An unbiased simply typed λ-calculus

Samuel Mimram
LIX, CNRS, École polytechnique, Institut Polytechnique de Paris, 91120 Palaiseau, France

Abstract

We introduce CCCaTT_1, a new type-theoretical definition of the simply-typed λ-calculus which is unbiased: it has a uniform family of generators and relations. As such, it unifies previously known formalisms such as combinatory logic or categorical combinators, and paves the way for higher-dimensional generalizations. The definition is based on a characterization of a class of simple types which are *contractible* in λ-calculus, in the sense that they contain a unique inhabitant modulo extensional equality. Our definition then consists in a dependent type theory which formally makes such types contractible, and we then observe that this is enough to derive the whole equational theory of simply typed λ-calculus. This approach is close to recently introduced type theoretical definitions of weak ω-categories. A proof assistant based on this work was implemented in order to prove some of the propositions developed here.

2012 ACM Subject Classification Theory of computation $\to$ Type theory

Keywords and phrases dependent type theory, lambda-calculus, combinators

Supplementary Material `https://github.com/smimram/cccatt`

Funding CoREACT ANR project (ANR-22-CE48-0015)

1 Introduction

The goal of this paper is to introduce CCCaTT_1, a new type-theoretic definition for simply-typed λ-calculus (STL) equipped with the extensional equality, where terms are identified up to $\beta\eta$-equivalence. It has this strange aspect that, contrarily to all the equivalent definitions we are aware of, it does not contain anything which vaguely resembles to a β-reduction or an η-expansion. This particularity should be sufficient to arouse your curiosity, but let us still provide some more motivations for this work.

Komori's conjecture. It can be observed that "most" λ-terms are the only ones in their most general type, up to extensional equality: in such a situation, we say that their most general type is *contractible*. For instance, the identity I and the first projection K are the only λ-terms of respective types $X \to X$ and $X \to Y \to X$. We could thus wonder whether all λ-terms could be obtained in this way. Of course, there are easy counter-examples to this: for instance, the Church numerals $2 \equiv \lambda fx.f(f(x))$ and $3 \equiv \lambda fx.f(f(f(x)))$ both have $(X \to X) \to X \to X$ as most general type. Observing that the previous type is not minimal (in the sense that it can be obtained from $Y \to Y$ by the substitution $Y := X \to X$), we can be lead to conjecture, as did Komori long before us [27, Problem 1.5], that minimally inhabited types are contractible: this would mean that minimal types generate, under type substitution, all inhabited types. This conjecture was shown not to hold [39, 48], but was quite influential because it was the motivation for a series of work providing necessary conditions to show that a type has at most one inhabitant [2, 3, 4, 11, 21, 28, 39, 41, 48]. We explain here that the conjecture can be fixed by also considering term substitutions: *the inhabitants of contractible types generate, under both type and term substitution, all simply typed λ-terms.* Note that our conjecture is more "constructive" than the one of Komori, since we shift from inhabitability to inhabitation proofs, i.e. λ-terms.

An unbiased λ-calculus. We provide here a definition of STL which is *unbiased*, in the sense that it does not a priori make the choice of a particular set of generators. This point of view is common in category theory where, for instance, one defines unbaised monoidal categories by taking n-ary tensors as generating operations, instead of only nullary and binary ones [32]. First, remark that the usual definitions of STL axiomatize it from a small number of operations: abstraction and application for λ-calculus [6], S, K (and sometimes I) and application for combinatory logic [20], a few generic morphisms, composition and pairing for categorical combinatory logic [13], and so on. Here, we provide a platform in which all of them can be recovered as subsystems, but which is still equivalent to STL. This is done by adopting a maximalist approach, and taking all the previous generators at once, and many more. The way we proceed is by providing one type theoretic rule which generates all those generators in a uniform way, as well as one other rule for generating the associated axioms. More precisely, those two rules respectively introduce an inhabitant for any contractible type and identify any two inhabitants of a contractible type: in other words, we simply formally make contractible types contractible. To be even more precise, we do not currently have a nice characterization of contractible types. Instead, we introduce a subclass of the contractible types, that we call *pasting types*, for which there is a simple definition which can efficiently be checked. Our definition actually formally make those contractible, which is enough for our purposes.

Weak higher cartesian closed categories. Another motivation comes from earlier work of ours, where we introduced CaTT [9, 17], a type-theoretic definition of weak ω-categories in the Grothendieck-Maltsiniotis variant [34], based on the idea that some collections of morphisms (the *pasting schemes*) should have a well-defined composition. We are looking for extensions of this to characterize categories with structure, such as (locally) cartesian closed categories: currently, the only known extension is the one of monoidal categories [8]. In order to start somewhere, we can take CaTT and "truncate" it to 1-categories: if we do so, we obtain an unbiased definition of 1-categories, which we can try to extend to a definition of cartesian closed 1-categories. This is precisely what the present paper does, and the style of type theoretic definition of STL we provide here draws strong inspiration from this point of view, although we cannot detail this here. Because our definition of STL is very generic, we have hope to extend it to higher dimensions and define weak cartesian closed categories.

Homotopy type theory. Another strong source of inspiration is homotopy type theory [49] (HOTT), from where originate the central properties for a type of being a proposition and contractible. Also, a motivation comes from one of the most important difficulties when providing semantics of HOTT in $(\infty, 1)$-toposes, the *coherence problem*, is due to the fact that many equalities (such as β-reduction) hold definitionally, whereas they only hold weakly in the semantics. This has been overcome by Shulman in [42] by showing that any topos can be suitably strictified, see also [14, 33]. Another approach could be to consider that definitional equality should be banned, and consider a type theory corresponding to locally cartesian closed categories with only propositional equality (it would have very nice theoretical properties, although it would be hell to use in practice). The present work is a first step in this direction.

Implementation. The type theory CCCaTT_1 is implemented in the form of an online proof assistant [35] whose source code is freely available [36]. All the derivations mentioned in the article have been mechanically checked with this tool.

Acknowledgment. We would like to warmly thank Thibaut Benjamin and Noam Zeilberger for the numerous discussions about the topics of this paper.

2 Contractible types in the simply typed λ-calculus

2.1 The simply-typed λ-calculus

We briefly recall here basic notations and properties for the λ-calculus. We refer to standard textbooks for details [18, 37, 45].

Intuitionistic logic. We suppose fixed a countable supply of *type variables*, which we denote as $X, Y, \ldots$ The *types*, or *formulas*, are generated by the grammar

$$A, B \quad ::= \quad X \quad | \quad A \to B$$

where X is an arbitrary type variable. A *context* Γ is a list of formulas; we sometimes write $\emptyset$ for the empty context. A *judgment* is a pair $\Gamma \vdash A$ consisting of a context Γ and a formula A. It is *provable* when it is derivable using the rules

$$\frac{}{\Gamma, A, \Delta \vdash A}\ (\mathrm{ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathrm{I}}) \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ (\to_{\mathrm{E}}) \qquad (1)$$

Terms. The λ-*terms* are generated by the grammar

$$t, u \quad ::= \quad x \quad | \quad \lambda x^A . t \quad | \quad t\, u$$

where x is a *variable* belonging to a fixed countable set. In a term of the form $\lambda x^A.t$, the variable x is *bound* in t. In the following, we consider terms up to α-*conversion*, i.e. renaming of bound variables. Given terms t and u and a variable x, we write $t[u/x]$ for the term obtained from t by substituting every occurrence of x by u, in a capture-avoiding way.

Typing. A *context* Γ is a list $x_1 : A_1, \ldots, x_n : A_n$ of pairs consisting of a variable x_i and a type A_i. The *domain* of such a context is the set of variables $\mathrm{dom}(\Gamma) \equiv \{x_1, \ldots, x_n\}$. A *typing judgment* $\Gamma \vdash t : A$ is a triple consisting of a context Γ, a term t and a type A. It is *valid* when it is derivable using the rules:

$$\frac{}{\Gamma, x : A, \Delta \vdash x : A}\ (\mathrm{ax}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A . t : A {\to} B}\ (\to_{\mathrm{I}}) \qquad \frac{\Gamma \vdash t : A {\to} B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\ (\to_{\mathrm{E}}) \quad (2)$$

The *principal premise* is the leftmost one. We say that t *has type* A in a context Γ, or that t *has type* $\Gamma \vdash A$, when $\Gamma \vdash t : A$ is derivable.

Curry-Howard correspondence. The Curry-Howard correspondence [23, 37, 45] is the observation that λ-terms encode proofs in the sense that

▶ **Proposition 1.** *Given a context Γ, there is a bijective correspondence between*
- *proofs of $\Gamma \vdash A$ according to (1),*
- *typing derivations of $\Gamma \vdash t : A$ for some λ-term t according to (2).*

The correspondence also extends to dynamics of proofs, based on the following results, which are implicitly used below:

▶ **Lemma 2.** *The following rules are admissible, where we suppose $x \notin \mathrm{dom}(\Gamma)$ for the second rule:*

$$\frac{\Gamma, x : A, \Delta \vdash t : B \qquad \Gamma, \Delta \vdash u : B}{\Gamma, \Delta \vdash t[u/x] : B} \qquad\qquad \frac{\Gamma \vdash t : A \to B}{\Gamma \vdash \lambda x^A . t\, x : B}$$

2.2 Extensional equivalence

In the following, we define various typed relations $\sim$ between λ-terms (β-reduction, η-expansion, etc.): such a relation is subset of 4-uples $\Gamma \vdash t \sim u : A$ consisting of a context Γ, two terms t and u, and a type A, and is defined inductively by inference rules. For every such a relation, the fact that $\Gamma \vdash t \sim u : A$ is derivable can be checked to imply that both $\Gamma \vdash t : A$ and $\Gamma \vdash u : A$ are derivable.

Reduction. The *β-reduction* is the typed relation $\rightsquigarrow_\beta$ satisfying the rule on the left below

$$\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda x^A.t)u \rightsquigarrow_\beta t[u/x] : B}\,(\beta) \qquad\qquad \frac{\Gamma \vdash t : A \to B}{\Gamma \vdash t \rightsquigarrow_\eta \lambda x^A.t\,x : A \to B}\,(\eta) \tag{3}$$

which is moreover compatible with context extension, i.e. satisfies

$$\frac{\Gamma, x : A \vdash t \rightsquigarrow_\beta t' : B}{\Gamma \vdash \lambda x^A.t \rightsquigarrow_\beta \lambda x^A.t' : A \to B} \qquad \frac{\Gamma \vdash t \rightsquigarrow_\beta t' : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u \rightsquigarrow_\beta t'u : B}$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u \rightsquigarrow_\beta u' : A}{\Gamma \vdash t\,u \rightsquigarrow_\beta tu' : B}$$

The *η-expansion* is the typed relation $\rightsquigarrow_\eta$ satisfying the rule on the right of (3), and which is moreover compatible with context extension. The $\beta\eta$-relation $\rightsquigarrow_{\beta\eta}$ is the union of the two above relations. The *extensional equivalence* $\overset{\text{ext}}{=}$ is the smallest typed equivalence relation containing both β-reduction and η-expansion. This relation is a congruence.

Long normal forms. A term is a *β-normal form* when it is not the source of any β-reduction. It is a *long* β-normal form when it is β-normal and any η-expansion gives rise to a non-β-normal term, i.e. each variable is applied to as many arguments as its type allows. In the following, since we are mostly interested in λ-terms up to extensional equivalence, we can restrict to terms in long β-normal form:

▶ **Proposition 3.** *Any term is extensionally equivalent to a long β-normal form.*

Proof. It is well-known that β-reduction is weakly normalizing for simply-typed λ-terms [18, Chapter 4]. The number of η-expansions one can perform while remaining in β-normal form is bounded by the number of arrows of types of subterms which are not application position. ◀

▶ Remark 4. The proposition shows that every extensional equivalence class of λ-terms contains at least one long β-normal form. With some more work, it can furthermore be shown that it contains exactly one long β-normal form, but this will not be needed here.

The terms in long β-normal form can be characterized explicitly.

▶ **Proposition 5.** *The λ-terms in long β-normal form are those generated by the two rules*

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A.t : A \to B}\,(\to_{\text{I}}^{\text{nl}})$$

and

$$\frac{x : A_1 \to \cdots \to A_n \to X \in \Gamma \qquad \Gamma \vdash t_1 : A_1 \qquad \ldots \qquad \Gamma \vdash t_n : A_n}{\Gamma \vdash x\,t_1 \ldots t_n : X}\,(\to_{\text{E}}^{\text{nl}})$$

where X denotes a type variable.

Through the Curry-Howard correspondence (Proposition 1), the terms in long β-normal form correspond to proofs in the following logical system with two rules:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathrm{I}}^{\mathrm{nl}}) \qquad \frac{A_1 \to \cdots \to A_n \to X \in \Gamma \qquad \Gamma \vdash A_1 \qquad \ldots \qquad \Gamma \vdash A_n}{\Gamma \vdash X}\ (\to_{\mathrm{E}}^{\mathrm{nl}}) \tag{4}$$

which we call here *normal logic* (because it is the logic of normal forms).

Based on the above observations, various algorithms have been proposed in order to describe all the λ-terms of a given type in (long) β-normal form [7, 47, 50], see [19, Chapter 8] for a general overview. Our result below can be thought of as a variant of those, specialized to recognize contractible types.

2.3 Contractible types

The following definitions are inspired from homotopy type theory [49, Chapter 3], by taking the point of view that any type A can conceptually be thought of as a space with the λ-terms of type A as points, in such a way that any two extensionally equivalent λ-terms are related by a path. To complete this picture, one should also specify what the are higher-dimensional paths, but this is left for future work (and the present paper can be considered as a first step in this direction).

Contractible types. A sequent $\Gamma \vdash A$ is a *proposition* if any two terms of this type are extensionally equivalent, i.e. the rule

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \overset{\mathrm{ext}}{=} u : A}$$

is admissible. It is *contractible* if it is a proposition which is inhabited, i.e. there is a term of type A in the context Γ. By extension, we say that a type A is a proposition (resp. contractible) when $\vdash A$ is a proposition (resp. contractible).

▶ **Example 6.** Consider the context $X : \star, Y : \star$. The types $X \to X$ and $X \to Y \to X$ are contractible (its only inhabitants are respectively the identity and the first projection). The type $X \to Y$ is a non-contractible proposition (there is no term of this type). The types $X \to X \to X$ and $(X \to X) \to X \to X$ are not propositions (they are respectively inhabited by the two projections and the Church numerals).

Deterministic types. We now present a condition on types which implies contractibility. By Proposition 3, in order to ensure that a type A is contractible, it is enough to ensure that it is the type of at most one long β-normal term, i.e. that it admits at most one proof in normal logic. This motivates considering the following definition of determinism, introduced in [11, Definition 5.4], although in the formalism of proof trees. A *partial* proof is one where some branches have not been completed, i.e., more formally, a proof using the additional derivation rule

$$\frac{}{\Gamma \vdash A}\ (\maltese)$$

A partial proof π is a *prefix* of another proof π' when π' can be obtained from π by replacing rules ($\maltese$) by partial proofs, what we write $\pi \sqsubseteq \pi'$. This equips partial proofs of $\Gamma \vdash A$ with a partial order, where (non-partial) proofs are maximal elements. A type A in a context Γ is *deterministic* when the set of partial proofs of $\Gamma \vdash A$ (wrt the rules of Proposition 5) is directed wrt $\sqsubseteq$, by which we mean that for every partial proofs π_1 and π_2 of $\Gamma \vdash A$ there is a partial proof π such that $\pi_1 \sqsubseteq \pi$ and $\pi_2 \sqsubseteq \pi$. The following is immediate from the definition [11, Corollary 5.6]:

▶ **Proposition 7.** *Any deterministic type is a proposition. Thus, any inhabited deterministic type is contractible.*

When looking for a proof of an arbitrary sequent $\Gamma \vdash A$ in normal logic, there are two possibilities:

1. either A is an arrow type $A \equiv A_1 \to A_2$ and the rule $(\to_{\mathrm{I}}^{\mathrm{nl}})$ is the only one to apply, in one possible way,
2. otherwise A is a variable $A \equiv X$ and the rule $(\to_{\mathrm{E}}^{\mathrm{nl}})$ is the only one that can apply, possibly in multiple ways.

In the second case, there are as many ways to apply the rule $(\to_{\mathrm{E}}^{\mathrm{nl}})$ as there are formulas with target X in the context. Here, given a formula A, its *target* $\mathrm{tgt}(A)$ is the rightmost type variable, which can be computed inductively by $\mathrm{tgt}(X) \equiv X$ and $\mathrm{tgt}(A \to B) \equiv \mathrm{tgt}(B)$. A type is thus deterministic precisely when, whenever looking for a proof of it in normal logic, whenever we are trying to prove a variable X, there is exactly one formula in the context with X as target. Informally, each "consumer" (a positive instance of a variable X that we are trying to prove) should have at most one "producer" (a negative instance of the same variable, which is available in the context).

▶ **Example 8.** The formula $X \to X \to X$ is provable but not deterministic; in fact, we have seen that it is not contractible since there are two proofs, which correspond to the following typing derivations:

$$\dfrac{\dfrac{}{x : X, y : X \vdash x : X}\,(\to_{\mathrm{E}}^{\mathrm{nl}})}{\vdash \lambda x^X y^X . x : X \to X \to X}\,(\to_{\mathrm{I}}^{\mathrm{nl}}) \qquad \dfrac{\dfrac{}{x : X, y : X \vdash y : X}\,(\to_{\mathrm{E}}^{\mathrm{nl}})}{\vdash \lambda x^X y^X . y : X \to X \to X}\,(\to_{\mathrm{I}}^{\mathrm{nl}})$$

(note that when applying $(\to_{\mathrm{E}}^{\mathrm{nl}})$ there are two formulas with target X in the context). The formula $X \to X$ is deterministic and provable, and thus contractible. The formula $(X \to Y) \to Y \to Y$ is not deterministic but contractible (and thus a proposition). The formula $Y \to X$ is deterministic but not provable. The formula $(X \to Y) \to (X \to Y) \to Y$ is not deterministic and not provable (and thus a proposition).

Other sufficient coherence conditions. For the sake of completeness let us recall other conditions which have been introduced in literature, which imply that a type is a proposition. A formula is *balanced* when no variable occurs more than once (there are variations where we suppose that, when a variable occurs twice, the two instances have different polarity), and *negatively non-duplicated* if every variable has at most one negative occurrence in it. It has been shown that every balanced formula is a proposition [5, 38, 39, 48], as well as every negatively non-duplicated formula [4, 10]. Another closely related condition is provability with *non-prime contraction* [2]. In fact, all those classes are included in the class of deterministic formulas, and the inclusion is strict [11, Section 5.1]. Finally, we note that, by Example 8, the class of deterministic formulas is strictly included in the class of propositions for which, to the best of our knowledge, there is no known characterization.

▶ **Open question 9.** *Provide a characterization of types which are propositions, as wells as those which are contractible.*

3 An unbiased type theory for simply-typed λ-calculus

3.1 Pasting types

Toward contractible types. Our aim is to identify a reasonably large subset of contractible types, which admits a nice and simple formalization. A natural candidate for this is to start from the notion of deterministic type, and restrict it so that we ensure provability. The discussion after Proposition 7 suggests that, in order to make sure that a formula is deterministic and inhabited, one can look for a normal proof of it, making sure that when we apply a rule $(\to_{\mathrm{E}}^{\mathrm{nl}})$ it was the only possible way of applying it, i.e. the goal variable X is the target of exactly one formula in the context (we fail if this is not the case). This method is not satisfactory yet because the proof search might not terminate. For instance, on the formula $(X \to X) \to X$, we will try to construct the "infinite" proof tree shown on the right, which informally corresponds the "infinite" λ-term $\lambda f^{X\to X}.f(f(f\ldots))$. In order to avoid such situations, it can be noted that, because we are looking for deterministic formulas, the proof search will loop if and only if we are trying to prove the same type variable twice in a branch of the proof tree (as in the above example). This suggests introducing a new variant of the proof system that we now describe.

$$\dfrac{\dfrac{\dfrac{\vdots}{X\to X\vdash X}\,(\to_{\mathrm{E}}^{\mathrm{nl}})}{X\to X\vdash X}\,(\to_{\mathrm{E}}^{\mathrm{nl}})}{\vdash (X\to X)\to X}\,(\to_{\mathrm{I}}^{\mathrm{nl}})$$

Pasting types. We consider sequents of the form $\Theta;\Gamma\vdash A$ where Θ is a list of type variables, Γ is a context (a list of formulas) and A is a formula. Here, Θ maintains the list of targets of formulas in the context Γ, and will be used in order to make sure that no two such formulas have the same target variable. Moreover, in order to avoid the "infinite" proof trees described above, when we use a formula from the context Γ as principal premise of a rule $(\to_{\mathrm{E}}^{\mathrm{nl}})$, we remove it from the context (but we do not remove its target variable from Θ): namely, if we have to use it twice in a branch, this means that we are trying to prove the same type variable twice and thus that the proof search will never end, as explained above. We thus propose a system with the following two rules:

$$\dfrac{\Theta,\mathrm{tgt}(A);\Gamma,A\vdash B}{\Theta;\Gamma\vdash A\to B}\,(\to_{\mathrm{I}}^{\mathrm{ps}}) \qquad \dfrac{\Theta;\Gamma,\Delta\vdash A_1 \quad \ldots \quad \Theta;\Gamma,\Delta\vdash A_n}{\Theta;\Gamma,A_1\to\cdots\to A_n\to X,\Delta\vdash X}\,(\to_{\mathrm{E}}^{\mathrm{ps}}) \tag{5}$$

In the rule $(\to_{\mathrm{I}}^{\mathrm{ps}})$, we have the following important side condition: we always suppose that the variable $\mathrm{tgt}(A)$ is not already present in Θ. A type A is a *pasting type*, what we write $\vdash_{\mathsf{ps}} A$, when the sequent $;\vdash A$ (with both Θ and Γ empty) is derivable with (5).

▶ **Example 10.** The types $X \to X$, $(X\to Y)\to X\to Y$ and $X\to Y\to X$ are pasting:

$$\dfrac{\dfrac{}{X;X\vdash X}\,(\to_{\mathrm{E}}^{\mathrm{ps}})}{;\vdash X\to X}\,(\to_{\mathrm{I}}^{\mathrm{ps}}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{Y,X;X\vdash X}\,(\to_{\mathrm{E}}^{\mathrm{ps}})}{Y,X;X\to Y,X\vdash Y}\,(\to_{\mathrm{E}}^{\mathrm{ps}})}{Y;X\to Y\vdash X\to Y}\,(\to_{\mathrm{I}}^{\mathrm{ps}})}{;\vdash (X\to Y)\to X\to Y}\,(\to_{\mathrm{I}}^{\mathrm{ps}}) \qquad \dfrac{\dfrac{\dfrac{}{X,Y;X,Y\vdash X}\,(\to_{\mathrm{E}}^{\mathrm{ps}})}{X;X\vdash Y\to X}\,(\to_{\mathrm{I}}^{\mathrm{ps}})}{;\vdash X\to Y\to X}\,(\to_{\mathrm{I}}^{\mathrm{ps}})$$

The type $(X\to X)\to X\to X$ is not pasting.

Our system has been designed so that proof search is extremely easy: at most one rule applies, and moreover no infinite proof tree can be constructed (the depth is bounded by the number of subformulas of the original sequent).

▶ **Proposition 11.** *Proof search in the system (5) is deterministic (when starting from an empty context) and terminating.*

By erasing the context Θ from a derivation using the rules (5) one obtains a normal proof, which is moreover pasting by construction and thus a proposition.

▶ **Proposition 12.** *Any pasting type is contractible.*

▶ Remark 13. The rules of pasting types ensure that no two formulas have the same target variable in a context, whereas the condition for deterministic formulas only requires this for variables which are actually being proved. We thus note that being pasting is thus slightly more restrictive than being deterministic (and slightly more general than being negatively non-duplicating), although this will not play a significant role in the following. For instance, the formula $X \to X \to Y \to Y$ is not pasting but is deterministic (because X is provided twice but never used), and the formula $((X \to X \to Y) \to Z) \to W \to W$ is pasting but not non-negatively duplicating.

Given a context $\Gamma \equiv A_1, \ldots, A_n$ and a type A, we write $\Gamma \to A$ as a shorthand for $A_1 \to \cdots \to A_n \to A$. The notion of pasting type can be generalized to take contexts in account by stating that A is a pasting type in context Γ, what we write $\Gamma \vdash_{\mathsf{ps}} A$ when we have $\vdash_{\mathsf{ps}} \Gamma \to A$ (according to the above definition). It is easily shown that terms of type $\Gamma \vdash A$ are in bijection with terms of type $\vdash \Gamma \to A$, up to extensional equivalence, and thus:

▶ **Proposition 14.** *Any pasting type in arbitrary context Γ is contractible.*

▶ **Example 15.** We have $X \to Y, X \vdash_{\mathsf{ps}} Y$. Namely, checking this amounts, by definition, to check that $(X \to Y) \to X \to Y$ is contractible, which is indeed the case by Example 10.

3.2 CCaTT$_1$

We now introduce the main type theory of this paper, that we call CCaTT_1 because, as we will see, it is the type theory for closed categories (the index 1 is here to indicate that we plan to generalize it from 1-categories to higher categories). It involves dependent types and is strongly inspired of the type theory CaTT for weak higher categories [17].

The type theory. We suppose fixed an infinite supply of variables $x, y, \ldots$ The *terms* are defined by the grammar

$$t, u \quad ::= \quad x \quad | \quad t \to u \quad | \quad \mathsf{coh}_{\Gamma \vdash A}[\sigma]$$

where Γ is a context, A is a type and σ is a substitution. We write $\mathrm{FV}(t)$ for the set of free variables of a term t:

$$\mathrm{FV}(x) \equiv \{x\} \qquad \mathrm{FV}(t \to u) \equiv \mathrm{FV}(t) \cup \mathrm{FV}(u) \qquad \mathrm{FV}(\mathsf{coh}_{\Gamma \vdash A}[\sigma]) \equiv \mathrm{FV}(\sigma)$$

The *substitutions* are defined by the grammar

$$\sigma \quad ::= \quad \langle\rangle \quad | \quad \langle \sigma, t \rangle$$

where t is a term. A substitution σ is thus a list of terms $\langle t_1, \ldots, t_n \rangle$, and its free variables are $\mathrm{FV}(\sigma) \equiv \bigcup_i \mathrm{FV}(t_i)$. The *types* are defined by the grammar

$$A \quad ::= \quad \star \quad | \quad \mathrm{El}(t)$$

A *context* Γ is a list $x_1 : A_1, \ldots, x_n : A_n$ of pairs consisting of a variable x_i and type A_i. We write $\mathrm{dom}(\Gamma) \equiv \{x_1, \ldots, x_n\}$ for its domain and $|\Gamma| \equiv n$ for its size.

The rules of our type theory are presented in Figure 1, and we now explain their meaning. As traditional in dependent type theory, it involves five kinds of judgments:

- $\Gamma \vdash$ means that the context Γ is well-formed,
- $\Gamma \vdash A$ means that the type A is well-formed in the context Γ,
- $\Gamma \vdash t : A$ means that t is a term of type A in the context Γ,
- $\Gamma \vdash \sigma : \Delta$ means that σ is a substitution from Γ to Δ,

$$\frac{}{\emptyset\vdash}\qquad\frac{\Gamma\vdash A}{\Gamma,x:A\vdash}\qquad\qquad\frac{x:A\in\Gamma}{\Gamma\vdash x:A}\qquad\qquad\frac{}{\Gamma\vdash\star}\qquad\frac{\Gamma\vdash A:\star}{\Gamma\vdash\mathrm{El}(A)}\qquad\qquad\frac{\Gamma\vdash A:\star\qquad\Gamma\vdash B:\star}{\Gamma\vdash A\to B:\star}$$

Rules for contexts	Axiom rule	Rules for types	Rules for $\star$

$$\frac{\Delta\vdash}{\Delta\vdash\langle\rangle:\emptyset}\qquad\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash A\qquad\Delta\vdash t:A[\sigma]}{\Delta\vdash\langle\sigma,t\rangle:(\Gamma,x:A)}\qquad\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash t:A}{\Delta\vdash t[\sigma]:A[\sigma]}\,(\mathrm{sub})$$

Rules for substitution

$$\frac{\Gamma\vdash t:A}{\Gamma\vdash t=t:A}\qquad\frac{\Gamma\vdash t=u:A}{\Gamma\vdash u=t:A}\qquad\frac{\Gamma\vdash t=u:A\qquad\Gamma\vdash u=v:A}{\Gamma\vdash t=v:A}$$

Rules for equality

$$\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash t=u:A}{\Delta\vdash t[\sigma]=u[\sigma]:A[\sigma]}\qquad\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash A\qquad\Delta\vdash u=u':A[\sigma]\qquad\Gamma,x:A\vdash t:B}{\Delta\vdash t[\langle\sigma,u\rangle]=t[\langle\sigma,u'\rangle]:B[\langle\sigma,u\rangle]}$$

Rules for congruence

$$\frac{\Gamma\vdash_{\mathsf{ps}}A}{\Gamma\vdash\mathsf{coh}_{\Gamma\vdash A}[\mathrm{id}_\Gamma]:A}\,(\mathsf{ps})\qquad\frac{\Gamma\vdash_{\mathsf{ps}}A\qquad\Gamma\vdash t:A\qquad\Gamma\vdash u:A}{\Gamma\vdash t=u:A}\,(\mathsf{ps}^=)$$

Rules for pasting types

Figure 1 Rules of the type theory CCaTT_1.

- $\Gamma\vdash t=u:A$ means that t and u are two equal terms of type A in the context Γ.

We did not include judgments of equality on types or on substitutions because those are not useful at this stage.

A term of type $\star$ corresponds to a type in λ-calculus, which we call a *simple type* in the following (in order to avoid confusion with types in our theory). For this reason, we usually write $X,Y,\ldots$ (resp. $A,B,\ldots$) for variables (resp. terms) of type $\star$. For instance, the sequent $X:\star,Y:\star\vdash X\to Y\to X:\star$ is easily checked to be derivable, which means that $X\to Y\to X$ is a valid simple type provided that X and Y are type variables. Here, the type $\star$ plays the role of a universe (which is closed under taking arrows) and, given a simple type A, we have a type $\mathrm{El}(A)$: the terms of this type can be thought of as λ-terms of simple type A. For instance, we will see that we will be able to derive a term $f\cdot x$ of type

$$X:\star,Y:\star,f:\mathrm{El}(X\to Y),x:\mathrm{El}(X)\vdash f\cdot x:\mathrm{El}(Y)$$

which given a function f and an element x associates the result of applying f to x. In the following, we usually simply write A instead of $\mathrm{El}(A)$, as traditional in type theory, thus implicitly allowing the cast of a simple type as a type.

A substitution σ with $\Delta\vdash\sigma:\Gamma$ is a list of terms $\langle t_1,\ldots,t_{|\Gamma|}\rangle$ with t_i being of type A_i, up to substituting previous variables by their value. Given a term t in the context Γ, we write $t[\sigma]$ for the term obtained from t by replacing every variable $x_i\in\mathrm{dom}(\Gamma)$ by t_i (and we define a similar operation $A[\sigma]$ for types A in the context Γ). Formally, we have

$$\begin{aligned}x_i[\sigma]&\equiv t_i &\qquad (t\to u)[\sigma]&\equiv t[\sigma]\to u[\sigma] &\qquad \mathsf{coh}_{\Gamma\vdash A}[\tau][\sigma]&\equiv\mathsf{coh}_{\Gamma\vdash A}[\sigma\circ\tau]\\ \star[\sigma]&\equiv\star & \mathrm{El}(t)[\sigma]&\equiv\mathrm{El}(t[\sigma])\end{aligned}$$

Given substitutions $\Delta\vdash\tau:\Gamma$ and $\Upsilon\vdash\sigma:\Delta$ with $\tau=t_1,\ldots,t_n[]$, we respectively write

$$\tau\circ\sigma\equiv\langle t_1[\sigma],\ldots,t_n[\sigma]\rangle\qquad\qquad\mathrm{id}_\Gamma\equiv\langle x_1,\ldots,x_{|\Gamma|}\rangle$$

for composite and identity substitutions.

▶ **Lemma 16.** *The following rules are admissible:*

$$\frac{\Gamma\vdash}{\Gamma\vdash\mathrm{id}_\Gamma:\Gamma}\qquad\frac{\Upsilon\vdash\sigma:\Delta\qquad\Delta\vdash\tau:\Gamma}{\Upsilon\vdash\tau\circ\sigma:\Delta}\qquad\frac{\Gamma\vdash t:A}{\Gamma\vdash t[\mathrm{id}_\Gamma]=t:A}\qquad\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash t:A}{\Delta\vdash t[\tau\circ\sigma]=t[\sigma][\tau]:A}$$

A context consists of m declarations of type variables X_i followed by n declarations of variables with simple type A_i:

▶ **Lemma 17.** *The well-formed contexts are, up to permutation, of the form*

$$X_1:\star,\ldots,X_m:\star,x_1:A_1,\ldots,x_n:A_n \tag{6}$$

where the A_i are simple types with $\mathrm{FV}(A_i)\subseteq\{X_1,\ldots,X_m\}$.

Similarly, a substitution $\Gamma\vdash\sigma:\Delta$ can be decomposed as a pair of substitutions $\langle\sigma_0,\sigma_1\rangle$ consisting of a substitution σ_0 assigning a simple type to the type variables X_i, and a substitution σ_1 assigning a term to the term variables x_i: it can thus be thought of as consisting both of a type refinement and a substitution of λ-terms. The rules for equality ensure that equality is a congruence: this is an equivalence relation, substituting equal terms give rise to equal terms, and applying substitutions consisting of equal terms give rise to equal terms.

All the rules described above are pretty standard for a dependent type theory. The main novelty of our type theory comes from the two rules for pasting types. Given a context Γ of the form (6) and a type A, we write $\Gamma\vdash_{\mathsf{ps}}A$ when the sequent $\Gamma\vdash A$ is derivable, $\{X_1,\ldots,X_n\}=\mathrm{FV}(A_1)\cup\ldots\cup\mathrm{FV}(A_n)$ (i.e. the set of declared variables is minimal), the type A is a simple type (i.e. not $\star$), and we have $A_1,\ldots,A_n\vdash_{\mathsf{ps}}A$ in the sense of Section 3.1. The rules for pasting types impose that such types are contractible, i.e. they are inhabited and any two inhabitants are equal. A term of the form $\mathsf{coh}_{\Gamma\vdash A}[\sigma]$, which can be constructed using the rule (ps), denotes the unique inhabitant of $\Gamma\vdash_{\mathsf{ps}}A$ under a substitution σ, i.e. the following rules are derivable:

$$\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash_{\mathsf{ps}}A}{\Delta\vdash\mathsf{coh}_{\Gamma\vdash A}[\sigma]:A[\sigma]}\qquad\qquad\frac{\Delta\vdash\sigma:\Gamma\qquad\Gamma\vdash_{\mathsf{ps}}A\qquad\Gamma\vdash t:A}{\Delta\vdash t[\sigma]=\mathsf{coh}_{\Gamma\vdash A}[\sigma]:A[\sigma]}$$

Note that, in the notation $\mathsf{coh}_{\Gamma\vdash A}[\sigma]$, both Γ and A are left invariant under substitution (they are only here to keep track of the pasting type which was contracted), and that the substitution σ is a “formal” one, i.e. it is part of the term and does not reduce.

3.3 Some derivations

Let us provide some first examples of derivations in our type theory. The type $X\to X$ is pasting by Example 10 and we thus have that $X:\star\vdash_{\mathsf{ps}}X\to X$ is derivable. By the rule (ps), we thus have a term $\mathsf{coh}_{X:\star\vdash X\to X}[\langle X\rangle]$ of type $X\to X$ (in context $X:\star$):

$$\frac{X:\star\vdash_{\mathsf{ps}}X\to X}{X:\star\vdash\mathsf{coh}_{X:\star\vdash X\to X}[\langle X\rangle]:X\to X}\,(\mathsf{ps})$$

This term should be thought of as the identity λ-term, and we will adopt the more traditional notation I_X for it. Similarly, we can define terms

$$\begin{array}{r}X:\star\vdash\mathsf{I}_X:X\to X\\ X:\star,Y:\star\vdash\mathsf{K}_{X,Y}:X\to Y\to X\\ X:\star,Y:\star,Z:\star\vdash\mathsf{S}_{X,Y,Z}:(X\to Y\to Z)\to(X\to Y)\to X\to Z\end{array}\tag{7}$$

(and we sometimes omit the indices when they can be inferred from the context). It can be observed that many of the "standard" λ-terms can be defined in this way, as the inhabitant of their most general type which is contractible, with the notable exception of $\omega \equiv \lambda x.xx$ (which is not typable) and Church natural numbers (whose type is not contractible). In particular, almost all Smullyan's 28 birds [43, p. 244] can also be defined like this[1], excepting the Jay ($\lambda fxyz.fx(fzy)$), the Mocking bird (ω), the Owl ($\lambda xy.y(xy)$), the Turing ($\lambda xy.y(xxy)$, not typable).

Those would be pretty useless if they could not interact, but fortunately we can also define application as the inhabitant of its type which is contractible, see Example 15:

$$X:\star, Y:\star, f:X\to Y, x:X \vdash f\cdot x : Y \tag{8}$$

Again, $f\cdot x$ is an informal but readable notation for $\mathsf{coh}_{X:\star,Y:\star,f:X\to Y,x:X\vdash Y}[\langle X,Y,f,x\rangle]$, where we leave X and Y implicit. As usual, this operation is implicitly bracketed on the left.

Finally, we can derive the expected relations by using the rule ($\mathsf{ps}^=$). For instance, we have $X:\star, x:X \vdash_{\mathsf{ps}} X$ which implies, by Proposition 14, that any two λ-terms in this context are equal. For instance, x and $\mathsf{I}_X\cdot x$ are both terms of this type, and are thus equal: $\mathsf{I}_X\cdot x = x$. Formally, we can derive

$$\frac{X:\star, x:X\vdash_{\mathsf{ps}} X}{X:\star,x:X\vdash \mathsf{I}_X\cdot x = x : X}\,(\mathsf{ps}^=)$$

Similarly, one can derive the following equalities:

$$\begin{gathered}
X:\star,x:X\vdash \mathsf{I}_X\cdot x = x : X\\
X:\star,Y:\star,x:X,y:Y\vdash \mathsf{K}_{X,Y}\cdot x\cdot y = x : X\\
X:\star,Y:\star,Z:\star,x:X\to Y\to Z,y:X\to Y,z:X\vdash \mathsf{S}_{X,Y,Z}\cdot x\cdot y\cdot z = (x\cdot y)\cdot(x\cdot z):Z
\end{gathered}\tag{9}$$

3.4 Recovering simply typed λ-calculus

As shown above we can derive terms which look like λ-terms, as well as some expected equations. Our aim is now to show that we have exactly simply-typed λ-calculus:

▶ **Theorem 18.** *Given type variables $X_1,\ldots,X_m$ and simple types $A_1,\ldots A_n, A$ using the previous variables, there is a bijection between*
1. *terms t such that $X_1:\star,\ldots,X_m:\star,x_1:A_1,\ldots,x_n:A_n\vdash t:A$ is derivable in CCaTT_1, up to derivable equalities,*
2. *λ-terms t such that $x_1:A_1,\ldots,x_n:A_n\vdash t:A$ is derivable in implicational simply-typed λ-calculus, up to extensional equality.*

We respectively write $C^m_{\Gamma\vdash A}$ and $\Lambda^m_{\Gamma\vdash A}$ for the set of CCaTT_1 terms and λ-terms of type A in context $\Gamma\equiv x_1:A_1,\ldots,x_n:A_n$ as above, and our aim is to construct a family of bijections

$$F : C^m_{\Gamma\vdash A} \rightleftarrows \Lambda^m_{\Gamma\vdash A} : G$$

Given CCaTT_1 term t, we define its image under F by induction on t as follows.
- If t is obtained by the rule (ps), we have $\Gamma\vdash_{\mathsf{ps}} A$ and the type $\Gamma\vdash A$ is contractible by Proposition 14. We take the only element of $\Lambda^m_{\Gamma\vdash A}$ as its image under F.
- If t is obtained by the rule (sub), it is of the form $u[\sigma]$ and we define its image as $(Fu)[F\sigma]$ where $F\sigma$ is the substitution obtained by applying F to every term of σ (which is defined by induction).

[1] See `https://cccatt.mimram.fr/#birds`.

This is well-defined on equivalence classes of terms because pasting types are contractible and extensional equality is a congruence on λ-terms. Conversely, we want to construct a map G from λ-terms. It is well-known that λ-terms can be generated by the combinators I, K and S and that extensional equality on λ-terms corresponds to the congruence generated by the *weak equality*, which is generated by axioms (9), as well as the following five "mysterious axioms"

$$\begin{array}{lrl}
(\mathrm{eq}_\eta) & \mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})\,\mathsf{K})(\mathsf{K}\,\mathsf{I}) = \mathsf{I} & :(X \to Y) \to X \to Y \\
(\mathrm{eq}_w) & \mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{K})(\mathsf{S}(\mathsf{K}\,\mathsf{S})\,\mathsf{K})))(\mathsf{K}\,\mathsf{K}) = \mathsf{S}(\mathsf{K}\,\mathsf{K}) & :(X \to Z) \to X \to Y \to Z \\
(\mathrm{eq}_\mathsf{I}) & \mathsf{S}(\mathsf{K}\,\mathsf{I}) = \mathsf{I} & :(X \to Y) \to X \to Y \\
(\mathrm{eq}_\mathsf{K}) & \mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{K})) = \mathsf{K} & :(X \to Z) \to (X \to Y) \to X \to Z \\
(\mathrm{eq}_\mathsf{S}) & \mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{S})))(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{S}))) = & \\
& \multicolumn{2}{l}{\mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{K})(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{S})))\,\mathsf{S}))))(\mathsf{K}\,\mathsf{S})} \\
& \multicolumn{2}{r}{:(X \to Y \to Z \to W) \to (X \to Y \to Z) \to (X \to X) \to X \to Z}
\end{array}$$

which respectively ensure that η-expansion is satisfied, extensional equality is compatible with weakening, and that weak equivalence is preserved under abstraction, see [20, Chapter 8] for details. Moreover, this correspondence between pure λ-calculus and combinatory logic extends to the simply typed setting, see for instance [1]. We can therefore define G as map from simply typed combinatory terms to CCaTT_1 terms. We have seen in (7) how to define the combinators I, K and S, and application in (8), with the rule (ps). Moreover, all the equations can be derived with the rule ($\mathsf{ps}^=$), by checking that their most general type is pasting (and thus contractible): we have seen this in (9) for weak equality, and we have indicated above the types of the mysterious axioms, which are easily checked to be pasting[2]. Finally, we should check that the functions F and G thus defined are mutually inverse. This follows from the following two observations: firstly, given a term t of a contractible type, we have $G \circ F(t) = t$ (resp. $F \circ G(t) = t$) because types are preserved by both F and G, and, secondly, both derivable equalities and extensional equality are congruences (and thus compatible with substitution).

3.5 Arrows vs substitution

It can be wondered how our arrow types interact with respect to substitution, i.e. whether a term of type $A \to B$ can be thought of as a function substituting its argument in a term. The situation is pretty much similar to combinatory logic: the abstraction is an admissible operation, but not a primitive one.

▶ **Proposition 19.** *The following rules are respectively derivable (for application) and admissible (for the three other rules):*

$$\frac{\Gamma \vdash t : A \to B}{\Gamma, x : A \vdash t \cdot x : B}\,(\to_\mathrm{E}) \qquad\qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda_x(t) : A \to B}\,(\to_\mathrm{I})$$

$$\frac{\Gamma \vdash t : A \to B}{\Gamma \vdash t = \lambda_x(t \cdot x) : A \to B}\,(\to_\eta) \qquad\qquad \frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash \lambda_x(t) \cdot u = t[\langle \mathrm{id}_\Gamma, u \rangle] : B}\,(\to_\beta)$$

The proposition of course follows from Theorem 18, but a direct proof can also be given. The rule ($\to_\mathrm{E}$) is easily derived from application. For the rule ($\to_\mathrm{I}$), what we mean here is that given a term t of type A, we can define a term $\lambda_x(t)$ of type $A \to B$ by induction on t: the base case being immediate since we defined $\Gamma \vdash_{\mathsf{ps}} A$ to be $\vdash_{\mathsf{ps}} \Gamma \to A$ in Section 3.1. The two other rules are shown by induction on t.

[2] See `https://cccatt.mimram.fr/#curry`.

3.6 Adding products

Previous type theory is limited to the implicational fragment, but finite products can be added. In order to do so, one extends the syntax of types with a terminal type 1, as well as the product $A \times B$ of two types A and B. We then need to extend the definition of pasting types of Section 3.1. In fact, the following observations show that we can almost directly reuse the previous definition. Recall from [12] that type isomorphisms in simply-typed λ-calculus are generated by the equations

$$\begin{array}{cccc}(A{\times}B){\times}C = A{\times}(B{\times}C) & 1{\times}A = A & A{\to}(B{\times}C) = (A{\to}B){\times}(A{\to}C) & A{\to}1 = 1 \\ A{\times}B = B{\times}A & A{\times}1 = A & (A{\times}B){\to}C = A{\to}B{\to}C & 1{\to}A = A\end{array}$$

Moreover, by orienting the above equations from left to right, one obtains a convergent rewriting system whose normal forms are of the form $A_1 \times \ldots \times A_n$ where the A_i are formulas in the implicational fragment [12, Proposition 2.8]. We define a type A (with products) to be a *pasting* whenever all the components A_i of its normal form are pasting in the sense of Section 3.1, what we write $\vdash_{\mathsf{ps}} A$. As before, this extends to types in context by defining $\Gamma \vdash_{\mathsf{ps}} A$ as $\vdash_{\mathsf{ps}} \Gamma \to A$.

▶ **Proposition 20.** *Any pasting type is contractible.*

Proof. This follows from Proposition 12, the fact that being contractible is invariant under type isomorphism, and that a product of types is contractible if and only if its components are contractible. The case of types in context is handled as in Proposition 14. ◀

▶ **Open question 21.** *Provide a direct definition of pasting types with finite products (without normalizing types first).*

We write CCCaTT_1 (the type theory for cartesian closed 1-categories) for the type theory obtained from Figure 1 by adding the two rules

$$\frac{\Gamma \vdash A : \star \qquad \Gamma \vdash B : \star}{\Gamma \vdash A \times B : \star} \qquad\qquad \frac{}{\Gamma \vdash 1 : \star}$$

which allow forming product and terminal types, and understanding the judgments $\Gamma \vdash_{\mathsf{ps}} A$ in the above sense. The expected generalization of Theorem 18 holds:

▶ **Theorem 22.** *There is a type-preserving bijection between terms derivable in CCCaTT_1 (up to derivable equality) and terms in the simply typed λ-calculus (up to extensional equality).*

The interpretation of CCCaTT_1 into λ-calculus can be performed as for Theorem 18 (we are contracting contractible types). In the other direction, we need to explain how to extend simply typed combinatory logic to products, which can be done as follows.

▶ **Proposition 23.** *Simply typed combinatory logic with finite products can be obtained from traditional combinatory logic (as described in Section 3.4) by adding four new combinators*

$$\mathsf{P} : X \to Y \to X \times Y \qquad \mathsf{P}_1 : X \times Y \to X \qquad \mathsf{P}_2 : X \times Y \to Y \qquad \mathsf{T} : 1$$

as well as the weak equality axioms

$$\mathsf{P}_1 \cdot (\mathsf{P} \cdot x \cdot y) = x \qquad \mathsf{P}_2 \cdot (\mathsf{P} \cdot x \cdot y) = x \qquad \mathsf{P} \cdot (\mathsf{P}_1 \cdot x) \cdot (\mathsf{P}_2 \cdot x) = x \qquad x = 1$$

(those are implicitly typed by the most general type of both members, in particular, we suppose $x : \mathsf{T}$ *in the last equation) and the mysterious axioms*

$$\begin{aligned}
\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{P}_1)))))(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{P}))) &= \mathsf{K} && : (X \to Y) \to (X \to Z) \to X \to Y \\
\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{P}_2)))))(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{P}))) &= \mathsf{K}\,\mathsf{I} && : (X \to Y) \to (X \to Z) \to X \to Z \\
\mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{P})))(\mathsf{S}(\mathsf{K}\,\mathsf{P}_1))))(\mathsf{S}(\mathsf{K}\,\mathsf{P}_2)) &= \mathsf{I} && : (X \to Y \times Z) \to X \to Y \times Z \\
\mathsf{K}\,\mathsf{T} &= \mathsf{I} && : 1 \to 1
\end{aligned}$$

The proof of the proposition can be performed by extending the traditional one [1, 20, 37]. Alternatively, one can show that the generators and relations for the formal theory of cartesian closed categories[3][4] [30, Section I.3], which is known to be equivalent to simply typed λ-calculus, can be derived. Another option consists in deriving[5] Curien's categorical combinators [13].

As mentioned above, we have an axiomatization for simply typed λ-calculus with finite products, and thus of cartesian closed categories, which are the corresponding categorical structure [30]. Adopting this point of view, the problem of characterizing contractible types amounts to proving a coherence theorem for cartesian closed categories, a well-studied problem [5, 24, 39, 40, 46]. In our context, this can be expressed as follows:

▶ **Theorem 24.** *Consider the free cartesian closed category* $\mathcal{X}^*$ *on a set* $\mathcal{X}$ *of variables: an object corresponds to a simple type on the variables. For any two objects* A *and* B *such that the type* $A \to B$ *is contractible (in particular, when it is inhabited and balanced, pasting, or deterministic) there is exactly one morphism in* $\mathcal{X}^*(A, B)$*.*

3.7 Models

Let us briefly present the semantics of our type theory, which is based on the traditional approach to dependent type theories [15, 22]. The *syntactic category* $\mathcal{S}$ has contexts as objects and substitutions as morphisms (with the identities and compositions described in Section 3.2 and Lemma 16). This category has a structure of category with family (cwf): to a context Γ we can associate the set Ty_Γ of well-formed types in this context, and to $A \in \mathrm{Ty}_\Gamma$ we can associate a set $\mathrm{Tm}_{\Gamma\vdash A}$ of terms of type A in the context Γ, and substitutions act in the expected way on those. The category **Set** has a canonical structure of a (large) cwf where, for any set X, Ty_X is the set of families $(Y_x)_{x\in X}$ of sets indexed by X and $\mathrm{Tm}_{X\vdash Y}$ is the set $\prod_{x\in X} Y_x$. A model of our type theory is a morphism of cwf $\mathcal{S} \to \mathbf{Set}$.

Any small cartesian closed category $\mathcal{C}$ induces a model of our type theory as follows. Given a morphism $f : A \to B$, we write $\ulcorner f \urcorner : 1 \to (A \to B)$ for the canonically associated morphism, sometimes called the *name* of f. The type $\star$ is interpreted as the set $\mathrm{Ob}(\mathcal{C})$ of objects of $\mathcal{C}$, a simple type $\mathrm{El}(A)$ (corresponding to an object A of $\mathcal{C}$) as the set of elements of A (i.e. the hom-set $\mathcal{C}(1, A)$), a term $t : A$ as an element of the interpretation of A, an equality $t = u : A$ as the fact that the interpretations of t and u are equal, the type $A \to B$ as the elements of the internal arrow type of the interpretations of A and B (and similarly for the product $A \times B$), a term $\mathsf{coh}_{\Gamma\vdash A}[\mathrm{id}_\Gamma] : A$ as the only element of $\mathcal{C}(1, A)$ given by Theorem 24, and so on. Above, when working in a context Γ, all the constructions are implicitly to be read as "families indexed over the interpretation of the context Γ". For instance,

[3] See `https://cccatt.mimram.fr/#ccc`.
[4] See `https://cccatt.mimram.fr/#ccc2`.
[5] See `https://cccatt.mimram.fr/#curien`.

- $X : \star \vdash (X \to X)$ is interpreted as the family $(\mathcal{C}(1, X \to X))_{X\in\mathrm{Ob}(C)}$,
- $X : \star \vdash \mathsf{coh} : X \to X$ is interpreted as the family $(\ulcorner\mathrm{id}_X\urcorner)_{X\in\mathrm{Ob}(C)}$,
- $X : \star, x : X \vdash \mathsf{coh} : X$ is interpreted as the family $(x)_{X\in\mathrm{Ob}(C),x\in\mathcal{C}(1,X)}$.

The converse can be shown, using same tools detailed in [9] for the case of CaTT:

▶ **Theorem 25.** *The models of the type theory CCCaTT$_1$ are precisely the small cartesian closed categories.*

This result also follows indirectly from the fact that our type theory corresponds to simply typed λ-calculus (Theorem 22), which in turn corresponds to cartesian closed categories [30]. We should detail elsewhere the fact that the models of the type theory for the implicational fragment (Section 3.2) correspond to closed clones [29].

3.8 Implementation

Our type theory is implemented in a small proof assistant which is available online [35], with freely available source code [36]. It actually implements small extensions of the type theory described above in order to be more convenient. The most useful feature is the presence of implicit arguments: most often, the type variable arguments can be inferred, which makes proofs much shorter by leaving uninteresting parts to a unification algorithm. We also implement some variants described in the next section: equality is unbiased, and we can optionally switch to subsystems of the theory such as a categories, monoidal categories or symmetric monoidal categories.

4 Extensions and future work

We have presented an unbiased type-theoretical definition of simply typed λ-calculus. There are many possible extensions of this work that we plan to investigate in a near future.

There are many interesting subsystems or variants of our type theory. We already have type theories for unbiased categories, monoidal categories, symmetric monoidal categories and cartesian categories, which should be detailed elsewhere: those are essentially obtained by keeping the same rules as those of Figure 1, but adapting the notion of $\Gamma \vdash_{\mathsf{ps}} A$ for deriving pasting types. We are still investigating the case of symmetric monoidal closed categories (which correspond to linear simply typed λ-calculus) for which we have a coherence theorem [26] and characterization of type isomorphisms [44], and we can also think of closed categories [16, 31], linear λ-calculus [38, 51], planar λ-calculus [52], or compact closed categories [25]. A fundamental extension of our work would consist in defining unbiased locally cartesian closed categories: this currently seems harder as no coherence theorem (in the sense of Theorem 24) seems to be known for those.

As indicated in the introduction, we would also like to generalize this work to weak higher categories, based on CaTT [9, 17]. A first step in this direction, consists in observing that equality is biased in our type theory (in the sense that we explicitly put rules for equivalence relations). We have already defined (and implemented) a variant where this is not so, by modifying the type theory so that there are explicit terms for proofs of identities. The first variant that seems within reach is the one of weak cartesian higher categories, and is currently being investigated together with Thibaut Benjamin.

Finally, it would be interesting to formalize the results of this paper, possibly based on the Agda developments [1].

References

1 Thorsten Altenkirch, Ambrus Kaposi, Artjoms Sinkarovs, and Tamás Végh. Combinatory logic and lambda calculus are equal, algebraically. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction (FSCD)*, volume 260 of *LIPIcs*, pages 24:1–24:19. Schloss Dagstuhl, 2023.

2 Takahito Aoto. Uniqueness of normal proofs in implicational intuitionistic logic. *Journal of Logic, Language and Information*, 8(2):217–242, 1999.

3 Takahito Aoto and Hiroakira Ono. Non-uniqueness of normal proofs for minimal formulas in implication-conjunction fragment of BCK. *Bulletin of the Section of Logic*, 23(3):104–112, 1994.

4 Takahito Aoto and Hiroakira Ono. Uniqueness of normal proofs in $\{\to, \land\}$-fragment of NJ. Technical Report IS-RR-94-0024F, School of Information Science, JAIST, 1994. Research report.

5 A. A. Babaev and S. V. Solov'ev. A coherence theorem for canonical morphisms in cartesian closed categories. *Journal of Soviet Mathematics*, 20:2263–2279, 1982.

6 Hendrik Pieter Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1985.

7 Choukri-Bey Ben-Yelles. *Type-assignment in the lambda-calculus: syntax and semantics*. PhD thesis, University of Wales Swansea, 1989.

8 Thibaut Benjamin. Monoidal weak ω-categories as models of a type theory. *Mathematical Structures in Computer Science*, 33(8):744–780, 2023.

9 Thibaut Benjamin, Eric Finster, and Samuel Mimram. Globular weak ω-categories as models of a type theory. *Higher Structures*, 8(2):1–69, 2024.

10 Pierre Bourreau and Sylvain Salvati. Game semantics and uniqueness of type inhabitance in the simply-typed λ-calculus. In *Typed Lambda Calculi and Applications*, volume 6690 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2011.

11 Sabine Broda and Luís Damas. On long normal inhabitants of a type. *Journal of Logic and Computation*, 15(3):353–390, 2005.

12 Kim B. Bruce, Roberto Di Cosmo, and Giuseppe Longo. Provable isomorphisms of types. *Mathematical Structures in Computer Science*, 2(2):231–247, 1992.

13 Pierre-Louis Curien. Categorical combinators. *Information and Control*, 69(1-3):188–254, 1986.

14 Pierre-Louis Curien, Richard Garner, and Martin Hofmann. Revisiting the categorical interpretation of dependent type theory. *Theoretical Computer Science*, 546:99–119, 2014.

15 Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs, International Workshop TYPES'95*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 1995.

16 Samuel Eilenberg and G. Max Kelly. Closed categories. In *Proceedings of the Conference on Categorical Algebra: La Jolla 1965*, pages 421–562. Springer, 1966.

17 Eric Finster and Samuel Mimram. A type-theoretical definition of weak ω-categories. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE Computer Society, 2017.

18 Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*, volume 7. Cambridge university press Cambridge, 1989.

19 J. Roger Hindley. *Basic simple type theory*. Number 42. Cambridge University Press, 1997.

20 J. Roger Hindley and Jonathan P. Seldin. *Lambda-calculus and combinators: an introduction*. Cambridge University Press, 2008.

21 Sachio Hirokawa. Principal types of BCK-lambda-terms. *Theoretical Computer Science*, 107(2):253–276, 1993.

22 Martin Hofmann and Martin Hofmann. Syntax and semantics of dependent types. *Extensional Constructs in Intensional Type Theory*, pages 13–54, 1997.

23 William A Howard. The formulae-as-types notion of construction. *To HB Curry: essays on combinatory logic, lambda calculus and formalism*, 44:479–490, 1980.

24 Barry Jay. The structure of free closed categories. *Journal of Pure and Applied Algebra*, 66(3):271–285, 1990.

25 Gregory M Kelly and Miguel L Laplaza. Coherence for compact closed categories. *Journal of pure and applied algebra*, 19:193–213, 1980.

26 Gregory Maxwell Kelly and Saunders MacLane. Coherence in closed categories. *Journal of Pure and Applied Algebra*, 1(1):97–140, 1971.

27 Yuichi Komori. BCK algebras and lambda calculus. In *Proceedings of the 10th Symposium on Semigroups*, pages 5–11, 1987.

28 Yuichi Komori and Sachio Hirokawa. The number of proofs for a BCK-formula. *Journal of Symbolic Logic*, 58(2):626–628, 1993.

29 Joachim Lambek. Multicategories revisited. *Contemporary Mathematics*, 92:217–239, 1989.

30 Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

31 Miguel L Laplaza. Coherence in nonmonoidal closed categories. *Transactions of the American Mathematical Society*, 230:293–311, 1977.

32 Tom Leinster. *Higher operads, higher categories.* Number 298. Cambridge University Press, 2004.

33 Peter LeFanu Lumsdaine and Michael A. Warren. The local universes model: an overlooked coherence construction for dependent type theories. *ACM Transactions on Computational Logic (TOCL)*, 16(3):1–31, 2015.

34 Georges Maltsiniotis. Grothendieck ∞-groupoids, and still another definition of ∞-categories. Preprint, 2010. `arXiv:1009.2331`.

35 Samuel Mimram. CCCaTT_1 online proof assistant. URL: `https://cccatt.mimram.fr/`.

36 Samuel Mimram. CCCaTT_1 GitHub repository. URL: `https://github.com/smimram/cccatt`.

37 Samuel Mimram. *Program = Proof.* 2020. URL: `http://pp.mimram.fr/`.

38 Grigory E. Mints. Closed categories and the theory of proofs. *Journal of Soviet Mathematics*, 15:45–62, 1981.

39 Grigory E. Mints. A simple proof of the coherence theorem for cartesian closed categories. In *Selected Papers in Proof Theory*, Studies in Proof Theory, pages 213–220. Bibliopolis, 1992.

40 Akira Mori and Yoshihiro Matsumoto. Coherence for cartesian closed categories: A sequential approach. In Nachum Dershowitz and Naomi Lindenstrauss, editors, *Conditional and Typed Rewriting Systems, 4th International Workshop, CTRS-94*, volume 968 of *Lecture Notes in Computer Science*, pages 276–295. Springer, 1994.

41 Gabriel Scherer and Didier Rémy. Which simple types have a unique inhabitant? In *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP*, pages 243–255. ACM, 2015.

42 Michael Shulman. All $(\infty, 1)$-toposes have strict univalent universes. Preprint, 2019. `arXiv: 1904.07004`.

43 Raymond M. Smullyan. *To Mock a Mockingbird: and other logic puzzles including an amazing adventure in combinatory logic.* Oxford University Press, USA, 2000.

44 Sergei Soloviev. A complete axiom system for isomorphism of types in closed categories. In Andrei Voronkov, editor, *Logic Programming and Automated Reasoning, 4th International Conference, LPAR'93*, volume 698 of *Lecture Notes in Computer Science*, pages 360–371. Springer, 1993.

45 Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism.* Elsevier, 2006.

46 Manfred E. Szabo. Coherence in cartesian closed categories and the generality of proofs. *Stud Logica*, 48(3):285–297, 1989.

47 Masako Takahashi, Yohji Akama, and Sachio Hirokawa. Normal proofs and their grammar. *Information and Computation*, 125(2):144–153, 1996.

48 Makoto Tatsuta. Uniqueness of normal proofs of minimal formulas. *Journal of Symbolic Logic*, 58(3):789–799, 1993.

49 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. 2013. URL: `https://homotopytypetheory.org/book`.

50 Marec Zaionc. The regular expression descriptions of unifier set in the typed λ-calculus. *Fundamenta Informaticae*, 10(3):309–322, 1987.

51 Noam Zeilberger. Balanced polymorphism and linear lambda calculus. *Talk at TYPES*, 2015.

52 Noam Zeilberger and Alain Giorgetti. A correspondence between rooted planar maps and normal planar lambda terms. *Logical Methods in Computer Science*, 11, 2015.