arXiv:1704.04747v1 [math.CT] 16 Apr 2017

# Dependent Cartesian Closed Categories

**Norihiro Yamada**
norihiro1988@gmail.com
University of Oxford

April 18, 2017

**Abstract**

We present a generalization of *cartesian closed categories (CCCs)* for *dependent types*, called *dependent cartesian closed categories (DCCCs)*, which also provides a reformulation of *categories with families (CwFs)*, an abstract semantics for *Martin-Löf type theory (MLTT)* which is very close to the syntax. Thus, DCCCs accomplish mathematical elegance as well as a direct interpretation of the syntax. Moreover, they capture the categorical counterpart of the generalization of the *simply-typed λ-calculus (STLC)* to MLTT in syntax, and give a systematic perspective on the relation between categorical semantics for these type theories. Furthermore, we construct a *term model* from the syntax, establishing the *completeness* of our interpretation of MLTT in DCCCs.

## Contents

**1 Introduction** **2**
1.1 Martin-Löf type theory . . . 2
1.2 Categories with families . . . 2
1.3 Problem to solve . . . 3
1.4 Dependent cartesian closed categories . . . 3
1.5 Related work and our contribution . . . 3
1.6 Overview . . . 4

**2 The syntax of Martin-Löf type theory** **4**
2.1 Contexts . . . 4
2.2 Structural rules . . . 5
2.3 The unit type . . . 5
2.4 Dependent function types . . . 5
2.5 Dependent pair types . . . 6
2.6 Context morphisms and generalized substitution . . . 6
2.7 The simply-typed λ-calculus . . . 7

**3 Categories with families** **7**
3.1 Interpretation of MLTT in CwFs . . . 10
3.2 Interpretation of STLC in CCCs . . . 11

**4 Dependent cartesian closed categories** **12**
4.1 Dependent categories . . . . . 12
4.2 Dependent cartesian categories . . . . . 12
4.3 Dependent cartesian closed categories . . . . . 15

**5 SDTC (simple and dependent types correspondence) theorem** **17**

**6 Conclusion and future work** **17**

# 1 Introduction

The present work introduces the notion of *dependent cartesian closed categories (DCCCs)* and their interpretation of *Martin-Löf type theory (MLTT)*, which form a generalization of *cartesian closed categories (CCCs)* and their interpretation of the *simply-typed λ-calculus (STLC)*. Our aim is primarily to capture the categorical counterpart of the path from STLC to MLTT in syntax but also to give a refinement of *categories with families (CwFs)* in order to obtain an abstract semantics for MLTT that is mathematically neat and close to the syntax.

## 1.1 Martin-Löf type theory

*MLTT* [ML82, ML84, ML98] is an extension of *STLC* [Chu40] that, under the *Curry-Howard isomorphism (CHI)* [SU06], corresponds to *intuitionistic predicate logic* [TS00], for which the extension is made by *dependent types* [Hof97], i.e., types that depend on terms. It was proposed by Martin-Löf as a foundation of constructive mathematics, but also it has been an object of active research in computer science because it can be seen as a programming language, and one may extract programs that are "correct by construction" from its proofs.

## 1.2 Categories with families

*Categorical logic* [Pit01, Jac99, LS88] is the study of connections between categories [ML13] and type theories [Pie02]. In categorical logic, both syntax and semantics of type theories are represented by categories possibly with additional structures, and interpretations by functors preserving the structures. For syntax, such categorical representations abstract tedious syntactic formalisms (such as *capture-free substitution*) and enable one to focus on structures in type theories; also, neat categorical structures in some sense "justify" the corresponding syntax. For semantics, on the other hand, such categorical semantics provides a convenient framework to establish concrete semantics because it is simpler and subsumes more interpretations than the traditional set-theoretic framework; also, it gives *internal languages*, i.e., formal expressions to talk about categorical phenomena beyond the usual language of categories.

*CwFs* [Hof97] are an abstract semantics for MLTT originally introduced in [Dyb96]. Compared to other categorical semantics for MLTT such as *categories with attributes* (*CwAs*) [Car86, Mog91, Pit01], *locally cartesian closed categories* (*LCCCs*) [See84] and ones based on *fibred categories* [Jac99] or *indexed categories* [Cur89, Obt89], CwFs are closer to the syntax, and so they directly exhibit semantic counterparts of syntactic phenomena. For this reason CwFs provide a convenient framework when one tries to find a concrete model of MLTT, e.g., the classic *groupoid model* [HS98] and the recent *game semantics* [AJV15] were both established by showing that they give rise to CwFs.

## 1.3 Problem to solve

However, although MLTT is a generalization of STLC (see Sec. 2.7), CwFs (and their interpretation of MLTT [Hof97]) are *not* a natural generalization of CCCs (and their interpretation of STLC [Jac99, Pit01]); e.g., CwFs do not provide a generalization of products or exponentials by *universal properties*. Hence, the path from CCCs to CwFs in categories does not completely capture the route from STLC to MLTT in syntax. On the other hand, CwFs refine the standard categorical semantics for STLC [Pit01, Jac99]: The former interprets contexts, context morphisms, types and terms by different entities in a CwF, while the latter does both contexts and types by objects and both context morphisms and terms by morphisms in a CCC.

In other words, it is a problem of conceptual and mathematical interests in its own right to give a natural generalization of CCCs and their interpretation of STLC that corresponds to MLTT. Moreover, we would like to obtain a systematic perspective on the difference between the two approaches to categorical logic, i.e., CwFs and CCCs, described above.

## 1.4 Dependent cartesian closed categories

Motivated by these considerations, in this paper we propose a generalization of CCCs, called *dependent cartesian closed categories* (DCCCs), and their interpretation of MLTT. A DCCC is a category $\mathcal{C}$ equipped with *dependent objects* and *dependent morphisms* (that respectively generalize objects and morphisms) as well as *dependent pair spaces* and *dependent map spaces* defined by familiar universal properties that directly generalize those for products and exponentials. As a result, DCCCs can be seen as a natural generalization of CCCs.

Moreover, DCCCs immediately induce CwFs that support 1-, $\Pi$- and $\Sigma$-types, giving a direct correspondence with MLTT. Although the converse does not hold, i.e., a CwF is not necessarily a DCCC, we claim that DCCCs are a *refinement* of CwFs since the *term model* and various instances of CwFs turn out to be DCCCs, which in particular establishes the *completeness* of our interpretation of MLTT in DCCCs.

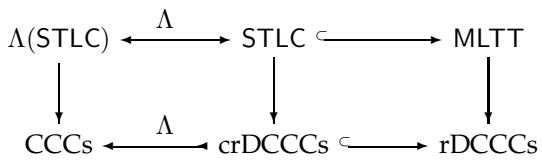
## 1.5 Related work and our contribution

The primary contribution of the present work is to give a categorical structure that forms a natural generalization of CCCs as well a refinement of CwFs. In fact, DCCCs are concisely formulated by universal properties, and various structures and axioms of CwFs are *derived*. Therefore we have captured the categorical counterpart of the route from simple types (i.e., non-dependent types) to dependent types, and obtained a simpler semantics for MLTT.

Note that some categorical models of MLTT are equivalent to CwFs [CD14], but none of them are close enough to CwFs to be called a *refinement* of CwFs. Also, they do not generalize CCCs and their interpretation of STLC in a systematic manner; e.g., morphisms $[\![A]\!] \to [\![B]\!]$ in a LCCC $\mathcal{C}$ [See84] interpret *closed terms* $\vdash M : A \Rightarrow B$ of MLTT, not *open* terms $x : A \vdash M' : B$, and the translation from LCCCs to CwFs takes a considerable amount of work [CD14].

Finally, we have clarified the relation between the interpretations of STLC in CwFs and CCCs: Since STLC (more precisely its term model) induces a *constant and recursive DCCC (crDCCC)*, and crDCCCs are in one-to-one correspondence $\Lambda$ with certain CCCs, the standard interpretation of STLC in CCCs actually maps $\Lambda(\mathsf{STLC})$, rather than STLC, into CCCs, but it "pretends" to interpret STLC itself. In contrast, DCCCs (and CwFs) rather directly model STLC. To

summarize, the following diagram commutes (see Thm. 5.2):




where *recursive DCCCs (rDCCCs)*, or more generally DCCCs, cannot be reduced to CCCs, which is why the standard semantics by CCCs is not applicable to dependent type theories.

### 1.6 Overview

The rest of the paper proceeds as follows. We review the syntax of MLTT in Sec. 2 and the interpretation of MLTT in CwFs in Sec. 3. Then Sec. 4 defines DCCCs, and Sec. 5 addresses the problem mentioned above. Finally, Sec. 6 makes a conclusion and proposes future work.

## 2 The syntax of Martin-Löf type theory

To describe how DCCCs model MLTT precisely, we first review its syntax, following the presentation in [Hof97]. See [NPS90, ML84] for a general introduction to MLTT.

***MLTT*** is a formal system to deduce ***judgements***, for which we usually write $\mathcal{J}$. There are the following six kinds of judgements (followed by the intended meanings):

- $\vdash \Gamma\ \mathsf{ctx}$ ($\Gamma$ is a ***context***)
- $\Gamma \vdash A\ \mathsf{type}$ ($A$ is a ***type*** in the context $\Gamma$)
- $\Gamma \vdash a : A$ ($a$ is a ***term*** of type $A$ in the context $\Gamma$)
- $\vdash \Gamma \equiv \Delta\ \mathsf{ctx}$ ($\Gamma$ and $\Delta$ are ***equal*** contexts)
- $\Gamma \vdash A \equiv B\ \mathsf{type}$ ($A$ and $B$ are ***equal*** types in $\Gamma$)
- $\Gamma \vdash a \equiv a' : A$ ($a$ and $a'$ are ***equal*** terms of type $A$ in $\Gamma$).

That is, MLTT consists of *axioms* $\frac{}{\mathcal{J}}$ and *(inference) rules* $\frac{\mathcal{J}_1\ \mathcal{J}_2 \dots \mathcal{J}_k}{\mathcal{J}_0}$, which are to make a *conclusion* from *hypotheses* by constructing a *derivation (tree)*. Under CHI, we may identify the words "*contexts*" with "*assumptions*", "*types*" with "*propositions*" and "*terms*" with "*proofs*"; thus, e.g., the judgement $\Gamma \vdash a : A$ can be read as "the proposition $A$ has the proof $a$ under the assumption $\Gamma$", etc. Below, we present MLTT with *strict* 1- $\Pi$- *and* $\Sigma$*-types*, which we call $\mathsf{MLTT}(1, \Pi, \Sigma)$.

### 2.1 Contexts

A ***context*** is a finite sequence $x_1 : A_1, x_2 : A_2, \dots, x_n : A_n$ of (variable : type)-pairs such that the variables $x_1, x_2, \dots, x_n$ are pair-wise distinct. It represents the "context" in which variables $x_i$ are assigned the types $A_i$. We have the following axiom and rules for contexts:

$$\frac{}{\vdash \diamondsuit\ \mathsf{ctx}}(\textsc{Ct-Emp}) \quad \frac{\Gamma \vdash A\ \mathsf{type}}{\vdash \Gamma, x : A\ \mathsf{ctx}}(\textsc{Ct-Ext}) \quad \frac{\vdash \Gamma \equiv \Delta\ \mathsf{ctx} \quad \Gamma \vdash A \equiv B\ \mathsf{type}}{\vdash \Gamma, x : A \equiv \Delta, y : B\ \mathsf{ctx}}(\textsc{Ct-Eq})$$

where $x$ (resp. $y$) does not occur in $\Gamma$ (resp. $\Delta$). Note that Ct-Eq is a *congruence rule*, i.e., it states that (judgmental) equality is *preserved* under "context extension". We will henceforth skip writing down congruence rules with respect to the other constructions.

## 2.2 Structural rules

Next, we collect the inference rules for all types as *structural rules*:

$$\frac{\vdash x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \ \mathsf{ctx} \ \ j \in \{1, 2, \ldots, n\}}{x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash x_j : A_j}(\textsc{Var}) \quad \frac{\vdash \Gamma \ \mathsf{ctx}}{\vdash \Gamma \equiv \Gamma \ \mathsf{ctx}}(\textsc{Ct-EqRefl})$$

$$\frac{\vdash \Gamma \equiv \Delta \ \mathsf{ctx}}{\vdash \Delta \equiv \Gamma \ \mathsf{ctx}}(\textsc{Ct-EqSym}) \quad \frac{\vdash \Gamma \equiv \Delta \ \mathsf{ctx} \ \vdash \Delta \equiv \Theta \ \mathsf{ctx}}{\vdash \Gamma \equiv \Theta \ \mathsf{ctx}}(\textsc{Ct-EqTrans})$$

$$\frac{\Gamma \vdash A \ \mathsf{type}}{\Gamma \vdash A \equiv A \ \mathsf{type}}(\textsc{Ty-EqRefl}) \quad \frac{\Gamma \vdash A \equiv B \ \mathsf{type}}{\Gamma \vdash B \equiv A \ \mathsf{type}}(\textsc{Ty-EqSym})$$

$$\frac{\Gamma \vdash A \equiv B \ \mathsf{type} \ \ \Gamma \vdash B \equiv C \ \mathsf{type}}{\Gamma \vdash A \equiv C \ \mathsf{type}}(\textsc{Ty-EqTrans}) \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A}(\textsc{Tm-EqRefl})$$

$$\frac{\Gamma \vdash a \equiv a' : A}{\Gamma \vdash a' \equiv a : A}(\textsc{Tm-EqSym}) \quad \frac{\Gamma \vdash a \equiv a' : A \ \ \Gamma \vdash a' \equiv a'' : A}{\Gamma \vdash a \equiv a'' : A}(\textsc{Tm-EqTrans})$$

$$\frac{\vdash \Gamma \equiv \Delta \ \mathsf{ctx} \ \ \Gamma \vdash A \ \mathsf{type}}{\Delta \vdash A \ \mathsf{type}}(\textsc{Ty-Con}) \quad \frac{\Gamma \vdash a : A \ \ \vdash \Gamma \equiv \Delta \ \mathsf{ctx} \ \ \Gamma \vdash A \equiv B \ \mathsf{type}}{\Delta \vdash a : B}(\textsc{Tm-Con})$$

The following ***weakening*** and ***substitution*** rules are *admissible* in MLTT, but it is convenient to present them explicitly:

$$\frac{\Gamma, \Delta \vdash \mathcal{J} \ \ \Gamma \vdash A \ \mathsf{type}}{\Gamma, x : A, \Delta \vdash \mathcal{J}}(\textsc{Weak}) \quad \frac{\Gamma, x : A, \Delta \vdash \mathcal{J} \ \ \Gamma \vdash a : A}{\Gamma, \Delta[a/x] \vdash \mathcal{J}[a/x]}(\textsc{Subst})$$

where $x$ does not occur in $\Gamma$ or $\Delta$, and $\mathcal{J}[a/x]$ (resp. $\Delta[a/x]$) denotes the *capture-free substitution* of $a$ for $x$ in $\mathcal{J}$ (resp. $\Delta$).

## 2.3 The unit type

The ***unit type*** $1$ is the simplest type because it has just one term $\star$ (we assume the uniqueness rule 1-Uniq below, which subsumes the usual elimination and computation rules for 1).

Its inference rules are as follows:

$$\frac{\vdash \Gamma \ \mathsf{ctx}}{\Gamma \vdash 1 \ \mathsf{type}}(\textsc{1-Form}) \quad \frac{\vdash \Gamma \ \mathsf{ctx}}{\Gamma \vdash \star : 1}(\textsc{1-Intro}) \quad \frac{\Gamma \vdash a : 1}{\Gamma \vdash a \equiv \star : 1}(\textsc{1-Uniq})$$

## 2.4 Dependent function types

A ***dependent function (Π-) type*** $\Pi_{x:A}B(x)$ represents the space of *dependent functions* $f : A \to \bigcup_{x \in A} B(x)$ such that $f(a) \in B(a)$ for all $a \in A$, generalizing *function types* in STLC.

Its inference rules are as follows:

$$\frac{\Gamma \vdash A \ \mathsf{type} \ \ \Gamma, x : A \vdash B \ \mathsf{type}}{\Gamma \vdash \Pi_{x:A} B \ \mathsf{type}}(\Pi\textsc{-Form}) \quad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x.b : \Pi_{x:A} B}(\Pi\textsc{-Intro})$$

$$\frac{\Gamma \vdash f : \Pi_{x:A} B \ \ \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}(\Pi\textsc{-Elim}) \quad \frac{\Gamma, x : A \vdash b : B \ \ \Gamma \vdash a : A}{\Gamma \vdash (\lambda x.b)(a) \equiv b[a/x] : B[a/x]}(\Pi\textsc{-Comp})$$

$$\frac{\Gamma \vdash f : \Pi_{x:A} B}{\Gamma \vdash \lambda x.f(x) \equiv f : \Pi_{x:A} B}(\Pi\textsc{-Uniq})$$

where we have included the uniqueness rule Π-Uniq as a generalization of *η-rule* in STLC.

## 2.5 Dependent pair types

A ***dependent pair (Σ-) type*** $\Sigma_{x:A}B(x)$ represents the space of *dependent pairs* $(a, b) : A \times \bigcup_{x \in A} B(x)$ such that $b \in B(a)$, which generalizes *pair types* in STLC.

Its inference rules are as follows:

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \Sigma_{x:A}B \text{ type}}(\Sigma\text{-FORM}) \quad \frac{\Gamma, x : A \vdash B \text{ type} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a, b) : \Sigma_{x:A}B}(\Sigma\text{-INTRO})$$

$$\frac{\Gamma, z : \Sigma_{x:A}B \vdash C \text{ type} \quad \Gamma \vdash p : \Sigma_{x:A}B \quad \Gamma, x : A, y : B \vdash g : C[(x, y)/z]}{\Gamma \vdash R^{\Sigma}(C, g, p) : C[p/z]}(\Sigma\text{-ELIM})$$

$$\frac{\Gamma, z : \Sigma_{x:A}B \vdash C \text{ type} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : B[a/x] \quad \Gamma, x : A, y : B \vdash g : C[(x, y)/z]}{\Gamma \vdash R^{\Sigma}(C, g, (a, b)) \equiv g[a/x, b/y] : C[(a, b)/z]}(\Sigma\text{-COMP})$$

$$\frac{\Gamma \vdash p : \Sigma_{x:A}B}{\Gamma \vdash (\pi_1[p], \pi_2[p]) \equiv p : \Sigma_{x:A}B}(\Sigma\text{-UNIQ})$$

where $\pi_1[p] \stackrel{\text{df.}}{\equiv} R^{\Sigma}(A, p, x)$, $\pi_2[p] \stackrel{\text{df.}}{\equiv} R^{\Sigma}(B, p, y)$. Again, we have included the uniqueness rule $\Sigma$-UNIQ for our MLTT to be in accordance with STLC and CCCs.

## 2.6 Context morphisms and generalized substitution

We have presented the syntax of $\mathsf{MLTT}(1, \Pi, \Sigma)$. Next, let us review a *derived* concept in the syntax: A ***context morphism*** [Hof97] from a context $\vdash \Delta$ ctx to another context $\vdash \Gamma$ ctx, where $\vdash \Gamma \equiv x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$ ctx, is a finite sequence $\mathbf{M} = \langle M_1, M_2, \ldots, M_n \rangle : \Delta \to \Gamma$ of terms

$$\begin{aligned}
&\Delta \vdash M_1 : A_1 \\
&\Delta \vdash M_2 : A_2[M_1/x_1] \\
&\quad \vdots \\
&\Delta \vdash M_n : A_n[M_1/x_1, M_2/x_2, \ldots, M_{n-1}/x_{n-1}].
\end{aligned}$$

In addition, we say that context morphisms $\mathbf{M}, \mathbf{M'} : \Delta \to \Gamma$ are ***(judgmentally) equal*** and write $\mathbf{M} \equiv \mathbf{M'} : \Delta \to \Gamma$ if so are their corresponding component terms.

Given any syntactic expression $E$, we define the ***generalized substitution*** $E[\mathbf{M}/\mathbf{x}]$ of $\mathbf{M}$ for $\mathbf{x}$ in $E$ (we often abbreviate it as $E[\mathbf{M}]$), where $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, to be the expression

$$E[M_1/x_1, M_2/x_2, \ldots, M_n/x_n]$$

i.e., what is obtained from $E$ by simultaneously substituting $M_i$ for $x_i$ in $E$ for $i = 1, 2, \ldots, n$. Then it is shown in [Hof97] that if $\Delta, \Theta \vdash E$ is a judgement, then so is $\Gamma, \Theta[\mathbf{M}] \vdash E[\mathbf{M}]$. Clearly, generalized substitution subsumes SUBST and WEAK.

Let $\mathbf{N} : \Theta \to \Delta$ be another context morphism. We define the ***composition*** $\mathbf{M} \circ \mathbf{N} : \Theta \to \Gamma$ by

$$\mathbf{M} \circ \mathbf{N} \stackrel{\text{df.}}{\equiv} \langle M_1[\mathbf{N}], M_2[\mathbf{N}], \ldots, M_n[\mathbf{N}] \rangle.$$

Also, we have the ***identity context morphisms***

$$id_\Gamma \stackrel{\text{df.}}{\equiv} \langle x_1, x_2, \ldots, x_n \rangle : \Gamma \to \Gamma.$$

It has been shown in [Hof97] that contexts and context morphisms modulo judgmental equality $\equiv$ form a category. This suggests that we may reformulate the *derived* notion of substitution as *primitive*, leading to CwFs (in Sec. 3).

## 2.7 The simply-typed $\lambda$-calculus

Since we shall compare our interpretation of MLTT in DCCCs with the standard interpretation of STLC (in the form of an equational theory) equipped with *finite product types*, which we call the ***equational theory*** $\boldsymbol{\lambda^{=}_{1,\times}}$, here we present its syntax as a sub-theory of MLTT.

The theory $\lambda^{=}_{1,\times}$ is obtained from $\mathsf{MLTT}(1,\Pi,\Sigma)$ by restricting types to *closed* ones and eliminating the rules for equalities between contexts and types as there are only the trivial equalities between them. Thus, it is a formal system to deduce the following judgements:

- $\vdash \Gamma\ \mathsf{ctx}$ ($\Gamma$ is a ***context***)
- $\vdash A\ \mathsf{type}$ ($A$ is a (*closed*) ***type***)
- $\Gamma \vdash a : A$ ($a$ is a ***term*** of type $A$ in the context $\Gamma$)
- $\Gamma \vdash a \equiv a' : A$ ($a$ and $a'$ are ***equal*** terms of type $A$ in $\Gamma$).

Speifically, $\lambda^{=}_{1,\times}$ consists of the following axioms and inference rules:

$$\frac{}{\vdash \diamondsuit\ \mathsf{ctx}}(\textsc{Ct-Emp}) \quad \frac{\vdash \Gamma\ \mathsf{ctx} \quad \vdash A\ \mathsf{type}}{\vdash \Gamma, x : A\ \mathsf{ctx}}(\textsc{Ct-Ext})$$

$$\frac{\vdash x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n\ \mathsf{ctx} \quad j \in \{1,2,\ldots,n\}}{x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash x_j : A_j}(\textsc{Var}) \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A}(\textsc{Tm-EqRefl})$$

$$\frac{\Gamma \vdash a \equiv a' : A}{\Gamma \vdash a' \equiv a : A}(\textsc{Tm-EqSym}) \quad \frac{\Gamma \vdash a \equiv a' : A \quad \Gamma \vdash a' \equiv a'' : A}{\Gamma \vdash a \equiv a'' : A}(\textsc{Tm-EqTrans})$$

$$\frac{}{\vdash 1\ \mathsf{type}}(1\text{-}\textsc{Form}) \quad \frac{\vdash A\ \mathsf{type} \quad \vdash B\ \mathsf{type}}{\vdash A \Rightarrow B\ \mathsf{type}}(\Rightarrow\text{-}\textsc{Form}) \quad \frac{\vdash A\ \mathsf{type} \quad \vdash B\ \mathsf{type}}{\vdash A \times B\ \mathsf{type}}(\times\text{-}\textsc{Form})$$

$$\frac{\vdash \Gamma\ \mathsf{ctx}}{\Gamma \vdash \star : 1}(1\text{-}\textsc{Intro}) \quad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x.b : A \Rightarrow B}(\Rightarrow\text{-}\textsc{Intro}) \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a,b) : A \times B}(\times\text{-}\textsc{Intro})$$

$$\frac{\Gamma \vdash f : A \Rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}(\Rightarrow\text{-}\textsc{Elim}) \quad \frac{\Gamma \vdash p : A_1 \times A_2}{\Gamma \vdash \pi_i[p] : A_i}(\times\text{-}\textsc{Elim}) \quad \frac{\Gamma \vdash c : C}{\Gamma \vdash c \equiv \star : C}(1\text{-}\textsc{Uniq})$$

$$\frac{\Gamma \vdash f : A \Rightarrow B}{\Gamma \vdash \lambda x.f(x) \equiv f : A \Rightarrow B}(\Rightarrow\text{-}\textsc{Uniq}) \quad \frac{\Gamma \vdash p : A \times B}{\Gamma \vdash (\pi_1[p], \pi_2[p]) \equiv p : A \times B}(\times\text{-}\textsc{Uniq})$$

$$\frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x.b)(a) \equiv b[a/x] : B}(\Rightarrow\text{-}\textsc{Comp}) \quad \frac{\Gamma \vdash a_i : A_i}{\Gamma \vdash \pi_i[(a_1,a_2)] \equiv a_i : A_i}(\times\text{-}\textsc{Comp})$$

where we again omit describing congruence rules. Following the convention, we write $A \Rightarrow B$, $A \times B$ for $\Pi_{x:A}B$, $\Sigma_{x:A}B$, respectively, since there is no dependent type in $\lambda^{=}_{1,\times}$. Note that $\times$-ELIM and $\times$-COMP are *admissible* in $\mathsf{MLTT}(1,\Pi,\Sigma)$ by $\Sigma$-ELIM and $\Sigma$-COMP as described in [Hof97]; the other rules are just inherited from $\mathsf{MLTT}(1,\Pi,\Sigma)$.

It is easy to see that $\lambda^{=}_{1,\times}$ coincides with the standard equational theory of STLC with respect to $\beta\eta$-*equivalence* equipped with finite product types [Cro93, Jac99, LS88].

# 3 Categories with families

Next, we quickly review the notion of *categories with families (CwFs)* and their *semantic type formers* as well as their interpretation of $\mathsf{MLTT}(1,\Pi,\Sigma)$.

▶ **Definition 3.1** (CwFs [Dyb96, Hof97]). A ***CwF*** is a tuple $\mathcal{C} = (\mathcal{C}, Ty, Tm, _\{_\}, T, _._, p, v, \langle_,_\rangle__)$, where:

- $\mathcal{C}$ is a category of ***contexts*** and ***context morphisms***
- $Ty$ assigns, to each $\Gamma \in \mathcal{C}$, a set $Ty(\Gamma)$ of ***types*** in $\Gamma$
- $Tm$ assigns, to each $\Gamma \in \mathcal{C}$, $A \in Ty(\Gamma)$, a set $Tm(\Gamma, A)$ of ***terms*** of type $A$ in $\Gamma$
- For each $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, $_\{_\}$ induces a function $_\{\phi\} : Ty(\Gamma) \to Ty(\Delta)$ and a family $(_\{\phi\}_A : Tm(\Gamma, A) \to Tm(\Delta, A\{\phi\}))_{A \in Ty(\Gamma)}$ of functions
- $T \in \mathcal{C}$ is a terminal object; $_._$ assigns, to each $\Gamma \in \mathcal{C}$, $A \in Ty(\Gamma)$, an object $\Gamma.A \in \mathcal{C}$ called the ***comprehension*** of $A$
- $p$ (resp. $v$) associates each $\Gamma \in \mathcal{C}$, $A \in Ty(\Gamma)$ with a morphism $p(A) : \Gamma.A \to \Gamma$ in $\mathcal{C}$ (resp. term $v_A \in Tm(\Gamma.A, A\{p(A)\})$) called the ***first projection*** (resp. ***second projection***) associated to $A$
- $\langle _, _ \rangle__$ assigns, to each $\phi : \Delta \to \Gamma$, $A \in Ty(\Gamma)$, $\tau \in Tm(\Delta, A\{\phi\})$, a morphism $\langle \phi, \tau \rangle_A : \Delta \to \Gamma.A$ in $\mathcal{C}$ called the ***extension*** of $\phi$ by $\tau$

that satisfy the following equations:

- (Ty-Id) $A\{id_\Gamma\} = A$
- (Ty-C) $A\{\phi \circ \psi\} = A\{\phi\}\{\psi\}$
- (Tm-Id) $f\{id_\Gamma\}_A = f$
- (Tm-C) $f\{\phi \circ \psi\}_A = f\{\phi\}_A\{\psi\}_{A\{\phi\}}$
- (Cons-L) $p(A) \circ \langle \phi, g \rangle_A = \phi$
- (Cons-R) $v_A\{\langle \phi, g \rangle_A\}_{A\{p(A)\}} = g$
- (Cons-Nat) $\langle \phi, g \rangle_A \circ \psi = \langle \phi \circ \psi, g\{\psi\}_{A\{\phi\}} \rangle_A$
- (Cons-Id) $\langle p(A), v_A \rangle_A = id_{\Gamma.A}$

for all $\Gamma, \Delta, \Theta \in \mathcal{C}$, $A \in Ty(\Gamma)$, $\phi : \Delta \to \Gamma$, $\psi : \Theta \to \Delta$, $f \in Tm(\Gamma, A)$, $g \in Tm(\Delta, A\{\phi\})$.

▶ Notation. We usually omit the subscript $A$ in $_\{_\}_A$ when it does not cause any ambiguity.

CwFs are very close to the syntax, which is best described by the following *term model*:

▶ **Definition 3.2** (Term model $\mathcal{T}$ [Hof97])**.** The CwF of contexts and context morphisms of $\mathsf{MLTT}(1, \Pi, \Sigma)$, called the ***term model*** $\mathcal{T}$, is defined as follows:

- The underlying category $\mathcal{T}$ consists of contexts and context morphisms of $\mathsf{MLTT}(1, \Pi, \Sigma)$ defined in Sec. 2 modulo (judgmental) equality $\equiv$. Let us write $[\Gamma]$ (resp. $[\mathsf{A}]$, $[\mathsf{a}]$) for the equivalence class of a context $\vdash \Gamma\ \mathsf{ctx}$ (resp. type $\Gamma \vdash \mathsf{A}\ \mathsf{type}$, term $\Gamma \vdash \mathsf{a} : \mathsf{A}$).
- For any object $[\Gamma] \in \mathcal{T}$, $Ty([\Gamma]) \stackrel{\text{df.}}{=} \{[\mathsf{A}] \mid \Gamma \vdash \mathsf{A}\ \mathsf{type}\}$ and $Tm([\Gamma], [\mathsf{A}]) \stackrel{\text{df.}}{=} \{[\mathsf{M}] \mid \Gamma \vdash \mathsf{M} : \mathsf{A}\}$. Below, we represent these equivalence classes by their arbitrary representatives.
- The functions $_\{_\}$ are generalized substitutions $_[_]$ on types and terms in Sec. 2.
- The terminal object is (the equivalence class of) the empty context $\vdash \diamondsuit\ \mathsf{ctx}$.
- Given $\vdash \Gamma\ \mathsf{ctx} \in \mathcal{T}$, $\Gamma \vdash \mathsf{A}\ \mathsf{type} \in Ty(\Gamma)$, $\Gamma.\mathsf{A} \stackrel{\text{df.}}{=} \vdash \Gamma, \mathsf{x} : \mathsf{A}\ \mathsf{ctx}$, where $\mathsf{x}$ is a *fresh variable*.

- Assume $\vdash \Gamma \equiv \mathsf{x}_1 : \mathsf{A}_1, \mathsf{x}_2 : \mathsf{A}_2, \ldots, \mathsf{x}_n : \mathsf{A}_n$ ctx. Then $p(\mathsf{A}) \stackrel{\text{df.}}{=} \langle \mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n \rangle : \Gamma.\mathsf{A} \to \Gamma$ and $v_{\mathsf{A}} \stackrel{\text{df.}}{=} \Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{x} : \mathsf{A} \in Tm(\Gamma.\mathsf{A}, \mathsf{A}[p(\mathsf{A})])$.
- Given $\phi : \Delta \to \Gamma$, $\Delta \vdash \mathsf{M} : \mathsf{A}[\phi] \in Tm(\Gamma, \mathsf{A}[\phi])$, $\langle \phi, \mathsf{M} \rangle_A \stackrel{\text{df.}}{=} \langle \phi, \mathsf{M} \rangle : \Gamma \to \Delta.\mathsf{A}$.

Nevertheless, CwFs model only the syntax common to all types; for a "full" interpretation of MLTT, we need to equip them with *semantic type formers* [Hof97]. This paper focuses on *strict* 1-, $\Pi$- *and* $\Sigma$*-types* for they correspond to the *cartesian closed structure* of categories.

**Definition 3.3** (CwFs with unit-type [Hof97])**.** A CwF $\mathcal{C}$ ***supports unit-type*** if:

- (Unit-Form) For any $\Gamma \in \mathcal{C}$, there is a type $1_\Gamma \in Ty(\Gamma)$
- (Unit-Intro) For any $\Gamma \in \mathcal{C}$, there is a term $\star_\Gamma \in Tm(\Gamma, 1_\Gamma)$
- (Unit-Subst) For any morphism $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, $1_\Gamma\{\phi\} = 1_\Delta$
- ($\star$-Subst) For any morphism $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, $\star_\Gamma\{\phi\} = \star_\Delta$.

It ***supports unit-type in the strict sense*** if (Unit-Uniq) for any $f \in Tm(\Gamma, 1_\Gamma)$, $f = \star_\Gamma$.

**Definition 3.4** (CwFs with $\Pi$-types [Hof97])**.** A CwF $\mathcal{C}$ ***supports $\Pi$-types*** if:

- ($\Pi$-Form) For each $\Gamma \in \mathcal{C}$, $A \in Ty(\Gamma)$, $B \in Ty(\Gamma.A)$, there is a type $\Pi(A, B) \in Ty(\Gamma)$
- ($\Pi$-Intro) For each $f \in Tm(\Gamma.A, B)$, there is a term $\lambda_{A,B}(f) \in Tm(\Gamma, \Pi(A, B))$
- ($\Pi$-Elim) For any $h \in Tm(\Gamma, \Pi(A, B))$, $a \in Tm(\Gamma, A)$, there is a term $App_{A,B}(h, a) \in Tm(\Gamma, B\{\overline{a}\})$, where $\overline{a} \stackrel{\text{df.}}{=} \langle id_\Gamma, a \rangle_A : \Gamma \to \Gamma.A$
- ($\Pi$-Comp) $App_{A,B}(\lambda_{A,B}(f), a) = f\{\overline{a}\}$
- ($\Pi$-Subst) For any $\Delta \in \mathcal{C}$, $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, $\Pi(A, B)\{\phi\} = \Pi(A\{\phi\}, B\{\phi^+\}$, where $\phi^+ \stackrel{\text{df.}}{=} \langle \phi \circ p(A\{\phi\}), v_{A\{\phi\}} \rangle_A : \Delta.A\{\phi\} \to \Gamma.A$
- ($\lambda$-Subst) For any $g \in Tm(\Gamma.A, B)$, $\lambda_{A,B}(g)\{\phi\} = \lambda_{A\{\phi\},B\{\phi^+\}}(g\{\phi^+\}) \in Tm(\Delta, \Pi(A\{\phi\}, B\{\phi^+\}))$
- (App-Subst) $App_{A,B}(h, a)\{\phi\} = App_{A\{\phi\},B\{\phi^+\}}(h\{\phi\}, a\{\phi\}) \in Tm(\Delta, B\{\overline{a} \circ \phi\})$.

It ***supports $\Pi$-type in the strict sense*** if ($\lambda$-Uniq) given $k \in Tm(\Gamma.A, \Pi(A, B)\{p(A)\})$, we have:

$$\lambda_{A\{p(A)\},B\{p(A)^+\}}(App_{A\{p(A)\},B\{p(A)^+\}}(k, v_A)) = k.$$

**Definition 3.5** (CwFs with $\Sigma$-types [Hof97])**.** A CwF $\mathcal{C}$ ***supports $\Sigma$-types*** if:

- ($\Sigma$-Form) For any $\Gamma \in \mathcal{C}$, $A \in Ty(\Gamma)$, $B \in Ty(\Gamma.A)$, there is a type $\Sigma(A, B) \in Ty(\Gamma)$
- ($\Sigma$-Intro) There is a morphism $Pair_{A,B} : \Gamma.A.B \to \Gamma.\Sigma(A, B)$ in $\mathcal{C}$
- ($\Sigma$-Elim) Given $P \in Ty(\Gamma.\Sigma(A, B))$, $f \in Tm(\Gamma.A.B, P\{Pair_{A,B}\})$, there is a term $\mathcal{R}^{\Sigma}_{A,B,P}(f) \in Tm(\Gamma.\Sigma(A, B), P)$
- ($\Sigma$-Comp) $\mathcal{R}^{\Sigma}_{A,B,P}(f)\{Pair_{A,B}\} = f$
- ($\Sigma$-Subst) For any $\Delta \in \mathcal{C}$, $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, $\Sigma(A, B)\{\phi\} = \Sigma(A\{\phi\}, B\{\phi^+\})$

- (Pair-Subst) $p(\Sigma(A,B)) \circ Pair_{A,B} = p(A) \circ p(B)$, $\phi^* \circ Pair_{A\{\phi\},B\{\phi^+\}} = Pair_{A,B} \circ \phi^{++}$, where $\phi^* \overset{\text{df.}}{=} \langle \phi \circ p(\Sigma(A,B)\{\phi\}), v_{\Sigma(A,B)\{\phi\}} \rangle_{\Sigma(A,B)} : \Delta.\Sigma(A,B)\{\phi\} \to \Gamma.\Sigma(A,B)$, $\phi^{++} \overset{\text{df.}}{=} \langle \phi^+ \circ p(B\{\phi^+\}), v_{B\{\phi^+\}} \rangle_B : \Delta.A\{\phi\}.B\{\phi^+\} \to \Gamma.A.B$
- ($\mathcal{R}^\Sigma$-Subst) $\mathcal{R}^\Sigma_{A,B,P}(f)\{\phi^*\} = \mathcal{R}^\Sigma_{A\{\phi\},B\{\phi^+\},P\{\phi^*\}}(f\{\phi^{++}\})$.

It ***supports $\Sigma$-types in the strict sense*** if ($\mathcal{R}^\Sigma$-Uniq) any $g \in Tm(\Gamma.\Sigma(A,B),P)$ with $g\{Pair_{A,B}\} = f$ satisfies $g = \mathcal{R}^\Sigma_{A,B,P}(f)$.

In light of Def. 3.2, it is rather straightforward to see these semantic type formers in the term model $\mathcal{T}$. Therefore we do not describe them here; see [Hof97] for their details.

## 3.1 Interpretation of MLTT in CwFs

Now, let us recall the interpretation of $\mathsf{MLTT}(1,\Pi,\Sigma)$ in CwFs $\mathcal{C}$ equipped with semantic 1-, $\Pi$-, $\Sigma$-types in the strict sense [Hof97]. Since a deduction of a judgement in MLTT is not unique in the presence of the rules TY-CON, TM-CON, we cannot define the interpretation by induction on deductions. For this problem, a standard approach is to define the interpretation $[\![_]\!]$ on *pre-syntax* which is *partial*, and show that it is well-defined on every valid syntax and preserves judgmental equality as the corresponding semantic equality [Hof97]. By this *soundness*, we may describe the interpretation $[\![_]\!]$ of the syntax by induction on derivation of judgements:

▶ **Definition 3.6** (Interpretation of MLTT in CwFs [Hof97])**.** The interpretation $[\![_]\!]$ of $\mathsf{MLTT}(1,\Pi,\Sigma)$ in a CwF $\mathcal{C} = (\mathcal{C}, Ty, Tm, _\{_\}, T, _._, p, v, \langle _,_ \rangle_)$ equipped with semantic type formers $1 = (1, \star)$, $\Pi = (\Pi, \lambda, App)$, $\Sigma = (\Sigma, Pair, R^\Sigma)$ in the strict sense is defined as follows:

- (CT-EMP) $[\![\vdash \diamondsuit\ \mathsf{ctx}]\!] \overset{\text{df.}}{=} T$
- (CT-EXT) $[\![\vdash \Gamma, \mathsf{x} : \mathsf{A}\ \mathsf{ctx}]\!] \overset{\text{df.}}{=} [\![\vdash \Gamma\ \mathsf{ctx}]\!].[\![\Gamma \vdash \mathsf{A}\ \mathsf{type}]\!]$
- (1-FORM) $[\![\Gamma \vdash 1\ \mathsf{type}]\!] \overset{\text{df.}}{=} 1_{[\![\Gamma]\!]}$
- ($\Pi$-FORM) $[\![\Gamma \vdash \Pi_{\mathsf{x}:\mathsf{A}}\mathsf{B}\ \mathsf{type}]\!] \overset{\text{df.}}{=} \Pi([\![\Gamma \vdash \mathsf{A}\ \mathsf{type}]\!], [\![\Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{B}\ \mathsf{type}]\!])$
- ($\Sigma$-FORM) $[\![\Gamma \vdash \Sigma_{\mathsf{x}:\mathsf{A}}\mathsf{B}\ \mathsf{type}]\!] \overset{\text{df.}}{=} \Sigma([\![\Gamma \vdash \mathsf{A}\ \mathsf{type}]\!], [\![\Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{B}\ \mathsf{type}]\!])$
- (VAR) $[\![\Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{x} : \mathsf{A}]\!] \overset{\text{df.}}{=} v_{[\![\mathsf{A}]\!]}$, $[\![\Gamma, \mathsf{x} : \mathsf{A}, \Delta, \mathsf{y} : \mathsf{B} \vdash \mathsf{x} : \mathsf{A}]\!] \overset{\text{df.}}{=} [\![\Gamma, \mathsf{x} : \mathsf{A}, \Delta \vdash \mathsf{x} : \mathsf{A}]\!]\{p([\![\Gamma, \mathsf{x} : \mathsf{A}, \Delta \vdash \mathsf{B}\ \mathsf{type}]\!])\}$
- (TY-CON) $[\![\Delta \vdash \mathsf{A}\ \mathsf{type}]\!] \overset{\text{df.}}{=} [\![\Gamma \vdash \mathsf{A}\ \mathsf{type}]\!]$
- (TM-CON) $[\![\Delta \vdash \mathsf{a} : \mathsf{B}]\!] \overset{\text{df.}}{=} [\![\Gamma \vdash \mathsf{a} : \mathsf{A}]\!]$
- (1-INTRO) $[\![\Gamma \vdash \star : 1]]\!] \overset{\text{df.}}{=} \star_{[\![\Gamma]\!]}$
- ($\Pi$-INTRO) $[\![\Gamma \vdash \lambda\mathsf{x}.\mathsf{b} : \Pi_{\mathsf{x}:\mathsf{A}}\mathsf{B}]\!] \overset{\text{df.}}{=} \lambda_{[\![\mathsf{A}]\!],[\![\mathsf{B}]\!]}([\![\Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{b} : \mathsf{B}]\!])$
- ($\Sigma$-INTRO) $[\![\Gamma \vdash (\mathsf{a},\mathsf{b}) : \Sigma_{\mathsf{x}:\mathsf{A}}\mathsf{B}]\!] \overset{\text{df.}}{=} Pair_{[\![\mathsf{A}]\!],[\![\mathsf{B}]\!]} \circ \langle \overline{[\![\Gamma \vdash \mathsf{a} : \mathsf{A}]\!]}, [\![\Gamma \vdash \mathsf{b} : \mathsf{B}[\mathsf{a}/\mathsf{x}]]\!] \rangle$
- ($\Pi$-ELIM) $[\![\Gamma \vdash \mathsf{f}(\mathsf{a}) : \mathsf{B}[\mathsf{a}/\mathsf{x}]]\!] \overset{\text{df.}}{=} App_{[\![\mathsf{A}]\!],[\![\mathsf{B}]\!]}([\![\Gamma \vdash \mathsf{f} : \Pi_{\mathsf{x}:\mathsf{A}}\mathsf{B}]\!], [\![\Gamma \vdash \mathsf{a} : \mathsf{A}]\!]\rangle)$
- ($\Sigma$-ELIM) $[\![\Gamma \vdash \mathsf{R}^\Sigma(\mathsf{C},\mathsf{g},\mathsf{p}) : \mathsf{C}[\mathsf{p}/\mathsf{z}]]\!] \overset{\text{df.}}{=} R^\Sigma_{[\![\mathsf{A}]\!],[\![\mathsf{B}]\!],[\![\mathsf{C}]\!]}([\![\Gamma, \mathsf{x} : \mathsf{A}, \mathsf{y} : \mathsf{B} \vdash \mathsf{g} : \mathsf{C}[(\mathsf{x},\mathsf{y})/\mathsf{z}]]\!]) \circ \overline{[\![\Gamma \vdash \mathsf{p} : \Sigma_{\mathsf{x}:\mathsf{A}}\mathsf{B}]\!]}$

where the hypotheses of the rules are as presented in Sec. 2, $\overline{[\![\Gamma \vdash a : A]\!]} \stackrel{\text{df.}}{=} \langle id_{[\![\Gamma]\!]}, [\![a]\!] \rangle : [\![\Gamma]\!] \to [\![\Gamma]\!].[\![A]\!]$ and $\overline{[\![\Gamma \vdash p : \Sigma_{x:A} B]\!]} \stackrel{\text{df.}}{=} \langle id_{[\![\Gamma]\!]}, [\![p]\!] \rangle : [\![\Gamma]\!] \to [\![\Gamma]\!].[\![\Sigma_{x:A} B]\!]$.

By the term model $\mathcal{T}$ in Def. 3.2, this interpretation is *complete*:

▶ **Theorem 3.7** (Completeness of CwFs [Hof97]). *For any contexts* $\vdash \Gamma$ ctx, $\vdash \Delta$ ctx *(resp. types* $\Gamma \vdash A$ type, $\Gamma \vdash B$ type*, terms* $\Gamma \vdash a : A$, $\Gamma \vdash a' : A$*) in MLTT, if* $[\![\Gamma]\!] = [\![\Delta]\!]$ *(resp.* $[\![A]\!] = [\![B]\!]$, $[\![a]\!] = [\![a']\!]$*) for the interpretation* $[\![_]\!]$ *in any CwF with the corresponding type formers, then there is a judgement* $\vdash \Gamma \equiv \Delta$ ctx *(resp.* $\Gamma \vdash A \equiv B$ type, $\Gamma \vdash a \equiv a' : A$*) in MLTT.*

## 3.2 Interpretation of STLC in CCCs

Since $\lambda^{=}_{1,\times}$ is a restriction of $\mathsf{MLTT}(1, \Pi, \Sigma)$, we may simply apply Def. 3.6 to interpret it in CwFs. However, the standard approach in categorical logic [Jac99, Pit01] is to interpret it in CCCs, where contexts and types are both interpreted by objects, and terms by morphisms:

▶ **Definition 3.8** (Interpretation of $\lambda^{=}_{1,\times}$ in CCCs [Pit01, Jac99, LS88]). The interpretation $\wr _ \wr$ of the equational theory $\lambda^{=}_{1,\times}$ in a CCC $\mathcal{C} = (\mathcal{C}, T, \times, \Rightarrow)$ is given as follows:

- (Ct-Emp) $\wr \vdash \diamondsuit \text{ ctx} \wr \stackrel{\text{df.}}{=} T$
- (Ct-Ext) $\wr \vdash \Gamma, x : A \text{ ctx} \wr \stackrel{\text{df.}}{=} \wr \vdash \Gamma \text{ ctx} \wr \times \wr \vdash A \text{ type} \wr$
- (1-Form) $\wr \vdash 1 \text{ type} \wr \stackrel{\text{df.}}{=} T$
- ($\Rightarrow$-Form) $\wr \vdash A \Rightarrow B \text{ type} \wr \stackrel{\text{df.}}{=} \wr \vdash A \text{ type} \wr \Rightarrow \wr \vdash B \text{ type} \wr$
- ($\times$-Form) $\wr \vdash A \times B \text{ type} \wr \stackrel{\text{df.}}{=} \wr \vdash A \text{ type} \wr \times \wr \vdash B \text{ type} \wr$
- (Var) $\wr x_1 : A_1, \ldots, x_j : A_j, \ldots, x_n : A_n \vdash x_j : A_j \wr \stackrel{\text{df.}}{=} \pi_j$
- (1-Intro) $\wr \Gamma \vdash \star : 1 \wr \stackrel{\text{df.}}{=} !_{\wr \Gamma \wr}$
- ($\Rightarrow$-Intro) $\wr \Gamma \vdash \lambda x.b : A \Rightarrow B \wr \stackrel{\text{df.}}{=} \lambda_{\wr A \wr, \wr B \wr}(\wr \Gamma, x : A \vdash b : B \wr)$
- ($\times$-Intro) $\wr \Gamma \vdash (a, b) : A \times B \wr \stackrel{\text{df.}}{=} \langle \wr \Gamma \vdash a : A \wr, \wr \Gamma \vdash b : B \wr \rangle$
- ($\Rightarrow$-Elim) $\wr \Gamma \vdash f(a) : B \wr \stackrel{\text{df.}}{=} ev_{\wr A \wr, \wr B \wr} \circ \langle \wr \Gamma \vdash f : A \Rightarrow B \wr, \wr \Gamma \vdash a : A \wr \rangle$
- ($\times$-Elim) $\wr \Gamma \vdash \pi_i[p] : A_i \wr \stackrel{\text{df.}}{=} \pi_i \circ \wr \Gamma \vdash p : A_1 \times A_2 \wr$

where the hypotheses of the rules are as presented in Sec. 2, and $\pi_j$, $\lambda_{\wr A \wr, \wr B \wr}$, $\langle _, _ \rangle$, $ev_{\wr A \wr, \wr B \wr}$ are the *$j^{th}$-projection*, *currying*, *pairing* and *evaluation* in $\mathcal{C}$, respectively.

▶ Remark. As each judgement in $\lambda^{=}_{1,\times}$ has a *unique* derivation, there is no problem in Def. 3.8.

In contrast, the term model or the ***classifying category*** $cl(\lambda^{=}_{1,\times})$ of $\lambda^{=}_{1,\times}$ has several variations. For instance, [Jac99, Cro93] both define it as the CCC of types and terms, while [Pit01] as the CCC of contexts and context morphisms. If we regard types $\vdash A$ type as contexts $\vdash \diamondsuit, x : A$ ctx, the latter appears very similar to the restriction $\mathcal{L}$ of $\mathcal{T}$ (in Def. 3.2) to $\lambda^{=}_{1,\times}$, but not exactly yet as, e.g., $\mathcal{T}$ does not have a product of any given two contexts.

Nevertheless, in light of the rather simple relation between $\mathsf{MLTT}(1, \Pi, \Sigma)$ and $\lambda^{=}_{1,\times}$, the gaps between the interpretations of $\lambda^{=}_{1,\times}$ in Def. 3.6, 3.8 and between the term models $\mathcal{L}$, $cl(\lambda^{=}_{1,\times})$ should not be the case. This is what has been called a problem to solve in the introduction.

# 4 Dependent cartesian closed categories

We have reviewed necessary backgrounds; we shall develop our central notion in this section.

## 4.1 Dependent categories

Let us begin with a generalization of categories to handle dependent types:

▶ **Definition 4.1** (Dependent categories)**.** A ***dependent category (DC)*** is a category $\mathcal{C}$ equipped with a pair $\mathscr{D}, _\{_\}$ such that:

- $\mathscr{D}$ assigns to each $\Gamma \in \mathcal{C}$ a class $\mathscr{D}(\Gamma)$ of ***dependent objects*** over $\Gamma$, and to each $A \in \mathscr{D}(\Gamma)$ a class $\mathscr{D}(\Gamma, A)$ of ***dependent morphisms*** from $\Gamma$ to $A$
- $_\{_\}$ assigns to each $\phi : \Delta \to \Gamma$ in $\mathcal{C}$ a (class) function $_\{\phi\} : \mathscr{D}(\Gamma) \to \mathscr{D}(\Delta)$ and a family $(_\{\phi\}_A : \mathscr{D}(\Gamma, A) \to \mathscr{D}(\Delta, A\{\phi\}))_{A \in \mathscr{D}(\Gamma)}$ of (class) functions, which are all called ***dependent compositions***, where any $A \in \mathscr{D}(\Gamma)$ (resp. $f \in \mathscr{D}(\Gamma, A)$) is ***constant*** if $A\{\phi_1\} = A\{\phi_2\}$ (resp. $f\{\phi_1\}_A = f\{\phi_2\}_A$) for all $\Delta \in \mathcal{C}$, $\phi_1, \phi_2 : \Delta \to \Gamma$ in $\mathcal{C}$

It is ***constant*** if its dependent objects are all constant.

▶ Notation. We write $f : \Gamma \twoheadrightarrow A$ and say $f$ is a dependent morphism from $\Gamma$ to $A$ in $\mathcal{C}$ if $f \in \mathscr{D}(\Gamma, A)$ in a DC $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$. We usually omit the subscript $A$ in $_\{_\}_A$.

▶ Example 4.2. Every category $\mathcal{C}$ is a DC as follows. Let $\mathscr{D}(\Gamma) \stackrel{\text{df.}}{=} \mathsf{ob}(\mathcal{C})$ for each $\Gamma \in \mathcal{C}$, and $\mathscr{D}(\Gamma, \Theta) \stackrel{\text{df.}}{=} \mathcal{C}(\Gamma, \Theta)$ for each $\Theta \in \mathscr{D}(\Gamma)$. Given $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, let the function $_\{\phi\} : \mathscr{D}(\Gamma) \to \mathscr{D}(\Delta)$ to be the identity function on $\mathsf{ob}(\mathcal{C})$, and the function $_\{\phi\}_\Theta : \mathscr{D}(\Gamma, \Theta) \to \mathscr{D}(\Delta, \Theta)$ for each $\Theta \in \mathscr{D}(\Gamma)$ to be the pre-composition function with $\phi$, i.e., $f \mapsto f \circ \phi$ for all $f \in \mathscr{D}(\Gamma, \Theta)$.

▶ Example 4.3. The category $\mathit{Sets}$ of sets and functions may be turned into a DC as follows. Let a dependent object over a set $X$, called a ***dependent set*** over $X$, to be a set $A = \{A(x) \in \mathit{Sets} \mid x \in X\}$, and a dependent morphism $X \twoheadrightarrow A$, called a ***dependent function*** from $\Gamma$ to $A$, to be a function $f : X \to \bigcup_{x \in X} A(x)$ with $f(x_0) \in A(x_0)$ for all $x_0 \in X$. Given a function $\phi : Y \to X$, let $A\{\phi\} \stackrel{\text{df.}}{=} \{A(\phi(y)) \mid y \in Y\}$ and $f\{\phi\} \stackrel{\text{df.}}{=} f \circ \phi : (y \in Y) \mapsto f(\phi(y)) \in A\{\phi\}(y)$.

## 4.2 Dependent cartesian categories

We may clarify a connection of DCs with CwFs by a generalization of finite products:

▶ **Definition 4.4** (DSCCs)**.** A ***dependent semi-cartesian category (DSCC)*** is a DC $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$ such that there exist a terminal object $T \in \mathcal{C}$, an object $\Gamma.A \in \mathcal{C}$, called the ***dependent pair space*** of $\Gamma$ and $A$, a morphism $\pi_1 : \Gamma.A \to \Gamma$, called the ***first projection***, and a dependent morphism $\pi_2 : \Gamma.A \twoheadrightarrow A\{\pi_1\}$, called the ***second projection***, for each $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, that satisfies the following axioms:

- (Bases) Given $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, if $A$ is constant, then there exists a unique $\hat{A} \in \mathscr{D}(T)$, called the ***base*** of $A$, such that $\hat{A}\{!_\Gamma\} = A$, where $!_\Gamma : \Gamma \to T$ is the unique morphism in $\mathcal{C}$
- (Unit law and associativity) Given $\Gamma, \Delta, \Theta \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, $\phi : \Delta \to \Gamma, \psi : \Theta \to \Delta$, $f : \Gamma \twoheadrightarrow A$ in $\mathcal{C}$, $A\{id_\Gamma\} = A$, $f\{id_\Gamma\}_A = f$, $A\{\phi \circ \psi\} = A\{\phi\}\{\psi\}$ and $f\{\phi \circ \psi\} = f\{\phi\}\{\psi\}$
- (Dependent pairings) Given $\Gamma, \Delta \in \mathcal{C}$, $\phi : \Delta \to \Gamma$, $g : \Delta \twoheadrightarrow A\{\phi\}$ in $\mathcal{C}$, there exists a unique morphism $\langle \phi, g \rangle : \Delta \to \Gamma.A$, called the ***dependent pairing*** of $\phi$ and $g$, that satisfies $\pi_1 \circ \langle \phi, g \rangle = \phi$ and $\pi_2\{\langle \phi, g \rangle\} = g$.

It is ***recursive*** if the map $(\Gamma \in \mathcal{C}, A \in \mathscr{D}(\Gamma)) \mapsto \Gamma.A$ is a bijection with codomain $\mathsf{ob}(\mathcal{C}) \setminus \{T\}$.

▶ Example 4.5. Every cartesian category (CC), i.e., a category $\mathcal{C}$ with finite products, is a DSCC since dependent pair spaces and projections are just products and projections in $\mathcal{C}$.

▶ **Theorem 4.6** (DSCCs as CwFs)**.** *Every DSCC is a CwF but not vice versa.*

*Proof.* Given a DSCC $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$, we define a CwF $\mathcal{C}$ as follows. Let $Ty(\Gamma) \stackrel{\text{df.}}{=} \mathscr{D}(\Gamma)$ for each $\Gamma \in \mathcal{C}$ and $Tm(\Gamma, A) \stackrel{\text{df.}}{=} \mathscr{D}(\Gamma, A)$ for each $A \in \mathscr{D}(\Gamma)$. We define substitutions to be the corresponding dependent compositions. Finally, we define the triple $\Gamma.A, p(A), v_A$ to be $\Gamma \stackrel{\pi_1}{\leftarrow} \Sigma(\Gamma, A) \stackrel{\pi_2}{\twoheadrightarrow} A\{\pi_1\}$ for each $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, and extensions to be the corresponding dependent pairings. It is then straightforward to verify the required equations. Since a CwF does not necessarily have bases or dependent pairings, the converse does not hold. ■

Therefore one may say that DSCCs are a *generalization of CCs* as well as a *refinement of CwFs*. See the following diagram that depicts the universal property of dependent pair spaces, which naturally generalizes that of products, where the arrow $A\{\pi_1\} \Rightarrow A\{\phi\}$ denotes the transformation of the codomain when $\langle \phi, g \rangle$ is substituted to $\pi_2$:

$$\begin{array}{ccccc} \Gamma & \xleftarrow{\phi} & \Delta & \xrightarrow{g} & A\{\phi\} \\ \| & & \vdots\,\langle \phi, g \rangle & & \Uparrow \\ \Gamma & \xleftarrow{\pi_1} & \Gamma.A & \xrightarrow{\pi_2} & A\{\pi_1\} \end{array}$$

Also, we have chosen the word "refinement" since the formulation of DSCCs naturally generalizes that of CCs, and most examples of a CwF are in fact DSCCs:

▶ Example 4.7. By the inverse of the translation from DSCCs to CwFs in the proof of Thm. 4.6, it is easy to see that the term model $\mathcal{T}$ is a DSCC. Similarly, the CwFs of groupoids in the classic paper [Hof97] and of games in the recent unpublished papers [Yam16, Yam17] are DSCCs.

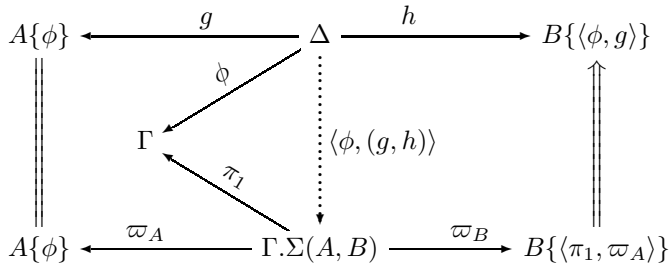
▶ Example 4.8. The DC $Sets$ may be equipped with a DSCC structure, in which the dependent product $X.A$ for each $X \in Sets$, $A \in \mathscr{D}(X)$ is defined to be the set $\{(x, a) \,|\, x \in X, a \in A(x)\}$ of ***dependent pairs*** in $X$ and $A$ with the obvious projections and dependent pairings.

▶ **Definition 4.9** (DCCs)**.** A ***terminal dependent object*** in a DSCC $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$ is a constant dependent object $1 \in \mathscr{D}(T)$, where $T \in \mathcal{C}$ is terminal, such that there exists just one dependent morphism from $\Gamma$ to $1\{!_\Gamma\}$ for each $\Gamma \in \mathcal{C}$. For each $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, $B \in \mathscr{D}(\Gamma.A)$, a ***dependent pair space*** of $A$ and $B$ in $\mathcal{C}$ is a dependent object $\Sigma(A, B) \in \mathscr{D}(\Gamma)$ such that $\Sigma(A, B)\{\phi\} = \Sigma(A\{\phi\}, B\{\phi^+\})$ for all $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, where $\phi^+ \stackrel{\text{df.}}{=} \langle \phi \circ \pi_1, \pi_2 \rangle : \Delta.A\{\phi\} \to \Gamma.A$, equipped with dependent morphisms $\varpi_A : \Gamma.\Sigma(A, B) \twoheadrightarrow A\{\pi_1\}$, $\varpi_B : \Gamma.\Sigma(A, B) \twoheadrightarrow B\{\langle \pi_1, \varpi_A \rangle\}$ in $\mathcal{C}$, called the ***(first and second) projections*** for $\Sigma(A, B)$, respectively, such that for any $\phi : \Delta \to \Gamma$, $g : \Delta \twoheadrightarrow A\{\phi\}$, $h : \Delta \twoheadrightarrow B\{\langle \phi, g \rangle\}$ in $\mathcal{C}$ there exists a unique dependent morphism $(g, h) : \Delta \twoheadrightarrow \Sigma(A, B)$ in $\mathcal{C}$, called the ***dependent pairing*** of $g$ and $h$, that satisfies $\varpi_A\{\langle \phi, (g, h) \rangle\} = g$ and $\varpi_B\{\langle \phi, (g, h) \rangle\} = h$. A ***dependent cartesian category (DCC)*** is a DSCC with a terminal dependent object and dependent pair spaces.

In terms of MLTT, a terminal dependent object and new dependent pair spaces in DCCs correspond respectively to 1- and $\Sigma$-types, while a terminal object and dependent pair spaces in DSCCs to the empty context and context extensions. Then naturally, every DSCC induced by a CC $\mathcal{C} = (\mathcal{C}, T, \times)$ as described in Exp. 4.5 forms a DCC since $\mathcal{C}$ has no distinction between objects

(or contexts) and dependent objects (or types), in which $1 \stackrel{\text{df.}}{=} T$, $\Sigma(A,B) \stackrel{\text{df.}}{=} A \times B$, $\varpi_A \stackrel{\text{df.}}{=} \pi_1 \circ \pi_2$, $\varpi_B \stackrel{\text{df.}}{=} \pi_2 \circ \pi_2$ and $(g,h) \stackrel{\text{df.}}{=} \langle g,h\rangle$ for each $\Gamma,\Delta \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, $B \in \mathscr{D}(\Gamma.A)$, $g : \Delta \to A$, $h : \Delta \to B$ in $\mathcal{C}$. In fact, this is why the standard approach to categorical logic [Pit01, Jac99] only needs finite products to interpret simple type theories in categories. However, it is not the case for dependent type theories, for which we need two kinds of (generalized) terminal objects and products in DCs. We shall come back to this point in Sec. 5.

One may see the similarity between the two kinds of dependent pair spaces in DCCs in the following diagram that depicts the universal property of the new one:




▶ **Theorem 4.10** (DCCs as CwFs with 1- and $\Sigma$-types)**.** *Every DSCC with a terminal dependent object (resp. dependent pairs spaces) is a CwF with* 1*-type (resp.* $\Sigma$*-types).*

*Proof.* We only handle $\Sigma$-types as 1-type is easy. Let $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$ be a DSCC with dependent pair spaces. By Thm. 4.6, it suffices to equip the induced CwF $\mathcal{C}$ with $\Sigma$-types.

- ($\Sigma$-Form) Let $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, $B \in \mathscr{D}(\Gamma.A)$; then, we have $\Sigma(A,B) \in \mathscr{D}(\Gamma)$.
- ($\Sigma$-Intro) We define $Pair_{A,B} \stackrel{\text{df.}}{=} \langle \pi_1 \circ \pi_1, (\pi_2\{\pi_1\}, \pi_2)\rangle : (\Gamma.A).B \to \Gamma.\Sigma(A,B)$. It is easy to see that $\langle\langle \pi_1, \varpi_A\rangle, \varpi_B\rangle : \Gamma.\Sigma(A,B) \to (\Gamma.A).B$ is the inverse $Pair^{-1}_{A,B}$.
- ($\Sigma$-Elim) Given $P \in \mathscr{D}(\Sigma(\Gamma,\Sigma(A,B)))$, $f \in Tm(\Sigma(\Sigma(\Gamma,A),B), P\{Pair_{A,B}\})$, we define $\mathcal{R}^{\Sigma}_{A,B,P}(f) \stackrel{\text{df.}}{=} f\{Pair^{-1}_{A,B}\} \in Tm(\Sigma(\Gamma,\Sigma(A,B)),P)$.
- ($\Sigma$-Comp) $\mathcal{R}^{\Sigma}_{A,B,P}(f)\{Pair_{A,B}\} = f\{Pair^{-1}_{A,B}\}\{Pair_{A,B}\} = f\{Pair^{-1}_{A,B} \circ Pair_{A,B}\} = f\{id_{(\Gamma.A).B}\} = f$.
- ($\Sigma$-Subst) $\Sigma(A,B)\{\phi\} = \Sigma(A\{\phi\}, B\{\phi^+\})$ just by the axiom of DCCs.
- (Pair-Subst) $p(\Sigma(A,B)) \circ Pair_{A,B} = \pi_1 \circ \langle \pi_1 \circ \pi_1, (\pi_2\{\pi_1\}, \pi_2)\rangle = \pi_1 \circ \pi_1 = p(A) \circ p(B)$;

$$\begin{aligned} \phi^\star \circ Pair_{A\{\phi\},B\{\phi^+\}} &= \langle \phi \circ \pi_1, \pi_2\rangle \circ \langle \pi_1 \circ \pi_1, (\pi_2\{\pi_1\}, \pi_2)\rangle \\ &= \langle \phi \circ \pi_1 \circ \pi_1, (\pi_2\{\pi_1\}, \pi_2)\rangle \\ &= \langle \pi_1 \circ \pi_1, (\pi_2\{\pi_1\}, \pi_2)\rangle \circ \langle\langle \phi \circ \pi_1, \pi_2\rangle \circ \pi_1, \pi_2\rangle \\ &= Pair_{A,B} \circ \phi^{++} \end{aligned}$$

where $\phi^\star \stackrel{\text{df.}}{=} \langle \phi \circ p(\Sigma(A,B)\{\phi\}), v_{\Sigma(A,B)\{\phi\}}\rangle : \Delta.\Sigma(A,B)\{\phi\} \to \Gamma.\Sigma(A,B)$, $\phi^{++} \stackrel{\text{df.}}{=} \langle \phi^+ \circ p(B\{\phi^+\}), v_{B\{\phi^+\}}\rangle : (\Delta.A\{\phi\}).B\{\phi^+\} \to (\Gamma.A).B$.

► ($\mathcal{R}^{\Sigma}$-Subst) Finally, we have:

$$\begin{aligned}
\mathcal{R}^{\Sigma}_{A,B,P}(f)\{\phi^{\star}\} &= f\{Pair^{-1}_{A,B}\}\{\langle \phi \circ p(\Sigma(A,B)\{\phi\}, v_{\Sigma(A,B)\{\phi\}}\rangle\} \\
&= f\{\langle\langle \pi_1, \varpi_A\rangle, \varpi_B\rangle \circ \langle \phi \circ \pi_1, \pi_2\rangle\} \\
&= f\{\langle\langle \phi \circ \pi_1, \varpi_A\{\langle \phi \circ \pi_1, \pi_2\rangle\}\rangle, \varpi_B\{\langle \phi \circ \pi_1, \pi_2\rangle\}\rangle\} \\
&= f\{\langle\langle \phi \circ \pi_1, \varpi_{A\{\phi\}}\rangle, \varpi_{B\{\phi^+\}}\rangle\} \\
&= f\{\langle \phi^+ \circ \langle \pi_1, \varpi_{A\{\phi\}}\rangle, \varpi_{B\{\phi^+\}}\rangle\} \\
&= f\{\langle \phi^+ \circ \pi_1, \pi_2\rangle\}\{\langle\langle \pi_1, \varpi_{A\{\phi\}}\rangle, \varpi_{B\{\phi^+\}}\rangle\} \\
&= f\{\phi^{++}\}\{Pair^{-1}_{A\{\phi\},B\{\phi^+\}}\} \\
&= \mathcal{R}^{\Sigma}_{A\{\phi\},B\{\phi^+\},P\{\phi^{\star}\}}(f\{\phi^{++}\})
\end{aligned}$$

which completes the proof. ■

Note that the structures and axioms of $\Sigma$-types in CwFs (see Def. 3.5) are reformulated much more concisely and "categorically", i.e., in terms of universal properties.

► Example 4.11. The DSCC $Sets$ is a DCC: The terminal dependent object $1 \in \mathscr{D}(\{\bullet\})$ is defined by $1(\bullet) \stackrel{\text{df.}}{=} \{\star\}$, where $\bullet, \star$ are arbitrary elements, and given $X \in Sets$, $A \in \mathscr{D}(X)$, $B \in \mathscr{D}(X.A)$, we define $\Sigma(A,B) \stackrel{\text{df.}}{=} \{A(x).B_x \,|\, x \in X\}$, where $B_x \stackrel{\text{df.}}{=} \{B(x,a) \,|\, a \in A(x)\}$.

► Example 4.12. The DSCCs of groupoids and games are easily turned into DCCs. It is also the case for the DSCC of the term model $\mathcal{T}$, in which a terminal dependent object is given by the rule 1-FORM, dependent pair spaces by $\Sigma$-FORM, projections by $\Sigma$-ELIM for terms $\Gamma, \mathsf{p} : \Sigma(\mathsf{A},\mathsf{B}) \vdash \pi_1[\mathsf{p}] : \mathsf{A}$, $\Gamma, \mathsf{p} : \Sigma(\mathsf{A},\mathsf{B}) \vdash \pi_2[\mathsf{p}] : \mathsf{B}$ and dependent pairings by $\Sigma$-INTRO.

## 4.3 Dependent cartesian closed categories

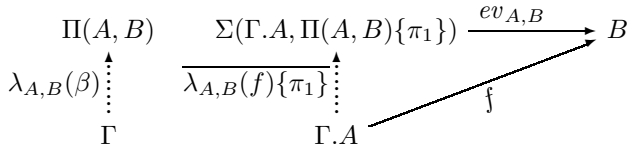
Now, we equip DCCs with a generalized *closed structure* to form a generalization of CCCs:

► **Definition 4.13** (DCCCs)**.** A ***dependent semi-cartesian closed category (DSCCC)*** is a DSCC $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$ such that given $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, $B \in \mathscr{D}(\Gamma.A)$ there exist a dependent object $\Pi(A,B) \in \mathscr{D}(\Gamma)$, called the ***dependent map space*** from $\Gamma$ to $A$, that satisfies $\Pi(A,B)\{\phi\} = \Pi(A\{\phi\}, B\{\phi^+\})$ for all $\phi : \Delta \to \Gamma$ in $\mathcal{C}$, and a dependent morphism $ev_{A,B} : \Sigma(\Gamma.A, \Pi(A,B)\{\pi_1\}) \twoheadrightarrow B\{\pi_1\}$, called the ***evaluation*** from $A$ to $B$, such that for any $f : \Gamma.A \twoheadrightarrow B$ in $\mathcal{C}$ there exists a unique dependent morphism $\lambda_{A,B}(f) : \Gamma \twoheadrightarrow \Pi(A,B)$ in $\mathcal{C}$, called the ***currying*** of $f$, that satisfies:

1. $ev_{A,B}\{\overline{\lambda_{A,B}(f)\{\pi_1\}}\} = f$, where $\overline{\lambda_{A,B}(f)\{\pi_1\}} \stackrel{\text{df.}}{=} \langle id_{\Gamma.A}, \lambda_{A,B}(f)\{\pi_1\}\rangle$
2. $\lambda_{A,B}(f)\{\phi\} = \lambda_{A\{\phi\},B\{\phi^+\}}(f\{\phi^+\})$ for all $\phi : \Delta \to \Gamma$ in $\mathcal{C}$.

It is a ***dependent cartesian closed category (DCCC)*** if $\mathcal{C}$ is a DCC.

It is easy to see that for a DCC induced by a CC, the new concepts coincide with the *closed structure*; thus, every CCC is a DCCC. The following diagram that depicts the new universal property illustrates this point:



In comparison with exponentials in categories, the point is that this new diagram in general may "carry the context $\Gamma$", gaining additional expressibility for dependent types.

▶ Example 4.14. The DCC $Sets$ forms a DCCC as follows. Given $X \in Sets$, $A \in \mathscr{D}(X)$, we define $\Pi(X, A)$ to be the set of all dependent functions from $X$ to $A$. For any dependent function $f : \Sigma(X, A) \to B$, its *currying* $\lambda_{A,B}(f) : X \to \Pi(A, B)$ maps $x \mapsto (a \mapsto f(x, a))$. The evaluation $ev_{A,B}$ is a dependent function that maps $((x, a), f) \mapsto f(a)$.

▶ Example 4.15. The DCCs of groupoids and games are also DCCCs. It is also the case for the DCC of the term model $\mathcal{T}$: Dependent map spaces are given by the rule $\Pi$-FORM, evaluations $ev_{A,B}$ by $\Pi$-ELIM for terms $\mathsf{\Gamma, x : A, f : \Pi(A, B) \vdash f(x) : B}$ and currying by $\Pi$-INTRO.

▶ **Theorem 4.16** (DSCCCs as CwFs with $\Pi$-types)**.** *Every DSCCC induces a CwF with* $\Pi$*-types.*

*Proof.* Let $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$ be a DSCCC. Again, by Thm. 4.6, it suffices to give $\Pi$-types.

- ($\Pi$-Form) Let $\Gamma \in \mathcal{C}$, $A \in \mathscr{D}(\Gamma)$, $B \in \mathscr{D}(\Sigma(\Gamma, A))$; then we have $\Pi(A, B) \in \mathscr{D}(\Gamma)$.
- ($\Pi$-Intro) For each $f \in Tm(\Sigma(\Gamma, A), B)$, we have $\lambda_{A,B}(f) \in Tm(\Gamma, \Pi(A, B))$.
- ($\Pi$-Elim) The map $ev_{A,B}\{\overline{(_)\{\pi_1\}}\} : Tm(\Gamma, \Pi(A, B)) \to Tm(\Sigma(\Gamma, A), B)$ is the inverse of $\lambda_{A,B}$, and so we write $\lambda^{-1}_{A,B}$ for it. Given $h \in Tm(\Gamma, \Pi(A, B))$, $a \in Tm(\Gamma, A)$, we define $App_{A,B}(h, a) \stackrel{\text{df.}}{=} \lambda^{-1}_{A,B}(h)\{\overline{a}\} \in Tm(\Gamma, B\{\overline{a}\})$, where $\overline{a} = \langle id_\Gamma, a\rangle : \Gamma \to \Sigma(\Gamma, A)$.
- ($\Pi$-Comp) $App_{A,B}(\lambda_{A,B}(f), a) = \lambda^{-1}_{A,B}(\lambda_{A,B}(f))\{\overline{a}\} = f\{\overline{a}\}$.
- ($\Pi$-, $\lambda$-Subst) Just by the corresponding axioms.
- (App-Subst) It is easy to see that:

$$\begin{aligned}
App_{A,B}(f, a)\{\phi\} &= \lambda^{-1}_{A,B}(f)\{\overline{a}\}\{\phi\} \\
&= \lambda^{-1}_{A,B}(f)\{\langle id_\Gamma, a\rangle \circ \phi\} \\
&= \lambda^{-1}_{A,B}(f)\{\langle \phi, a\{\phi\}\rangle\} \\
&= \lambda^{-1}_{A,B}(f)\{\phi^+ \circ \overline{a\{\phi\}}\} \\
&= \lambda^{-1}_{A,B}(f)\{\phi^+\}\{\overline{a\{\phi\}}\} \\
&= \lambda^{-1}_{A\{\phi\},B\{\phi^+\}}(f\{\phi\})\{\overline{a\{\phi\}}\} \text{ (by the second axiom of } \lambda) \\
&= App_{A\{\phi\},B\{\phi^+\}}(f\{\phi\}, a\{\phi\})
\end{aligned}$$

  where $\overline{a\{\phi\}} \stackrel{\text{df.}}{=} \langle id_\Delta, a\{\phi\}\rangle : \Delta \to \Sigma(\Delta, A\{\phi\})$.

■

To summarize, DCCCs are a generalization of CCCs as well as a refinement of CwFs with strict 1-, $\Sigma$- and $\Pi$-types (as promised in the introduction):

▶ **Corollary 4.17** (CCCs, DCCCs and CwFs)**.** *Every CCC is a DCCC, and every DCCC is a CwF with strict* 1*-,* $\Sigma$*- and* $\Pi$*-types.*

We prove a kind of converse in the next section: A constant and recursive DCCC is a CCC. Moreover, since the CwF $\mathcal{T}$ of the syntax *faithfully* induces a DCCC, we have obtained:

▶ **Theorem 4.18** (Completeness of DCCCs)**.** *The interpretation of MLTT in DCCCs (via CwFs as described in Def. 3.6) is complete in the same sense as Thm. 3.7.*

# 5 SDTC (simple and dependent types correspondence) theorem

Now, recall the diagram in Sec. 1.5, in which the right square clearly commutes. This section is devoted to the commutativity of the left square, i.e., a faithful translation from crDCCCs to CCCs, which in particular establishes a bijection between the two interpretations of STLC.

▶ **Definition 5.1** (Translation from crDCCCs to CCCs)**.** Given a crDCCC $\mathcal{C} = (\mathcal{C}, \mathscr{D}, _\{_\})$, we define a CCC $\Lambda(\mathcal{C}) = (\mathcal{C}, T, \times, \Lambda)$ as follows:

- The underlying category and the terminal object are those of $\mathcal{C}$
- Products and projections are inductively defined by: (basis) $\Delta \times T \overset{\text{df.}}{=} \Delta$, $\pi_1 \overset{\text{df.}}{=} id_\Delta : \Delta \times T \to \Delta$, $\pi_2 \overset{\text{df.}}{=} !_\Delta : \Delta \times T \to T$; (inductive step) $\Delta \times (\Gamma.A) \overset{\text{df.}}{=} (\Delta \times \Gamma).\hat{A}\{!_{\Delta\times\Gamma}\}$, $\pi_1 \overset{\text{df.}}{=} (\Delta \times \Gamma).\hat{A}\{!_{\Delta\times\Gamma}\} \overset{\pi_1}{\to} \Delta \times \Gamma \overset{\pi_1}{\to} \Delta$, $\pi_2 \overset{\text{df.}}{=} (\Delta \times \Gamma).\hat{A}\{!_{\Delta\times\Gamma}\} \overset{\langle \pi_2 \circ \pi_1, \pi_2 \rangle}{\to} \Gamma.A$
- Pairings are inductively defined by: (basis) given $\psi : \Theta \to \Delta$, $!_\Theta : \Theta \to T$, $\langle \psi, !_\Theta \rangle \overset{\text{df.}}{=} \psi$; (inductive step) given $\psi : \Theta \to \Delta$, $\langle \varphi, g \rangle : \Theta \to \Gamma.A$, $\langle \psi, \langle \varphi, g \rangle \rangle \overset{\text{df.}}{=} \langle \langle \psi, \varphi \rangle, g \rangle$
- Given $\Gamma \in \mathcal{C} \setminus \{T\}$, we define $\oint \Gamma \in \mathcal{C}$ as a preliminary concept inductively by: (basis) $\oint T.A \overset{\text{df.}}{=} T.A$; (inductive step 1) $\oint (T.A).B \overset{\text{df.}}{=} T.\Sigma(A, B)$; (inductive step 2) $\oint ((\Delta.C).A).B \overset{\text{df.}}{=} \oint (\oint ((\Delta.C).A).B)$. Since $\oint \Gamma$ is of the form $T._$ for all $\Gamma \in \mathcal{C} \setminus \{T\}$, we write $\oint \Gamma = T.\ell(\Gamma)$.
- Exponentials are then inductively defined by: (basis 1) $\Delta^T \overset{\text{df.}}{=} \Delta$; (basis 2) $T^\Gamma \overset{\text{df.}}{=} T$; (inductive step) $\Gamma.A \Rightarrow \Delta.B \overset{\text{df.}}{=} T.\Pi(\ell(\Gamma.A), \ell(\Delta.B)\{!_{\oint(\Gamma.A)}\})$
- Evaluations and currying are taken from the corresponding dependent map spaces.

▶ **Theorem 5.2** (SDTC)**.** *For any crDCCC $\mathcal{C}$, $\Lambda(\mathcal{C})$ forms a CCC, and the composition $\mathcal{L} \overset{[\![_]\!]}{\to} \mathcal{C} \overset{\Lambda}{\to} \Lambda(\mathcal{C})$ of $\Lambda$ on $\mathcal{C}$ and an interpretation $[\![_]\!] : \mathcal{L} \to \mathcal{C}$ (of Def. 3.6) coincides with another $\mathcal{L} \overset{\Lambda}{\to} \Lambda(\mathcal{L}) \overset{\wr_\int}{\to} \Lambda(\mathcal{C})$ of the corresponding $\wr_\int : \Lambda(\mathcal{L}) \to \Lambda(\mathcal{C})$ (of Def. 3.8) and $\Lambda$ on $\mathcal{L}$.*

*Proof.* It is straightforward to see that $\Lambda(\mathcal{C})$ is a well-defined CCC. The equality $[\![_]\!] \circ \Lambda = \Lambda \circ \wr_\int$ is shown by induction on the structure of the term model $\mathcal{L}$. ■

# 6 Conclusion and future work

We have given a generalization of CCCs, viz., DCCCs, that forms a semantics for MLTT, and analyzed the relation between the standard interpretation of STLC in CCCs (Def. 3.8) and ours in DCCCs (Def. 3.6), viz., there is a bijection $\Lambda$ between them (Def. 5.1, Thm. 5.2).

As future work, it would be interesting to characterize the CCCs that constitute the image of $\Lambda$, so that they are in one-to-one correspondence with crDCCCs. This may lead to a refinement of the standard semantics of simple type theories in CCCs. Moreover, it remains to develop machineries for the usual "theory-category correspondence" [Pit01].

**Acknowledgements.** The author was supported by Funai Overseas Scholarship. Also, he is grateful to Samson Abramsky for fruitful discussions.

## References


[AJV15] Samson Abramsky, Radha Jagadeesan, and Matthijs Vákár. Games for dependent types. In *Automata, Languages, and Programming*, pages 31–43. Springer, 2015.

[Car86] John Cartmell. Generalized algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32(3):209–243, 1986.

[CD14] Pierre Clairambault and Peter Dybjer. The biequivalence of locally cartesian closed categories and Martin-löf type theories. *Mathematical Structures in Computer Science*, 24(05):e240501, 2014.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *The journal of symbolic logic*, 5(02):56–68, 1940.

[Cro93] Roy L Crole. *Categories for types*. Cambridge University Press, 1993.

[Cur89] Pierre-Louis Curien. Alpha-conversion, conditions on variables and categorical logic. *Studia Logica*, 48(3):319–360, 1989.

[Dyb96] Peter Dybjer. Internal type theory. In *Types for Proofs and Programs*, pages 120–134. Springer, 1996.

[Hof97] Martin Hofmann. Syntax and semantics of dependent types. In *Extensional Constructs in Intensional Type Theory*, pages 13–54. Springer, 1997.

[HS98] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. *Twenty-five years of constructive type theory (Venice, 1995)*, 36:83–111, 1998.

[Jac99] Bart Jacobs. *Categorical Logic and Type Theory*, volume 141. Elsevier, 1999.

[LS88] Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

[ML82] Per Martin-Löf. Constructive mathematics and computer programming. *Studies in Logic and the Foundations of Mathematics*, 104:153–175, 1982.

[ML84] Per Martin-Löf. *Intuitionistic Type Theory: Notes by Giovanni Sambin of a series of lectures given in Padova, June 1980*. 1984.

[ML98] Per Martin-Löf. An intuitionistic theory of types. *Twenty-five years of constructive type theory*, 36:127–172, 1998.

[ML13] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[Mog91] Eugenio Moggi. A category-theoretic account of program modules. *Mathematical Structures in Computer Science*, 1(01):103–139, 1991.

[NPS90] Bengt Nordström, Kent Petersson, and Jan M Smith. Programming in Martin-löfs type theory, volume 7 of international series of monographs on computer science, 1990.

[Obt89] Adam Obtułowicz. Categorical and algebraic aspects of martin-löf type theory. *Studia Logica*, 48(3):299–317, 1989.

[Pie02] Benjamin C Pierce. *Types and programming languages*. MIT press, 2002.

[Pit01] Andrew M Pitts. Categorical logic. In *Handbook of logic in computer science*, pages 39–123. Oxford University Press, 2001.

[See84] Robert AG Seely. Locally cartesian closed categories and type theory. In *Mathematical proceedings of the Cambridge philosophical society*, volume 95, pages 33–48. Cambridge Univ Press, 1984.

[SU06] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149. Elsevier, 2006.

[TS00] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory*. Number 43. Cambridge University Press, 2000.

[Yam16] Norihiro Yamada. Game semantics for Martin-Löf type theory. *arXiv preprint arXiv:1610.01669*, 2016.

[Yam17] Norihiro Yamada. Game-theoretic investigation of intensional equalities. *arXiv preprint arXiv:1703.02015*, 2017.