SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Deciding isomorphisms in Cartesian closed categories

av

Hjalmar Wijk

2019 - No K28

Deciding isomorphisms in Cartesian closed categories

Hjalmar Wijk

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Erik Palmgren

2019

Acknowledgements

I would like to thank my advisor Erik Palmgren for showing me the world of category theory and guiding me to many wonderful insights.

Abstract

Category theory can be used to generalize the notion of isomorphism which exists in many areas of mathematics and study the 'isomorphism problem', of determining whether two objects are isomorphic, in a very general setting. In this thesis, we will look at a particular class of categories called the Cartesian closed categories, where we will find a remarkably simple but complete decision procedure for determining which isomorphisms must exist. We will also study the subclass of such categories which contain objects similar to the set of natural numbers, where we can demonstrate isomorphisms analogous to the pairing functions $\mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$ in much greater generality.

Contents

1 Introduction

A common problem in many areas of mathematics is to determine whether two objects A and B (which might be sets, groups, manifolds, varieties etc.) are isomorphic - meaning that there are morphisms (suitably defined) $f : A \to B$ and $g : B \to A$ such that $g \circ f$ is the identity morphism on A, and $f \circ g$ is the identity morphism on B. This will be written $A \cong B$. Some examples of common decision problems of this type are:

- Determining whether two sets are isomorphic.
- The group isomorphism problem, of determining whether two finitely presented groups are isomorphic.
- The homeomorphism problem, of determining whether two manifolds are homeomorphic.
- The isomorphism problem for varieties, of determining whether two varieties over the algebraic numbers are isomorphic.

Category theory, which studies categories made up precisely of objects and 'morphism-like' arrows between them, can provide general treatments for problems of this kind. This theory will be presented properly later, but for now an informal understanding of concrete categories will help.

Definition 1.0.1. A concrete category consists of *objects*, which are sets with some kind of structure (such as a group structure), and *arrows*, which are functions between objects that preserve that structure. Among the arrows there is an identity arrow for each object A, referred to as 1_A. We can also compose arrows just like we can compose functions.

All the examples above can be viewed as concrete categories in this way, for example we have the categories **Set** of sets with no structure and the functions between them, and its restriction to only finite sets called **Finset**. These are particularly nice categories since determining whether two objects are isomorphic only comes down to determining whether they have the same cardinality or size.

We might hope to use category theory to answer general questions about isomorphisms, such as whether all categories where we can define something like a Cartesian product (do not worry about the details for now) will exhibit the isomorphism $(A \times B) \times C \cong A \times (B \times C)$ for arbitrary objects A,B,C. More ambitiously we could consider looking for sound and complete theories or effective decision methods specifying exactly which isomorphisms must hold for all categories which have certain properties. In general this will not be possible; for a lot of categories determining what isomorphisms must hold is an undecidable problem, as will be seen in example 2.1.6.

It turns out, however, that if we work close to the well-behaved **Set** and **Finset** we can do better. As an illustrative example let us return to the case of categories with some kind of products similar to the Cartesian ones in **Set** and see if designing a decision procedure for isomorphisms might be possible. This would tell us whether, for instance

$$(A \times B) \times (C \times A) \cong (A \times A) \times (B \times C), \tag{1}$$

holds in all categories with products. It turns out that we can fairly easily prove associativity and commutativity for all categories with products, and armed with those it takes only a few steps to prove that the above isomorphism is valid. This provides a sound axiomatic theory of isomorphisms for categories with products $Th_{\times}$:

(i) $A \times B \cong B \times A$

(ii) $(A \times B) \times C \cong A \times (B \times C)$

We can even propose an effective decision method by taking any two objects and putting them in a 'normal form' where the basic objects are put in alphabetic order and the product is done starting from the left, then determining that they are equal if they have the same normal form. For the example objects in (1) the normal form for both would be

$$((A \times A) \times B) \times C,$$

and then we would immediately see that they are isomorphic. While this looks promising, the theory might not be complete, there might still be other isomorphisms which will always hold, but cannot be derived from just associativity and commutativity. In terms of our decision procedure this would mean that there are normal forms which seem different but are actually isomorphic. How could we prove that that is not the case?

One way, which will be used several times in this thesis, is to simply find a category with products where no other isomorphisms hold, thus proving that no other isomorphisms can hold for all such categories. This method certainly is not guaranteed to work, since there might be no such category, but in this case there is - **Finset**. This will be left without proof, but it turns out that two different normal forms interpreted in **Finset** will never be isomorphic since there is always some assignment of sets to the arbitrary variables $A, B, C, ...$ which gives the two normal forms different cardinalities.

Much of this thesis will be devoted to replicating this basic methodology in much more detail for increasingly specific classes of categories, where more and more interesting isomorphisms can be shown to always exist.

If we have arbitrary sets A, B in **Set** there is another common way of forming new sets which generalizes very well to categorical structures, namely forming the set of functions

$A \to B$, often written B^A. In category theory these objects are often called exponential objects. Without going into details, any category which, for all objects A, B, has both products $A \times B$ and exponential objects B^A which are sufficiently well-behaved is called a Cartesian closed category (CCC). Obvious examples are once again **Set** and **Finset**. CCC:s are also of special interest because they form models for typed λ-calculi, a commonly used formal system for describing computation. This means that finding decision methods for isomorphisms in CCC:s immediately gives decision methods for isomorphisms between types in typed λ-calculi, which has important applications in computer science.

Using a very similar method to the one illustrated for categories with just products this thesis will in parts 2 and 3 present a proof that there is a sound and complete axiomatization of isomorphisms which hold in all CCC:s, and that this can be used to build effective decision methods. In fact we will, similar to the product case, show that all CCC:s contain at least those isomorphisms found in **Finset**, which is to say that **Finset** only contains the minimum number of isomorphisms needed to be a CCC. This proof is based largely on Solov'ev's from 1983.

This thesis will then try to extend some of these results by noticing that in **Set** there are many isomorphisms such as $A \times A \cong A$ which hold for infinite sets but which the theory of CCC:s is too general to capture since it includes e.g. **Finset** where no such objects exist. We could thus wonder if these isomorphisms in **Set** might hold in all CCC:s which have objects somehow corresponding to infinite sets.

There turns out to be a standard extension of CCC which does something like this, namely CCC:s with natural number objects, objects which act like $\mathbb{N}$ does in **Set**. In the part 4 we will show that adding this requirement does indeed lead to new valid isomorphisms that look much like those in **Set**. This also has interesting applications since typed λ-calculi are very often extended with a corresponding natural number type.

2 Background on CCC:s and λ-calculi

In this section we will go through the bare-minimum background in category theory and λ-calculus necessary for understanding the results, establish the basic notations and definitions used in the thesis and finally look at the link between λ-calculi and Cartesian closed categories. For a more detailed exposition on category theory see [1], and for a thorough look at the link to λ-calculus see [5]. This presentation is essentially a much shortened mix of the two.

2.1 Categories and isomorphisms

In the introduction we saw concrete categories such as **Set**, but the notion of category does not actually need the objects to be sets or the arrows to be actual functions.

Definition 2.1.1. A *category* consists of two classes: the class of *arrows* and the class of *objects*. It also has two functions from the class of arrows to the class of objects, called *source* and *target*. If f is an arrow we write $f : A \to B$ to mean 'source$(f) = A$' and 'target$(f) = B$'.

In addition every object A must have a specified *identity arrow* $1_A : A \to A$, and each pair of arrows $f : A \to B, g : B \to C$ must have a *composite arrow* $gf : A \to C$. Finally, for any $f : A \to B, g : B \to C, h : C \to D$ we must have:

$$f1_A = f = 1_B f, \quad (hg)f = h(gf)$$

All concrete categories are categories, but this definition allows for many other kinds.

Example 2.1.2. While we have already seen the concrete category of sets, an individual set can also be seen as a category itself, which has the elements of the set as objects and no arrows except the required identity arrows.

Example 2.1.3. Any preordered set can be seen as a category, where the objects are the elements of the set and there is exactly one arrow $A \to B$ if and only if $A \leq B$. Composition of arrows corresponds to transitivity of the order, and the identity arrows corresponds to reflexivity.

From this definition we can formally define a very general notion of isomorphism.

Definition 2.1.4. An arrow $f : A \to B$ is called an isomorphism if there exists an *inverse* arrow $g : B \to A$ such that

$$gf = 1_A, \quad fg = 1_B.$$

Two objects A, B are called isomorphic, written $A \cong B$, if there exists an isomorphism between them

If $g : B \to A$ and $g' : B \to A$ are both inverses to $f : A \to B$ then

$$gf = 1_A \Rightarrow gfg' = g' \Rightarrow g1_B = g' \Rightarrow g = g',$$

so inverses are unique and we are justified in writing $g = f^{-1}$.

Before moving on we will prove a general property of isomorphisms.

Proposition 2.1.5. *The isomorphism relation $\cong$ is an equivalence relation, meaning if A, B, C are objects in any category then*

(i) $A \cong A$

(ii) *If* $A \cong B$, *also* $B \cong A$.

(iii) *If* $A \cong B$ *and* $B \cong C$, *then* $A \cong C$.

Proof. (i) For any object A the identity arrow 1_A provides the isomorphism.

(ii) If $A \cong B$ then there is some isomorphism $f : A \to B$. The inverse $f^{-1} : B \to A$ is then also an isomorphism (with f as inverse) and shows $B \cong A$.

(iii) If $A \cong B$ and $B \cong C$ there are isomorphisms $f : A \to B, g : B \to C$. Then $gf : A \to C$ is an isomorphism with inverse $f^{-1}g^{-1}$, showing $A \cong C$.

□

We can now look at an example of where the 'isomorphism problem', of determining whether two objects in a category are isomorphic, is undecidable. This will build on the standard notion of a Turing machine and the halting problem - if those are unfamiliar this example can be safely skipped.

Example 2.1.6. Let $\mathbf{T}$ be a category where

- the objects are 1-tape Turing machines along with their current state, head position and a complete description of the tape.
- the relation $\rightarrowtail$ holds between two objects $t_1 \rightarrowtail t_2$ exactly if simulating t_1 for one step gives you t_2. The relation $=_H$ holds between any two objects which are in the halting state. Note that objects which are not in a halting state can never be related to any object by $=_H$ and objects that are in a halting state can never be related to any object by $\rightarrowtail$. Finally we write $\overset{*}{=}$ for the reflexive, symmetric and transitive closure of the union of these relations.
- there is a unique arrow $t_1 \to t_2$ in $\mathbf{T}$ if and only if $t_1 \overset{*}{=} t_2$. Since $\overset{*}{=}$ is reflexive and transitive this fulfills the definition of a category.

Since $\overset{*}{=}$ is symmetric there will be arrows $f : t_1 \to t_2$, $g : t_2 \to t_1$ precisely when $t_1 \overset{*}{=} t_2$, and furthermore $gf = 1_{t_1}, fg = 1_{t_2}$ by the uniqueness of arrows, meaning f and g form the two directions of an isomorphism. Thus a decision method for the isomorphism problem in $\mathbf{T}$ will also be a decision method for determining if $t_1 \overset{*}{=} t_2$ for any objects t_1, t_2. But such a decision method could then be used to solve the halting problem:

(i) Given any turing machine in its initial configuration t, pick any turing machine in a halting state h and consider both as objects of $\mathbf{T}$.

(ii) Decide whether $t \overset{*}{=} h$.

(iii) If it is true, then it is easy to see that there has to be a sequence of transitions $t \rightarrowtail t_1 \rightarrowtail ... \rightarrowtail t_n =_H ... =_H h$, and that in particular t eventually reaches a halting state t_n.

(iv) If it is false, then no sequence of transitions $\rightarrowtail t_1 \rightarrowtail t_2 \rightarrowtail ...$ can ever reach a halting state, since then that state would be related to h by $\overset{*}{=}$ so also t would. Thus t never halts.

Since the halting problem is undecidable, this shows that there cannot be a decision method for determining isomorphisms in **T**.

Luckily we will be studying more well-behaved categories.

2.2 Products and terminal objects

The sets with just one element are important in **Set** since functions from a one-element set can be identified with the elements of the target set. This is useful since it provides a way to talk about 'elements' in categorical language. A way to characterize a one-element set is that there is a unique function from any other set into it. This notion can be generalized and turns out to be quite useful.

Definition 2.2.1. In any category **C** an object 1 is *terminal* if for any object A there is a unique arrow $A \to 1$.

There might be many terminal objects in a category (as there are in **Set**), but we will often privilege one, arbitrarily, and call it 1. The unique arrows $A \to 1$ for any object A will then be written $\bigcirc_A$. This is unproblematic since it turns out they are unique up to isomorphism.

Proposition 2.2.2. *Any two terminal objects T and T' in a category **C** are isomorphic*

Proof. In fact, there is a unique isomorphism $T \to T'$. First notice that there are unique arrows $u : T \to T'$ and $v : T' \to T$ since they are terminal. Since both are terminal there must also be just one unique arrow $T \to T$ and $T' \to T'$, which means

$$1_T = vu, \quad 1_{T'} = uv$$

so u and v are isomorphisms, and $T \cong T'$ □

This is our first example of using general categorical constructions to prove that certain isomorphisms must always hold. Many more will follow throughout this thesis.

Continuing the theme of generalizing notions from **Set**, let us look closer at the Cartesian product. Given two sets A, B in the category **Set** the Cartesian product $A \times B$ consists of all pairs (a, b) with $a \in A, b \in B$. There are also natural projection functions

$$A \xleftarrow{p_1} A \times B \xrightarrow{p_2} B$$

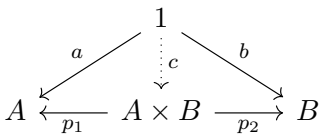
with

$$p_1(a, b) = a, \quad p_2(a, b) = b$$

The important property of the Cartesian product is that given elements $a \in A, b \in B$ there is a unique element $c \in A \times B$ such that

$$p_1(c) = a, \quad p_2(c) = b.$$

In other words, if we have a one-element set 1 and functions a, b like this:



then there is a unique function $c : 1 \to A \times B$ such that the diagram *commutes*, meaning any path yields the same function e.g. $p_1 c = a$.

This definition is almost ready to generalize directly to any category, however in categories that are not concrete, so that the objects are not sets, it might not be 'enough' to only consider the arrows from a terminal object 1, as it is in **Set**. Instead we require that arrows from any object should have this property.

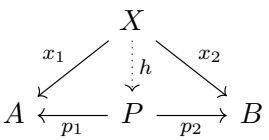
Definition 2.2.3. In a category **C** a *product diagram* for the objects A, B consists of an object P and arrows

$$A \xleftarrow{p_1} P \xrightarrow{p_2} B$$

such that for any X and arrows

$$A \xleftarrow{x_1} X \xrightarrow{x_2} B$$

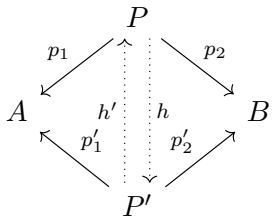
there is a **unique** arrow h making the following diagram commute



Once again an object could have many products, for instance in **Set** we could use different representations of the ordered pair and still have a valid product. Luckily this is not a problem, as is proved elegantly in [1].

Proposition 2.2.4. *Products are unique up to isomorphism.*

Proof. Suppose A and B are objects with two product diagrams:



Then we have unique arrows $h : P \to P', h' : P' \to P$ making the diagram above commute. We then have $p_1(h'h) = p_1$ and $p_2(h'h) = p_2$, something which is also accomplished by 1_P, and since such an arrow must be unique by the definition of a product, $h'h = 1_P$. Similarly we can show $1_{P'} = hh'$. So $P \cong P'$. □

Thus if A, B have a product diagram we can feel safe writing

$$A \xleftarrow{p_1} A \times B \xrightarrow{p_2} B$$

for one arbitrarily selected diagram. And given X, x_1, x_2 as in the definition we write $\langle x_1, x_2 \rangle$ for the unique $h : X \to A \times B$.

Definition 2.2.5. A category which has a product diagram for any two objects A, B is said to *have binary product*. If we have arrows $f : A \to B$, $f' : A' \to B'$ in a category with binary product, we write

$$f \times f' := \langle fp_1, f'p_2 \rangle : A \times A' \to B \times B'.$$

In **Set**, $f \times f'$ would be the function which takes an element (a, a') to $(f(a), f'(a'))$.

Definition 2.2.6. We can also define *n-ary products*

$$A_1 \times A_2 \times A_3 \times ... \times A_n$$

by a similar construction to the binary one. Given any object X and arrows $x_1 : X \to A_1, ..., x_n : X \to A_n$ there should be a unique $u : X \to A_1 \times A_2 \times ... \times A_n$ such that for all $i \leq n$, $p_i u = x_i$. Observe that in particular, a *null-ary product*, for $n = 0$, is exactly a terminal object, and the unary product of just A with itself is A. A category is said to *have all finite products* if it has n-ary products for all n.

As before, all n-ary products are clearly unique up to isomorphism. Now we will, as in [1], show that working with n-ary products is in some sense unnecessary.

Proposition 2.2.7. *If a category* $\mathbf{C}$ *has a terminal object and all binary products then it has all finite products.*

While specifying the exact projections is slightly bothersome, it should be fairly clear that setting

$$A_1 \times A_2 \times A_3 = (A_1 \times A_2) \times A_3$$

will fulfill the construction for 3-ary (ternary) products. Repeating this construction lets you write any n-ary product for $n \geq 3$ in terms of just the binary product.

Corollary 2.2.8. *The binary product operation is associative, meaning*

$$(A_1 \times A_2) \times A_3 \cong A_1 \times (A_2 \times A_3).$$

Proof. Since $A_1 \times (A_2 \times A_3)$ with suitable projections is also a ternary product of A_1, A_2, A_3 the result follows from ternary products being unique up to isomorphism. □

2.3 Hom-sets and exponentials

There is one more aspect of **Set** which we wish to generalize, namely the idea of 'sets of functions $A \to B$', often written B^A. First we can notice that forming such a set 'outside' any category is easy.

Definition 2.3.1. For objects A, B in a category $\mathbf{C}$, let

$$\mathrm{Hom}_{\mathbf{C}}(A, B) = \{f \in \mathbf{C} \mid f : A \to B\}.$$

Such a set of arrows is called a *Hom-set*.

However, we would like an object **within** the category to represent this notion, not just a set outside it. To define that we need to understand what makes function sets interesting in **Set**.

In **Set** we can do *evaluation*, meaning there is a function eval:$C^B \times B \to C$, defined by $\mathrm{eval}(f, b) = f(b)$. An important property of eval is that given any function $f(y) : B \to C$ there is a unique function $f^* \in C^B$ such that

$$\mathrm{eval}(f^*, y) = f^*(y) = f(y),$$

where f^* is called the *transpose* of f. If we wish to talk about this in the language of category theory however, we need to consider f^* as an arrow into C^B instead of an element of it. While it would be enough to consider it as an arrow $1 \to C^B$ in **Set** we will in general

need to consider arrows $A \to C^B$ from any other object A. This means we instead consider a function $f(x,y) : (A \times B) \to C$ to have a unique transpose $f^*(x) : A \to C^B$ so that

$$\text{eval}(f^*(x), y) = f^*(x)(y) = f(x,y),$$

which we get by $f^*(a) = f(a,y)$ - it essentially 'fixes' $x = a$ and then returns the remaining function in just y. Since f^* is unique this defines an isomorphism

$$\text{Hom}_{\mathbf{Set}}(A \times B, C) \cong \text{Hom}_{\mathbf{Set}}(A, C^B)$$

Where given a function $f : A \times B \to C$ you can construct $f^* : A \to C^B$, and given a function $h : A \to C^B$ you can construct $\text{eval}(h(a), b) : A \times B \to C$.

Definition 2.3.2. Let $\mathbf{C}$ be a category with finite products. An *exponential* of objects B and C consists of an object

$$C^B$$

and an arrow $\epsilon : C^B \times B \to C$, such that for any object A and arrow

$$f : A \times B \to C$$

there is a unique arrow

$$f^* : A \to C^B$$

such that

$$\epsilon(f^* \times 1_B) = f.$$

This definition ensures that we have an isomorphism of sets

$$Hom_{\mathbf{C}}(A \times B, C) \cong Hom_{\mathbf{C}}(A, C^B),$$

by taking $f : A \times B \to C$ to (the unique) $f^* : A \to C^B$, and taking $g : A \to C^B$ to $\epsilon(g \times 1_B) : A \times B \to C$.

Definition 2.3.3. We say that a category with finite products *has exponentials* if there is an exponential object B^A and evaluation arrow $\epsilon_{A,B} : B^A \times A$ for every pair of objects A, B in the category.

2.4 Cartesian closed categories

Definition 2.4.1. We call a category $\mathbf{C}$ *Cartesian closed* if it has all finite products and exponentials. Cartesian closed categories will sometimes be abbreviated CCC.

Examples are of course **Set** and **Finset**, but an additional example of a CCC which will be useful in the final section is ω−**cpo**. It is another concrete category, but one with much more structure than **Set**.

Example 2.4.2. ω-**cpo** is a Cartesian closed category with

- **objects** being ω-complete partially ordered sets with a least element $\bot$. These are sets with some order $\leq$ such that given any increasing sequence (possibly infinite)

  $$a_0 \leq a_1 \leq a_2 \leq ...$$

  there is a supremum a^* such that $a_i \leq a^*$ and $a^* \leq a'$ for all other upper bounds. As mentioned they must also have a least element $\bot$ with $\bot \leq a$ for all elements a of the underlying set.

- **arrows** being ω-continuous functions. These are functions between ω-complete partially ordered sets which preserve increasing sequences and their suprema. In particular they are monotone.

- **terminal object** $\{\bot\}$, which is a trivial $\omega - cpo$ and since it is a singleton there will be a unique arrow into it from any other object.

- **binary products** $A \times B$ defined by taking the Cartesian products of the underlying sets and ordering it by $(a, b) \leq (a', b') \Leftrightarrow a \leq_A a' \wedge b \leq_B b'$. Projection is the standard, $(a, b) \overset{p_1}{\mapsto} a, (a, b) \overset{p_2}{\mapsto} b$. Of course $(\bot_A, \bot_B)$ is the new least element.

  To see that $A \times B$ is ω-complete, take any increasing sequence

  $$(a_1, b_1) \leq (a_2, b_2) \leq ...$$

  We can then find the supremums a^* and b^* of the increasing sequences $a_1 \leq a_2 \leq ...$ and $b_1 \leq b_2 \leq ...$ and (a^*, b^*) will clearly be the supremum we are looking for. This also shows that projections are ω-continuous, since a supremum (a^*, b^*) will indeed get projected onto the corresponding supremum in A or B. Finally it is also clear that $(\bot_A, \bot_B)$ is a least element.

- **exponentials** B^A defined by taking the set of ω-continuous functions $A \to B$ and ordering them pointwise, so that

  $$f : A \to B \leq g : A \to B \Leftrightarrow \forall a \in A : \ f(a) \leq_B g(a).$$

  Evaluation $\epsilon_{A,B} : B^A \times A \to B$ is also standard, with $f, a \mapsto f(a)$. The least element is given by $f_\bot : A \to B, a \mapsto \bot_B$ which is trivially ω-continuous.

  The exponentials are also ω-complete - given an increasing sequence of functions

  $$f_1 \leq f_2 \leq f_3 \leq ...$$

  we can construct a function f^* by letting $f^*(a)$ be the supremum of the increasing sequence $f_1(a) \leq f_2(a) \leq ...$ in B. f^* will then be a supremum in B^A. We also need to show that evaluation preserves suprema. If we have an increasing sequence

  $$(f_1, a_1) \leq (f_2, a_2) \leq ...$$

with suprema (f^*, a^*), then using the monotonicity of ω-continuous functions $f_i(a_i) \leq f_i(a^*) \leq f^*(a^*)$. In addition any other upper bound $(\hat{f}, \hat{a})$ will have $f^*(a^*) \leq f^*(\hat{a}) \leq \hat{f}(\hat{a})$ since a^*, f^* are least upper bounds. Finally we have that for any function $f : A \to B$ and element $a \in A$, $f_\perp(a) = \perp_B \leq f(a)$ so $f_\perp$ is indeed a least element.

2.5 λ-calculus

Now we will look at another way of thinking about collections of objects and functions with the notions of *pairing*, *projection*, *application* and *transposition* - namely the language of λ-calculus. We will then sketch a proof that λ-calculi directly correspond to Cartesian closed categories. See [5, p. 72–80] for a much more precise and detailed coverage of the topic, though much of this presentation is taken from the less formal coverage in [1, p. 143].

Informally λ-calculus is a formalism for specifying functions, which uses the λ character to show binding of variables. For instance the function which takes x to x^2 would be represented as $\lambda x.x^2$.

Definition 2.5.1. A typed λ-calculus consists of

(i) Types: The class of types contains some basic types, including the type 1, and is closed under two type forming operations: if A and B are types then so are $A \times B$, $A \Rightarrow B$.

(ii) Terms: The class of terms contains variables $x, y, z, ... : A$ of each type (where $x : A$ means x is of type A), the constant $* : 1$ and possibly some other typed constants $a, b, c, ... : A$. These are then used to form new terms according to the following rules:

  (a) If $a : A$ and $b : B$ are terms then $\langle a, b\rangle : A \times B$ is a term.

  (b) If $c : A \times B$ is a term, then $\pi_1(c) : A$ and $\pi_2(c) : B$ are terms.

  (c) If $c : A \Rightarrow B$ and $a : A$ are terms, then $ca : B$ is a term.

  (d) If $x : A$ is a variable and $b : B$ is a term, then $\lambda x.b : A \Rightarrow B$ is a term.

  There may be other terms not indicated by these rules.

(iii) Equations: Before listing the equations which must hold, it is worth mentioning that intuitively $\langle -, -\rangle$ means pairing, cx means c applied to x and $\lambda x.b(x)$ is a function taking x to $b(x)$. The λ character *binds* variables in a similar fashion to a quantifier, and variables which are not bound, such as y in the term $\lambda x.yx$, are called *free*. A term with no free variables is called *closed*. Another important notion is that of substitution: if a, b are terms then $a[b/x]$ is the result of replacing every free occurrence of x in a by b. This can cause problems if b contains free variables which become bound (also referred to as caught) in the new term, which we will take care to avoid.

(a) $a = *$ for all $a : 1$.

(b) $\pi_1(\langle a, b\rangle) = a$.

(c) $\pi_2(\langle a, b\rangle) = b$.

(d) $\langle \pi_1(c), \pi_2(c)\rangle = c$.

(e) $(\lambda x.b)a = b[a/x]$.

(f) $\lambda x.cx = c$, as long as there are no occurrences of x in c.

(g) $\lambda x.b = \lambda y.b[y/x]$ (as long as there was not already an occurrence of y in b)

There might also be further equations in any particular λ-calculus. We also have, in line with the standard notion of 'equality':

- Substitution rules, where $a = b$ means also $fa = fb$ for any term f of suitable type and $\lambda x.a = \lambda x.b$ for a variable x.
- Reflexivity, symmetry and transitivity.

It turns out that we can consider λ-calculi to be in some sense languages describing Cartesian closed categories (or phrased differently, that Cartesian closed categories form models for λ-calculi). Seeing this connection might help interpreting the language.

Definition 2.5.2. Given a λ-calculus $\mathscr{D}$, the category of types $\mathbf{C}(\mathscr{D})$ is defined as:

(i) The objects are the types of the λ-calculus.

(ii) The arrows $A \to B$ are equivalence classes of closed terms of type $A \Rightarrow B$. We write $[c]$ for the equivalence class of the term c.

(iii) In particular, the identity arrow 1_A for an object A is $[\lambda x.x]$ where $x : A$.

(iv) The composition gf for arrows $f : A \to B = [c], g : B \to C = [d]$ is given by $[\lambda x.d(cx)]$.

Proposition 2.5.3. *For any typed λ-calculus $\mathscr{D}$, $\boldsymbol{C}(\mathscr{D})$ is a category.*

Proof. First off we will argue that the arrows are well-defined in that they don't depend on the choice of representative. If $x : A$ and $y : A$ then $\lambda x.x = \lambda y.y$ by (g), so $[\lambda x.x] = [\lambda y.y]$ and the identity arrow is well-defined. If $c : A \to B = c' : A \to B$, $d : B \to C = d' : B \to C$ and $x : A, y : A$, then also $\lambda x.d(cx) = \lambda y.d'(c'y)$ by the substitution rule and (g).

With that out of the way we need to check that it fulfills the two equations in definition 2.1.1. If $f : A \to B = [c] : A \Rightarrow B$ is an arbitrary arrow, then indeed

$$f1_A = [\lambda x.c((\lambda y.y)x)] = [\lambda x.cx] = [c] = f$$
$$1_A f = [\lambda x.(\lambda y.y)(cx)] = [\lambda x.cx] = [c] = f$$

so the unit laws hold. For associativity, let $f : A \to B = [a], g : B \to C = [b], h : C \to D = [c]$. Then

$$\begin{aligned} h(gf) =& [\lambda x.c((\lambda y.b(ay))x)] \\ =& [\lambda x.c(b(ax))] \\ =& [\lambda x.(\lambda y.c(by))(ax)] \\ =& (hg)f. \end{aligned}$$

□

Since the careful work of keeping track of the equivalence classes is quite bothersome, from now on we will often omit it and simply identify arrows with representatives of the class. This will not cause any problems.

Theorem 2.5.4. *For any λ-calculus $\mathscr{D}$ the category of types $\boldsymbol{C}(\mathscr{D})$ is a Cartesian closed category.*

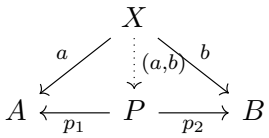
Proof. We need to show that $\mathbf{C}(\mathscr{D})$ has a terminal object, binary products and exponentials.

The terminal object is the object corresponding to the type 1. For any object A the unique arrow $\bigcirc_A : A \to 1$ is given by $\lambda x.*$ for $x : A$. Equation (iii)(a) ensures that any other term of this type will be in the same equivalence class.

Given two objects A, B the projections from $A \times B$ are given by

$$p_1 = \lambda z.(\pi_1(z)), \quad p_2 = \lambda z.(\pi_2(z)),$$

where $z : A \times B$. Given any arrows a, b as in



let $(a, b) = \lambda x.\langle ax, bx\rangle$. Then we have

$$\begin{aligned} p_1(a,b) =& \lambda y.(\lambda z.(\pi_1(z)))(\lambda x.\langle ax, bx\rangle y) \\ =& \lambda y.(\lambda z.(\pi_1(z)))(\langle ay, by\rangle) \\ =& \lambda y.(\pi_1(\langle ay, by\rangle)) \\ =& \lambda y.ay \\ =& a \end{aligned}$$

and similar for p_2. This shows that the required arrow (a, b) exists, it remains to show that it is unique. For any $c : A \times B \to X$, if it makes the above diagram commute then

$$a = p_1 c = \lambda x.p_1(cx) = \lambda x.(\lambda z.(\pi_1(z)))(cx) = \lambda x.(\pi_1(cx))$$

and similarly $b = \lambda x.(\pi_2(cx))$. This means

$$\begin{aligned}(a,b) =& \lambda x.\langle ax, bx\rangle \\ =& \lambda x.\langle(\lambda y.\pi_1(cy))x, (\lambda z.\pi_2(cz))x\rangle \\ =& \lambda x.\langle(\pi_1(cx)), (\pi_2(cx))\rangle \\ =& \lambda x.cx \\ =& c\end{aligned}$$

so (a,b) is unique.

It only remains to show that there are exponentials. Given any objects A, B let B^A be the object corresponding to the type $A \Rightarrow B$. Then we define evaluation by

$$\epsilon = \lambda z.\pi_1(z)\pi_2(z),$$

where $z : (A \Rightarrow B) \times A$. Given any function $f : A \times B \to C$ we take as the transpose

$$f^* = \lambda x.\lambda y.f\langle x, y\rangle,$$

with $x : A$, $y : B$.

Then we wish to show

$$\epsilon(f^* \times 1_B) = f$$

Translating the definition into a λ-term we get $g \times h = \lambda x.\langle g\pi_1(x), h\pi_2(x)\rangle$. We then get

$$\begin{aligned}\epsilon(f^* \times 1_B) =& \lambda w.(\lambda z.\pi_1(z)\pi_2(z))(\lambda x.\langle f^*\pi_1(x), 1_B\pi_2(x)\rangle w) \\ =& \lambda w.(\lambda z.\pi_1(z)\pi_2(z))(\lambda x.\langle(\lambda v.\lambda y.f\langle v, y\rangle)\pi_1(x), \pi_2(x)\rangle w) \\ =& \lambda w.(\lambda z.\pi_1(z)\pi_2(z))(\lambda x.\langle\lambda y.f\langle\pi_1(x), y\rangle, \pi_2(x)\rangle w) \\ =& \lambda w.(\lambda z.\pi_1(z)\pi_2(z))(\langle\lambda y.f\langle\pi_1(w), y\rangle, \pi_2(w)\rangle) \\ =& \lambda w.(\pi_1(\langle\lambda y.f\langle\pi_1(w), y\rangle, \pi_2(w)\rangle)\pi_2(\langle\lambda y.f\langle\pi_1(w), y\rangle, \pi_2(w)\rangle)) \\ =& \lambda w.((\lambda y.f\langle\pi_1(w), y\rangle)(\pi_2(w))) \\ =& \lambda w.f\langle\pi_1(w), \pi_2(w)\rangle \\ =& \lambda w.fw \\ =& f\end{aligned}$$

We have shown that ϵ is such that for any function a transpose exists but not that the transpose is unique.

By doing the same kind of routine λ-calculation as above we can (but to avoid tedium we will not) show

$$(\epsilon(g \times 1_B))^* = g$$

Armed with this we assume that there is some other transposition operation which takes $f : A \times B \to C$ to $f' : A \to C^B$, such that $\epsilon(f' \times 1_B) = f$. Then we have

$$f^* = (\epsilon(f' \times 1_B))^* = f'$$

and so f^* is unique.

With this we have shown that $\mathbf{C}(\mathscr{D})$ has finite products and exponentials, and is Cartesian closed. □

While this is all that will be shown in this thesis, [5, p. 72–80] in fact show that the "meta-categories" $\mathbf{Cart}_N$ of Cartesian closed categories and λ-**Calc** of λ-calculi are isomorphic in a very 'nice' respect which makes them equivalent as categories. We will not go into detail on this, but you can conceptualize it as λ-calculi providing a language for encoding or describing the arrows in a corresponding Cartesian closed category.

3 Deciding isomorphisms in CCC:s

This section will present a sound and complete theory $Th^1_{\times T}$ for deciding isomorphisms that hold in all Cartesian closed categories. The name for the theory has been standardized in the literature, and $\times, T$ indicate the presence of products and terminal objects respectively. We will also see a working decision procedure for concretely determining this, though there are much faster methods such as the one presented in [4]. First we will prove a sound theory for valid isomorphisms, as is done in [2] except we will develop a lot more detail. We will then show, loosely following [6], that no other isomorphisms hold in the category **Finset** so that no other isomorphisms can hold in all CCC:s.

3.1 A sound theory of isomorphisms

We have already seen that for any category with binary products and any objects A, B, C,

$$(A \times B) \times C \cong A \times (B \times C).$$

This section will demonstrate a number of other isomorphisms of this type, which hold regardless of which basic objects are involved. But first we wish to formalize what this means.

Definition 3.1.1. The *object variables* O_v of $\mathbf{C}$ are letters a, b, c, possibly with subscripts, representing arbitrary objects in $\mathbf{C}$. The *object-schemes* O are either object variables, the object-scheme 1 or products/exponentials built from other object-schemes. An *assignment* I assigns to each object variable an object of the category. An assignment can then be uniquely extended to a *full* assignment which assigns objects to all object-schemes, respecting products and exponentials and always assigning the object 1 to the object-scheme 1. We say that two object-schemes are isomorphic if and only if they are assigned isomorphic objects by all full assignments.

It is precisely isomorphisms between object-schemes which stand a chance of generalizing to all CCC:s. We can immediately see that isomorphisms between object-schemes act much like those on objects.

Proposition 3.1.2. $\cong$ *forms an equivalence relation on object-schemes.*

Proof. First we note that by 2.1.5 $\cong$ is an equivalence relation on objects.

- For any object-scheme A and assignment I, $I(A) \cong I(A)$ so also $A \cong A$.
- If A, B are object-schemes such that $A \cong B$ then for any assignment I we have $I(A) \cong I(B)$, which means also $I(B) \cong I(A)$. From this we can conclude $B \cong A$.
- If A, B, C are object-schemes such that $A \cong B$ and $B \cong C$ then for any assignment I we have $I(A) \cong I(B)$ and $I(B) \cong I(C)$ which means also $I(A) \cong I(C)$. From this we can conclude $A \cong C$.

□

An advantage of working with object-schemes is that we can easily talk about specific 'parts' of them.

Definition 3.1.3. If A is an object-scheme with $A = A_0 \times A_1$ or $A = A_0^{A_1}$, then we call A_0, A_1 object-parts of A. Furthermore the object-part relation is reflexive (A is an object-part of A) and transitive (if A is an object-part of B which is an object-part of C, then also A is an object-part of C). We will often want to talk about a specific occurrence of an object-part X in some object-scheme A, which will be given by a *position*, a sequence $(x_0, x_2, x_3, ..., x_n)$ where each x_i is either 0 or 1. Saying that X occurs at position p in A then means

- X occurs at position $p = ()$ in A, where p is the empty sequence, if and only if $A = X$.
- Let $A = A_0 \times A_1$ or $A = A_0^{A_1}$, and $p = (x_0, p')$ where p' is a sequence. Then X occurs at position p in A if and only if X occurs at position p' in A_{x_0}.

This inductive definition just formalizes the idea that the numbers are 'instructions' telling you whether the left (0) or right (1) part of A contain the occurrence of X we are looking for, and that after finishing all the instructions in the sequence you will arrive at X. We call a position valid for an object-scheme A if there is some object-part that occurs there. Finally we write $A[Y/p]$ for the result of replacing the object-part at position p with the object-scheme Y, which can also be defined inductively

- If p is the empty sequence, then $A[Y/p] = Y$.

- If $A = A_0 \times A_1$ or $A = A_0^{A_1}$, $p = (x_0, p')$ where p' is a sequence, then you get $A[Y/p]$ by replacing A_{x_0} by $A_{x_0}[Y/p']$.

While somewhat complicated, this precise definition of object-parts and occurrences will quickly pay off. First though, let us look in some detail at a useful lemma which is proved very briefly in [2].

Lemma 3.1.4. *Products and exponentials in a CCC 'preserve' isomorphisms of object-schemes, meaning if $A \cong B$ and $C \cong D$ then also*

(i) $A \times C \cong B \times D$

(ii) $C^A \cong D^B$

Proof. (i) If A, B, C, D are object-schemes with $A \cong B$ and $C \cong D$ then there are isomorphisms $f : I(A) \to I(B), g : I(C) \to I(D)$ for any full assignment I. Then $f \times g : I(A) \times I(C) \to I(B) \times I(D)$ is an isomorphism with inverse $f^{-1} \times g^{-1}$. This can be shown by a straight-forward λ-calculation

$$\begin{aligned}(f \times g)(f^{-1} \times g^{-1}) =& \lambda z.(\lambda x.\langle f\pi_1(x), g\pi_2(x)\rangle)(\lambda y.\langle f^{-1}\pi_1(y), g^{-1}\pi_2(y)\rangle z)\\ =& \lambda z.(\lambda x.\langle f\pi_1(x), g\pi_2(x)\rangle)(\langle f^{-1}\pi_1(z), g^{-1}\pi_2(z)\rangle)\\ =& \lambda z.(\langle f(f^{-1}\pi_1(z)), g(g^{-1}\pi_2(z))\rangle)\\ =& \lambda z.(\langle \pi_1(z), \pi_2(z))\rangle)\\ =& \lambda z.z\\ =& 1_{I(A)\times I(C)}.\end{aligned}$$

Very similarly you can show $(f^{-1} \times g^{-1})(f \times g) = 1_{I(B)\times I(D)}$. Thus we find that $I(A \times C) = I(A) \times I(C) \cong I(B) \times I(D) = I(B \times D)$ for any full assignment I.

(ii) With the same situation as in (i), we now want to associate to each arrow $s : I(A) \to I(C)$ an arrow $t : I(B) \to I(D)$, using f, g. But this could be done by setting $t = gsf^{-1}$. In other words, by picking a t such that this diagram commutes.

$$\begin{array}{ccc} I(A) & \xrightarrow{s} & I(C) \\ {\scriptstyle f^{-1}}\uparrow \;\downarrow{\scriptstyle f} & & {\scriptstyle g^{-1}}\uparrow \;\downarrow{\scriptstyle g} \\ I(B) & \xrightarrow{t} & I(D) \end{array}$$

This also has an obvious inverse, from a given t pick the s that makes this diagram commute. With $b : I(B)$ the arrow corresponding to $\lambda s.\lambda b.g(s(f^{-1}b)) : (I(A) \Rightarrow I(C)) \Rightarrow (I(B) \Rightarrow I(D))$ will carry out this process and turns out to indeed be an isomorphism, though we will skip carrying out the precise λ-calculation this time.

This shows that for any full assignment I, $I(C^A) = I(C)^{I(A)} \cong I(D)^{I(B)} = I(D^B)$ which shows the lemma.

□

Using the idea of object-parts developed earlier we can now extend this lemma.

Corollary 3.1.5. *If $X \cong Y$ and X is an object-part of A occurring at p, then $A \cong A[Y/p]$.*

Proof. The proof will be by induction on the length of p.

- If p is the empty sequence, then $A = X$ and $A[Y/p] = Y$ so the result follows immediately.
- If the result holds for position sequences of length n, then assume $p = (x_0, p')$ where p' is a sequence of max length n. We also have either $A = A_0 \times A_1$ or $A = A_0^{A_1}$, and w.l.o.g. we can assume $x_0 = 0$. Then we note that X is an object-part occurring at position p' in A_0 so by induction $A_0[Y/p] \cong A_0$. Then we apply Lemma 3.1.4 and get that also $A = A_0 \times A_1 \cong A_0[Y/p] \times A_1 = A[Y/p]$ or similarly if A is an exponential. This shows the induction step.

□

This brings us to our first main result, which follows [2, Th. 2.6], though we will develop the proof in much more detail.

Theorem 3.1.6. *In any CCC, if the theory $Th^1_{\times T}$ of equality over object-schemes, given by the axioms*

(i) $A \times B = B \times A$

(ii) $(A \times B) \times C = A \times (B \times C)$

(iii) $C^{A \times B} = (C^B)^A$

(iv) $(B \times C)^A = B^A \times C^A$

(v) $A \times 1 = A$

(vi) $A^1 = A$

(vii) $1^A = 1$

proves $M = N$ for any object-schemes M, N, then $M \cong N$.

Proof. The proof will be an induction on the length of the proof. The statement holds trivially for proofs with 0 steps. If we assume that any proof in less than k steps of $M = N$ means $M \cong N$, then if a proof in k steps shows $M = N$ the final step must be either

(i) A direct application of one of the above axioms.

(ii) An application of the transitive or symmetric property using an earlier proof that $N = M$ or $M = X$ and $X = N$.

(iii) An application of the substitution rule, using an earlier proof of $X = Y$ for some object-part X occurring at position p in M to show $M = M[Y/p]$ which is then identical to N, or vice versa.

We showed in prop. 3.1.2 that isomorphisms form an equivalence relation on object-schemes, which shows that in case (ii) the induction hypothesis will let us conclude our isomorphism $M \cong N$, since earlier proofs are shorter than k steps. Using corollary 3.1.5 we immediately see that in case (iii) $X \cong Y \Rightarrow M \cong M[Y/p] = N$. It remains to show that for any assignment I there are valid isomorphisms corresponding to the 7 axioms. In most cases this will be done by specifying a λ-term representing the isomorphism.

(i) Let $x : I(A) \times I(B), y : I(B) \times I(A)$ be variables. Then

$$f = \lambda x.\langle \pi_2(x), \pi_1(x)\rangle : I(A) \times I(B) \Rightarrow I(B) \times I(A)$$

and

$$g = \lambda y.\langle \pi_2(y), \pi_1(y)\rangle : I(B) \times I(A) \Rightarrow I(A) \times I(B)$$

provide the two directions of the desired isomorphism, since (with $z : I(B) \times I(A)$)

$$\begin{aligned} fg =& \lambda z.(\lambda x.\langle \pi_2(x), \pi_1(x)\rangle)(\lambda y.\langle \pi_2(y), \pi_1(y)\rangle z) \\ =& \lambda z.(\lambda x.\langle \pi_2(x), \pi_1(x)\rangle)(\langle \pi_2(z), \pi_1(z)\rangle) \\ =& \lambda z.(\langle \pi_2(\langle \pi_2(z), \pi_1(z)\rangle), \pi_1(\langle \pi_2(z), \pi_1(z)\rangle)\rangle) \\ =& \lambda z.(\langle \pi_1(z), \pi_2(z)\rangle) \\ =& \lambda z.z \\ =& 1_{I(B)\times I(A)} \end{aligned}$$

and by almost exactly the same steps $gf = 1_{I(A)\times I(B)}$.

(ii) See Corollary 2.2.8.

(iii) Let $x : (I(A) \times I(B)) \Rightarrow I(C), y : I(A) \Rightarrow (I(B) \Rightarrow I(C)), a : I(A), b : I(B), d : I(A) \times I(B)$ be variables. Then

$$f = \lambda x.\lambda a.\lambda b.x\langle a, b\rangle : ((I(A) \times I(B)) \Rightarrow I(C)) \Rightarrow (I(A) \Rightarrow (I(B) \Rightarrow I(C)))$$

and

$$g = \lambda y.\lambda d.(y\pi_1(d))\pi_2(d) : (I(A) \Rightarrow (I(B) \Rightarrow I(C))) \Rightarrow ((I(A) \times I(B)) \Rightarrow I(C))$$

provide the two directions of the desired isomorphism, since (with $z : (I(A)\times I(B)) \Rightarrow I(C)$)

$$\begin{aligned} gf =& \lambda z.(\lambda y.\lambda d.(y\pi_1(d))\pi_2(d))(\lambda x.\lambda a.\lambda b.x\langle a,b\rangle z) \\ =& \lambda z.(\lambda y.\lambda d.(y\pi_1(d))\pi_2(d))(\lambda a.\lambda b.z\langle a,b\rangle) \\ =& \lambda z.(\lambda d.(\lambda a.\lambda b.z\langle a,b\rangle\pi_1(d))\pi_2(d)) \\ =& \lambda z.(\lambda d.(\lambda b.z\langle \pi_1(d),b\rangle)\pi_2(d)) \\ =& \lambda z.(\lambda d.(z\langle \pi_1(d),\pi_2(d)\rangle)) \\ =& \lambda z.(\lambda d.zd) \\ =& \lambda z.z \\ =& 1_{(I(A)\times I(B))\Rightarrow I(C)} \end{aligned}$$

and similarly $fg = 1_{I(A)\Rightarrow(I(B)\Rightarrow I(C)}$.

(iv) Let $x : I(A) \Rightarrow (I(B) \times I(C)), y : (I(A) \Rightarrow I(B)) \times (I(A) \Rightarrow I(C)), a : I(A)$ be variables. Then

$$f = \lambda x.\langle \lambda a.\pi_1(xa), \lambda a.\pi_2(xa)\rangle : (I(A) \Rightarrow (I(B)\times I(C))) \Rightarrow ((I(A) \Rightarrow I(B))\times(I(A) \Rightarrow I(C)))$$

and

$$g = \lambda y.\lambda a.\langle \pi_1(y)a, \pi_2(y)a\rangle : (I(A) \Rightarrow I(B))\times(I(A) \Rightarrow I(C)) \Rightarrow (I(A) \Rightarrow (I(B)\times I(C)))$$

provide the two directions of the desired isomorphism. This can be proved by another two long but straight-forward λ-calculations showing $gf = 1_{I(A)\Rightarrow(I(B)\times I(C))}, fg = 1_{(I(A)\Rightarrow I(B))\times(I(A)\Rightarrow I(C))}$ which we will skip.

(v) Let $x : I(A) \times 1, y : I(A)$ be variables. Then

$$f = \lambda x.\pi_1(x) : I(A) \times 1 \Rightarrow I(A)$$

and

$$g = \lambda y.\langle y, *\rangle : I(A) \Rightarrow I(A) \times 1$$

provide the two directions of the desired isomorphism. With $z : I(A) \times 1$ we have

$$\begin{aligned} gf =& \lambda z.(\lambda y.\langle y,*\rangle)(\lambda x.\pi_1(x)z) \\ =& \lambda z.(\lambda y.\langle y,*\rangle)\pi_1(z) \\ =& \lambda z.(\langle \pi_1(z),*\rangle) \\ =& \lambda z.(\langle \pi_1(z),\pi_2(z)\rangle) \\ =& \lambda z.z \\ =& 1_{I(A)\times 1} \end{aligned}$$

This proof uses that since $\pi_2(z)$ is a term of type 1, equation (iii)(a) forces $\pi_2(z) = *$. The proof of $fg = 1_I(A)$ is on the other hand entirely trivial and will thus be skipped.

(vi) Let $x : 1 \Rightarrow I(A), y : I(A), a : 1$ be variables. Then as usual we construct the two directions of the isomorphism

$$f = \lambda x.(x*) : (1 \Rightarrow I(A)) \Rightarrow I(A)$$

and

$$g = \lambda y.\lambda a.y : I(A) \Rightarrow (1 \Rightarrow I(A)).$$

We will only prove the direction that uses the special properties of 1 (with $z : 1 \Rightarrow I(A)$):

$$\begin{aligned} gf =& \lambda z.(\lambda y.\lambda a.y)(\lambda x.(x*)z) \\ =& \lambda z.(\lambda y.\lambda a.y)(z*) \\ =& \lambda z.\lambda a.(z*) \\ =& \lambda z.\lambda a.(za) \\ =& \lambda z.z \\ =& 1_{1 \Rightarrow I(A)} \end{aligned}$$

(vii) Let $x : 1, y : I(A) \Rightarrow 1, a : I(A)$ be variables. Then

$$f = \lambda x.\lambda a.x : 1 \Rightarrow (I(A) \Rightarrow 1)$$

and

$$g = \lambda y.* : (I(A) \Rightarrow 1) \Rightarrow 1$$

form the isomorphism. I(A)s before we will prove the direction using the unique property of 1 (with $z : I(A) \Rightarrow 1$):

$$\begin{aligned} gf =& \lambda z.(\lambda x.\lambda a.x)((\lambda y.*)z) \\ =& \lambda z.(\lambda x.\lambda a.x)* \\ =& \lambda z.\lambda a.* \\ =& \lambda z.\lambda a.(za) \\ =& \lambda z.z \\ =& 1_{I(A) \Rightarrow 1} \end{aligned}$$

which uses that $za : 1 = * : 1$.

This shows that also proofs with k steps allow us to conclude $M \cong N$, which then shows it for all proofs by induction. □

3.2 A decision-procedure for $Th^1_{\times T}$

In this section we will describe a procedure to determine whether $Th^1_{\times T}$ proves $A = B$ for any object-schemes A and B. This will be done by 'reducing' them to the normal form presented in [6].

To do this we will give directions to the equalities of $Th^1_{\times T}$.

Definition 3.2.1. Let the one-step-reduction relation $\rightarrowtail$ be defined by

(i) $(C^A)^B \rightarrowtail C^{A\times B}$

(ii) $(B \times C)^A \rightarrowtail B^A \times C^A$

(iii) $A \times 1 \rightarrowtail A$

(iv) $1 \times A \rightarrowtail A$

(v) $A^1 \rightarrowtail A$

(vi) $1^A \rightarrowtail 1$

(vii) $A \times (B \times C) \rightarrowtail (A \times B) \times C$

Commutativity is notably missing from this theory and will be dealt with later, for now notice the extra reduction relation for $1 \times A$.

We then define $\overset{*}{\rightarrowtail}$ as the substitutional closure of $\rightarrowtail$, meaning that if $X \overset{*}{\rightarrowtail} Y$ then if X occurs as an object-part in A at position p, $A \overset{*}{\rightarrowtail} A[Y/p]$. Finally let $\tilde{\rightarrowtail}$ be the reflexive and transitive closure of $\overset{*}{\rightarrowtail}$.

Note that clearly, if $A\tilde{\rightarrowtail}B$ then $Th^1_{\times T}$ also proves $A = B$, since every form of reduction preserves equality. The following two lemmas are based on [2, prop. 2.8], though they use a slightly different normal form and only prove the first lemma.

Lemma 3.2.2. *This reduction process is strongly normalizing, meaning that repeatedly applying reduction rules (in any order) to an object-scheme will eventually give you a normal form where no more reductions are possible.*

Proof. Rules (i) and (ii) both reduce the number of products and exponentiations present in the 'base' of an exponent object-part. No rules increase this number, so they can only be applied a finite number of times. Rules (iii)-(vi) all reduce the size of the formula, so between uses of (i) and (ii) these can only be applied a finite number of times. Between uses of all other rules there are only a finite number of uses of (vii) possible, which won't increase the size of the formula either. Thus you will eventually run out of ways to apply all rules. □

Lemma 3.2.3. *This reduction process is locally confluent, meaning if $A \overset{*}{\rightarrowtail} B$ and $A \overset{*}{\rightarrowtail} C$ then there is some object-scheme D such that $B\tilde{\rightarrowtail}D$ and $C\tilde{\rightarrowtail}D$.*

This is important because it hints that the order of applying the reduction steps does not matter, you will end up with the same object anyway.

Proof. The proof is somewhat involved but unimportant for the larger narrative, so it can be found in Appendix I. □

Since we already showed that this reduction was strongly normalizing Newman's lemma gives that the reduction is *globally confluent*, so that if $A \tilde{\rightarrowtail} B$ and $A \tilde{\rightarrowtail} C$, then there is some object D such that $B \tilde{\rightarrowtail} D$ and $C \tilde{\rightarrowtail} D$. That also means that every object A has a unique normal form, since otherwise there would need to be ways of further reducing the two different normal forms to the same object which is obviously impossible. We will refer to the normal form of A as $nf(A)$.

Remark 3.2.4. If A is in normal form all exponent object-parts X^Y will have X be an object-variable, and either $A = 1$ or there are no occurrences of 1 as an object-part in A. Any product will also be nested from the left as in $(...((A_1 \times A_2) \times A_3) \times ...)$, and we will occasionally write just $A_1 \times A_2 \times A_3 \times ...$ for short. If any of this was not the case there would be further reductions possible.

Finally let us state a very simple lemma about commutativity.

Lemma 3.2.5. *If commutativity shows $nf(A) = nf(B)$ then $Th^1_{\times T}$ shows $A = B$.*

Proof. This is obvious since $Th^1_{\times T}$ proves $A = nf(A)$ and $B = nf(A)$, and commutativity is an axiom of this theory so it also proves $nf(A) = nf(B)$. □

We've thus shown the soundness of a simple decision procedure for determining whether A and B are isomorphic: reduce both to normal form and exhaustively search for ways of commuting the object-parts to make them equal. It remains to show the completeness, that if we cannot find a way to commute the normal forms which make them equal then there is some assignment in some CCC which makes them non-isomorphic.

3.3 Showing the completeness of $Th^1_{\times T}$

In this section we will show the converse of Theorem 3.1.6, namely that if $Th^1_{\times T}$ cannot prove $M = N$, then there is a Cartesian closed categories with an assignment I such that $I(M) \ncong I(N)$. We will also demonstrate that this leads to a complete decision procedure. As in [6] this will be done by looking at **Finset**.

Definition 3.3.1. Finset is the category with finite sets as objects and regular functions between them as arrows. It is a Cartesian closed category by interpreting the product $A \times B$ as the Cartesian product, 1 as some one-element set $\{*\}$ and the exponential B^A as the set of functions $f : A \to B$. We also define a mapping card : finite sets $\to \mathbb{N}$ taking objects of **Finset** to their cardinality, or number of elements.

Proposition 3.3.2.

- *For any sets A and B, $A \cong B$ if and only if card(A) = card(B).*
- *If A and B are sets with card$(A) = n$, card(B)=m then card$(A \times B) = n \cdot m$ and card$(B^A) = m^n$.*

These results are well-known and will be left without proof.

Make note that by composing a full assignment with card we can get a *cardinality assignment* $J : O \to \mathbb{N}$ which assigns to each object-scheme a cardinality. It is simple to show that two object-schemes are isomorphic if and only if they are assigned the same cardinality by every cardinality assignment.

We now wish to show that if A and B are object-schemes such that you cannot show $nf(A) = nf(B)$ using commutativity, then $A \ncong B$. This will require a rather technical lemma.

Before getting into it let us look at a motivational idea. Say the object variables used in the object-schemes are included among $a_0, a_1, ..., a_n$, and that the maximum arity of any product in A or B is K. Then, if we could pick a cardinality assignment J such that $J(a_0) > 1$ and $J(a_{i+1}) > J(a_i)^K$ for all i, the cardinality of two product order-schemes would be equal if and only if they contain exactly the same multiplicities of all object-variables. Indeed, if there was some maximum index s such that the multiplicity of a_s is greater in A than it is in B, then no increase in the multiplicity of object-variables with lower index in B could compensate for the cardinality J would assign the extra a_s, without breaking the assumption that no products have more than K factors. What we have done is essentially assign each object-variables such a different cardinality that we could never trade several of one for another, which forces the products to contain the exact same multiplicities of everything in order to be equal.

Formalizing this process and expanding it to deal with exponentials will be done through a technical lemma, which will then easily show the completeness of $Th^1_{\times T}$. The core idea of this lemma, if not the exact execution, comes from [6, Lemma 7]

Lemma 3.3.3. *Let A and B be object-schemes in normal form such that they cannot be proven equal by commutativity. Let $V = \{a_0, a_1, ..., a_n\}$ include all the object variables occurring in A and B. Furthermore let K be the largest arity of any product in either object-scheme, let $L = K + 1$ and let*

$$J_c : V \to \mathbb{N}, a_i \mapsto c^{L^i},$$

be a cardinality assignment for each c. Then for any $\delta > 0$ there is a $p > 0$ such that J_c will, when extended to a full assignment, fulfill either $J_c(A) > \delta \cdot J_c(B)$ for all $c > p$ or $J_c(B) > \delta \cdot J_c(A)$ for all $c > p$.

The proof will be by induction on the *exponential depth* of object-schemes, meaning the maximum number of nested exponentiations. For an object-scheme A let $\mathscr{D}(A)$ refer to this exponential depth.

Proof. **Base case:** If $\max(\mathscr{D}(A), \mathscr{D}(B)) = 0$ then both A and B are products of object variables. Let s be the greatest index such that a_s has a higher multiplicity in one of the

products, and w.l.o.g. we assume that it has multiplicity m_s in B and $m_s + m'$ in A, where $m' \geq 1$. This is guaranteed to be possible since otherwise commutativity would prove $A = B$.

If $s = 0$ then $A = a_0^{m_0+m'} = c^{m_0+m'}$ while $B = c^{m_0}$, so picking $p = \max(1, \delta)$ is enough to conclude $J_c(A) > \delta \cdot J_c(B)$ for $c > \delta$. We now assume $s > 0$ and write

$$X_c = J_c(a_s)^{m_s} \cdot J_c(a_{s+1})^{m_{s+1}} \cdot ... \cdot J_c(a_n)^{m_n},$$

where $m_{s+1}, ..., m_n$ are the multiplicities of these factors in A and B (which are the same since s was the highest index where they differ). Finally we set $p = \max(1, \delta)$ and get for $c > p$

$$\begin{aligned} J_c(A) \geq & X_c \cdot J_c(a_s)^{m'} \\ \geq & X_c \cdot c^{L^s} \\ = & X_c \cdot (c^{L^{s-1}})^{K+1} \\ > & \delta \cdot X_c \cdot (c^{L^{s-1}})^K \\ \geq & \delta J_c(B), \end{aligned}$$

which shows the lemma.

Inductive step: If the lemma holds for any object-schemes A', B' with $\max(\mathscr{D}(A'), \mathscr{D}(B')) < d$, then we will show that it also holds for any object-schemes A, B with $d = \max(\mathscr{D}(A), \mathscr{D}(B))$.

In general A, B will be products (possibly unary) of exponential objects of the form $a_i^{A'}$ with A' being an object-scheme with $\mathscr{D}(A') < d$. This is because the 'base' of any exponential object-part of a normal form will be an object variable, see remark 3.2.4.

The simplest case is when they are both unary products, so $A = a_i^{A'}$ and $B = a_j^{B'}$. Here we will show a slight strengthening of the lemma, namely that for any δ there is a p such that either $J_c(A) > \delta J_c(B)^K$ for all $c > p$ or $J_c(B) > \delta J_c(A)^K$ for all $c > p$.

If commutativity shows $A' = B'$ then by the soundness of the theory we must have for any c that $J_c(A') = J_c(B')$ and in order for it not to show $A = B$ we must have $i \neq j$. W.l.o.g. we assume $i > j \geq 0$ and then set $p = max(1, \delta)$ giving us for all $c > p$

$$\begin{aligned} J_c(A) = J_c(a_i)^{J_c(A')} = c^{L^i \cdot J_c(A')} = c^{L^{i-1} \cdot J_c(A')(K+1)} \\ > \delta (c^{L^{i-1} \cdot J_c(A')})^K \geq \delta (c^{L^j \cdot J_c(B')})^K = \delta J_c(B)^K, \end{aligned}$$

which shows the (stronger version of) the lemma. If instead commutativity cannot show $A' = B'$ the induction hypothesis applies. We pick $\delta_1 = L^{j-i+1}$ and then assume w.l.o.g. that $J(A') > \delta_1 J(B')$ for $c > p_1$ for some p_1. If also $i > j$ then showing that $J_c(a_i^{A'}) > \delta J_c(a_j^{B'})$ for large enough c is trivial. The interesting case is when $j > i$, so that the base and the exponent are 'pushing in different directions'. In that case pick $p = \max(p_1, \delta, 1)$.

This gives that for all $c > p$

$$J_c(A) = J_c(a_i)^{J_c(A')} = c^{L^i \cdot J_c(A')} > c^{L^i \cdot L^{j-i+1} J_c(B')}$$
$$= c^{L^j \cdot J_c(B') \cdot (K+1)} > \delta (c^{L^j \cdot J_c(B')})^K = \delta J_c(B)^K,$$

which shows the (stronger version of the) lemma in this case also.

If they are not unary products then they can be written on the form $A = a_{i_1}^{A'_1} \times a_{i_2}^{A'_2} \times ... \times a_{i_k}^{A'_k}$, $B = a_{i_{k+1}}^{A'_{k+1}} \times a_{i_{k+2}}^{A'_{k+2}} \times ... \times a_{j_t}^{A'_t}$.

We then take any δ and do pair-wise applications of the earlier proof that the (stronger version of the) lemma holds for comparing object-schemes with exponentiation as the outer-most operator, and find that every pair $a_{i_s}^{A'_s}, a_{i_q}^{A'_q}$ can either be proven equal by commutativity so that $J_c(a_{i_s}^{A'_s}) = J_c(a_{i_q}^{A'_q})$ for all c, or there is some $p_{s,q}$ such that either $J_c(a_{i_s}^{A'_s}) > \delta J_c(a_{i_q}^{A'_q})^K$ for all $c > p$ or the reverse. This means that the order $<_J$ on the factors $a_{i_s}^{A'_s}$ induced by J_c with $c > p^* = \max(p_{1,1}, p_{1,2}, ..., p_{2,1}, ..., p_{t,t})$ will not depend on which such c is picked. We write $a_{i_s}^{A'_s} =_J a_{i_q}^{A'_q}$ if J_c assigns the same cardinality to $a_{i_s}^{A'_s}$ and $a_{i_q}^{A'_q}$ for $c > p^*$, and note that this implies that $a_{i_s}^{A'_s} = a_{i_q}^{A'_q}$ is provable from commutativity. We should also notice that clearly the order $<_J$ is a strict total order up to $=_J$.

If we say that the multiplicity of a factor in A or B is the number of other factors there are in the object-scheme which are equal by $=_J$, then we can let S be the greatest factor (by $<_J$) which exists in higher multiplicity in one object-scheme than the other. This is guaranteed to exist since equality in this order means provable equality by commutativity, so if all factors have the same multiplicity $A = B$ can be shown by commutativity. We can assume w.l.o.g. that S has higher multiplicity in A, and let Q be a greatest factor in B less than S by $<_J$. Note that this means

$$J_c(S) > \delta J_c(Q)^K.$$

Again let X_c be the cardinality assigned by J_c to the product of all factors equal to or greater than S in B (by $<_J$). There will be corresponding factors in A assigned equal cardinality to those in X_c since A has higher multiplicity of factors equal to S, and S was the greatest where they differed in multiplicity. Then we have, for $c > p^*$

$$J_c(A) \geq X_c \cdot J_c(S) > \delta X_c \cdot J_c(Q)^K \geq J_c(B),$$

which completes the inductive step. □

Theorem 3.3.4. *If it is true in all CCC:s that there is an isomorphism $A \cong B$ then commutativity shows $nf(A) = nf(B)$ and $Th^1_{\times T}$ proves $A = B$.*

Proof. If it is true in all CCC:s then in particular it is true in **Finset** for the equivalent object-schemes, and also for their normal forms $nf(A) \cong nf(B)$. But if commutativity did not show $nf(A) = nf(B)$ then by 3.3.3 there is an assignment which proves that $nf(A) \ncong nf(B)$ in **Finset**. This is a contradiction, so clearly we must have that commutativity shows $nf(A) = nf(B)$, and by 3.2 also $Th^1_{\times T}$ proves $A = B$. □

Corollary 3.3.5. *For any objects A and B reducing them to normal form and checking if there is some way to commute $nf(A)$ to get $nf(B)$ constitutes a valid decision procedure for determining if $A \cong B$ in any CCC.*

Proof. If you can commute $nf(A)$ to get $nf(B)$ then by 3.2 $Th^1_{\times T}$ proves $A = B$, and by 3.1.6 this implies $A \cong B$ in all CCC:s. The other direction was just proved above. □

4 Extending to infinity

After obtaining a good understanding of the isomorphisms which hold in **all** Cartesian closed categories we will now try to restrict ourselves to only those which contain 'infinite' objects. The simplest or most natural example of an infinite set is probably $\mathbb{N}$, and we will similarly consider what are called *natural number objects*, based on the presentation in [5].

4.1 Natural number objects and types

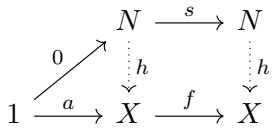
Definition 4.1.1. An object N in a Cartesian closed category along with arrows $0 : 1 \to N$ and $s : N \to N$ is called a *strong* natural number object (NNO) if, given any diagram of the form

$$1 \xrightarrow{a} X \xrightarrow{f} X$$

there is a unique arrow $h : N \to X$ such that

$$h0 = a, \quad hS = fh,$$

as in this commutative diagram



In **Set** of course $\mathbb{N}$ is a natural number object with the map from a singleton set to 0 being the arrow $0 : 1 \to \mathbb{N}$, and the successor function being $s : \mathbb{N} \to \mathbb{N}$. The unique function $h : \mathbb{N} \to X$ will take 0 to a and then take n to $f(f(f(...a...)))$ repeated n times. A *weak natural number object* is an object like above where h is not necessarily unique. An example of a weak NNO in **Set** is $\mathbb{N} \cup \{-1\}$ with the standard successor function, since where h maps the -1 is irrelevant and there are thus many valid choices.

It is shown in [5, Cor. 9.2] that a CCC with weak NNO N will, for any objects B, have arrows $I_B : B \times B^B \times N \to B$ with the property that for any arrows $a : 1 \to B$, $f : 1 \to B^B$, $n : 1 \to N$,

$$I_B\langle a, f, 0\rangle = a, \quad I_B\langle a, f, Sn\rangle = fI_B\langle a, f, n\rangle.$$

You can think of this as essentially iterating f on a a total of n times.

We can use this to define a natural number type in λ-calculus.

Definition 4.1.2. A typed λ-calculus with natural number type follows definition 2.5.1 except in addition it has:

- The type N
- The constant $0 : N$, and the term forming rules:
  - If $n : N$ is a term, then $S(n) : N$ is a term.
  - If $a : A, f : A \Rightarrow A, n : N$ are terms, then $I_A(a, f, n) : A$ is a term.
- The equations:
  - $I_A(a, f, 0) = a.$ for all types A
  - $I_A(a, f, S(n)) = fI_A(a, f, n)$ for all types A.

Proposition 4.1.3. *For a λ-calculus $\mathscr{L}$ with natural number type, its corresponding category of types $\boldsymbol{C}(\mathscr{L})$ is a CCC with a (weak) natural number object.*

Proof. We already know that it is a CCC. The natural number object is obviously the object corresponding to the type N, and if $z : 1, n : N$ are variables then the arrow $\mathbf{0}$: $1 \to N$ is given by $[\lambda z.0]$ and the arrow $S : N \to N$ is given by $[\lambda n.S(n)]$. Finally, given arrows $\mathbf{a} : 1 \to A = [a]$, $\mathbf{f} : A \to A = [f]$ we get the arrow $h : N \to A$ from $[\lambda n.I(f, a*, n)]$, and a routine λ-calculation will show that it fulfills definition 4.1.1. □

As we might have expected, λ-calculi with natural number types and CCC:s with weak natural number objects are in fact equivalent, as shown in [5, p. 72-80]. If you add that equality of terms should be closed under *term induction*:

- Let $h(n), h'(n) : A$ be terms with a free variable $n : N$. Then if $h(0) = h'(0)$ and there is an $f : A \Rightarrow A$ such that $h(S(n)) = fh(n), h'(S(n)) = fh'(n)$, we conclude $h(n) = h'(n)$.

then it will correspond to a CCC with a strong natural number object, since that exactly corresponds to the uniqueness of the arrow h in definition 4.1.1. It is easy to see that term induction will not be valid with only a weak NNO when considering examples of weak NNO:s like $\mathbb{N} \cup \{-1\}$, where there are elements which are not of the form $S(S(S(...0...)))$. h and h' might act very differently on -1 in this case, while still fulfilling the requirements for term induction.

In particular notice that if $A = N, f = S$ and $h'(n) = n$ then $h(0) = 0$ and $h(S(n)) = S(h(n))$ is enough to conclude $h(n) = n$. This is the particular instance of the rule which we will use later.

4.2 Primitive recursive functions in λ-calculus

In order to understand isomorphisms in CCC:s with natural number objects we will, following [5], try to understand which numerical functions are *representable* in a λ-calculus with natural number type. We will specifically consider the *primitive recursive functions*, a famous and very well-behaved set of functions which take some number of natural number arguments and map them to a single natural number. We say that a function taking k natural number arguments is $k-$ary.

Definition 4.2.1. The *basic* primitive recursive functions are:

(i) the 0-ary **constant function** 0, which takes no arguments and always takes the value 0.

(ii) the 1-ary **successor function** S, which maps $n \mapsto n + 1$.

(iii) the **projections** p_i^k, which are $k-$ary functions mapping $(n_1, n_2, ..., n_k) \mapsto n_i$. In other words they pick the i:th element of the $k-$tuple.

The set of primitive recursive functions is the smallest set of functions containing the basic ones and being closed under the following two schemes for generating new functions:

(i) **Composition**, if f, a $k-$ary function, and $g_1, g_2, ..., g_k$, all $m-$ary functions, are all primitive recursive, then the $m-$ary function h which maps an $m-$tuple $a \in N^m$ to $f(g_1(a), g_2(a), ..., g_k(a))$ is primitive recursive.

(ii) **Primitive recursion**, if f, a $k-$ary function, and g, a $(k+2)$-ary function, are both primitive recursive, then also the $(k+1)$-ary h with

$$h(0, x_1, .., x_k) = f(x_1, ..., x_k) \quad h(S(n), x_1, ..., x_k) = g(n, h(n, x_1, ..., x_k), x_1, ..., x_k),$$

is primitive recursive.

Primitive recursion can be interpreted as repeatedly applying g to f (both parameterized by $x_1, ..., x_k$), except g also takes the current 'loop-number' as a first argument. This turns out to be a very powerful construction, and many commonly used numeric functions are primitive recursive.

Remark 4.2.2. In particular the following functions are primitive recursive:

- The 2-ary functions of addition, multiplication and exponentiation.
- The 1-ary function sgn given by

$$sgn(n) = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{otherwise} \end{cases}$$

- The 2-ary function "monus" (written $\dot{-}$) given by

$$x \dot{-} y = \begin{cases} x - y, & \text{if } x > y \\ 0, & \text{otherwise} \end{cases}$$

- The 2-ary functions of integer division and remainder.

For a proof of this see [3, p. 10–15].

Now we will look at how this relates to λ-calculi (cf. [5, p. 257]).

Definition 4.2.3. Let $\#n$ be the term $\underbrace{S(S(S(}_{\text{n times}}...0...)))$. We say that a $k-$ary function f is representable in a λ-calculus with natural number type $\mathscr{L}$ if there is a closed term $F : N^k \Rightarrow N$ such that

$$F\langle ...\langle \#n_1, \#n_2\rangle, ..., \#n_k\rangle = \#f(n_1, n_2, ..., n_k)$$

for every k-tuple $(n_1, n_2, ..., n_k) \in \mathbb{N}^k$.

We now present [5, Th. 2.4].

Theorem 4.2.4. *All primitive recursive functions are representable in any λ-calculus with natural number type.*

Proof. First we show that the basic primitive recursive functions are representable:

(i) The constant function 0 is represented by the term 0.

(ii) The successor function S is represented by the term $\lambda n.S(n)$, where $n : N$.

(iii) The projections are given by terms applying π_1, π_2 correctly to navigate to the position you wish to project. For instance $p_2^3 = \lambda x.\pi_2(\pi_1(x))$, where $x : (N \times N) \times N$.

It is obvious that we can do composition, but showing that primitive recursion is possible will take some more work. Given terms $F : N^k \Rightarrow N, G : ((N \times N) \times N^k) \Rightarrow N$ representing f and g we write $F_x = Fx$ and $G_x = \lambda u.G\langle u, x\rangle$, where $x : N^k, u : N \times N$ are variables. Obtaining the term $H : (N \times N^k) \Rightarrow N$ so that it matches the h generated by primitive recursion could now almost be done by using the iterator construction $I_{N \times N^k}$ to iterate G_x on F_x n times - however G_x still needs to know which loop it is on as a second argument. We can solve this by iterating $K_x = \lambda u.\langle S(\pi_1(u)), G_x u\rangle$ on $C_x = \langle 0, F_x\rangle$ instead, using the left part to keep track of which iteration we are on. If we write $I_x(n) = I_{(N \times N) \times N^k}(c_x, k_x, n)$, the iterator equations are:

- $I_x(0) = C_x = \langle 0, F_x\rangle$
- $I_x(S(n)) = K_x I_x(n) = \langle S(\pi_1(I_x(n))), G_x I_x(n)\rangle$

These are easier to understand if we split the expression into two parts - so that $I_x^1(n) = \pi_1(I_x(n))$, $I_x^2(n) = \pi_2(I_x(n))$. Then the above equations become

(i) $I_x^1(0) = 0, \quad I_x^1(S(n)) = S(I_x^1(n)),$

(ii) $I_x^2(0) = F_x, \quad I_x^2(S(n)) = G_x\langle I_x^1(n), I_x^2(n)\rangle.$

Since we have (for any x) $I_x^1(\#0) = \#0$ and $I_x^1(\#(n+1)) = S(L_x(\#n))$ simple induction on n shows that $I_x^1(\#n) = \#n$. Thus $I_x^2(\#(n+1)) = G_x\langle \#n, I_x^2(\#n)\rangle$ which is starting to look a lot like the iteration we want. It is now simple to engineer the term H so that

$$H\langle n, x\rangle = I_x^2(n)$$

We then write $\#a = \langle ...\langle \#a_1, \#a_2\rangle, ..., \#a_k\rangle$ and prove that H represents h by induction, starting with the base case:

$$H\langle \#0, \#a\rangle = I_{\#a}^2(\#0) = F_{\#a} = F\#a = \#f(a_1, a_2, ..., a_k).$$

Now we assume that $H\langle \#n, \#a\rangle = \#h(n, a_1, ..., a_k)$, and get

$$\begin{aligned} H\langle \#(n+1), \#a\rangle &= I_{\#a}^2(\#(n+1)) = G_{\#a}\langle \#n, I_x^2(\#n)\rangle \\ =&G\langle\langle \#n, H\langle n, x\rangle\rangle, \#a\rangle = \#g(n, h(n, a_1, ..., a_k), a_1, ..., a_k). \end{aligned}$$

Thus we have shown that H represents h by induction and completed the proof. □

It is worth noting that (i) implies $I^1_x(n) = n$ for a variable $n : N$ if we use term induction, and that I^2_x is thus the unique (up to equivalence) term fulfilling (ii). Furthermore the proof gives a term $R_{F,G}$, labeled H in the proof, which is the unique term fulfilling

$$R_{F,G}\langle 0, x\rangle = Fx \quad R_{F,G}\langle S(n), x\rangle = G\langle\langle n, R_{F,G}\langle n, x\rangle\rangle, x\rangle.$$

We thus see that primitive recursion and composition are both possible do to *within* a λ-calculus with natural number type and term induction. That is, you do not just get terms that represent primitive recursive functions but terms that exactly mirror their equational definition for variables $n : N$. We will also take the time to state a simple lemma we will use later building on this result.

Lemma 4.2.5. *If $P, Q : N \Rightarrow N$ are terms of a typed λ-calculus with natural number type and term induction so that $P0 = Q0$ and $PS(n) = QS(n)$ for all n, then $P = Q$.*

Proof. Let $x : N \times N$ be a variable and define $G = \lambda x.PS(\pi_1(x)) : (N \times N) \Rightarrow N$. Then $R_{P0,G} : N \Rightarrow N$ is the *unique* term with the property

$$R_{P0,G}0 = P0, \quad R_{P0,G}S(n) = G\langle n, R_{P0,G}\langle n, x\rangle\rangle = PS(n).$$

Since both P and Q also fulfill this equation we must have $R_{P0,G} = P = Q$. □

Finally we get around to stating the main result of this section.

Theorem 4.2.6. *In any CCC with strong natural number object N, $N \times N \cong N$.*

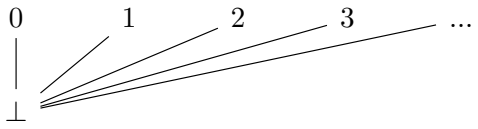
Proof. The cantor pairing function $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by

$$\pi(k_1, k_2) := \frac{1}{2}(k_1 + k_2)(k_1 + k_2 + 1) + k_2$$

is bijective and primitive recursive, as shown in [3, prop. 5.4]. The inverses $s_1, s_2 : \mathbb{N} \to \mathbb{N}$, while slightly more difficult to state, are also primitive recursive. The proof uses only the primitive recursive definitions of the functions, and they imply that $\pi(s_1(n), s_2(n)) = n$ and for any 2-tuple a, $(s_1(\pi(a)), s_2(\pi(a))) = a$. Since we have a strong NNO, which is equivalent to a λ-calculus with term induction, we can exactly mirror these definitions within the category and get terms P, S_1, S_2 so that P and $\lambda n.\langle S_1 n, S_2 n\rangle$ are in fact the two directions of an isomorphism $N \times N \cong N$. □

A Cartesian closed category with a weak NNO N is not guaranteed to have this property. As an example we look back at the CCC ω-**cpo**, which was defined in 2.4.2.

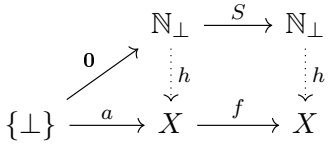
Given any set S we write $S_\perp$ for the $\omega - cpo$ with underlying set $S \cup \{\perp\}$ where $a \leq b$ if and only if $a = b$ or $a = \perp$. It is essentially S as an unordered set with $\perp$ inserted as a least element. For instance we have $\mathbb{N}_\perp$, presented here as a Hasse-diagram where there is a line *upward* from a to b if $a < b$ and there is no z such that $a < z < b$:



We can now show that $\omega - \mathbf{cpo}$ in fact has a weak NNO.

Proposition 4.2.7. *$\mathbb{N}_\perp$ with arrows $\boldsymbol{0} : \{\perp\} \to \mathbb{N}$ which maps $\perp \mapsto 0$ and $S : \mathbb{N}_\perp \to \mathbb{N}_\perp$ which maps $n \mapsto n+1$ and $\perp \mapsto \perp$, is a weak NNO in $\omega - \boldsymbol{cpo}$.*

Proof. Since there are no strictly increasing infinite sequences in $\mathbb{N}$ ω-completeness will always hold, and clearly $\perp$ is a least element. Furthermore, $\mathbf{0}$ and S are both ω-continuous since they are monotone and there are no tricky suprema to preserve. Given any X, f, a as in 4.1.1 we can let $h(\perp_N) = \perp_X$ and $h(n) = f(f(f(...a...)))$ repeated n times, completing the commuting diagram

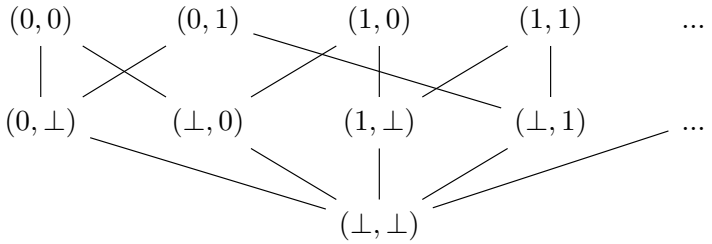


so $\mathbb{N}_\perp$ is a weak NNO. However, there will be many cases where we can map $\perp_N$ to something other than the given least element in the other set. It might for instance have multiple least elements we could pick from. This means that in general h will not be unique, so $\mathbb{N}_\perp$ is not a strong NNO. □

Finally we get to the point of all this - providing an exception to the rule $N \times N \cong N$.

Proposition 4.2.8. *In the category ω-**cpo**, $\mathbb{N}_\perp \times \mathbb{N}_\perp \ncong \mathbb{N}_\perp$*

Proof. If we look at a Hasse-diagram of some select elements of $\mathbb{N}_\perp \times \mathbb{N}_\perp$



we can notice that the height of the diagram is now 3 instead of 2. If we imagine an isomorphism $f : \mathbb{N}_\perp \times \mathbb{N}_\perp \to \mathbb{N}_\perp$, it would have to map e.g. the sequence $(\perp, \perp) \leq$

$(\perp, 0) \leq (0, 0)$ to some sequence $a \leq b \leq c$, but since there are no strictly increasing sequences of three elements in $\mathbb{N}_\perp$ we would have $a = b$ or $b = c$, contradicting that f is an isomorphism. □

Reflecting on this we realize that there must be, and indeed is, a realization P of the cantor pairing function in this category, since only a weak NNO is required for that. It maps all the pairs of natural number elements of $\mathbb{N}_\perp \times \mathbb{N}_\perp$ to the natural number elements of $\mathbb{N}_\perp$ bijectively. However, in $\omega - \mathbf{cpo}$ this is not enough, it would also have to map all the pairs which include $\perp$ in order to be an isomorphism, which we have shown to be impossible.

4.3 Toward a decision procedure for N-objects

We will now look closer at the implications of $N \times N \cong N$ in CCC:s with strong natural number objects.

Definition 4.3.1. An *N-object* is an object which is constructed from just 1 and N as basic objects, using products and exponentials.

This allows us to prove a very general valid isomorphism.

Theorem 4.3.2. *For any N-object A in any CCC with a strong NNO, $A \times A \cong A$.*

Proof. In remark 3.2.4 we saw that $nf(A)$ will either be 1 or have no occurrences of 1 and that only basic objects will be in the 'base' of exponentials in A. Combining these we find that either $A \times A \cong 1 \times 1 \cong 1$, or all exponential object-parts of A are of the form N^X for some X. In general we then have

$$A = N^{X_1} \times N^{X_2} \times ... \times N^{X_n} \times N \times ... \times N.$$

We then remember that for any X rule (iv) of $Th^1_{\times T}$ gives that $N^X \times N^X \cong (N \times N)^X \cong N^X$. This means, using commutativity and associativity to rearrange the product

$$A \times A \cong (N^{X_1} \times N^{X_1}) \times (N^{X_2} \times N^{X_2}) \times ... \times (N^{X_n} \times N^{X_n}) \times (N \times N) \times ... \times (N \times N)$$
$$\cong N^{X_1} \times N^{X_2} \times ... \times N^{X_n} \times N \times ... \times N = A$$

□

On the other hand, we can use **Set** to demonstrate some isomorphisms which are not valid.

Proposition 4.3.3. *If A, B are N-objects whose normal forms have different exponential depth, then they are not isomorphic in all CCC:s with strong NNO:s.*

Proof. **Set** with $\mathbb{N}$ is a CCC with a strong NNO. We will show that if A, B are on normal form and $\mathscr{D}(A) > \mathscr{D}(B)$, then also $\text{card}(A) > \text{card}(B)$. This will be done by induction on $\mathscr{D}(A)$, and we will make heavy use of the fact that any N-objects A, B on normal form are either infinite or isomorphic to 1, which means

$$\text{card}(A \times B) = \max(\text{card}(A), \text{card}(B)), \tag{2}$$

will always apply (assuming axiom of choice).

- First assume $\mathscr{D}(A) = 1$, $\mathscr{D}(B) = 0$. Then B is a finite product $N \times ... \times N$, and by (2) $\text{card}(B) = \text{card}(N)$. A on the other hand is a finite product of N^N and N. Notice that Cantor's theorem gives that for any object A $\text{card}(A) < \text{card}(N^A)$, so (2) gives $\text{card}(A) = \text{card}(N^N) > \text{card}(N)$.
- Now assume that for any A', B' with $\mathscr{D}(A') < k, \mathscr{D}(B') < \mathscr{D}(A')$ it is true that $\text{card}(A') > \text{card}(B')$. Then assume $\mathscr{D}(A) = k$, so that in general A is a product $A_1 \times ... \times A_n$ where we can write any A_i on the form $N^{A_i'}$ if we let $A_i' = 1$ when $A_i = N$. If $\text{card}(A_j') \geq \text{card}(A_i')$ for all i then by the induction hypothesis $\mathscr{D}(A_j) = k - 1$, and by (2) we have $\text{card}(A) = \text{card}(N^{A_j'})$.

  By a similar argument we show that for any B with $\mathscr{D}(B) = l < k$ we have $\text{card}(B) = \text{card}(N^{B_i'})$ with $\mathscr{D}(B_i') = l - 1 < k - 1$. So by the induction hypothesis $\text{card}(A_j') > \text{card}(B_i')$ and thus $\text{card}(A) > \text{card}(B)$.

Since two sets with different cardinality by definition cannot be isomorphic, $A \ncong B$ if they have different exponential depth. □

Thus we are only uncertain about which isomorphisms are valid between two objects of the same exponential depth. The earlier equation (2) can quite easily be used to show that in **Set** any two N-objects with the same exponential depth are isomorphic, and we already know that (2) works in any CCC with strong NNO for products $A \times A$. The problem is determining whether it also holds when the two factors are different. We can in fact prove a special case of this in the general setting.

Proposition 4.3.4. *In any CCC with strong natural number object N we have $N^N \times N \cong N^N$.*

Proof. Given any term $\langle F, n\rangle : N^N \times N$ and variable $k : N$ we can construct the term

$$P(F, n) = \lambda k.(n \cdot (S(0) \dot{-} \text{sgn}(k)) + f(k \dot{-} S(0)) \cdot \text{sgn}(k)) : N^N$$

using terms for $+, \dot{-}, \text{sgn}, \cdot$ as in remark 4.2.2, defined exactly like the corresponding primitive recursive functions, where of course $a + b$ is here short for $+\langle a, b\rangle$ etc.

The term F_n thus represents the function

$$f_n(k) = \begin{cases} n, & \text{if } k = 0 \\ f(k-1), & \text{otherwise} \end{cases}$$

where f is the function represented by F. Finally let

$$P^* = \lambda x.P(\pi_1(x), \pi_2(x)) : (N^N \times N) \Rightarrow N^N,$$

where $x : N^N \times N$.

For the other direction, given a term $G : N^N$, we can construct $\langle Q_1(G), Q_2(G)\rangle : N^N \times N$ by letting $Q_1(G) = \lambda k.G(S(k))$ and $Q_2(G) = G0$. Then let

$$Q^* = \lambda y.\langle Q_1(y), Q_2(y)\rangle : N^N \Rightarrow (N^N \times N),$$

where $y : N^N$.

We must now show that indeed P^*, Q^* provide the two directions of an isomorphism.

$$\begin{aligned} &\lambda y.P^*(Q^*y) = \lambda y.P(\pi_1(\langle Q_1(y), Q_2(y)\rangle), \pi_2(\langle Q_1(y), Q_2(y)\rangle)) \\ =&\lambda y.P(Q_1(y), Q_2(y)) = \lambda y.P(\lambda k_1.y(S(k_1)), y0) \\ =&\lambda y.\lambda k_2.(y0 \cdot (S(0)\dot{-}\mathrm{sgn}(k_2)) + (\lambda k_1.y(S(k_1))(k_2\dot{-}S(0))) \cdot \mathrm{sgn}(k_2)) \\ =&\lambda y.\lambda k_2.(y0 \cdot (S(0)\dot{-}\mathrm{sgn}(k_2)) + y(S(k_2\dot{-}S(0))) \cdot \mathrm{sgn}(k_2)) \end{aligned} \tag{3}$$

Now we notice that

$$\begin{aligned} &\lambda k_2.(y0 \cdot (S(0)\dot{-}\mathrm{sgn}(k_2)) + y(S(k_2\dot{-}S(0))) \cdot \mathrm{sgn}(k_2))0 \\ =&y0 \cdot (S(0)\dot{-}\mathrm{sgn}(0)) + y(S(0\dot{-}S(0))) \cdot \mathrm{sgn}(0) \\ =&y0 \cdot (S(1)\dot{-}0) + y(S(0)) \cdot 0 \\ =&y0 \cdot (S(0)) \\ =&y0 \end{aligned}$$

and

$$\begin{aligned} &\lambda k_2.(y0 \cdot (S(0)\dot{-}\mathrm{sgn}(k_2)) + y(S(k_2\dot{-}S(0))) \cdot \mathrm{sgn}(k_2))S(n) \\ =&y0 \cdot (S(0)\dot{-}\mathrm{sgn}(S(n))) + y(S(S(n)\dot{-}S(0))) \cdot \mathrm{sgn}(S(n)) \\ =&y0 \cdot (S(0\dot{-}S(0)) + y(S(n)) \cdot S(0) \\ =&y0 \cdot 0 + y(S(n)) \\ =&yS(n) \end{aligned}$$

which shows by 4.2.5 that $\lambda k_2.(y\#0 \cdot (\#1\dot{-}\mathrm{sgn}(k_2)) + y(S(k_2\dot{-}\#1)) \cdot \mathrm{sgn}(k_2)) = y$, so that (3) is in fact equal to $\lambda y.y$. The other direction is similar. □

Beyond this however the tools of primitive recursive functions and term induction seems to be lacking, and this thesis will not show any more valid isomorphisms.

5 Final notes

In this thesis we have first shown that there is a sound and complete theory and decision procedure for determining which isomorphisms hold in all CCC:s. We have then tried to understand which further isomorphisms hold in all CCC:s with weak/strong NNO:s, where only limited progress has been made. While we have narrowed the space of uncertain isomorphisms in CCC:s with strong NNO:s from both directions we are far from completely determining them. There is reason to believe that in fact the strong result from **Set** will not hold in all cases, and that there can be objects with the same exponential depth which are not isomorphic. This is because showing, for instance, $N^{N^N} \times N \cong N^{N^N}$ would seem to require functions which can do case-splitting on functions. An example solution in the vein of 4.3.4 would be taking $(f : N^N \to N, n)$ to

$$f_n(g) = \begin{cases} n, & \text{if g is constant 0} \\ f(g-1), & \text{if g is some other constant function} \\ f(g), & \text{otherwise} \end{cases}$$

however there is no computable function which can check if g is constant, since that would involve checking every function value. In [5, Th. 2.7] they show that any representable function is computable, so the above could never work as a solution. This is of course not a proof that no isomorphism could be constructed, but it would require some other method.

Perhaps more likely is that there are indeed CCC:s with strong NNO:s which do not have these isomorphisms, and it would be very interesting to find examples of such.

Another question left open by this thesis is whether a modified version of theorem 4.3.2 requiring only that $A \times A \cong A$ if $\mathscr{D}(A) \geq k$ holds for weak NNO:s with some k. The reason this is left open is that the result does seem to hold in $\omega - \mathbf{cpo}$ for $k = 1$, even though we showed that it did not for $k = 0$. Finding examples of weak NNO:s where this does not hold for higher k or proving that for some k this is indeed a valid theorem would both be very interesting results.

References

[1] Steve Awodey. *Category Theory*. 2nd. New York, NY, USA: Oxford University Press, Inc., 2010. ISBN: 0199237182, 9780199237180.

[2] Kim Bruce, Roberto Di Cosmo, and Giuseppe Longo. "Provable Isomorphisms of Types". In: *Mathematical Structures in Computer Science* 2 (June 1992), pp. 231–247. DOI: 10.1017/S0960129500001444.

[3] S C. Russen. "Mathematical Logic: A Course with Exercises Part II: Recursion Theory, Gödel's Theorems, Set Theory, Model Theory by René Cori; Daniel Lascar; Donald H. Pelletier". In: *The Mathematical Gazette* 88 (Jan. 2004), pp. 187–188. DOI: 10.2307/3621399.

[4] Joseph Gil and Yoav Zibin. "Efficient algorithms for isomorphisms of simple types". In: *Mathematical Structures in Computer Science* 15 (Oct. 2005), pp. 917–957. DOI: 10.1145/640128.604146.

[5] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*. New York, NY, USA: Cambridge University Press, 1986. ISBN: 0-521-24665-2.

[6] S. V. Solov'ev. "The category of finite sets and Cartesian closed categories". In: *Journal of Soviet Mathematics* 22.3 (June 1983), pp. 1387–1400. ISSN: 1573-8795. DOI: 10.1007/BF01084396. URL: https://doi.org/10.1007/BF01084396.

Appendix I

The proof of 3.2.3 follows, which references the rules in definition 3.2.1.

Proof. Let $A \overset{*}{\rightarrowtail} B$ by applying rule (j) to an object-part at position p, and $A \overset{*}{\rightarrowtail} C$ by applying rule (k) to an object-part at position q. If p and q do not overlap, meaning neither form an initial segment of the other, then we can obviously apply rule (k) to B at position q and rule (j) to C at position p and get the same result, showing confluence. It becomes more interesting when the reductions overlap, so that p is an initial segment of q or vice versa. Without loss of generality we can assume p is an initial segment of q so that $q = (p, q_2)$. Then it will be sufficient to restrict our attention to the object-part at p, which we might call M, where $M \rightarrowtail M_j$ by rule (j) applied to the whole of M, and then let the other reduction (k) work on the object-part at position q_2 of M. We will go through cases according to the possible values of j, k and q_2.

- If (j)=(i) then $M = (D^C)^B$ for some object-schemes B, C, D. First we will illustrate an important point with an example - if q_2 is $(0, 1)$ then applying (k) to the object part at that position will simply be a reduction $M \overset{*}{\rightarrowtail} M_k = (D^{C'})^B$ where $C \rightarrowtail C'$ by (k). We can then apply (i) and get $M_k \rightarrowtail M^* = D^{C' \times B}$. We can also apply (k) to $M_j = D^{C \times B}$ at position $(1, 0)$ and get $M_j \overset{*}{\rightarrowtail} D^{C' \times B} = M^*$ showing confluence. This same process can done if q_2 has any of $(0, 0), (1)$ or $(0, 1)$ as an initial segment, since such reductions won't mess with the structure required to apply (i). This leaves two interesting cases.

  - If $q_2 = ()$ then (k) is also being applied to the whole of M, which is an exponential, so rules (i),(ii),(v) and (vi) could hope to apply. However, D^C could never equal $E \times F$ for any object-schemes E, F or equal 1, so (ii) and (vi) are out. Using the same rule at the same position is of course gonna give the same result so we can ignore (i). This leaves (k) =(v), and in order to apply it we need $B = 1$, giving us

    $$M = (D^C)^1 \rightarrowtail M_k = D^C \quad M \rightarrowtail M_j = D^{C \times 1}.$$

    These can both be reduced to $M^* = D^C$, by doing nothing in the case of M_k or applying rule (iii) at position (1) in the case of M_j.
  - If $q_2 = (0)$ then the object-part at the position (D^C) is again an exponential, however we cannot ignore (i),(ii) and (vi) this time.
    To apply rule (i) we need $D = E^F$ for some object-schemes E, F and we get

    $$M = ((E^F)^C)^B \overset{*}{\rightarrowtail} M_k = (E^{F \times C})^B \quad M \rightarrowtail M_j = (E^F)^{C \times B}.$$

    But both can be reduced to $M^* = E^{(F \times C) \times B}$, using rule (i) again and, when reducing M_j, an application of (vii).

To apply rule (ii) we need $D = E \times F$ and we get

$$M = ((E \times F)^C)^B \overset{*}{\rightarrowtail} M_k = (E^C \times F^C)^B \quad M \rightarrowtail M_j = (E \times F)^{C \times B}.$$

But both of these can be reduced to $M^* = E^{C \times B} \times F^{C \times B}$, by applying rule (ii) again in both cases and then, in the case of M_k, applying rule (i) at position (0) and position (1).
To apply rule (v) we need $C = 1$, and we get

$$M = (D^1)^B \overset{*}{\rightarrowtail} M_k = D^B \quad M \rightarrowtail M_j = D^{1 \times B}.$$

But both can be reduced to $M^* = D^B$ by applying rule (iv) to M_j at position (1).
Finally, to apply rule (vi) we need $D = 1$, and we get

$$M = (1^C)^B \overset{*}{\rightarrowtail} M_k = 1^B \quad M \rightarrowtail M_j = 1^{C \times B}.$$

But both can be reduced to $M^* = 1$ by applying (vi) again.

Thus we have shown confluence in every case when (j)=(i).

- If (j)=(ii), then $M = (D \times C)^B$ and once again any position that has $(0,0), (0,1)$ or (1) as an initial segment is uninteresting. This leaves two cases:
  - If $q_2 = ()$ then the object-part at that position is an exponential so once again we deal with rules (i),(ii),(v),(vi). We can ignore (ii) since it is the same rule at the same position, and since we will never have $D \times C = E^F$ nor $D \times C = 1$ we can also ignore (i) and (vi).
    The only remaining case is (v), and in order to apply it we need $B = 1$, giving us
    $$M = (D \times C)^1 \rightarrowtail M_k = D \times C \quad M \rightarrowtail M_j = D^1 \times C^1.$$
    But both can be reduced to $M^* = D \times C$ by applying rule (v) at positions (0) and (1) in M_j.
  - If $q_2 = (0)$ then the object-part at that position $(D \times C)$ is a product, meaning we can apply rules (iii),(iv),(vii).
    In order to apply (iii) we need $C = 1$, and we get
    $$M = (D \times 1)^B \overset{*}{\rightarrowtail} M_k = D^B \quad M \rightarrowtail M_j = D^B \times 1^B.$$
    But both can be reduced to $M^* = D^B$, where you reduce M_j by first applying rule (vi) at position (1), then applying rule (iii) at position ().
    (iv) is essentially the same as (iii), so we skip to (vii), which needs $C = E \times F$. We get
    $$M = (D \times (E \times F))^B \overset{*}{\rightarrowtail} M_k = ((D \times E) \times F)^B \quad M \rightarrowtail M_j = D^B \times (E \times F)^B.$$
    But both can be reduced to $M^* = (D^B \times E^B) \times F^B$, by applying (ii) twice to M_k and by reducing M_j through an application of (ii) at position (1) followed by an application of (vii).

Thus we have shown confluence in every case when (j)=(ii).

- If (j)=(iii) then $M = A \times 1$. No reduction can be applied to 1, and as before any position with initial segment (0) is uninteresting, so there is only one relevant position - namely ().

  The object-part at that position is a product, so rules (iii),(iv) and (vii) have a chance of being applied. As usual we can ignore (iii) since it is the same rule at the same position, and in the case of (iv) we need $M = 1 \times 1$ and both rules reduce to 1, so we have confluence. Finally (vii) would need $1 = B \times C$ which is impossible, so it cannot actually be applied. This leaves no other cases and we have confluence when (j)=(iii).

- If (j)=(iv) then $M = 1 \times A$, and as before the only interesting position is (), where the only interesting rule to apply is (vii), since (iii) requires $M = 1 \times 1$ which was already covered.

  In order to apply (vii) we need $A = B \times C$, which gives

  $$M = 1 \times (B \times C) \rightarrowtail M_k = (1 \times B) \times C \quad M \rightarrowtail M_j = B \times C.$$

  But both can be reduced to $M^* = B \times C$, by applying rule (iv) to position (0) of M_k.

  This covers all cases so we have confluence for (j)=(iv).

- If (j)=(v) then $M = A^1$, and the only interesting position is once again (). The object-part at that position is an exponential so rules (i),(ii),(v),(vi) could be applicable. As usual we ignore (v).

  To apply (i) we need $A = D^C$ for some object-schemes D and C, which gives $M = (D^C)^1$, a case that was already covered. The same is true of (ii) which requires $M = (D \times C)^1$.

  That leaves (vi) which requires $M = 1^1$ and both rules reduce it to 1, showing confluence for (j)=(v).

- If (j)=(vi) then $M = 1^A$ and the only interesting position is (). The object-part at that position is an exponential, but rules (i) and (ii) are both inapplicable since we can't have $1 = D^C$ or $1 = D \times C$ for any D, C. To apply rule (v) we need $M = 1^1$ which was already covered, and (vi) is fine since it is the same rule at the same position. This covers all cases and shows confluence for (j)=(vi).

- Finally, if (j) =(vii) then $M = A \times (B \times C)$, and the interesting positions are () and (1).

  - If $q_2 = ()$, then the object-part at the position (M) is a product and it might be possible to apply rules (iii),(iv) and (vii). We ignore (vii) as usual, (iii) is impossible since $B \times C \neq 1$, and (iv) requires $M = 1 \times (B \times C)$ which was already covered.

– If $q_2 = (1)$ then we have a product once again. This time however, all three product rules are applicable.
To apply (iii) we need $C = 1$ which gives

$$M = A \times (B \times 1) \overset{*}{\rightarrowtail} M_k = A \times B \quad M \rightarrowtail M_j = (A \times B) \times 1.$$

But both can be reduced to $M^* = A \times B$ by applying rule (iii) to M_j.
To apply (iv) we need $B = 1$ which gives

$$M = A \times (1 \times C) \overset{*}{\rightarrowtail} M_k = A \times C \quad M \rightarrowtail M_j = (A \times 1) \times C.$$

But both can be reduced to $M^* = A \times C$ by applying rule (iii) to M_j at position (0).
To apply (vii) we need $C = D \times E$, and we get

$$M = A \times (B \times (D \times E)) \overset{*}{\rightarrowtail} M_k = A \times ((B \times D) \times E) \quad M \rightarrowtail M_j = (A \times B) \times (D \times E).$$

But both can be reduced to $M^* = ((A \times B) \times D) \times E$ by further applications of rule (vii).

Thus we have confluence also for (j)=(vii).

This finally covers all cases and shows local confluence for all values of j, k and q_2. We have thus shown that even if the two reductions apply to overlapping positions p and (p, q_2) we can further reduce both B and C to $D = A[M^*/p]$, showing local confluence. □