequent construction is then exactly the same and cannot be simplified. Crucially underlying is a pigeonhole principle, which is formulated using duplicate-free lists. We work with an inductive definition of a duplicate-freeness predicate $\#l$ as

$$\frac{}{\#[]} \qquad \frac{x \notin l \qquad \#l}{\#(x :: l)}$$

Lemma 5.1. *Given a discrete type X there is a function* php $: \mathbb{L}X \to \mathbb{L}X \to \mathbb{O}X$ *such that if l_1 is duplicate-free and $|l|_1 > |l|_2$, then* php $l_1\ l_2 =$ Some x *and $x \in l_1$ but $x \notin l_2$.*

Proof. Let d decide equality on X. The proof relies on a function remove $: X \to \mathbb{L}X \to \mathbb{L}X$ with the expected properties. We can then define

$$\begin{aligned} \text{php } []\ l_2 &:= \text{None} \\ \text{php } (x :: l_1)\ l_2 &:= \text{Some } x && (\text{if } x \notin l_2) \\ \text{php } (x :: l_1)\ l_2 &:= \text{php } l1\ (\text{remove } x\ l_2) && (\text{if } x \in l_2) \end{aligned}$$

The proof is then by induction on the derivation of $\#l_1$, with l_2 generalised.

The case $l_1 = []$ is contradictory, since $|[]| = 0 > |l_2|$ is impossible.

Let $x \notin l_1$ and $\#l_1$. Case analysis whether $x \in l_2$, possible using d. If $x \notin l_2$, the claim is immediate. If $x \in l_2$, the claim follows from the induction hypothesis for l_2 with x removed. □

Lemma 5.2. *Let f be an injection from X to Y There is a function* $\mathsf{find}\colon \mathbb{L}(X \times Y) \to X \to Y$ *such that if* $C : \mathbb{L}(X \times Y)$ *has duplicate-free projections and* $x_0 \notin_1 C$, *then* $\mathsf{find}\ C\ x_0 \notin_2 C$.

Proof. Define

$$\begin{aligned}\mathsf{find}\ C\ x := \textbf{if}\ \mathsf{php}\ &(fx :: \mathsf{map}(\lambda(x', y).\ fx')C)\\ &(\mathsf{map}\ (\lambda(x, y).\ y)\ C)\\ \textbf{is}\ \mathsf{Some}\ y\ \textbf{then}\ y\ \textbf{else}\ &fx.\end{aligned}$$

By the properties of php we know that the **else** case is irrelevant given the pre-conditions of the lemma. The rest immediately follows. □

Theorem 5.3 (Computational Cantor-Bernstein)**.** *For enumerable discrete types X, Y with injections $X \to Y$ and $Y \to X$ there are functions $X \to Y$ and $Y \to X$ inverting each other.*

Recall the definition of f and g for the running example.

A	B	C	D	E	F	G	H
↓	↓	↓	↓	↓	↓	↓	↓
4	3	6	1	5	8	2	7

1	2	3	4	5	6	7	8
↓	↓	↓	↓	↓	↓	↓	↓
C	E	A	B	D	H	F	G

We obtain the following bijection:

AC	BE	xx	Dx	xx	FH	Gx	xx
↕↕	↕↕		↕	↕	↕↕	↕	
41	32	xx	5x	xx	86	7x	xx

In the first and second stage, everything is regular and the bijection is built following f and g. For the third stage, C and 3 are already present. For the fourth, one cannot add (D, fD) because $fD = 1$ and 1 is already present. Thus, we add $(D, 5)$ because $fE = 4$ and E is the first letter when searching (backwards in the order of depiction) which does not yet have an assignment. For the fifth stage, nothing has to be added. Adding $(F, 8)$ and $(H, 6)$ is regular. For the sixth stage, (G, fG) cannot be added because $fG = 2$ and 2 is already present. Thus, $(G, 7)$ is added because $fH = 7$ and H is the first letter when searching which is unassigned.

Note how, when comparing with the bijection from the previous section D gets a different assignment due to the differing search, and G gets the same but with a different search procedure.

Different implementations of the function in the pigeonhole principle are possible and lead to different bijections. The bijection as defined in this section is the same as the one obtained by Kirst [10] in his work on generalising the contributions of the present paper to obtain a back-and-forth proof method also applying to Cantor's isomorphism theorem regarding countable dense unbounded linear orders.

6 Cantor-Bernstein Using List Enumerators

We now give a proof of the Cantor-Bernstein theorem using list enumerators, where the constructed bijection only depends on the given enumeration functions and not on the injections.

For the running example we end up with the most natural bijection

A	B	C	D	E	F	G	H
↕	↕	↕	↕	↕	↕	↕	↕
1	2	3	4	5	6	7	8

Abstractly, F maps the n-th element of X to the n-th element of Y.

In the computation, given an element x, F will first compute the duplicate-free list $[x_0, \dots, x_n]$ of all elements x_i occurring until $x = x_n$ in the enumeration of X. By applying f pointwise, we obtain the list $[fx_0, \dots, fx_n]$ with $n+1$ many elements. Consequently, we can use the enumerator of Y to also enumerate a list of that many elements, which will be $[y_0, \dots, y_n]$. F then maps x to y_n.

Formally, we first prove that every enumerator gives rise to a function $L : \mathbb{N} \to \mathbb{L}X$ which returns duplicate-free, cumulative lists, where it can be computed in which list any element x occurs. This construction is entirely standard and frequently used to construct enumerators in Coq.

Lemma 6.1. *Let X be enumerable and discrete. Then X is list-enumerable with a duplicate-free list enumerator: There are* $L\colon \mathbb{N} \to \mathbb{L}X$ *and* $\mathsf{occ}\colon X \to \mathbb{N}$ *such that*

1. *for all n, Ln is duplicate-free,*
2. *given $n \le m$, Ln is a (not necessarily strict) prefix of Lm, and*
3. *for all x, $x \in L(\mathsf{occ}\ x)$.*

Proof. Let e enumerate X. Define occ as the function resulting from using Lemma 2.4 on $\forall x.\exists n.\ ex = \mathsf{Some}\ n$, $L0 := []$ and

$$\begin{aligned} L(\mathsf{S}\ n) &:= Ln +\!\!+ [x] && \text{if } en = \mathsf{Some}\ x \text{ and } x \notin Ln\\ L(\mathsf{S}\ n) &:= Ln && \text{otherwise} \end{aligned}$$

□

All further proofs will not need to rely on any choice principles, not even 2.4. We then prove that the index of any element x can also be computed, i.e. its position in $L(\mathsf{occ}\ x)$.

Lemma 6.2. *Let X be list-enumerable with a duplicate-free list enumerator L. Then there are functions* $\mathsf{index}\colon X \to \mathbb{N}$ *and* $\mathsf{gen}\colon \mathbb{L}X \to \mathbb{N}$ *such that*

1. *if $x \in Ln$ or $|Ln| > \mathsf{index}\ x$, then $Ln[\mathsf{index}\ x] = \mathsf{Some}\ x$,*
2. *if $Ln[m] = \mathsf{Some}\ x$, then $m = \mathsf{index}\ x$, and*
3. *given a duplicate-free $l\colon \mathbb{L}X$ we have $|L(\mathsf{gen}\ l)| \ge |l|$.*

Proof. Define $\mathsf{gen}\ l$ as the maximum of all $\mathsf{occ}\ x$ for $x \in l$ and $\mathsf{index}\ x$ as the position of x in $L(\mathsf{occ}\ x)$. □

We now put together these ingredients to construct the bijection according to the intuition.

Theorem 6.3. *Given two retracts X and Y of $\mathbb{N}$ with injections $X \to Y$ and $Y \to X$ there are functions $X \to Y$ and $Y \to X$ inverting each other.*

Proof. Let L_X, occ_X, L_Y, and occ_Y be obtained by applying Lemma 6.1 and index_X, gen_X, index_Y, and gen_Y by applying Lemma 6.2 for both X and Y.

Let f be an injection $X \to Y$. Define $F : X \to Y$ as

$$Fx := \mathsf{if}\ (L_Y(\mathrm{gen}_Y(\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))))) [\mathrm{index}_X x] \\ \mathsf{is\ Some}\ y\ \mathsf{then}\ y\ \mathsf{else}\ fx$$

We prove that only the then case matters, i.e. that

$$L_Y(\mathrm{gen}_Y(\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))))[\mathrm{index}_X x] \neq \mathsf{None}.$$

To do so, it suffices to prove that

$$|L_Y(\mathrm{gen}_Y(\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))))| > \mathrm{index}_X x.$$

We have that $L_X(\mathrm{occ}_X x)[\mathrm{index}_X x] = \mathsf{Some}\, x$, meaning $|L_X(\mathrm{occ}_X x)| > \mathrm{index}_X x$, and that $\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))$ is duplicate-free because f is injective and $L_X(\mathrm{occ}_X x)$ is duplicate-free. Now

$$\begin{aligned} &|L_Y(\mathrm{gen}_Y(\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))))| \\ \geq &|\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))| && \textit{by Lemma 6.2 since the list is duplicate-free} \\ \geq &|L_X(\mathrm{occ}_X x)| && \textit{because } f \textit{ is injective} \\ > &\mathrm{index}_X x && \textit{by Lemma 6.1 (3) we have } x \in L(\mathrm{occ} x)\textit{, thus Lemma 6.2 (1) applies} \end{aligned}$$

as needed.

Let $G : Y \to X$ be defined symmetrically. We have that $G(Fx) = x$: Let $(L_Y(\mathrm{gen}_Y(\mathsf{map}\ f\ (L_X(\mathrm{occ}_X x))))) [\mathrm{index}_X x] = \mathsf{Some}\, y$. By Lemma 6.2 (2), $\mathrm{index}_X x = \mathrm{index}_Y y$.

We then have

$$\begin{aligned} &(L_X(\mathrm{gen}_X(\mathsf{map}\ g\ (L_Y(\mathrm{occ}_Y y))))) [\mathrm{index}_Y y] \\ =&(L_X(\mathrm{gen}_X(\mathsf{map}\ g\ (L_Y(\mathrm{occ}_Y y))))) [\mathrm{index}_X x] \\ =&\mathsf{Some}\, x \end{aligned}$$

because $|L_X(\mathrm{gen}_X(\mathsf{map}\ g\ (L_Y(\mathrm{occ}_Y y))))| \geq |L_Y(\mathrm{occ}_Y y)| > \mathrm{index}_Y y = \mathrm{index}_X x$. □

7 Conclusion

We have presented a proof of the Myhill isomorphism theorem and three proofs of the Cantor-Bernstein theorem for enumerable discrete types in constructive type theory.

The proofs are formalised in Coq, but can be translated to other constructive type theories, in particular to MLTT (as implemented e.g. in Agda) and HoTT (as implemented e.g. in Cubical Agda). Morally more classically inclined type-theoretic proof assistants such as Lean cannot really benefit from our analysis, since the result is trivial to prove classically.

The central result is a novel result with a compact proof shedding interesting light on the computational content of the Cantor-Bernstein theorem.

The Coq proofs themselves are unremarkable: They do not require advanced features. The proof of the Myhill isomorphism theorem becomes a little more elegant by using the Equations package [18] (but then still does not rely on Axiom K), but the use could be easily circumvented as well. The proof of CB from the Myhill isomorphism theorem and the proof via the pigeonhole principle both require around 280 lines of code, whereas the proof via list enumerators requires 220 lines.

We mention here the comment by the reviewers of this paper that our proof based on list enumerators in section 6 is reminiscent of Cantor's original proof of Cantor's theorem concerning dense linear orders [3], whereas newer proofs of Cantor's theorem work with back-and-forth methods as used in the Myhill isomorphism theorem and our proof based on a computable pigeonhole principle in section 5, see the historical discussion by Silver [17] and the uniform presentation by Kirst [10].

It is an interesting open problem how and whether the conditions of enumerability and discreteness can be relaxed for type-theoretic versions of CB, or whether incomparable properties of types exist that allow proving CB. A possible relaxation of discreteness could be weak discreteness (i.e. the predicate $x_1 \neq x_2$ being computationally decidable), whereas a possible relaxation for enumerability could be some form of constructive well-ordering, as used in the definition of ordinals in the HoTT book [19, §10.3]. It is possible that such a generalised version of CB for enumerable discrete types would only be provable for total functional relations rather than functions. However, since any total functional relation on enumerable discrete types can, as discussed, be turned into a function again (Lemma 2.4), such a theorem would in particular imply our result for enumerable discrete types. Furthermore, it is possible that extensionality principles (such as functional extensionality, propositional extensionality, or even univalence) would be needed, which do not play a role in our proof. We leave it to future work to analyse whether such a generalised presentation is possible.

Acknowledgments

We want to thank Dominik Kirst and Dominique Larchey-Wendling for many discussions around CB and Δ^0_1-choice, Lennard Gäher for feedback on an early draft of parts of this paper, Martín Escardó for raising the question on twitter, ultimately motivating us to go back to our first proof of CB and obtaining the others, and the anonymous reviewers for their very helpful suggestions and corrections.

Yannick Forster received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101024493.

References

[1] Felix Bernstein. Untersuchungen aus der Mengenlehre. *Mathematische Annalen*, 61(1):117–155, March 1905. doi:10.1007/bf01457734.

[2] Georg Cantor. Mitteilungen zur Lehre vom Transfiniten. In *Zeitschrift für Philosophie und philosophische Kritik 91*, 1887.

[3] Georg Cantor. Beiträge zur begründung der transfiniten mengenlehre. *Mathematische Annalen*, 46(4):481–512, 1895. doi:10.1007/BF02124929.

[4] Martín Hötzel Escardó. The Cantor–Schröder–Bernstein Theorem for ∞-groupoids. *J. Homotopy Relat. Struct.*, 16(3):363–366, jun 2021. URL: https://doi.org/10.1007%2Fs40062-021-00284-6, doi:10.1007/s40062-021-00284-6.

[5] Martín Hötzel Escardó (@EscardoMartin). "Challenge: how much beyond finite sets with decidable equality can CSB apply to in constructive mathematics?" 12:47 PM - 3 Feb 2022. Tweet. https://web.archive.org/web/20220203205238/https://twitter.com/EscardoMartin/status/1489339787005906958, 2022. Accessed: (September 14th).

[6] Yannick Forster. *Computability in Constructive Type Theory*. PhD thesis, Saarland University, 2021. doi:10.22028/D291-35758.

[7] Yannick Forster and Felix Jahn. Constructive and Synthetic Reducibility Degrees: Post's Problem for Many-one and Truth-table Reducibility in Coq. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, Dagstuhl, Germany. doi:10.4230/LIPIcs.CSL.2023.16.

[8] Yannick Forster, Felix Jahn, and Gert Smolka. A Constructive and Synthetic Theory of Reducibility: Myhill's Isomorphism Theorem and Post's Problem for Many-one and Truth-table Reducibility in Coq (Full Version). preprint, February 2022. URL: https://hal.inria.fr/hal-03580081.

[9] Felix Jahn. *Synthetic One-One, Many-One, and Truth-Table Reducibility in Coq*. Bachelor's thesis, Saarland University, 2020. URL: https://ps.uni-saarland.de/~jahn/bachelor.php.

[10] Dominik Kirst. Computational Back-And-Forth Arguments in Constructive Type Theory. In June Andronick and Leonardo de Moura, editors, *13th International Conference on Interactive Theorem Proving (ITP 2022)*, volume 237 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:12, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16731, doi:10.4230/LIPIcs.ITP.2022.22.

[11] Alwin R. Korselt. Über einen beweis des Äquivalenzsatzes. *Math. Ann.*, 70(2):294–296, jun 1911. URL: https://doi.org/10.1007%2Fbf01461161, doi:10.1007/bf01461161.

[12] Jules König (Gyula Kőnig). Sur la théorie des ensembles. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 143:110–112, 1906. URL: https://gallica.bnf.fr/ark:/12148/bpt6k30977.image.f110.langEN.

[13] John Myhill. Creative sets. 1957. doi:10.1002/malq.19550010205.

[14] Pierre Pradic and Chad E. Brown. Cantor-Bernstein implies excluded middle. *CoRR*, abs/1904.09193, 2019. arXiv:1904.09193.

[15] Hartley Rogers. Theory of recursive functions and effective computability. 1987.

[16] Ernst Schröder. Über G. Cantorsche Satze. *Jahresberichtder Deutsche Mathematiker-Vereinigung*, 5:81–82, 1896.

[17] Charles L Silver. Who invented cantor's back-and-forth argument? *Modern Logic*, 4(1):74–78, 1994.

[18] Matthieu Sozeau and Cyprien Mangin. Equations reloaded: High-level dependently-typed functional programming and proving in coq. *Proceedings of the ACM on Programming Languages*, 3(ICFP):1–29, 2019. doi:10.1145/3341690.

[19] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[20] Wikipedia contributors. Schröder–Bernstein theorem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Schr%C3%B6der%E2%80%93Bernstein_theorem&oldid=1095347368, 2022. [Online; accessed 14-September-2022].

Received 2022-09-21; accepted 2022-11-21