Combinatory logic and lambda calculus are equal, algebraically

Thorsten Altenkirch
School of Computer Science, University of Nottingham, UK

Ambrus Kaposi
Eötvös Loránd University, Budapest, Hungary

Artjoms Šinkarovs
Heriot-Watt University, Scotland, UK

Tamás Végh
Eötvös Loránd University, Budapest, Hungary

Abstract

It is well-known that lambda calculus is equivalent to combinatory logic extended with four extensionality equations. In this paper we describe a formalisation of this fact in Cubical Agda. The distinguishing features of our formalisation are the following: (i) Both languages are defined as generalised algebraic theories, the syntaxes are intrinsically typed and quotiented by conversion; we never mention preterms or break the quotients in our construction. (ii) Typing is a parameter, thus the un(i)typed and simply typed variants are special cases of the same proof. (iii) We define syntaxes as quotient inductive-inductive types (QIITs) in Cubical Agda; we prove the equivalence and (via univalence) the equality of these QIITs; we do not rely on any axioms, the conversion functions all compute and can be experimented with.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Combinatory logic, lambda calculus, quotient inductive types, Cubical Agda

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material *Software (Formalisation)*: `https://bitbucket.org/akaposi/combinator`

Funding The first two authors were supported by the "Application Domain Specific Highly Reliable IT Solutions" project which has been implemented with support from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.
Artjoms Šinkarovs: Supported by the Engineering and Physical Sciences Research Council through the grant EP/N028201/1.

1 Introduction

It is well-known that lambda calculus with β, η and combinatory logic with four extra conversion rules are equivalent. Proofs of this fact (e.g. [10, 5, 11, 6]) generally use the traditional untyped definition of presyntax and define the bracket abstraction algorithm on combinator preterms. There are hints in the literature [22, 12] that the correspondance can be proven in an algebraic setting. This would be contrasted with the textbook proofs which work on particular representations of the syntax.

It is clear what combinatory logic is as an algebraic theory: there is a single sort of terms, two nullary operations S and K, one binary operation $- \cdot -$ and two equations $\mathsf{S} \cdot t \cdot u \cdot v = t \cdot v \cdot (u \cdot v)$ and $\mathsf{K} \cdot u \cdot v = u$. Models of this theory are called combinatory algebras. What about the lambda calculus? Castellan, Clairambault and Dybjer [9] suggest that the syntax of lambda calculus should be defined as the initial category with families

(CwF) with extra structure. This representation is generalised algebraic [8], and by indexing terms by their typing contexts, it avoids the problems related to the ξ rule [22]. In short, untyped lambda calculus is a uni-typed CwF with an isomorphism

$$\mathsf{lam} : \mathsf{Tm}\,(m+1) \cong \mathsf{Tm}\,m$$

natural in m, where Tm is the sort of terms which is indexed by the possible number of free variables. The left to right direction is abstraction, the right to left direction is application, the fact that the two roundtrips are identities are the β and η laws. In simply typed CwFs, terms are also indexed by types, the simply typed lambda calculus has an arrow type former $- \Rightarrow - : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty}$ and the above isomorphism becomes

$$\mathsf{lam} : \mathsf{Tm}\,(\Gamma \triangleright A)\,B \cong \mathsf{Tm}\,\Gamma\,(A \Rightarrow B).$$

Here terms are indexed both by contexts and types and $\Gamma \triangleright A$ is the context Γ extended with one variable of type A. In fact, the untyped case is a special case of the typed one where we assume all elements of Ty to be equal.

Quotient inductive-inductive type (QIITs) [14, 16] are inductive types where later sorts can be indexed over previous ones, and where equality constructors are also allowed: a QIIT is freely generated by its constructors and is quotiented at the same time by its equality constructors. The sorts, constructors and equality constructors of a QIIT can be also seen as the sorts, operations and equations of a generalised algebraic theory. Given a generalised algebraic theory, the corresponding QIIT is its initial algebra. The elimination principle of the QIIT correponds exactly to the universal property (initiality) of the initial algebra. When a language is defined as an algebraic theory, the corresponding QIIT is its intrinsic (well-formed, well-typed) syntax quotiented by conversion. That is, convertible terms are equal in such a syntax. This approach to the syntax is very natural for dependently typed languages [2] where conversion cannot be defined separately from typing, but in this paper we apply it in a simply typed setting. Cubical Agda [25] is currently the only implementation of type theory with native support for QIITs. It features the more general higher inductive-inductive types (HIITs [16]). A QIIT is a HIIT with constructors truncating each sort to be a set, in the sense of homotopy type theory [19]. This means that any two equalities between equalities of elements of a QIIT are equal. Cubical Agda also features the univalence axiom which turns an isomorphism (bijection) into an equality.

In this paper we prove that combinatory terms of a given type are isomorphic to lambda terms of the same type, thus by univalence we obtain an equality $\mathsf{Tm}_{\mathsf{C}}\,A = \mathsf{Tm}_{\mathsf{L}} \diamond A$. The subscripts denote combinatory and lambda terms, respectively, and $\diamond$ is the empty context. Here Tm_{C} also features four equations expressing extensionality in addition to the computation rules of S and K mentioned above. In the proof, we make use of an auxiliary theory Cwk which is a variant of C featuring contexts, an operation q (the last variable in a context, a.k.a. the zero De Bruijn index) and a weakening operation (which is also the successor operation for De Bruijn indices). We define lambda abstraction by induction on terms in Cwk. An illustration of Cwk is that extensionality can be expressed by the implication $\mathsf{wk}\,t \cdot \mathsf{q} = \mathsf{wk}\,t' \cdot \mathsf{q} \to t = t'$.

Equality of combinatory and lambda terms ensures that lambda terms can be replaced by combinatory terms in any construction and vica versa. This is indeed the case in Cubical Agda as equality of the two different sets of terms is proof-relevant, and we can transport over it. Our proof is parameterised by a set of types Ty closed under arrow $- \Rightarrow -$, thus it applies to both the un(i)-typed and simply typed cases.

The main contribution of this paper is an algebraic proof of the equivalence of combinatory logic and lambda calculus which does not mention representations. Further contributions are the typing generality and the formalisation in Cubical Agda.

1.1 Structure of the paper

After summarising related work and describing our notations, in Section 2 we define the three theories C, Cwk and L. In Section 3 we prove the isomorphism $\mathsf{Tm}_{\mathsf{C}}\,A \cong \mathsf{Tm}_{\mathsf{Cwk}}\diamond A$. We define the lambda calculus operations using the syntax of Cwk in Section 4. Using these, in Section 5 we prove the isomorphism $\mathsf{Tm}_{\mathsf{Cwk}}\,\Gamma\,A \cong \mathsf{Tm}_{\mathsf{L}}\,\Gamma\;A$. To represent open lambda terms as combinator terms, we introduce an arrow type with a context domain. $\Gamma \Rightarrow^* A$ is defined as $\diamond \Rightarrow^* A :\equiv A$ and $(\Gamma \triangleright B) \Rightarrow^* A :\equiv \Gamma \Rightarrow^* B \Rightarrow A$. In Section 6 we prove $\mathsf{Tm}_{\mathsf{L}}\diamond(\Gamma \Rightarrow^* A) \cong \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A$. Putting together everything we obtain our main theorem $\mathsf{Tm}_{\mathsf{C}}\,(\Gamma \Rightarrow^* A) \cong \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A$. We conclude in Section 7.

1.2 Related work

It is usual to describe the semantics of languages as (generalised or essentially) algebraic theories, see e.g. [17]. The syntax is however usually given by abstract syntax trees. There are few textbooks which use well-typed unquotiented syntax trees, e.g. [26]. Several important constructions on the syntax of typed lambda calculi can be performed on intrinsic quotiented terms, e.g. normalisation [1], parametricity [2], typechecking [13] and unification [15]. In this paper we show that the bracket abstraction algorithm can be also defined in the typed and quotiented setting.

Selinger [22] remarks that extensional models of lambda calculus do not form an algebraic variety because the subalgebra of closed terms is not extensional. This does not apply in our setting using CwFs because closed terms do not form a subalgebra. We use the algebraic description of lambda calculus by Castellan, Clairambault and Dybjer (uni-typed CwF) [9]. Hyland [12] describes lambda calculus in a way equivalent to ours using notions from categorical universal algebra, but omits the connection to combinatory logic for reasons of space.

Swierstra [23] defines a correct-by-construction conversion of combinators into lambda terms. He uses intrinsically typed unquotiented terms indexed by their semantics using a trick by McBride [18].

The relationship of combinatory logic and lambda calculus is still an active research area, for example a rewriting relation for combinator terms equivalent to β reduction was investigated in [20, 21], using preterms and typing relations. Combinators are used in realisability semantics in the form of partial combinatory algebras, for example in [4].

1.3 Metatheory and formalisation

We work with notations close to Agda's. The universe of types is written Set, we don't write universe indices, however we work in a predicative setting. Dependent functions are written $(x : A) \to B$ where B can use x, application is juxtaposition. We write implicit arguments as $\{x : A\} \to B$ or we simply omit them and just write B. When f is an implicit function, we can supply arguments in curly brackets as $f\,\{a\}$. Implicit functions are used a lot for readability, but they are just a concise notation, formally they are always specified. Σ types are written using infix $\times$, the unit type $\top$ has one definitionally unique element $\mathtt{tt}$. We write definitional equality as $\equiv$, definitions using $:\equiv$, propositional equality as $=$. We assume definitional function extensionality, that is, a propositional equality of two functions is proven as a pointwise equality. We use equational reasoning notation when proving equalities. We define isomorphism as the following iterated Σ type (named record type). We overload the name of the isomorphism and the function in the forward direction.

$$(f : A \cong B) :\equiv (f : A \to B) \times (\mathsf{f}^{-1} : B \to A) \times (f^{\beta} : \mathsf{f}^{-1}\,(\mathsf{f}\,a) = a) \times (f^{\eta} : \mathsf{f}\,(\mathsf{f}^{-1}\,b) = b)$$

We use QIITs (set-truncated HIITs) and eliminate from them by pattern matching. All the pattern matching definitions can be defined using the elimination principles of the QIITs. Because most of the equalities that we prove or use are propositions, we do not prove equalities of equalities in the paper. This is the main difference between the paper and the Cubical Agda formalisation. In the formalisation most of the line count comes from boilerplate proofs proving equalities of equalites using the isSet constructors of QIITs. We define special cases of the elimination principles for some of the QIITs and use them to reduce this boilerplate. For example, when proving a proposition by induction on a QIIT, we do not need to provide methods for the equality constructors. The only non-propositional equalities we prove are coming from isomorphisms via univalence. The formalisation is available as supplementary material.

This paper can be understood without knowing Cubical Agda or even homotopy type theory.

2 Three theories

We parameterise Sections 2–6 by a $\mathsf{Ty} : \mathsf{Set}$ and $- \Rightarrow - : \mathsf{Ty} \to \mathsf{Ty} \to \mathsf{Ty}$. We define contexts inductively by the following constructors.

$$\begin{aligned}
&\mathsf{Con} &&: \mathsf{Set}\\
&\diamond &&: \mathsf{Con}\\
&- \triangleright - &&: \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Con}
\end{aligned}$$

$\diamond$ denotes the empty context, $\Gamma \triangleright A$ is the context Γ extended by the type A. A context of length three containing types A, B, C is written $\diamond \triangleright A \triangleright B \triangleright C$.

In Figures 1, 2, 3 we define the theories C, Cwk and L, respectively. The syntaxes (initial models/algebras) for these theories are implemented in Cubical Agda using inductive types with equality constructors. The syntaxes of C and Cwk are indexed quotient inductive types (QIT), the syntax of L is given by two mutually defined indexed QITs. The operators become constructors, the equations become equality constructors, and each type has an extra isSet constructor ensuring that the higher equality structure is trivial.

In the rest of this section we explain the operations of the three theories, and define some derivable operations and equations in each.

2.1 Combinator logic with extensionality

Theory C is defined by the indexed sort, three operations and six equations in Figure 1. Some of the operators have implicit parameters. For example, application $- \cdot -$ takes the types A and B implicitly, its fully explicit type is $\{A : \mathsf{Ty}\}\{B : \mathsf{Ty}\} \to \mathsf{Tm}\,(A \Rightarrow B) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,B$. K has two, S has three implicit type parameters. $\mathsf{K}\beta$ has two implicit type parameters and two implicit term parameters, and we understand $\mathsf{K}\beta$ with the most general implicit parameters, i.e. $u : \mathsf{Tm}\,A$, $v : \mathsf{Tm}\,B$ where A and B don't have to be the same. Similarly, we use the

$$
\begin{aligned}
&\mathsf{Tm} &&: \mathsf{Ty} \to \mathsf{Set}\\
&- \cdot - &&: \mathsf{Tm}\,(A \Rightarrow B) \to \mathsf{Tm}\,A \to \mathsf{Tm}\,B\\
&\mathsf{K} &&: \mathsf{Tm}\,(A \Rightarrow B \Rightarrow A)\\
&\mathsf{S} &&: \mathsf{Tm}\,\big((A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C\big)\\
&\mathsf{K}\beta &&: \mathsf{K} \cdot u \cdot v = u\\
&\mathsf{S}\beta &&: \mathsf{S} \cdot t \cdot u \cdot v = t \cdot v \cdot (u \cdot v)\\
&\mathsf{lamK}\beta &&: \mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})\big) = \mathsf{K}\\
&\mathsf{lamS}\beta &&: \mathsf{S} \cdot \Big(\mathsf{K} \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S})\big)\Big) \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S})\big)\Big) =\\
& && \quad \mathsf{S} \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}))) \cdot \mathsf{S})\big)\Big)\Big) \cdot (\mathsf{K} \cdot \mathsf{S})\\
&\mathsf{lamwk}\cdot &&: \mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) = \mathsf{S} \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \mathsf{K})\big)\Big) \cdot (\mathsf{K} \cdot \mathsf{K})\\
&\eta &&: \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K} = \mathsf{S} \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \mathsf{K}\big) \cdot \big(\mathsf{K} \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K})\big)
\end{aligned}
$$

Figure 1 Theory C: combinator logic with extensionality equations. Note that Ty, $- \Rightarrow -$ are parameters, Con is defined inductively (beginning of Section 2).

$$
\begin{aligned}
&\mathsf{Tm} &&: \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Set}\\
&- \cdot - &&: \mathsf{Tm}\,\Gamma\,(A \Rightarrow B) \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,B\\
&\mathsf{K} &&: \mathsf{Tm}\,\Gamma\,(A \Rightarrow B \Rightarrow A)\\
&\mathsf{S} &&: \mathsf{Tm}\,\Gamma\,\big((A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C\big)\\
&\mathsf{K}\beta &&: \mathsf{K} \cdot u \cdot v = u\\
&\mathsf{S}\beta &&: \mathsf{S} \cdot t \cdot u \cdot v = t \cdot v \cdot (u \cdot v)\\
&\mathsf{q} &&: \mathsf{Tm}\,(\Gamma \triangleright A)\,A\\
&\mathsf{wk} &&: \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,(\Gamma \triangleright B)\,A\\
&\mathsf{wk}\cdot &&: \mathsf{wk}\,(t \cdot u) = \mathsf{wk}\,t \cdot \mathsf{wk}\,u\\
&\mathsf{wkK} &&: \mathsf{wk}\,\mathsf{K} = \mathsf{K}\\
&\mathsf{wkS} &&: \mathsf{wk}\,\mathsf{S} = \mathsf{S}\\
&\mathsf{lamK}\beta &&: \mathsf{S}\,\{\diamond\} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})\big) = \mathsf{K}\\
&\mathsf{lamS}\beta &&: \mathsf{S}\,\{\diamond\} \cdot \Big(\mathsf{K} \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S})\big)\Big) \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S})\big)\Big) =\\
& && \quad \mathsf{S}\,\{\diamond\} \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}))) \cdot \mathsf{S})\big)\Big)\Big) \cdot (\mathsf{K} \cdot \mathsf{S})\\
&\mathsf{lamwk}\cdot &&: \mathsf{S}\,\{\diamond\} \cdot (\mathsf{K} \cdot \mathsf{K}) = \mathsf{S} \cdot \Big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \mathsf{K})\big)\Big) \cdot (\mathsf{K} \cdot \mathsf{K})\\
&\eta &&: \mathsf{S}\,\{\diamond\} \cdot \mathsf{K} \cdot \mathsf{K} = \mathsf{S} \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \mathsf{K}\big) \cdot \big(\mathsf{K} \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K})\big)
\end{aligned}
$$

Figure 2 Theory Cwk: combinator logic with variables, weakenings and extensionality equations. Note that the extensionality equations are all given in the empty context. Ty and $- \Rightarrow -$ are parameters, Con is defined inductively (beginning of Section 2).

most general versions of the the other equations. The last four equations don't have terms as parameters, but do have implicit type parameters.

Using implicit parameters, we can write typed combinatory terms the same way as we would write untyped ones. For example, the identity combinator is defined as follows.

$$\mathsf{I} : \mathsf{Tm}\,(A \Rightarrow A)$$
$$\mathsf{I} :\equiv \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}$$

If we write out the implicit parameters of the S and Ks (but not the $- \cdot -$ applications), this becomes $\mathsf{S}\,\{A\}\{A \Rightarrow A\}\{A\} \cdot \mathsf{K}\,\{A\}\{A \Rightarrow A\} \cdot \mathsf{K}\,\{A\}\{A\}$. If we provide Agda with the information that I will have type $\mathsf{Tm}\,(A \Rightarrow A)$, it is enough to specify the second parameter of the second K. This uniquely determines the other implicit parameters, so in Agda we write $\mathsf{I} :\equiv \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}\,\{\}\{A\}$. We prove the β law for I using equational reasoning.

$$\mathsf{I}\beta : \mathsf{I} \cdot t \equiv \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K} \cdot t \overset{\mathsf{S}\beta}{=} \mathsf{K} \cdot t \cdot (\mathsf{K} \cdot t) \overset{\mathsf{K}\beta}{=} t$$

We say that C has extensionality because of the last four equations. We add these so as to be equivalent to Cwk which includes these equations so as to be equivalent to L. The origin of these equations will be partly revealed in Subsection 2.2 and fully revealed in Section 4.

2.2 Combinator logic with variables, weakenings and extensionality

Theory Cwk is defined in Figure 2. The sort of terms is now indexed by both contexts and types. Application, K, S, $\mathsf{K}\beta$ and $\mathsf{S}\beta$ are just like for C with the difference that all these work in an arbitrary context. We have two extra operations which correspond to the Peano constructors of De Bruijn indices: q is the zero index, wk is successor. For example De Bruijn index 2 is written $\mathsf{wk}\,(\mathsf{wk}\,\mathsf{q}) : \mathsf{Tm}\,(\Gamma \triangleright A \triangleright B \triangleright C)\,A$. Note that Γ, A and B are implicit arguments of wk. As wk can be applied to any term, we add equations expressing that it commutes with $- \cdot -$, K and S. Finally, the three equations $\mathsf{lamK}\beta$, $\mathsf{lamS}\beta$, $\mathsf{lamwk}\cdot$ are needed so that we can define lambda abstraction (lam) by recursion on the syntax, and η corresponds to the η rule in L (see Section 4 for their usage). We limit these equations to the empty context because lam then automatically preserves them (the input of lam is in an extended context). Limiting to the empty context is not a real limitation when contexts are defined inductively. We prove that the equations hold in any context by adding wk as many times as the length of the context. Because wk commutes with K, S and $- \cdot -$ and the four equations do not contain anything else, the weakened terms will be the same as the original terms, just in a different context. Thus we obtain the general primed versions $\mathsf{lamK}\beta'$, $\mathsf{lamS}\beta'$, $\mathsf{lamwk}\cdot'$ and η'. For example, $\mathsf{lamK}\beta'$ is defined as follows. For readability, we merged some steps.

$$\mathsf{lamK}\beta' : \{\Gamma : \mathsf{Con}\} \to \mathsf{S}\,\{\Gamma\} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})\big) = \mathsf{K}$$

$$\begin{aligned}
&\mathsf{lamK}\beta'\,\{\diamond\} &&: \mathsf{S}\,\{\diamond\} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})\big) \overset{\mathsf{lamK}\beta}{=} \mathsf{K}\,\{\diamond\} \\
&\mathsf{lamK}\beta'\,\{\Gamma \triangleright A\} &&: \mathsf{S}\,\{\Gamma \triangleright A\} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})\big) && =(\mathsf{wkS}, \mathsf{wkK} \text{ 3x each}) \\
& && \mathsf{wk}\,(\mathsf{S}\,\{\Gamma\}) \cdot (\mathsf{wk}\,\mathsf{K} \cdot \mathsf{wk}\,\mathsf{S}) \cdot \big(\mathsf{wk}\,\mathsf{S} \cdot (\mathsf{wk}\,\mathsf{K} \cdot \mathsf{wk}\,\mathsf{K})\big) && =(\mathsf{wk}\cdot \text{ twice}) \\
& && \mathsf{wk}\,(\mathsf{S}\,\{\Gamma\}) \cdot (\mathsf{wk}\,(\mathsf{K} \cdot \mathsf{S})) \cdot \big(\mathsf{wk}\,\mathsf{S} \cdot (\mathsf{wk}\,(\mathsf{K} \cdot \mathsf{K}))\big) && =(\mathsf{wk}\cdot \text{ twice}) \\
& && \mathsf{wk}\,\big(\mathsf{S}\,\{\Gamma\} \cdot (\mathsf{K} \cdot \mathsf{S})\big) \cdot \mathsf{wk}\,(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})) && =(\mathsf{wk}\cdot) \\
& && \mathsf{wk}\,\big(\mathsf{S}\,\{\Gamma\} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}))\big) && =(\mathsf{lamK}\beta'\,\{\Gamma\}) \\
& && \mathsf{wk}\,(\mathsf{K}\,\{\Gamma\}) && =(\mathsf{wkK}) \\
& && \mathsf{K}\,\{\Gamma \triangleright A\}
\end{aligned}$$

$$
\begin{array}{llll}
\mathsf{Sub} & : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} & -,- & : \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Tm}\,\Delta\,A \to \mathsf{Sub}\,\Delta\,(\Gamma \triangleright A) \\
- \circ - & : \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Theta\,\Gamma & \mathsf{p} & : \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma \\
\mathsf{ass} & : (\sigma \circ \rho) \circ \tau = \sigma \circ (\rho \circ \tau) & \mathsf{q} & : \mathsf{Tm}\,(\Gamma \triangleright A)\,A \\
\mathsf{id} & : \mathsf{Sub}\,\Gamma\,\Gamma & \triangleright\beta_1 & : \mathsf{p} \circ (\sigma, t) = \sigma \\
\mathsf{idl} & : \sigma \circ \mathsf{id} = \sigma & \triangleright\beta_2 & : \mathsf{q}[\sigma, t] = t \\
\mathsf{idr} & : \mathsf{id} \circ \sigma = \sigma & \triangleright\eta & : \{\sigma : \mathsf{Sub}\,\Delta\,(\Gamma \triangleright A)\} \to \sigma = (\mathsf{p} \circ \sigma, \mathsf{q}[\sigma]) \\
\epsilon & : \mathsf{Sub}\,\Gamma\,\diamond & \mathsf{lam} & : \mathsf{Tm}\,(\Gamma \triangleright A)\,B \to \mathsf{Tm}\,\Gamma\,(A \Rightarrow B) \\
\diamond\eta & : \{\sigma : \mathsf{Sub}\,\Gamma\,\diamond\} \to \sigma = \epsilon & - \cdot - & : \mathsf{Tm}\,\Gamma\,(A \Rightarrow B) \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,B \\
\mathsf{Tm} & : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Set} & \Rightarrow\beta & : \mathsf{lam}\,t \cdot u = t[\mathsf{id}, u] \\
-[-] & : \mathsf{Tm}\,\Gamma\,A \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Tm}\,\Delta\,A & \Rightarrow\eta & : \{t : \mathsf{Tm}\,\Gamma\,(A \Rightarrow B)\} \to t = \mathsf{lam}\,(t[\mathsf{p}] \cdot \mathsf{q}) \\
[\circ] & : A[\sigma \circ \rho] = A[\sigma][\rho] & \mathsf{lam}[] & : (\mathsf{lam}\,t)[\sigma] = \mathsf{lam}\,(t[\sigma \circ \mathsf{p}, \mathsf{q}]) \\
[\mathsf{id}] & : A[\mathsf{id}] = A & \cdot[] & : (t \cdot u)[\sigma] = (t[\sigma]) \cdot (u[\sigma])
\end{array}
$$

Figure 3 Theory L: lambda calculus. A concise description using categorical terminology: a category with terminal object where objects are Con and the terminal object is $\diamond$; for each $A : \mathsf{Ty}$ a locally representable presheaf $\mathsf{Tm} - A$ over the category where $- \triangleright -$ generates the new objects; an isomorphism $\mathsf{Tm}\,(\Gamma \triangleright A)\,B \cong \mathsf{Tm}\,\Gamma\,(A \Rightarrow B)$ natural in Γ. Note that Ty and $- \Rightarrow -$ are parameters, Con is defined inductively (beginning of Section 2).

We derive another version of each of the four equations which we call the pointful variants. These are the versions that will be actually used when defining lam.

$$
\begin{array}{ll}
\mathsf{lamK}\beta'' & : \mathsf{S} \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot u) \cdot v = u \\
\mathsf{lamS}\beta'' & : \mathsf{S} \cdot \big(\mathsf{S} \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot t) \cdot u\big) \cdot v = \mathsf{S} \cdot (\mathsf{S} \cdot t \cdot u) \cdot (\mathsf{S} \cdot u \cdot v) \\
\mathsf{lamwk}\cdot'' & : \mathsf{K} \cdot (t \cdot u) = \mathsf{S} \cdot (\mathsf{K} \cdot t) \cdot (\mathsf{K} \cdot u) \\
\eta'' & : t = \mathsf{S} \cdot (\mathsf{K} \cdot t) \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K})
\end{array}
$$

We obtain the pointful versions by applying $\mathsf{K}\beta$, $\mathsf{S}\beta$ multiple times to the pointfree version. Here is the proof for $\mathsf{lamK}\beta$, see Appendix A or the formalisation for the other equations.

$$
\begin{array}{lll}
\mathsf{lamK}\beta'' : & \mathsf{S} \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot u) \cdot v & =(\mathsf{K}\beta) \\
& \mathsf{K} \cdot \mathsf{S} \cdot u \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot u\big) \cdot v & =(\mathsf{S}\beta) \\
& \mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K})\big) \cdot u \cdot v & =\mathsf{lamK}\beta' \\
& \mathsf{K} \cdot u \cdot v & =(\mathsf{K}\beta) \\
& u &
\end{array}
$$

2.3 Lambda calculus

Theory L is defined in Figure 3. Lambda calculus can be seen as a second order theory with one sort Tm indexed by Ty and an isomorphism $(\mathsf{Tm}\,A \to \mathsf{Tm}\,B) \cong \mathsf{Tm}\,(A \Rightarrow B)$. The left to right direction is the binder lam which takes a function as an input. We turn this second order theory into a first order theory using a substitution calculus with term-variables in which the second order operation lam becomes a first order operation with an input in an extended context. We describe all the operations in detail: Con, Sub form a category with terminal object $\diamond$. $\diamond$ is the empty context, $\mathsf{Sub}\,\Delta\,\Gamma$ is called a substitution from Δ to Γ. It is a list of terms, where all terms have free variables in Δ and their types are in Γ. For example, a $\mathsf{Sub}\,\Delta\,(\diamond \triangleright A \triangleright B)$ corresponds to a $\mathsf{Tm}\,\Delta\,A$ together with a $\mathsf{Tm}\,\Delta\,B$. Terms can be instantiated by substitutions: if we have a $t : \mathsf{Tm}\,\Gamma\,A$ and a substitution $\sigma : \mathsf{Sub}\,\Delta\,\Gamma$, then we obtain a term $t[\sigma]$ which we call the instantiation of t by σ. This operation replaces all free variables in t by terms in σ, which in turn have free variables declared by Δ. The instantiation operation is functorial, witnessed by $[\circ]$ and $[\mathsf{id}]$. A substitution can either be the unique empty substitution ϵ which targets the empty context $\diamond$ or a substitution built using $-,-$ which targets an extended context. The operations $-,-$, p, q and equations $\triangleright\beta_1$, $\triangleright\beta_2$, $\triangleright\eta$ can be summarised as an isomorphism $\mathsf{Sub}\,\Delta\,(\Gamma \triangleright A) \cong \mathsf{Sub}\,\Delta\,\Gamma \times \mathsf{Tm}\,\Delta\,A$. The variables are typed De Bruijn indices built by q and $-[\mathsf{p}]$, the latter takes the role of wk. We have application $- \cdot -$ and abstraction lam with the computation rule $\Rightarrow\beta$ and uniqueness rule $\Rightarrow\eta$. The rules for the arrow type are summarised by the isomorphism $\mathsf{Tm}\,(\Gamma \triangleright A)\,B \cong \mathsf{Tm}\,\Gamma\,(A \Rightarrow B)$ natural in Γ. Naturality is expressed by $\mathsf{lam}[]$ and $\cdot[]$. Again we stress that most operations in L take implicit arguments, e.g. lam takes Γ, A and B implicitly before its first and only explicit argument.

Naturality of substitution extension holds in any model of L:

$$
\begin{aligned}
,\circ : (\sigma, t) \circ \rho \quad & =(\triangleright\eta) \\
\big(\mathsf{p} \circ ((\sigma, t) \circ \rho), \mathsf{q}[(\sigma, t) \circ \rho]\big) \quad & =(\mathsf{ass}, [\circ]) \\
\big((\mathsf{p} \circ (\sigma, t)) \circ \rho, \mathsf{q}[\sigma, t][\rho]\big) \quad & =(\triangleright\beta_1, \triangleright\beta_2) \\
(\sigma \circ \rho, t[\rho]) &
\end{aligned}
$$

In every model of L, pointwise equal functions are equal, we call this property funext:

$$
\begin{aligned}
&\mathsf{funext} : t[\mathsf{p}] \cdot \mathsf{q} = t'[\mathsf{p}] \cdot \mathsf{q} \to t = t' \\
&\mathsf{funext}\,e : t \overset{\Rightarrow\eta}{=} \mathsf{lam}\,(t[\mathsf{p}] \cdot \mathsf{q}) \overset{e}{=} \mathsf{lam}\,(t'[\mathsf{p}] \cdot \mathsf{q}) \overset{\Rightarrow\eta}{=} t'
\end{aligned}
$$

3 Combinators with and without weakenings are equal

In this section we prove the equivalence of the syntaxes of C and Cwk. We will define an isomorphism $\mathsf{f} : \mathsf{Tm}_{\mathsf{C}}\,A \cong \mathsf{Tm}_{\mathsf{Cwk}}\,\diamond\,A$, and via univalence obtain $\mathsf{Tm}_{\mathsf{C}}\,A = \mathsf{Tm}_{\mathsf{Cwk}}\,\diamond\,A$. The four extensionality equations do not play a role in this section. In fact, any number of closed equations are preserved by f, as long as the same equations hold in Cwk. For readability, we will just say C when we mean the syntax of C, and similarly for Cwk.

$$
\begin{aligned}
&\mathsf{f} : \mathsf{Tm}_{\mathsf{C}}\,A \to \mathsf{Tm}_{\mathsf{Cwk}}\,\diamond\,A \\
&\mathsf{f}\,(t \cdot u) :\equiv \mathsf{f}\,t \cdot \mathsf{f}\,u \\
&\mathsf{f}\,\mathsf{K} \quad :\equiv \mathsf{K} \\
&\mathsf{f}\,\mathsf{S} \quad :\equiv \mathsf{S}
\end{aligned}
\qquad\qquad
\begin{aligned}
&\mathsf{f}^{-1} : \mathsf{Tm}_{\mathsf{Cwk}}\,\diamond\,A \to \mathsf{Tm}_{\mathsf{C}}\,A \\
&\mathsf{f}^{-1}\,(t \cdot u) :\equiv \mathsf{f}^{-1}\,t \cdot \mathsf{f}^{-1}\,u \\
&\mathsf{f}^{-1}\,\mathsf{K} \quad :\equiv \mathsf{K} \\
&\mathsf{f}^{-1}\,\mathsf{S} \quad :\equiv \mathsf{S}
\end{aligned}
$$

Figure 4 The proof-relevant parts of the isomorphism $\mathsf{f} : \mathsf{Tm}_{\mathsf{C}}\,A \cong \mathsf{Tm}_{\mathsf{Cwk}}\,\diamond\,A$. In the f^{-1} direction we don't have to provide cases for constructors q and wk because they are not in the empty context: Con is defined inductively, hence we know that $\diamond \neq \Gamma \triangleright A$ for any Γ and A. We treat the cases for equality constructors in the main text.

In Figure 4 we define the forward and backward directions of f by recursion on Tm_{C} and $\mathsf{Tm}_{\mathsf{Cwk}}$, respectively. We use pattern matching notation. As C is included in Cwk we just return the same operations. We overload the constructors of the two syntaxes, e.g. we write K both for K_C and K_Cwk, it should be clear from the context which is meant. With implicit arguments, the line for K is $\mathsf{f}\,(\mathsf{K}\,\{A\}\{B\}) :\equiv \mathsf{K}\,\{\diamond\}\{A\}\{B\}$, so we choose the output K to be in the empty context. In the other direction we know that the input is in the empty context, hence we define $\mathsf{f}^{-1}\,(\mathsf{K}\,\{\diamond\}\{A\}\{B\}) :\equiv \mathsf{K}\,\{A\}\{B\}$.

Part of the definitions of f and f^{-1} are that they preserve the equality constructors. f maps each equality constructor of C to the corresponding equality constructor of Cwk. We spell out the implicit arugments for $\mathsf{K}\beta$: $\mathsf{f}\,(\mathsf{K}\beta\,\{A\}\{B\}\{u\}\{v\}) :\equiv \mathsf{K}\beta\,\{\diamond\}\{A\}\{B\}\{\mathsf{f}\,u\}\{\mathsf{f}\,v\}$. On the Cwk side we again use the empty context, the same types and we apply f to the term arguments. The other cases are simply $\mathsf{f}\,\mathsf{S}\beta :\equiv \mathsf{S}\beta$, $\mathsf{f}\,\mathsf{lamK}\beta :\equiv \mathsf{lamK}\beta$, $\dots$, $\mathsf{f}\,\eta :\equiv \eta$.

f^{-1} also preserves all the equations in Cwk by their corresponding equations in C. The three extra equations of $\mathsf{wk}\cdot$, wkK and wkS are preserved vacously because they are equating terms in open contexts.

When proving that $\mathsf{f}\circ\mathsf{f}^{-1} = \lambda t.t$ and vice versa, we only have to compare the results of the proof irrelevant parts shown in Figure 4 because the other components are equal by isSet. We prove f^β and f^η by trivial inductions on Tm_C and Tm_Cwk, respectively.

4 Lambda calculus operations in Cwk

In this section we derive the operations of L in the syntax of Cwk. We first define the operations in Figure 5.

$$
\begin{aligned}
&\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}\\
&\mathsf{Sub}\,\Delta\,\diamond \quad :\equiv \top\\
&\mathsf{Sub}\,\Delta\,(\Gamma \triangleright A) :\equiv \mathsf{Sub}\,\Delta\,\Gamma \times \mathsf{Tm}\,\Delta\,A
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{wks} : \{\Gamma : \mathsf{Con}\} \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Sub}\,(\Delta \triangleright A)\,\Gamma\\
&\mathsf{wks}\,\{\diamond\}\,\mathsf{tt} \quad :\equiv \mathsf{tt}\\
&\mathsf{wks}\,\{\Gamma \triangleright A\}\,(\sigma, t) :\equiv (\mathsf{wks}\,\{\Gamma\}\,\sigma, \mathsf{wk}\,t)
\end{aligned}
$$

$$
\begin{aligned}
&-[-] : \mathsf{Tm}\,\Gamma\,A \to \mathsf{Sub}\,\Delta\,\Gamma \to \mathsf{Tm}\,\Delta\,A\\
&(t \cdot u)[\sigma] \quad :\equiv (t[\sigma]) \cdot (u[\sigma])\\
&\mathsf{K}[\sigma] \quad :\equiv \mathsf{K}\\
&\mathsf{S}[\sigma] \quad :\equiv \mathsf{S}\\
&\mathsf{q}[\sigma, u] \quad :\equiv u\\
&(\mathsf{wk}\,t)[\sigma, u] :\equiv t[\sigma]
\end{aligned}
$$

$$
\begin{aligned}
&- \circ - : \{\Gamma : \mathsf{Con}\} \to \mathsf{Sub}\,\Delta\,\Gamma \to\\
&\qquad \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Theta\,\Gamma\\
&\mathsf{tt} \quad \circ^{\{\diamond\}} \quad \rho :\equiv \mathsf{tt}\\
&(\sigma, t) \circ^{\{\Gamma \triangleright A\}} \rho :\equiv (\sigma \circ^{\{\Gamma\}} \rho, t[\rho])
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{id} : \{\Gamma : \mathsf{Con}\} \to \mathsf{Sub}\,\Gamma\,\Gamma\\
&\mathsf{id}\,\{\diamond\} \quad :\equiv \mathsf{tt}\\
&\mathsf{id}\,\{\Gamma \triangleright A\} :\equiv (\mathsf{wks}\,(\mathsf{id}\,\{\Gamma\}), \mathsf{q})
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{p} : \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma\\
&\mathsf{p} :\equiv \mathsf{wks}\,\mathsf{id}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{lam} : \mathsf{Tm}\,(\Gamma \triangleright A)\,B \to \mathsf{Tm}\,\Gamma\,(A \Rightarrow B)\\
&\mathsf{lam}\,(t \cdot u) :\equiv \mathsf{S} \cdot \mathsf{lam}\,t \cdot \mathsf{lam}\,t\\
&\mathsf{lam}\,\mathsf{K} \quad :\equiv \mathsf{K} \cdot \mathsf{K}\\
&\mathsf{lam}\,\mathsf{S} \quad :\equiv \mathsf{K} \cdot \mathsf{S}\\
&\mathsf{lam}\,\mathsf{q} \quad :\equiv \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}\\
&\mathsf{lam}\,(\mathsf{wk}\,t) :\equiv \mathsf{K} \cdot t
\end{aligned}
$$

Figure 5 The definitions of L operations in the syntax of Cwk. See the text for the proof-irrelevant parts.

Substitutions are defined by recursion on the target context. Then we define weakening of these substitutions by iterating wk for terms, again by recursion on the target context. Instantiation of Cwk terms by a substitution is defined by recursion on terms. We show that instantiation preserves the equations as follows.

$$
\begin{aligned}
&\mathsf{K}\beta[\sigma] && : (\mathsf{K} \cdot u \cdot v)[\sigma] \equiv \mathsf{K} \cdot (u[\sigma]) \cdot (v[\sigma]) \overset{\mathsf{K}\beta}{=} u[\sigma] \\
&\mathsf{S}\beta[\sigma] && : (\mathsf{S} \cdot t \cdot u \cdot v)[\sigma] \equiv \mathsf{S} \cdot (t[\sigma]) \cdot (u[\sigma])\,(v[\sigma]) \overset{\mathsf{S}\beta}{=} t[\sigma] \circ (v[\sigma]) \circ (u[\sigma] \circ (v[\sigma])) \\
&\mathsf{wk}\cdot[\sigma] && : (\mathsf{wk}\,(t \cdot u))[\sigma, v] \equiv (t[\sigma]) \cdot (u[\sigma]) \equiv (\mathsf{wk}\,t \cdot \mathsf{wk}\,u)[\sigma, v] \\
&\mathsf{wkK}[\sigma] && : (\mathsf{wk}\,\mathsf{K})[\sigma] \equiv \mathsf{K} \equiv \mathsf{K}[\sigma] \\
&\mathsf{wkS}[\sigma] && : (\mathsf{wk}\,\mathsf{S})[\sigma] \equiv \mathsf{S} \equiv \mathsf{S}[\sigma] \\
&\mathsf{lamK}\beta[\sigma] && :\equiv \mathsf{lamK}\beta' \\
&\mathsf{lamS}\beta[\sigma] && :\equiv \mathsf{lamS}\beta' \\
&\mathsf{lamwk}\cdot[\sigma] && :\equiv \mathsf{lamwk}\cdot' \\
&\eta[\sigma] && :\equiv \eta'
\end{aligned}
$$

The last four equations use the generalised versions of the extensionality equations which work in arbitrary contexts (defined in Subsection 2.2). Composition $- \circ^{\Gamma} -$ is defined by induction on the target context and uses instantiation. We write the implicit argument Γ in superscript for readability. The identity substitution id is defined by induction on the context. The first projection p is the weakening of identity. lam is also defined by recursion on Cwk terms. This is usually called the bracket abstraction algorithm. Lambda of application applies the S combinator to the results of the recursive calls, lambda of q is the identity combinator, lambda of K and S are constant K and S, respectively, while lambda of a weakened term is constantly that term. The fact that lam preserves the equations is the source of the three equations $\mathsf{lamK}\beta$, $\mathsf{lamS}\beta$ and $\mathsf{lamwk}\cdot$. The naming of these equations is now revealed.

$$
\begin{aligned}
&\mathsf{lam}\,\mathsf{K}\beta && : \mathsf{lam}\,(\mathsf{K} \cdot u \cdot v) && \equiv \\
& && \quad \mathsf{S} \cdot \mathsf{lam}\,(\mathsf{K} \cdot u) \cdot \mathsf{lam}\,v && \equiv \\
& && \quad \mathsf{S} \cdot \big(\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{K}) \cdot \mathsf{lam}\,u\big) \cdot \mathsf{lam}\,v && = (\mathsf{lamK}\beta'') \\
& && \quad \mathsf{lam}\,u \\
&\mathsf{lam}\,\mathsf{S}\beta && : \mathsf{lam}\,(\mathsf{S} \cdot t \cdot u \cdot v) && \equiv \\
& && \quad \mathsf{S} \cdot \mathsf{lam}\,(\mathsf{S} \cdot t \cdot u) \cdot \mathsf{lam}\,v && \equiv \\
& && \quad \mathsf{S} \cdot \big(\mathsf{S} \cdot \mathsf{lam}\,(\mathsf{S} \cdot t) \cdot \mathsf{lam}\,u\big) \cdot \mathsf{lam}\,v && \equiv \\
& && \quad \mathsf{S} \cdot \big(\mathsf{S} \cdot (\mathsf{S} \cdot (\mathsf{K} \cdot \mathsf{S}) \cdot \mathsf{lam}\,t) \cdot \mathsf{lam}\,u\big) \cdot \mathsf{lam}\,v && = (\mathsf{lamS}\beta'') \\
& && \quad \mathsf{S} \cdot (\mathsf{S} \cdot \mathsf{lam}\,t \cdot \mathsf{lam}\,u) \cdot (\mathsf{S} \cdot \mathsf{lam}\,u \cdot \mathsf{lam}\,v) && \equiv \\
& && \quad \mathsf{S} \cdot \mathsf{lam}\,(t \cdot u) \cdot \mathsf{lam}\,(u \cdot v) && \equiv \\
& && \quad \mathsf{lam}\,\big(t \cdot v \cdot (u \cdot v)\big)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{lam}\,\mathsf{wk}\cdot && : \mathsf{lam}\,(\mathsf{wk}\,(t \cdot u)) \equiv \mathsf{K} \cdot (t \cdot u) \overset{\mathsf{lamwk}\cdot''}{=} \mathsf{S} \cdot (\mathsf{K} \cdot t) \cdot (\mathsf{K} \cdot u) \equiv \mathsf{lam}\,(\mathsf{wk}\,t \cdot \mathsf{wk}\,u) \\
&\mathsf{lam}\,\mathsf{wkK} && : \mathsf{lam}\,(\mathsf{wk}\,\mathsf{K}) \equiv \mathsf{K} \cdot \mathsf{K} \equiv \mathsf{lam}\,\mathsf{K} \\
&\mathsf{lam}\,\mathsf{wkS} && : \mathsf{lam}\,(\mathsf{wk}\,\mathsf{K}) \equiv \mathsf{K} \cdot \mathsf{S} \equiv \mathsf{lam}\,\mathsf{S}
\end{aligned}
$$

We make use of the double-primed non-closed and pointful variants, but we put the closed pointfree variants in the definition of Cwk because they are vacuously preserved by lam as lam operates on terms in the nonempty context. The last of the four extensionality equations η will be used to prove the η law for the defined lam.

Now we prove all the equations of L in the following order. The proofs are straightforward, see Appendix B or the formalisation for the details.

$[\circ]$	$: \{t : \mathsf{Tm}\,\Gamma\,A\} \to t[\sigma \circ \rho] = t[\sigma][\rho]$	induction on t
ass	$: \{\Gamma : \mathsf{Con}\}\{\sigma : \mathsf{Sub}\,\Delta\,\Gamma\} \to (\sigma \circ \rho) \circ \tau = \sigma \circ (\rho \circ \tau)$	induction on Γ
$\mathsf{wks}\circ$	$: \{\Gamma : \mathsf{Con}\} \to \mathsf{wks}\,\{\Gamma\}\,\sigma \circ (\rho, u) = \sigma \circ \rho$	induction on Γ
idl	$: \{\Gamma : \mathsf{Con}\} \to \mathsf{id}\,\{\Gamma\} \circ \sigma = \sigma$	induction on Γ
$[\mathsf{wks}]$	$: \{t : \mathsf{Tm}\,\Gamma\,A\} \to t[\mathsf{wks}\,\sigma] = \mathsf{wk}\,(t[\sigma])$	induction on t
$[\mathsf{id}]$	$: \{t : \mathsf{Tm}\,\Gamma\,A\} \to t[\mathsf{id}] = t$	induction on t
idr	$: \{\Gamma : \mathsf{Con}\} \to \sigma \circ^{\Gamma} \mathsf{id} = \sigma$	induction on Γ
$\diamond\eta$	$: \{\sigma : \mathsf{Sub}\,\Gamma\,\diamond\} \to \sigma = \epsilon$	holds by definition
$\triangleright\beta_1$	$: \mathsf{p} \circ (\sigma, t) \equiv \mathsf{wks}\,\mathsf{id} \circ (\sigma, t) \overset{\mathsf{wks}\circ}{=} \mathsf{id} \circ \sigma \overset{\mathsf{idl}}{=} \sigma$	derivable
$\triangleright\beta_2$	$: \mathsf{q}[\sigma, t] \equiv t$	holds by definition
$\triangleright\eta$	$: (\sigma, t) \overset{\triangleright\beta_1}{=} (\mathsf{p} \circ (\sigma, t), t) \equiv (\mathsf{p} \circ (\sigma, t), \mathsf{q}[\sigma, t])$	derivable
$\Rightarrow\beta$	$: \{t : \mathsf{Tm}\,(\Gamma \triangleright A)\,B\} \to \mathsf{lam}\,t \cdot v = t[\mathsf{id}, v]$	induction on t
$\Rightarrow\eta$	$: t = \mathsf{lam}\,(t[\mathsf{p}] \cdot \mathsf{q})$	derivable using η''
$\mathsf{lam}[]$	$: (\mathsf{lam}\,t)[\sigma] = \mathsf{lam}\,(t[\sigma \circ \mathsf{p}, \mathsf{q}])$	induction on t
$\cdot[]$	$: (t \cdot u)[\sigma] \equiv (t[\sigma]) \cdot (u[\sigma])$	holds by definition

Finally we show how instantiation interacts with the combinators K and S:

$$\mathsf{K}[] : \mathsf{K}[\sigma] \equiv \mathsf{K}[\mathsf{wks}\,\mathsf{id}] \overset{[\mathsf{wks}]}{=} \mathsf{wk}\,(\mathsf{K}[\mathsf{id}]) \overset{[\mathsf{id}]}{=} \mathsf{wk}\,\mathsf{K} \overset{\mathsf{wkK}}{=} \mathsf{K}$$

$$\mathsf{S}[] : \mathsf{S}[\sigma] \equiv \mathsf{S}[\mathsf{wks}\,\mathsf{id}] \overset{[\mathsf{wks}]}{=} \mathsf{wk}\,(\mathsf{S}[\mathsf{id}]) \overset{[\mathsf{id}]}{=} \mathsf{wk}\,\mathsf{S} \overset{\mathsf{wkS}}{=} \mathsf{S}$$

5 Combinators with weakenings and lambda terms are equal

The proof-relevant parts of $\mathsf{g} : \mathsf{Tm}_{\mathsf{Cwk}}\,\Gamma\,A \cong \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A$ are defined in Figure 6.

$$
\begin{aligned}
&\mathsf{g} : \mathsf{Tm}_{\mathsf{Cwk}}\,\Gamma\,A \to \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A \\
&\mathsf{g}\,(t \cdot u) :\equiv \mathsf{g}\,t \cdot \mathsf{g}\,u \\
&\mathsf{g}\,\mathsf{K} \quad :\equiv \mathsf{lam}\,(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}])) \\
&\mathsf{g}\,\mathsf{S} \quad :\equiv \mathsf{lam}\Big(\mathsf{lam}\,\big(\mathsf{lam}\,(\mathsf{q}[\mathsf{p} \circ \mathsf{p}] \cdot \mathsf{q} \cdot (\mathsf{q}[\mathsf{p}] \cdot \mathsf{q}))\big)\Big) \\
&\mathsf{g}\,\mathsf{q} \quad :\equiv \mathsf{q} \\
&\mathsf{g}\,(\mathsf{wk}\,t) :\equiv (\mathsf{g}\,t)[\mathsf{p}] \\
\\
&\mathsf{g} : \{\Gamma : \mathsf{Con}\} \to \mathsf{Sub}_{\mathsf{Cwk}}\,\Delta\,\Gamma \to \mathsf{Sub}_{\mathsf{L}}\,\Delta\,\Gamma \\
&\mathsf{g}\,\{\diamond\} \quad \mathsf{tt} \quad :\equiv \epsilon \\
&\mathsf{g}\,\{\Gamma \triangleright A\}\,(\sigma, t) :\equiv (\mathsf{g}\,\sigma, \mathsf{g}\,t)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{g}^{-1} : \mathsf{Sub}_{\mathsf{L}}\,\Delta\,\Gamma \to \mathsf{Sub}_{\mathsf{Cwk}}\,\Delta\,\Gamma \\
&\mathsf{g}^{-1} : \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A \to \mathsf{Tm}_{\mathsf{Cwk}}\,\Gamma\,A \\
&\mathsf{g}^{-1}\,(\sigma \circ \rho) :\equiv \mathsf{g}^{-1}\,\sigma \circ \mathsf{g}^{-1}\,\rho \\
&\mathsf{g}^{-1}\,\mathsf{id} \quad :\equiv \mathsf{id} \\
&\mathsf{g}^{-1}\,\epsilon \quad :\equiv \mathsf{tt} \\
&\mathsf{g}^{-1}\,(t[\sigma]) \quad :\equiv (\mathsf{g}^{-1}\,t)[\mathsf{g}^{-1}\,\sigma] \\
&\mathsf{g}^{-1}\,(\sigma, t) \quad :\equiv (\mathsf{g}^{-1}\,\sigma, \mathsf{g}^{-1}\,t) \\
&\mathsf{g}^{-1}\,\mathsf{p} \quad :\equiv \mathsf{p} \\
&\mathsf{g}^{-1}\,\mathsf{q} \quad :\equiv \mathsf{q} \\
&\mathsf{g}^{-1}\,(\mathsf{lam}\,t) :\equiv \mathsf{lam}\,(\mathsf{g}^{-1}\,t) \\
&\mathsf{g}^{-1}\,(t \cdot u) \quad :\equiv (\mathsf{g}^{-1}\,t) \cdot (\mathsf{g}^{-1}\,u)
\end{aligned}
$$

Figure 6 The proof-relevant parts of the isomorphism $\mathsf{g} : \mathsf{Tm}_{\mathsf{Cwk}}\,\Gamma\,A \cong \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A$. We treat the cases for equality constructors in the main text.

Mapping Cwk terms to L terms (left hand side of Figure 6) is easy for those accustomed to the lambda calculus. The combinatory operations have straightforward implementations using lambda terms. It is similarly straightforward to show that g preserves the equations of Cwk. As an example we show all the steps in proving preservation of $\mathsf{K}\beta$. This also illustrates working with the equational theory of L, or in other words working within a CwF. The preservation of the other equations is proven in an analogous way.

$$
\begin{aligned}
\mathsf{g}\,\mathsf{K}\beta : \mathsf{g}\,(\mathsf{K}\cdot u\cdot v) \quad & \equiv \\
\mathsf{lam}\,\big(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}])\big)\cdot \mathsf{g}\,u\cdot \mathsf{g}\,v \quad & =(\Rightarrow\beta) \\
\mathsf{lam}\,(\mathsf{q}[\mathsf{p}])[\mathsf{id},\mathsf{g}\,u]\cdot \mathsf{g}\,v \quad & =(\mathsf{lam}[]) \\
\mathsf{lam}\,\big(\mathsf{q}[\mathsf{p}][(\mathsf{id},\mathsf{g}\,u)\circ\mathsf{p},\mathsf{q}]\big)\cdot \mathsf{g}\,v \quad & =([\circ]) \\
\mathsf{lam}\,\big(\mathsf{q}[\mathsf{p}\circ((\mathsf{id},\mathsf{g}\,u)\circ\mathsf{p},\mathsf{q})]\big)\cdot \mathsf{g}\,v \quad & =(\triangleright\beta_1) \\
\mathsf{lam}\,\big(\mathsf{q}[(\mathsf{id},\mathsf{g}\,u)\circ\mathsf{p}]\big)\cdot \mathsf{g}\,v \quad & =(,\circ) \\
\mathsf{lam}\,\big(\mathsf{q}[\mathsf{id}\circ\mathsf{p},\mathsf{g}\,u[\mathsf{p}]]\big)\cdot \mathsf{g}\,v \quad & =(\triangleright\beta_2) \\
\mathsf{lam}\,\big((\mathsf{g}\,u)[\mathsf{p}]\big)\cdot \mathsf{g}\,v \quad & =(\Rightarrow\beta) \\
(\mathsf{g}\,u)[\mathsf{p}][\mathsf{id},\mathsf{g}\,v] \quad & =([\circ]) \\
(\mathsf{g}\,u)[\mathsf{p}\circ(\mathsf{id},\mathsf{g}\,v)] \quad & =(\triangleright\beta_1) \\
(\mathsf{g}\,u)[\mathsf{id}] \quad & =([\mathsf{id}]) \\
\mathsf{g}\,u \quad &
\end{aligned}
$$

We also define g on $\mathsf{Sub}_{\mathsf{Cwk}}$ by induction on the target context, and we prove that g preserves the lambda operations defined in Section 4 as follows, in the following order.

$$
\begin{aligned}
&\mathsf{gwks} : \{\sigma : \mathsf{Sub}\,\Delta\,\Gamma\} \to \mathsf{g}\,(\mathsf{wks}\,\sigma) = \mathsf{g}\,\mathsf{wks}\circ\mathsf{p} && \text{induction on } \Gamma \\
&\mathsf{gid} \quad : \{\Gamma : \mathsf{Con}\} \to \mathsf{g}\,(\mathsf{id}\,\{\Gamma\}) = \mathsf{id} && \text{induction on } \Gamma \\
&\mathsf{gp} \quad : \mathsf{g}\,\mathsf{p} \equiv \mathsf{g}\,(\mathsf{wks}\,\mathsf{id}) \overset{\mathsf{gwks}}{=} \mathsf{g}\,\mathsf{id}\circ\mathsf{p} \overset{\mathsf{gid}}{=} \mathsf{id}\circ\mathsf{p} \overset{\mathsf{idl}}{=} \mathsf{p} && \text{derivable} \\
&\mathsf{g}[] \quad : \mathsf{g}\,(t[\sigma]) = (\mathsf{g}\,t)[\mathsf{g}\,\sigma] && \text{induction on } t \\
&\mathsf{g}\circ \quad : \{\sigma : \mathsf{Sub}\,\Delta\,\Gamma\} \to \mathsf{g}\,(\sigma\circ\rho) = \mathsf{g}\,\sigma\circ\mathsf{g}\rho && \text{induction on } \Gamma \\
&\mathsf{glam} : \mathsf{g}\,(\mathsf{lam}\,t) = \mathsf{lam}\,(\mathsf{g}\,t) && \text{induction on } t
\end{aligned}
$$

In the other direction we use the lambda calculus operations defined in Cwk in Section 4. g^{-1} is defined by mutual recursion on $\mathsf{Sub}_{\mathsf{L}}$ and Tm_{L} on the right hand side of Figure 6. Preservation of the equations correspond to the counterparts of the equations in Cwk that we proved in Section 4. So $\mathsf{g}^{-1}\,\mathsf{ass}_{\mathsf{L}} :\equiv \mathsf{ass}_{\mathsf{Cwk}}, \ \dots, \ \mathsf{g}^{-1}\,\cdot[]_{\mathsf{L}} :\equiv \cdot[]_{\mathsf{Cwk}}$.

The first roundtrip is proven by induction on $\mathsf{Tm}_{\mathsf{Cwk}}$ as follows.

$$
\begin{aligned}
&\mathsf{g}^{\beta} : \{t : \mathsf{Tm}_{\mathsf{Cwk}}\,\Gamma\,A\} \to \mathsf{g}^{-1}\,(\mathsf{g}\,t) = t \\
&\mathsf{g}^{\beta}\,\{t\cdot u\} : \mathsf{g}^{-1}\,(\mathsf{g}\,(t\cdot u)) \equiv \mathsf{g}^{-1}\,(\mathsf{g}\,t)\cdot\mathsf{g}^{-1}\,(\mathsf{g}\,t) \overset{\mathsf{g}^{\beta}\,\{t\},\mathsf{g}^{\beta}\,\{t\}}{=} t\cdot u \\
&\mathsf{g}^{\beta}\,\{\mathsf{K}\} \quad : \mathsf{g}^{-1}\,(\mathsf{g}\,\mathsf{K}) \equiv \mathsf{g}^{-1}\,(\mathsf{lam}\,(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}]))) \equiv \mathsf{lam}\,(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}])) \overset{\mathsf{funext}\,(\mathsf{funext}\,\mathsf{K}^{=})}{=} \mathsf{K} \\
&\mathsf{g}^{\beta}\,\{\mathsf{S}\} \quad : \mathsf{g}^{-1}\,(\mathsf{g}\,\mathsf{S}) \equiv \mathsf{lam}\left(\mathsf{lam}\left(\mathsf{lam}\,\big(\mathsf{q}[\mathsf{p}\circ\mathsf{p}]\cdot\mathsf{q}\cdot(\mathsf{q}[\mathsf{p}]\cdot\mathsf{q})\big)\right)\right) \overset{\mathsf{funext}\,(\mathsf{funext}\,(\mathsf{funext}\,\mathsf{S}^{=}))}{=} \mathsf{S} \\
&\mathsf{g}^{\beta}\,\{\mathsf{q}\} \quad : \mathsf{g}^{-1}\,(\mathsf{g}\,\mathsf{q}) \equiv \mathsf{g}^{-1}\,\mathsf{q} \equiv \mathsf{q} \\
&\mathsf{g}^{\beta}\,\{\mathsf{wk}\,t\} : \mathsf{g}^{-1}\,(\mathsf{g}\,(\mathsf{wk}\,t)) \equiv (\mathsf{g}^{-1}\,(\mathsf{g}\,t))[\mathsf{p}] \overset{\mathsf{g}^{\beta}\,\{t\}}{=} t[\mathsf{p}] \equiv t[\mathsf{wks}\,\mathsf{id}] \overset{[\mathsf{wks}]}{=} \mathsf{wk}\,(t[\mathsf{id}]) \overset{[\mathsf{id}]}{=} \mathsf{wk}\,t
\end{aligned}
$$

The interesting cases are K and S: here we use function extensionality which holds in any model of L (thus in Cwk as shown in Section 4). To prove that the implementation of K using $\mathsf{lam}\,(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}]))$ is equal to K we apply funext twice, and then we can use well-known reasoning with CwF combinators. The proof of $\mathsf{S}^{=}$ is analogous.

$$
\begin{aligned}
\mathsf{K}^{=} : \Big(&(\mathsf{lam}\,(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}])))[\mathsf{p}]\cdot\mathsf{q}\Big)[\mathsf{p}]\cdot\mathsf{q} && =(\text{CwF reasoning})\\
&\mathsf{lam}\,(\mathsf{lam}\,(\mathsf{q}[\mathsf{p}]))\cdot\mathsf{q}[\mathsf{p}]\cdot\mathsf{q} && =(\Rightarrow\beta \text{ twice and more CwF reasoning})\\
&\mathsf{q}[\mathsf{p}] && =(\mathsf{K}\beta)\\
&\mathsf{K}\cdot\mathsf{q}[\mathsf{p}]\cdot\mathsf{q} && =(\mathsf{K}[]\text{ twice})\\
&\mathsf{K}[\mathsf{p}][\mathsf{p}]\cdot\mathsf{q}[\mathsf{p}]\cdot\mathsf{q} && =(\cdot[])\\
&(\mathsf{K}[\mathsf{p}]\cdot\mathsf{q})[\mathsf{p}]\cdot\mathsf{q}
\end{aligned}
$$

The second roundtrip is a mutual induction on Sub_L and Tm_L. We make use of the fact equations saying that g preserves the lambda operations .

$$
\begin{aligned}
&\mathsf{g}^{\eta} : \{\sigma : \mathsf{Sub}_\mathsf{L}\,\Delta\,\Gamma\} \to \mathsf{g}\,(\mathsf{g}^{-1}\,\sigma) = \sigma\\
&\mathsf{g}^{\eta} : \{t : \mathsf{Tm}_\mathsf{L}\,\Gamma\,A\} \to \mathsf{g}\,(\mathsf{g}^{-1}\,t) = t\\
&\mathsf{g}^{\eta}\,\{\sigma\circ\rho\} : \mathsf{g}\,(\mathsf{g}^{-1}\,(\sigma\circ\rho)) \equiv \mathsf{g}\,(\mathsf{g}^{-1}\,\sigma\circ\mathsf{g}^{-1}\,\rho) \overset{\mathsf{g}\circ}{=} \mathsf{g}\,(\mathsf{g}^{-1}\,\sigma)\circ\mathsf{g}\,(\mathsf{g}^{-1}\,\rho) \overset{\mathsf{g}^{\eta}\,\{\sigma\},\mathsf{g}^{\eta}\,\{\rho\}}{=} \sigma\circ\rho\\
&\mathsf{g}^{\eta}\,\{\mathsf{id}\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,\mathsf{id}) \equiv \mathsf{g}\,\mathsf{id} \overset{\mathsf{gid}}{=} \mathsf{id}\\
&\mathsf{g}^{\eta}\,\{\epsilon\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,\epsilon) \equiv \mathsf{g}\,\{\diamond\}\mathsf{tt} \equiv \epsilon\\
&\mathsf{g}^{\eta}\,\{t[\sigma]\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,(t[\sigma])) \equiv \mathsf{g}\,\big((\mathsf{g}^{-1}\,t)[\mathsf{g}^{-1}\,\sigma]\big) \overset{\mathsf{g}[]}{=} (\mathsf{g}\,(\mathsf{g}^{-1}\,t))[\mathsf{g}\,(\mathsf{g}^{-1}\,\sigma)] \overset{\mathsf{g}^{\eta}\,\{t\},\mathsf{g}^{\eta}\,\{\rho\}}{=} t[\sigma]\\
&\mathsf{g}^{\eta}\,\{\sigma,t\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,(\sigma,t)) \equiv \mathsf{g}\,\{\Gamma\triangleright A\}\,(\mathsf{g}^{-1}\,\sigma,\mathsf{g}^{-1}\,t) \equiv \big(\mathsf{g}\,(\mathsf{g}^{-1}\,\sigma),\mathsf{g}\,(\mathsf{g}^{-1}\,t)\big) \overset{\mathsf{g}^{\eta}}{=} (\sigma,t)\\
&\mathsf{g}^{\eta}\,\{\mathsf{p}\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,\mathsf{p}) \equiv \mathsf{g}\,\mathsf{p} \overset{\mathsf{gp}}{=} \mathsf{p}\\
&\mathsf{g}^{\eta}\,\{\mathsf{q}\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,\mathsf{q}) \equiv t\\
&\mathsf{g}^{\eta}\,\{\mathsf{lam}\,t\} : \mathsf{g}\,(\mathsf{g}^{-1}\,(\mathsf{lam}\,t)) \equiv \mathsf{g}\,\big(\mathsf{lam}\,(\mathsf{g}^{-1}\,t)\big) \overset{\mathsf{glam}}{=} \mathsf{lam}\,(\mathsf{g}\,(\mathsf{g}^{-1}\,t)) \overset{\mathsf{g}^{\eta}\,\{t\}}{=} \mathsf{lam}\,t\\
&\mathsf{g}^{\eta}\,\{t\cdot u\} \quad : \mathsf{g}\,(\mathsf{g}^{-1}\,(t\cdot u)) \equiv \mathsf{g}\,(\mathsf{g}^{-1}\,t)\cdot\mathsf{g}\,(\mathsf{g}^{-1}\,u) \overset{\mathsf{g}^{\eta}\,t,\mathsf{g}^{\eta}\,u}{=} t\cdot u
\end{aligned}
$$

6 Lambda terms can be moved to the empty context

We define $\mathsf{h} : \mathsf{Tm}_\mathsf{L}\,\diamond\,(\Gamma \Rightarrow^{*} A) \cong \mathsf{Tm}_\mathsf{L}\,\Gamma\,A$. Both directions and the roundtrips are defined by induction on Γ.

$$
\begin{aligned}
&\mathsf{h} : \{\Gamma : \mathsf{Con}\} \to \mathsf{Tm}_\mathsf{L}\,\diamond\,(\Gamma \Rightarrow^{*} A) \to \mathsf{Tm}_\mathsf{L}\,\Gamma\,A\\
&\mathsf{h} \quad \{\diamond\} \qquad t :\equiv t\\
&\mathsf{h} \quad \{\Gamma\triangleright B\}\,t :\equiv (\mathsf{h}\,\{\Gamma\}\,t)[\mathsf{p}]\cdot\mathsf{q}\\
&\mathsf{h}^{-1} : \{\Gamma : \mathsf{Con}\} \to \mathsf{Tm}_\mathsf{L}\,\Gamma\,A \to \mathsf{Tm}_\mathsf{L}\,\diamond\,(\Gamma \Rightarrow^{*} A)\\
&\mathsf{h}^{-1}\,\{\diamond\} \qquad t :\equiv t\\
&\mathsf{h}^{-1}\,\{\Gamma\triangleright B\}\,t :\equiv \mathsf{h}^{-1}\,\{\Gamma\}\,(\mathsf{lam}\,t)\\
&\mathsf{h}^{\beta} : \{\Gamma : \mathsf{Con}\} \to \mathsf{h}^{-1}\,\{\Gamma\}\,(\mathsf{h}\,\{\Gamma\}\,t) = t\\
&\mathsf{h}^{\beta} \quad \{\diamond\} \qquad : \mathsf{h}^{-1}\,\{\diamond\}\,(\mathsf{h}\,\{\diamond\}\,t) \equiv t
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{h}^{\beta}\ \{\Gamma \rhd B\} : \mathsf{h}^{-1}\,\{\Gamma \rhd B\}\,(\mathsf{h}\,\{\Gamma \rhd B\}\,t) \equiv \mathsf{h}^{-1}\,\big(\mathsf{lam}\,((\mathsf{h}\,t)[\mathsf{p}]\cdot \mathsf{q})\big) \overset{\Rightarrow\eta}{=} \mathsf{h}^{-1}\,(\mathsf{h}\,t) \overset{\mathsf{h}^{\beta}\,\{\Gamma\}}{=} t \\
&\mathsf{h}^{\eta} : \{\Gamma : \mathsf{Con}\} \to \mathsf{h}\,\{\Gamma\}\,(\mathsf{h}^{-1}\,\{\Gamma\}\,t) = t \\
&\mathsf{h}^{\eta}\ \{\diamond\} \qquad : \mathsf{h}\,\{\diamond\}\,(\mathsf{h}^{-1}\,\{\diamond\}\,t) \equiv t \\
&\mathsf{h}^{\eta}\ \{\Gamma \rhd B\}\ : \mathsf{h}\,\{\Gamma \rhd B\}\,(\mathsf{h}^{-1}\,\{\Gamma \rhd B\}\,t) \equiv \\
&\qquad\qquad \big(\mathsf{h}\,(\mathsf{h}^{-1}\,(\mathsf{lam}\,t))\big)[\mathsf{p}]\cdot \mathsf{q} \qquad =(\mathsf{h}^{\eta}\,\{\Gamma\}) \\
&\qquad\qquad (\mathsf{lam}\,t)[\mathsf{p}]\cdot \mathsf{q} \qquad =(\mathsf{lam}[]) \\
&\qquad\qquad \mathsf{lam}\,(t[\mathsf{p}\circ\mathsf{p},\mathsf{q}])\cdot \mathsf{q} \qquad =(\Rightarrow\beta) \\
&\qquad\qquad t[\mathsf{p}\circ\mathsf{p},\mathsf{q}][\mathsf{id},\mathsf{q}] \qquad =(\text{CwF reasoning}) \\
&\qquad\qquad t[\mathsf{id}] \qquad =([\mathsf{id}]) \\
&\qquad\qquad t
\end{aligned}
$$

Putting together f from Section 3, g from Section 5 and h, we obtain $\mathsf{Tm}_{\mathsf{C}}\,(\Gamma \Rightarrow^{*} A) \cong \mathsf{Tm}_{\mathsf{L}}\,\Gamma\,A$.

7 Conclusions and further work

We proved the equivalence of the syntax of combinator logic and lambda calculus in an abstract setting. In this algebraic setting we do not refer to specific representations of the syntaxes. In particular, the bracket abstraction algorithm (defining lambda for combinators) preserves all equations. We believe that avoiding talking about representations is benificial because the proof is more general, it applies to all representations. Thus we can extend the title of Selinger's paper [22] saying that lambda calculus is algebraic and (at least some) proofs about lambda calculus can be done algebraically. Moreover, we can run all the proofs even in this abstract setting using QIITs of Cubical Agda. For example, given a formalisation of normalisation for lambda calculus, we also obtain an algorithm for normalisation of combinator terms up to extensionality. In the future, it would be interesting to characterise normal forms of extensional combinatory logic using this technique.

There are several remaining open questions regarding the algebraic presentation of combinatory logic. For example, we do not know how to define lambda by recursion on the syntax of the following theories, even though they are all equivalent to Cwk: extensional combinatory logic with only variables; combinatory logic without the four extensionality equations but with funext; simply typed CwF with application, K, S and extensionality. In the future we would like to describe combinatory logic algebraically with ξ but without η following [7]. In the other direction, we would like to define an algebraic presentation of lambd calculus that is equivalent to combinatory logic without the extensionality equations. We only related the syntaxes of lambda calculus and combinator logic, but more generally, we can turn any model of lambda calculus into a model of combinatory logic, while the other way we have to restrict to definable terms. It would be interesting to formalise this more general correspondance.

Another future research direction is the algebraic presentation of dependently typed combinatory logic following [24, 3].

References

1 Thorsten Altenkirch and Ambrus Kaposi. Normalisation by evaluation for dependent types. In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal*, volume 52

of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/LIPIcs.FSCD.2016.6`.

2 Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 18–29. ACM, 2016. `doi:10.1145/2837614.2837638`.

3 Thorsten Altenkirch, Ambrus Kaposi, Artjoms Šinkarovs, and Tamás Végh. The Münchhausen method and combinatory type theory. In Delia Kesner and Pierre-Marie Pédrot, editors, *28th International Conference on Types for Proofs and Programs (TYPES 2022)*. University of Nantes, 2022. URL: `https://types22.inria.fr/files/2022/06/TYPES_2022_paper_8.pdf`.

4 Robert Atkey. Syntax and semantics of quantitative type theory. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 56–65. ACM, 2018. `doi:10.1145/3209108.3209189`.

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

6 Katalin Bimbó. *Combinatory Logic: Pure, Applied and Typed*. Taylor & Francis, 2011.

7 Martin W. Bunder, J. Roger Hindley, and Jonathan P. Seldin. On adding (xi) to weak equality in combinatory logic. *J. Symb. Log.*, 54(2):590–607, 1989. `doi:10.2307/2274872`.

8 John Cartmell. Generalised algebraic theories and contextual categories. *Ann. Pure Appl. Log.*, 32:209–243, 1986. `doi:10.1016/0168-0072(86)90053-9`.

9 Simon Castellan, Pierre Clairambault, and Peter Dybjer. Categories with families: Unityped, simply typed, and dependently typed. *CoRR*, abs/1904.00827, 2019. URL: `http://arxiv.org/abs/1904.00827`, `arXiv:1904.00827`.

10 Haskell B. Curry, Robert Feys, and William Craig. Combinatory logic, volume i. *Philosophical Review*, 68(4):548–550, 1959. `doi:10.2307/2182503`.

11 J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, USA, 2 edition, 2008.

12 J. M. E. Hyland. Classical lambda calculus in modern dress. *Math. Struct. Comput. Sci.*, 27(5):762–781, 2017. `doi:10.1017/S0960129515000377`.

13 Ambrus Kaposi. Formalisation of type checking into algebraic syntax. `https://bitbucket.org/akaposi/tt-in-tt/src/master/Typecheck.agda`, 2018.

14 Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. *Proc. ACM Program. Lang.*, 3(POPL):2:1–2:24, 2019. `doi:10.1145/3290315`.

15 András Kovács. Elaboration with first-class implicit function types. *Proc. ACM Program. Lang.*, 4(ICFP):101:1–101:29, 2020. `doi:10.1145/3408983`.

16 András Kovács. *Type-Theoretic Signatures for Algebraic Theories and Inductive Types*. PhD thesis, Eötvös Loránd University, Hungary, 2022. URL: `https://andraskovacs.github.io/pdfs/phdthesis_compact.pdf`.

17 J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, USA, 1986.

18 Conor McBride. Outrageous but meaningful coincidences: dependent type-safe syntax and evaluation. In Bruno C. d. S. Oliveira and Marcin Zalewski, editors, *Proceedings of the ACM SIGPLAN Workshop on Generic Programming, WGP 2010, Baltimore, MD, USA, September 27-29, 2010*, pages 1–12. ACM, 2010. `doi:10.1145/1863495.1863497`.

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

20 Jonathan P. Seldin. The search for a reduction in combinatory logic equivalent to $\lambda\beta$-reduction. *Theor. Comput. Sci.*, 412(37):4905–4918, 2011. `doi:10.1016/j.tcs.2011.02.002`.

21 Jonathan P. Seldin. The search for a reduction in combinatory logic equivalent to $\lambda\beta$-reduction, part II. *Theor. Comput. Sci.*, 663:34–58, 2017. `doi:10.1016/j.tcs.2016.12.016`.

22 Peter Selinger. The lambda calculus is algebraic. *J. Funct. Program.*, 12(6):549–566, 2002. `doi:10.1017/S0956796801004294`.

23 Wouter Swierstra. A correct-by-construction conversion to combinators, 2021. `https://webspace.science.uu.nl/~swier004/publications/2021-jfp-submission-2.pdf`https://webspace.science.uu.nl/ swier004/publications/2021-jfp-submission-2.pdf.

24 William W Tait. Variable-free formalization of the Curry-Howard Theory. In *Twenty Five Years of Constructive Type Theory*. Oxford University Press, 10 1998. `arXiv:https://academic.oup.com/book/0/chapter/355196043/chapter-pdf/43763003/isbn-9780198501275-book-part-16.pdf`, `doi:10.1093/oso/9780198501275.003.0016`.

25 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical agda: A dependently typed programming language with univalence and higher inductive types. *J. Funct. Program.*, 31:e8, 2021. `doi:10.1017/S0956796821000034`.

26 Philip Wadler, Wen Kokke, and Jeremy G. Siek. *Programming Language Foundations in Agda*. August 2022. URL: `https://plfa.inf.ed.ac.uk/20.08/`.

A Derivation of the pointful versions of the extensionality equations from the pointfree variants

In this section we derive $\mathsf{lamS}\beta''$ from $\mathsf{lamS}\beta'$, $\mathsf{lamwk}\text{·}\beta''$ from $\mathsf{lamwk}\text{·}\beta'$, η'' from η. See Subsection 2.2 for the derivation of $\mathsf{lamK}\beta'$ from $\mathsf{lamK}\beta''$.

$$
\begin{aligned}
\mathsf{lamS}\beta'' \ :\ & \mathsf{S}\cdot\big(\mathsf{S}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot t)\cdot u\big)\cdot v && =(\mathsf{K}\beta)\\
& \mathsf{K}\cdot\mathsf{S}\cdot u\cdot\Big(\mathsf{S}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot t)\cdot u\Big)\cdot v && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\Big(\mathsf{S}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot t)\Big)\cdot u\cdot v && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\Big(\mathsf{K}\cdot\mathsf{S}\cdot t\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot t)\Big)\cdot u\cdot v && =(\mathsf{K}\beta,\mathsf{S}\beta)\\
& \mathsf{K}\cdot\big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\big)\cdot t\cdot\Big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S}))\cdot t\Big)\cdot u\cdot v && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot\Big(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S}))\Big)\cdot\Big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S}))\Big)\cdot t\cdot u\cdot v && =(\mathsf{lamS}\beta')\\
& \mathsf{S}\cdot\Big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\Big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{K})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})))\cdot\mathsf{S}))\Big)\Big)\cdot\\
& (\mathsf{K}\cdot\mathsf{S})\cdot t\cdot u\cdot v && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\Big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{K})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})))\cdot\mathsf{S}))\Big)\cdot t\cdot\\
& (\mathsf{K}\cdot\mathsf{S}\cdot t)\cdot u\cdot v && =(\mathsf{S}\beta,\mathsf{K}\beta)\\
& \mathsf{S}\cdot\Big(\mathsf{K}\cdot\mathsf{K}\cdot t\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})))\cdot\mathsf{S})\cdot t)\Big)\cdot\mathsf{S}\cdot u\cdot v && =(\mathsf{K}\beta,\mathsf{S}\beta)\\
& \mathsf{S}\cdot\Big(\mathsf{K}\cdot(\mathsf{K}\cdot\mathsf{S}\cdot t\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})))\cdot\mathsf{S}\cdot t))\Big)\cdot\mathsf{S}\cdot u\cdot v && =(\mathsf{K}\beta,\mathsf{S}\beta)\\
& \mathsf{S}\cdot\Big(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S}))\cdot t\cdot(\mathsf{S}\cdot t)))\Big)\cdot\mathsf{S}\cdot u\cdot v && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot\Big(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot t)))\Big)\cdot\mathsf{S}\cdot u\cdot v && =(\mathsf{S}\beta)\\
& \mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot t)))\cdot u\cdot(\mathsf{S}\cdot u)\cdot v && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot t))\cdot(\mathsf{S}\cdot u)\cdot v && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot t)\cdot v\cdot(\mathsf{S}\cdot u\cdot v) && =(\mathsf{S}\beta)
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{K}\cdot\mathsf{S}\cdot v\cdot(\mathsf{S}\cdot t\cdot v)\cdot(\mathsf{S}\cdot u\cdot v) && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot(\mathsf{S}\cdot t\cdot u)\cdot(\mathsf{S}\cdot u\cdot v)\\
\mathsf{lamwk}{\cdot}'' :\ & \mathsf{K}\cdot(t\cdot u) && =(\mathsf{K}\beta)\\
& \mathsf{K}\cdot\mathsf{K}\cdot u\cdot(t\cdot u) && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{K})\cdot t\cdot u && =(\mathsf{lamwk}{\cdot}')\\
& \mathsf{S}\cdot\Big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{K})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\mathsf{K}))\Big)\cdot(\mathsf{K}\cdot\mathsf{K})\cdot t\cdot u && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{K})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\mathsf{K})\big)\cdot t\cdot(\mathsf{K}\cdot\mathsf{K}\cdot t)\cdot u && =(\mathsf{S}\beta,\mathsf{K}\beta)\\
& \mathsf{K}\cdot\mathsf{S}\cdot t\cdot\big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{K})\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\mathsf{K})\cdot t\big)\cdot\mathsf{K}\cdot u && =(\mathsf{K}\beta,\mathsf{S}\beta)\\
& \mathsf{S}\cdot\big(\mathsf{K}\cdot\mathsf{K}\cdot t\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\mathsf{K}\cdot t)\big)\cdot\mathsf{K}\cdot u && =(\mathsf{K}\beta,\mathsf{S}\beta)\\
& \mathsf{S}\cdot\big(\mathsf{K}\cdot(\mathsf{K}\cdot\mathsf{S}\cdot t\cdot(\mathsf{K}\cdot t))\big)\cdot\mathsf{K}\cdot u && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot\big(\mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot t))\big)\cdot\mathsf{K}\cdot u && =(\mathsf{S}\beta)\\
& \mathsf{K}\cdot(\mathsf{S}\cdot(\mathsf{K}\cdot t))\cdot u\cdot(\mathsf{K}\cdot u) && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot t)\cdot(\mathsf{K}\cdot u)\\
\eta'' :\ & t && =(\mathsf{K}\beta)\\
& \mathsf{K}\cdot t\cdot(\mathsf{K}\cdot t) && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot\mathsf{K}\cdot\mathsf{K}\cdot t && =(\eta')\\
& \mathsf{S}\cdot\big(\mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\mathsf{K}\big)\cdot\big(\mathsf{K}\cdot(\mathsf{S}\cdot\mathsf{K}\cdot\mathsf{K})\big)\cdot t && =(\mathsf{S}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot\mathsf{S})\cdot\mathsf{K}\cdot t\cdot\big(\mathsf{K}\cdot(\mathsf{S}\cdot\mathsf{K}\cdot\mathsf{K})\cdot t\big) && =(\mathsf{S}\beta,\mathsf{K}\beta)\\
& \mathsf{K}\cdot\mathsf{S}\cdot t\cdot(\mathsf{K}\cdot t)\cdot(\mathsf{S}\cdot\mathsf{K}\cdot\mathsf{K}) && =(\mathsf{K}\beta)\\
& \mathsf{S}\cdot(\mathsf{K}\cdot t)\cdot(\mathsf{S}\cdot\mathsf{K}\cdot\mathsf{K})
\end{aligned}
$$

B Proofs of lambda calculus equations in Cwk

We prove all the equations stated in Section 4.

$$
\begin{aligned}
& [\circ] : \{t : \mathsf{Tm}\,\Gamma\,A\} \to t[\sigma\circ\rho] = t[\sigma][\rho]\\
& [\circ]\,\{t\cdot u\} && : (t\cdot u)[\sigma\circ\rho] \equiv (t[\sigma\circ\rho])\cdot(u[\sigma\circ\rho]) \overset{[\circ]}{=} (t[\sigma][\rho])\cdot(u[\sigma][\rho]) \equiv (t\cdot u)[\sigma][\rho]\\
& [\circ]\,\{\mathsf{K}\} && : \mathsf{K}[\sigma\circ\rho] \equiv \mathsf{K} \equiv \mathsf{K}[\sigma][\rho]\\
& [\circ]\,\{\mathsf{S}\} && : \mathsf{S}[\sigma\circ\rho] \equiv \mathsf{S} \equiv \mathsf{S}[\sigma][\rho]\\
& [\circ]\,\{\mathsf{q}\} && : \mathsf{q}[(\sigma,u)\circ\rho] \equiv u[\rho] \equiv \mathsf{q}[\sigma,u][\rho]\\
& [\circ]\,\{\mathsf{wk}\,t\} && : (\mathsf{wk}\,t)[(\sigma,u)\circ\rho] \equiv t[\sigma\circ\rho] \overset{[\circ]}{=} t[\sigma][\rho] \equiv (\mathsf{wk}\,t)[\sigma,u][\rho]
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{ass} : \{\Gamma : \mathsf{Con}\}\{\sigma : \mathsf{Sub}\,\Delta\,\Gamma\} \to (\sigma\circ\delta)\circ\tau = \sigma\circ(\delta\circ\tau)\\
& \mathsf{ass}\,\{\diamond\} && : (\mathsf{tt}\circ\rho)\circ\tau \equiv \rho\circ\tau \equiv \mathsf{tt}\circ(\rho\circ\tau)\\
& \mathsf{ass}\,\{\Gamma\triangleright A\} && : ((\sigma,t)\circ\rho)\circ\tau && \equiv\\
& && \ \ ((\sigma\circ\rho)\circ\tau, t[\rho][\tau]) && =(\mathsf{ass}\,\{\Gamma\},[\circ])\\
& && \ \ (\sigma\circ(\rho\circ\tau), t[\rho\circ\tau]) && \equiv\\
& && \ \ (\sigma,t)\circ(\rho\circ\tau)
\end{aligned}
$$

$$
\begin{aligned}
& \mathsf{wks}\circ : \{\Gamma : \mathsf{Con}\} \to \mathsf{wks}\,\{\Gamma\}\,\sigma\circ(\rho,u) = \sigma\circ\rho\\
& \mathsf{wks}\circ\,\{\diamond\} && : \mathsf{wks}\,\{\diamond\}\,\mathsf{tt}\circ(\rho,u) \equiv \mathsf{tt}\circ(\rho,u) \equiv \mathsf{tt} \equiv \mathsf{tt}\circ\rho\\
& \mathsf{wks}\circ\,\{\Gamma\triangleright A\} && : \mathsf{wks}\,\{\Gamma\triangleright A\}\,(\sigma,t)\circ(\rho,u) && \equiv
\end{aligned}
$$

$$\begin{array}{lll}
 & (\mathsf{wks}\,\sigma, \mathsf{wk}\,t) \circ (\rho, u) & \equiv \\
 & (\mathsf{wks}\,\{\Gamma\}\,\sigma \circ (\rho, u), t[\rho]) & =(\mathsf{wks}{\circ}\,\{\Gamma\}) \\
 & (\sigma \circ \rho, t[\rho]) & \equiv \\
 & (\sigma, t) \circ \rho & \\
\multicolumn{3}{l}{\mathsf{idl} : \{\Gamma : \mathsf{Con}\} \to \mathsf{id}\,\{\Gamma\} \circ \sigma = \sigma} \\
\mathsf{idl}\,\{\diamond\} & : \mathsf{id}\,\{\diamond\} \circ \mathsf{tt} \equiv \mathsf{tt} \circ \mathsf{tt} \equiv \mathsf{tt} & \\
\mathsf{idl}\,\{\Gamma \rhd A\} & : \mathsf{id}\,\{\Gamma \rhd A\} \circ (\sigma, t) & \equiv \\
 & (\mathsf{wks}\,\mathsf{id}, \mathsf{q}) \circ (\sigma, t) & \equiv \\
 & (\mathsf{wks}\,\mathsf{id} \circ (\sigma, t), t) & =(\mathsf{wks}{\circ}) \\
 & (\mathsf{id}\,\{\Gamma\} \circ \sigma, t) & =(\mathsf{idl}\,\{\Gamma\}) \\
 & (\sigma, t) & \\
\multicolumn{3}{l}{[\mathsf{wks}] : \{t : \mathsf{Tm}\,\Gamma\,A\} \to t[\mathsf{wks}\,\sigma] = \mathsf{wk}\,(t[\sigma])} \\
[\mathsf{wks}]\,\{t \cdot u\} & : (t \cdot u)[\mathsf{wks}\,\sigma] & \equiv \\
 & (t[\mathsf{wks}\,\sigma]) \cdot (u[\mathsf{wks}\,\sigma]) & =([\mathsf{wks}]\,\{t\}, [\mathsf{wks}]\,\{u\}) \\
 & \mathsf{wk}\,(t[\sigma]) \cdot \mathsf{wk}\,(u[\sigma]) & =(\mathsf{wk}{\cdot}) \\
 & \mathsf{wk}\,((t[\sigma]) \cdot (u[\sigma])) & \equiv \\
 & \mathsf{wk}\,((t \cdot u)[\sigma]) & \\
[\mathsf{wks}]\,\{\mathsf{K}\} & : \mathsf{K}[\mathsf{wks}\,\sigma] \equiv \mathsf{K} \overset{\mathsf{wkK}}{=} \mathsf{wk}\,\mathsf{K} \equiv \mathsf{wk}\,(\mathsf{K}[\sigma]) & \\
[\mathsf{wks}]\,\{\mathsf{S}\} & : \mathsf{S}[\mathsf{wks}\,\sigma] \equiv \mathsf{S} \overset{\mathsf{wkS}}{=} \mathsf{wk}\,\mathsf{S} \equiv \mathsf{wk}\,(\mathsf{S}[\sigma]) & \\
[\mathsf{wks}]\,\{\mathsf{q}\} & : \mathsf{q}[\mathsf{wks}\,(\sigma, u)] \equiv \mathsf{q}[\mathsf{wks}\,\sigma, \mathsf{wk}\,u] \equiv \mathsf{wk}\,u \equiv \mathsf{wk}\,(\mathsf{q}[\sigma, u]) & \\
[\mathsf{wks}]\,\{\mathsf{wk}\,t\} & : (\mathsf{wk}\,t)[\mathsf{wks}\,(\sigma, u)] & \equiv \\
 & (\mathsf{wk}\,t)[\mathsf{wks}\,\sigma, \mathsf{wk}\,u] & \equiv \\
 & t[\mathsf{wks}\,\sigma] & =([\mathsf{wks}]\,\{t\}) \\
 & \mathsf{wk}\,(t[\sigma]) & \equiv \\
 & \mathsf{wk}\,((\mathsf{wk}\,t)[\sigma, u]) & \\
\multicolumn{3}{l}{[\mathsf{id}] : \{t : \mathsf{Tm}\,\Gamma\,A\} \to t[\mathsf{id}] = t} \\
[\mathsf{id}]\,\{t \cdot u\} & : (t \cdot u)[\mathsf{id}] \equiv (t[\mathsf{id}]) \cdot (\mathsf{id}) \overset{[\mathsf{id}]\,\{t\},[\mathsf{id}]\,\{u\}}{=} t \cdot u & \\
[\mathsf{id}]\,\{\mathsf{K}\} & : \mathsf{K}[\mathsf{id}] \equiv \mathsf{K} & \\
[\mathsf{id}]\,\{\mathsf{S}\} & : \mathsf{S}[\mathsf{id}] \equiv \mathsf{S} & \\
[\mathsf{id}]\,\{\mathsf{q}\} & : \mathsf{q}[\mathsf{id}] \equiv \mathsf{q}[\mathsf{wks}\,\mathsf{id}, \mathsf{q}] \equiv \mathsf{q} & \\
[\mathsf{id}]\,\{\mathsf{wk}\,t\} & : (\mathsf{wk}\,t)[\mathsf{id}] \equiv (\mathsf{wk}\,t)[\mathsf{wks}\,\mathsf{id}, \mathsf{q}] \equiv t[\mathsf{wks}\,\mathsf{id}] \overset{[\mathsf{wks}]}{=} \mathsf{wk}\,(t[\mathsf{id}]) \overset{[\mathsf{id}]\,\{t\}}{=} \mathsf{wk}\,t & \\
\multicolumn{3}{l}{\mathsf{idr} : \{\Gamma : \mathsf{Con}\} \to \sigma \circ^{\Gamma} \mathsf{id} = \sigma} \\
\mathsf{idr}\,\{\diamond\} & : \mathsf{tt} \circ \mathsf{id} \equiv \mathsf{tt} & \\
\mathsf{idr}\,\{\Gamma \rhd A\} & : (\sigma, t) \circ^{\Gamma \rhd A} \mathsf{id} \equiv (\sigma \circ^{\Gamma} \mathsf{id}, t[\mathsf{id}]) \overset{\mathsf{idr}\,\{\Gamma\},[\mathsf{id}]}{=} (\sigma, t) & \\
\multicolumn{3}{l}{{\Rightarrow}\beta : \{t : \mathsf{Tm}\,(\Gamma \rhd A)\,B\} \to \mathsf{lam}\,t \cdot v = t[\mathsf{id}, v]} \\
{\Rightarrow}\beta\,\{t \cdot u\} & : \mathsf{lam}\,(t \cdot u) \cdot v & \equiv \\
 & \mathsf{S} \cdot \mathsf{lam}\,t \cdot \mathsf{lam}\,u \cdot v & =(\mathsf{S}\beta) \\
 & \mathsf{lam}\,t \cdot v \cdot (\mathsf{lam}\,u \cdot v) & =({\Rightarrow}\beta\,\{t\}, {\Rightarrow}\beta\,\{u\})
\end{array}$$

$$
\begin{array}{lll}
 & (t[\mathsf{id}, v]) \cdot (u[\mathsf{id}, v]) & \equiv \\
 & (t \cdot u)[\mathsf{id}, v] & \\
\Rightarrow\beta\,\{\mathsf{K}\} & : \mathsf{lam}\,\mathsf{K} \cdot v \equiv \mathsf{K} \cdot \mathsf{K} \cdot v \overset{\mathsf{K}\beta}{=} \mathsf{K} \equiv \mathsf{K}[\mathsf{id}, v] & \\
\Rightarrow\beta\,\{\mathsf{S}\} & : \mathsf{lam}\,\mathsf{S} \cdot v \equiv \mathsf{K} \cdot \mathsf{S} \cdot v \overset{\mathsf{K}\beta}{=} \mathsf{S} \equiv \mathsf{S}[\mathsf{id}, v] & \\
\Rightarrow\beta\,\{\mathsf{q}\} & : \mathsf{lam}\,\mathsf{q} \cdot v \equiv \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K} \cdot v \overset{\mathsf{S}\beta}{=} \mathsf{K} \cdot v \cdot (\mathsf{K} \cdot v) \overset{\mathsf{K}\beta}{=} v \equiv \mathsf{q}[\mathsf{id}, v] & \\
\Rightarrow\beta\,\{\mathsf{wk}\,t\} & : \mathsf{lam}\,(\mathsf{wk}\,t) \cdot v \equiv \mathsf{K} \cdot t \cdot v \overset{\mathsf{K}\beta}{=} t \overset{[\mathsf{id}]}{=} t[\mathsf{id}] \equiv (\mathsf{wk}\,t)[\mathsf{id}, v] & \\
\multicolumn{3}{l}{\Rightarrow\eta : t = \mathsf{lam}\,(t[\mathsf{p}] \cdot \mathsf{q})} \\
\Rightarrow\eta & : t & =(\eta'') \\
 & \mathsf{S} \cdot (\mathsf{K} \cdot t) \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}) & \equiv \\
 & \mathsf{S} \cdot \mathsf{lam}\,(\mathsf{wk}\,t) \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}) & =([\mathsf{id}]) \\
 & \mathsf{S} \cdot \mathsf{lam}\,(\mathsf{wk}\,(t[\mathsf{id}])) \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}) & =([\mathsf{wks}]) \\
 & \mathsf{S} \cdot \mathsf{lam}\,(t[\mathsf{wks}\,\mathsf{id}]) \cdot (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K}) & \equiv \\
 & \mathsf{lam}\,(t[\mathsf{p}] \cdot \mathsf{q}) & \\
\multicolumn{3}{l}{\mathsf{lam}[] : (\mathsf{lam}\,t)[\sigma] = \mathsf{lam}\,(t[\sigma \circ \mathsf{p}, \mathsf{q}])} \\
\mathsf{lam}[]\,\{t \cdot u\} & : (\mathsf{lam}\,(t \cdot u))[\sigma] & \equiv \\
 & \mathsf{S} \cdot ((\mathsf{lam}\,t)[\sigma]) \cdot ((\mathsf{lam}\,u)[\sigma]) & =(\mathsf{lam}[]\,\{t\}, \mathsf{lam}[]\,\{u\}) \\
 & \mathsf{S} \cdot \mathsf{lam}\,(t[\sigma \circ \mathsf{p}, \mathsf{q}]) \cdot \mathsf{lam}\,(u[\sigma \circ \mathsf{p}, \mathsf{q}]) & \equiv \\
 & \mathsf{lam}\,((t[\sigma \circ \mathsf{p}, \mathsf{q}]) \cdot (u[\sigma \circ \mathsf{p}, \mathsf{q}])) & \equiv \\
 & \mathsf{lam}\,((t \cdot u)[\sigma \circ \mathsf{p}, \mathsf{q}]) & \\
\mathsf{lam}[]\,\{\mathsf{K}\} & : (\mathsf{lam}\,\mathsf{K})[\sigma] \equiv (\mathsf{K} \cdot \mathsf{K})[\sigma] \equiv \mathsf{K} \cdot \mathsf{K} \equiv \mathsf{lam}\,\mathsf{K} \equiv \mathsf{lam}\,(\mathsf{K}[\sigma \circ \mathsf{p}, \mathsf{q}]) & \\
\mathsf{lam}[]\,\{\mathsf{S}\} & : (\mathsf{lam}\,\mathsf{S})[\sigma] \equiv (\mathsf{K} \cdot \mathsf{S})[\sigma] \equiv \mathsf{K} \cdot \mathsf{S} \equiv \mathsf{lam}\,\mathsf{S} \equiv \mathsf{lam}\,(\mathsf{S}[\sigma \circ \mathsf{p}, \mathsf{q}]) & \\
\mathsf{lam}[]\,\{\mathsf{q}\} & : (\mathsf{lam}\,\mathsf{q})[\sigma] \equiv (\mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K})[\sigma] \equiv \mathsf{S} \cdot \mathsf{K} \cdot \mathsf{K} \equiv \mathsf{lam}\,\mathsf{q} \equiv \mathsf{lam}\,(\mathsf{q}[\sigma \circ \mathsf{p}, \mathsf{q}]) & \\
\mathsf{lam}[]\,\{\mathsf{wk}\,t\} & : (\mathsf{lam}\,(\mathsf{wk}\,t))[\sigma] & \equiv \\
 & (\mathsf{K} \cdot t)[\sigma] & \equiv \\
 & \mathsf{K} \cdot (t[\sigma]) & \equiv \\
 & \mathsf{lam}\,(\mathsf{wk}\,(t[\sigma])) & =([\mathsf{id}]) \\
 & \mathsf{lam}\,(\mathsf{wk}\,(t[\sigma][\mathsf{id}])) & =([\mathsf{wks}]) \\
 & \mathsf{lam}\,(t[\sigma][\mathsf{wks}\,\mathsf{id}]) & =([\circ]) \\
 & \mathsf{lam}\,(t[\sigma \circ \mathsf{wks}\,\mathsf{id}]) & \equiv \\
 & \mathsf{lam}\,(t[\sigma \circ \mathsf{p}]) & \equiv \\
 & \mathsf{lam}\,(\mathsf{wk}\,t[\sigma \circ \mathsf{p}, \mathsf{q}]) &
\end{array}
$$