C-system of a module over a monad on sets[1]

Vladimir Voevodsky[2,3]

August 2014

Abstract

This is the second paper in a series started in [15] which aims to provide mathematical descriptions of objects and constructions related to the first few steps of the semantical theory of dependent type systems.

We construct for any pair (R, LM), where R is a monad on sets and LM is a left module over R, a C-system ("contextual category") $CC(R, LM)$ and describe, using the results of [15] a class of sub-quotients of $CC(R, LM)$ in terms of objects directly constructed from R and LM. In the special case of the monads of expressions associated with nominal signatures this construction gives the C-systems of general dependent type theories when they are specified by collections of judgements of the four standard kinds.

1 Introduction

The first few steps in all approaches to the semantics of dependent type theories remain insufficiently understood. The constructions which have been worked out in detail in the case of a few particular type systems by dedicated authors are being extended to the wide variety of type systems under consideration today by analogy. This is not acceptable in mathematics. Instead we should be able to obtain the required results for new type systems by *specialization* of general theorems formulated and proved for abstract objects the instances of which combine together to produce a given type system.

One such class of objects is the class of C-systems introduced in [3] (see also [4]) under the name "contextual categories". A modified axiomatics of C-systems and the construction of new C-systems as sub-objects and regular quotients of the existing ones in a way convenient for use in type-theoretic applications are considered in [15].

Modules over monads were introduced in [6] in the context of syntax with binding and substitution.

In the present paper, after some general comments about monads on $Sets$ and their modules, we construct for any such monad R and a left module LM over R a C-system (contextual category) $CC(R, LM)$. We describe, using the results of [15], all the C-subsystems of $CC(R, LM)$ in terms of objects directly associated with R and LM.

We then define two additional operations σ and $\widetilde{\sigma}$ on $CC(R, LM)$ and describe the regular congruence relations (see [15]) on C-subsystems of $CC(R, LM)$ which are compatible in a certain sense with σ and $\widetilde{\sigma}$.

Of a particular interest is the case of "syntactic" pairs where $R(\{x_1, \ldots, x_n\})$ and $LM(\{x_1, \ldots, x_n\})$ are the sets of expressions of some kind with free variables from $\{x_1, \ldots, x_n\}$ modulo an equivalence relation such as α-equivalence.

[1] *2000 Mathematical Subject Classification: 18D99, 08C99, 03B15*

[2] School of Mathematics, Institute for Advanced Study, Princeton NJ, USA. e-mail: vladimir@ias.edu

[3] Work on this paper was supported by NSF grant 1100938.

The simplest class of syntactic pairs where $LM = R$ arises from signatures considered in [6, p.228]. To any such signature Σ one associates a class of expressions with bindings and $R(\{x_1, \ldots, x_n\})$ is the set of such expressions with free variables from the set $\{x_1, \ldots, x_n\}$ modulo the α-equivalence.

Suppose now that we are given a type theory based on the syntax of expressions specified by Σ that is formulated in terms of the four kinds of basic judgements originally introduced by Per Martin-Lof in [11, p.161].

Since we are only interested in the α-equivalence classes of judgements we may assume that the variables declared in the context are taken from the set of natural numbers such that the first declared variable is 1, the second is 2 etc. Then, the set of judgements of the form

$$(1 : A_1, \ldots, n : A_n \vdash A\, type)$$

(in the notation of Martin-Lof "$A\, type\, (1 \in A_1, \ldots, n \in A_n)$") can be identified with the set of judgements of the form

$$(1 : A_1, \ldots, n : A_n, n+1 : A \rhd)$$

stating that the context $(1 : A_1, \ldots, n : A_n, n+1 : A)$ is well-formed.

With this identification the type theory is specified by four sets $C, \widetilde{C}, Ceq$ and $\widetilde{Ceq}$ where

$$C \subset \coprod_{n \geq 0} LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n-1\})$$

$$\widetilde{C} \subset \coprod_{n \geq 0} LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n-1\}) \times R(\{1, \ldots, n\}) \times LM(\{1, \ldots, n\})$$

$$Ceq \subset \coprod_{n \geq 0} LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n-1\}) \times LM(\{1, \ldots, n\})^2$$

$$\widetilde{Ceq} \subset \coprod_{n \geq 0} LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n-1\}) \times R(\{1, \ldots, n\})^2 \times LM(\{1, \ldots, n\})$$

On the other hand we show that any pair $(CC, \sim)$, where CC is a sub-C-system of $CC(R, LM)$ and $\sim$ is a regular congruence relation on CC, defines four subsets of such form. Proposition 6.2 spells out the necessary and sufficient conditions that the sets $C, \widetilde{C}, Ceq, \widetilde{Ceq}$ should satisfy in order to correspond to a pair $(CC, \sim)$.

A wider class of syntactic pairs (R, LM) that arises from nominal signatures is considered in Section 7.

This is one the papers extending the material which I started to work on in [14]. I would like to thank the Institute Henri Poincare in Paris and the organizers of the "Proofs" trimester for their hospitality during the preparation of this paper. The work on this paper was facilitated by discussions with Richard Garner and Egbert Rijke.

Notation: For morphisms $f : X \to Y$ and $g : Y \to Z$ we denote their composition as $f \circ g$. For functors $F : \mathcal{C} \to \mathcal{C}'$, $G : \mathcal{C}' \to \mathcal{C}''$ we use the standard notation $G \circ F$ for their composition.

2 Left modules over monads

Recall (cf. [6]) that a monad on a category $\mathcal{C}$ is a functor $M : \mathcal{C} \to \mathcal{C}$ together with two families of morphisms:

1. for any $X \in \mathcal{C}$, a morphism $\eta_X : X \to R(X)$,
2. for any $X \in \mathcal{C}$, a morphism $\mu_X : R(R(X)) \to R(X)$

which satisfy certain conditions. For objects X, X' and a morphism $f : X' \to R(X)$, the composition $R(X') \overset{R(f)}{\to} R(R(X)) \overset{\mu_X}{\to} R(X)$ is a morphism $bind(f) : R(X') \to R(X)$. This allows one to describe monads as follows:

Lemma 2.1 *The construction outline above defines an equivalence between (the type of) monads on $\mathcal{C}$ and (the type of) collections of data of the form:*

1. *for every object X an object $R(X)$,*
2. *for every object X a morphism $\eta_X : X \to R(X)$,*
3. *for every two objects X, X' and a morphism $f : X \to R(X')$, a morphism $bind(f) : R(X) \to R(X')$*

which satisfy the following conditions:

1. *for an object X, $bind(\eta_X) = id_{R(X)}$,*
2. *for a morphism $f : X \to X'$, $\eta_X \circ bind(f) = f$,*
3. *for two morphisms $f : X \to R(X')$, $g : X' \to R(X'')$, $bind(f \circ bind(g)) = bind(f) \circ bind(g)$.*

Proof: Straightforward. Cf. [12], [8, Prop. 1].

Lemma 2.2 *Let R be a monad on the product category $\mathcal{C} \times \mathcal{D}$. Let $A \in \mathcal{D}$. Then the functor $R_{A,1} : X \mapsto pr_{\mathcal{C}}(R(X, A))$ has a natural structure of a monad on $\mathcal{C}$.*

Proof: One defines the morphisms $\eta_X : X \to R_{A,1}(X)$ by

$$\eta_X := pr_{\mathcal{C}}(\eta_{(X,A)})$$

and morphisms $bind(f) : R_{A,1}(X) \to R_{A,1}(X')$ for $f : X \to R_{A,1}(X')$ by

$$bind(f) := pr_{\mathcal{C}}(bind(f, pr_{\mathcal{D}}(\eta_{(X,A)})))$$

The verification of the conditions of Lemma 2.1 is straightforward.

The concept of a module over a monad was first explicitly introduced in [6].

Definition 2.3 *Let R be a monad on a category $\mathcal{C}$. A left module over R with values in a category $\mathcal{D}$ is a functor $LM : \mathcal{C} \to \mathcal{D}$ together with, for all $X, X' \in \mathcal{C}$ and $f : X \to R(X')$, a morphism $\rho(f) : LM(X) \to LM(X')$ such that*

1. *$\rho(\eta_X) = Id_{LM(X)}$,*

2. for $f : X \to R(X')$, $g : X' \to R(X'')$, $\rho(f)\rho(g) = \rho(f\, bind(g))$.

One verifies easily (cf. [8, Def. 9]) that a left R-module structure on LM is the same as a natural transformation $LM \circ R \to LM$ which satisfies the expected compatibility conditions with respect to $Id \to R$ and $R \circ R \to R$.

Lemma 2.4 *Let R be a monad on a category $\mathcal{C}$. Then one has:*

1. *If LM_1, LM_2 are left R-modules with values in $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively then the functor $X \mapsto (LM_1(X), LM_2(X))$ has a natural structure of a left R-module with values in $\mathcal{D}_1 \times \mathcal{D}_2$.*
2. *If LM is a left R-module with values in $\mathcal{D}$ and $F : \mathcal{D} \to \mathcal{D}'$ is a functor then $F \circ LM$ has a natural structure of a left R-module with values in $\mathcal{D}'$.*

Proof: Straightforward.

Lemma 2.5 *Under the assumptions and in the notation of Lemma 2.2 the morphisms*

$$\rho(f : X \to R_{A,1}(X')) = bind(f, pr_{\mathcal{D}}(\eta_{(X',A)})) : R(X, A) \to R(X', A)$$

define a structure of a left $R_{A,1}$-module with values in $\mathcal{C} \times \mathcal{D}$ on the functor

$$M_{A,1} : X \mapsto R(X, A)$$

Proof: Direct verification of the conditions of Definition 2.3.

In the case of a monad R on $Sets$ and a left R-module LM with values in $Sets$, for $E \in LM(\{x_1, \ldots, x_n\})$ or $E \in R(\{1, \ldots, m\})$ and $f : \{x_1, \ldots, x_n\} \to R(X')$ such that $f(x_i) = f_i$ we write $\rho(f)(E)$ as $E(f_1/x_1, \ldots, f_n/x_n)$.

For $E \in LM(\{1, \ldots, m\})$ and $n \geq 1$ we set:

$$t_n(E) = E[n + 1/n, n + 2/n + 1, \ldots, m + 1/m]$$

$$s_n(E) = E[n/n + 1, n + 1/n + 2, \ldots, m - 1/m]$$

If we were numbering elements of a set with n elements from 0 then we would have $t_n = LM(\partial_{n-1})$ and $s_n = LM(\sigma_{n-1})$ where ∂_i and σ_i are the usual generators of the simplicial category.

For a monad R on $Sets$ we let $R - cor$ ("R-correspondences") to be the full subcategory of the Kleisli category of R whose objects are finite sets. Recall, that the set of morphisms from X to Y in $R - cor$ is the set of maps from X to $R(Y)$ i.e. $R(Y)^X$ (in other words, $R - cor$ is the category of free, finitely generated R-algebras).

We further let $C(R)$ denote the pre-category[4] with

$$Ob(C(R)) = \mathbf{N}$$

$$Mor(C(R)) = \coprod_{m,n \in \mathbf{N}} R(\{1, \ldots, m\})^n$$

which is equivalent, as a category, to $(R - cor)^{op}$.

[4]See the introduction to [15].

Remark 2.6 A finitary monad (on sets) is a monad $R : Sets \rightarrow Sets$ that, as a functor, commutes with filtering colimits. Since any set is, canonically, the colimit of the filtering diagram of its finite subsets, a functor $Sets \rightarrow Sets$ that commutes with filtering colimits can be equivalently described as a functor $FSets \rightarrow Sets$ where $FSets$ is the category of finite sets. Furthermore, Lemma 2.1 can be used to show that finitary monads on $Sets$ can be defined as collections of data of the form:

1. for every finite set X a set $R(X)$,
2. for every finite set X a function $\eta_X : X \rightarrow R(X)$,
3. for every finite sets X, X' and a function $f : X \rightarrow R(X')$, a function $bind(f) : R(X) \rightarrow R(X')$

which satisfy the conditions:

1. for a finite set X, $bind(\eta_X) = id_{R(X)}$,
2. for a function of finite sets $f : X \rightarrow X'$, $\eta_X \circ bind(f) = f$,
3. for two functions $f : X \rightarrow R(X')$, $g : X' \rightarrow R(X'')$, $bind(f \circ bind(g)) = bind(f) \circ bind(g)$.

This description shows that for any monad R the restriction of R to a functor $R^{fin} : FSets \rightarrow Sets$ is a finitary monad.

Similar observations apply to left R-modules. The constructions of this paper, while done for a general pair (R, LM), only depend on the corresponding finitary pair (R^{fin}, LM^{fin}).

Remark 2.7 The correspondence $R \mapsto C(R)$ defines an equivalence between the type of the finitary monads on $Sets$ and the type of the pre-category structures on $\mathbf{N}$ that extend the pre-category structure of finite sets and where the addition remains to be the coproduct.

Remark 2.8 A finitary sub-monad of R is the same as a sub-pre-category in $C(R)$ which contains all objects. Intersection of two sub-monads is a sub-monad which allows one to speak of sub-monads generated by a set of elements.

3 The C-system $CC(R, LM)$.

Let R be a monad on $Sets$ and LM a left module over R with values in $Sets$. Let $CC(R, LM)$ be the pre-category whose set of objects is $Ob(CC(R, LM)) = \amalg_{n\geq 0} Ob_n$ where

$$Ob_n = LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n-1\})$$

and the set of morphisms is

$$Mor(CC(R, LM)) = \coprod_{m,n\geq 0} Ob_m \times Ob_n \times R(\{1, \ldots, m\})^n$$

with the obvious domain and codomain maps. The composition of morphisms is defined in the same way as in $C(R)$ such that the mapping $Ob(CC(R, LM)) \rightarrow \mathbf{N}$ which sends all elements of Ob_n to n, is a functor from $CC(R, LM)$ to $C(R)$. The associativity of compositions follows immediately from the associativity of compositions in $R - cor$.

Note that if $LM(\emptyset) = \emptyset$ then $CC(R, LM) = \emptyset$ and otherwise the functor $CC(R, LM) \to C(R)$ is an equivalence, so that in the second case $C(R)$ and $CC(R, LM)$ are indistinguishable as categories. However, as pre-categories they are quite different unless $LM = (X \mapsto pt)$ in which case the functor $CC(R, LM) \to C(R)$ is an isomorphism.

The pre-category $CC(R, LM)$ is given the structure of a C-system as follows. The final object is the only element of Ob_0, the map ft is defined by the rule

$$ft(T_1, \ldots, T_n) = (T_1, \ldots, T_{n-1}).$$

The canonical pull-back square defined by an object $(T_1, \ldots, T_{n+1})$ and a morphism

$$(f_1, \ldots, f_n) : (R_1, \ldots, R_m) \to (T_1, \ldots, T_n)$$

is of the form:

$$\begin{tikzcd}
(R_1, \ldots, R_m, T_{n+1}(f_1/1, \ldots, f_n/n)) \arrow[r, "(f_1,\ldots,f_n,m+1)"] \arrow[d, "(1,\ldots,m)"'] & (T_1, \ldots, T_{n+1}) \arrow[d, "(1,\ldots,n)"] \\
(R_1, \ldots, R_m) \arrow[r, "(f_1,\ldots,f_n)"] & (T_1, \ldots, T_n)
\end{tikzcd} \qquad (1)$$

Proposition 3.1 *With the structure defined above $CC(R, LM)$ is a C-system.*

Proof: Straightforward.

Remark 3.2 There is another construction of a pre-category from (R, LM) which takes as an additional parameter a set Var which is called the set of variables. Let $F_n(Var)$ be the set of sequences of length n of pair-wise distinct elements of Var. Define the pre-category $CC(R, LM, Var)$ as follows. The set of objects of $CC(R, LM, Var)$ is

$$Ob(CC(R, LM, Var)) = \amalg_{n \geq 0} \amalg_{(x_1,\ldots,x_n) \in F_n(Var)} LM(\emptyset) \times \ldots \times LM(\{x_1, \ldots, x_{n-1}\})$$

For compatibility with the traditional type theory we will write the elements of $Ob(CC(R, LM, X))$ as sequences of the form $x_1 : E_1, \ldots, x_n : E_n$. The set of morphisms is given by

$$Hom_{CC(R,LM,Var)}((x_1 : E_1, \ldots, x_m : E_m), (y_1 : T_1, \ldots, y_n : T_n)) = R(\{x_1, \ldots, x_m\})^n$$

The composition is defined in such a way that the projection

$$(x_1 : E_1, \ldots, x_n : E_n) \mapsto (E_1, E_2(1/x_1), \ldots, E_n(1/x_1, \ldots, n-1/x_{n-1}))$$

is a functor from $CC(R, LM, Var)$ to $CC(R, LM)$.

This functor is clearly an equivalence of categories but not an isomorphism of pre-categories.

There are an obvious final object and the map ft on $CC(R, LM, Var)$.

There is however a real problem in making it into a C-system which is due to the following. Consider an object $(y_1 : T_1, \ldots, y_{n+1} : T_{n+1})$ and a morphism $(f_1, \ldots, f_n) : (x_1 : R_1, \ldots, x_m : R_m) \to (y_1 :$

$T_1, \ldots, y_n : T_n)$. In order for the functor to $CC(R, LM)$ to be a C-system morphism the canonical square build on this pair should have the form

$$\begin{tikzcd}
(x_1 : R_1, \ldots, x_m : R_m, x_{m+1} : T_{n+1}(f_1/1, \ldots, f_n/n)) \arrow[r, "(f_1,\ldots,f_n,x_{n+1})"] \arrow[d] & (y_1 : T_1, \ldots, y_{n+1} : T_{n+1}) \arrow[d] \\
(x_1 : R_1, \ldots, x_m : R_m) \arrow[r, "(f_1,\ldots,f_n)"] & (y_1 : T_1, \ldots, y_n : T_n)
\end{tikzcd}$$

where x_{m+1} is an element of Var which is distinct from each of the elements $x_1, \ldots, x_m$. Moreover, we should choose x_{m+1} in such a way the the resulting construction satisfies the C-system axioms for $(f_1, \ldots, f_n) = Id$ and for the compositions $(g_1, \ldots, g_m) \circ (f_1, \ldots, f_n)$. One can easily see that no such choice is possible for a finite set Var. At the moment it is not clear to me whether or not it is possible for an infinite Var.

Recall from [15] that for a C-system CC one defines $\widetilde{Ob}(CC)$ as the subset of $Mor(CC)$ which consists of morphisms s of the form $ft(X) \to X$ such that $l(X) > 0$ and $s \circ p_X = Id_{ft(X)}$.

Lemma 3.3 *One has:*

$$\widetilde{Ob}(CC(R, LM)) \cong \coprod_{n \geq 0} LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n\}) \times R(\{1, \ldots, n\})$$

Proof: An element of $\widetilde{Ob}(CC(R, LM))$ is a section s of the canonical morphism $p_\Gamma : \Gamma \to ft(\Gamma)$. It follows immediately from the definition of $CC(R, LM)$ that for $\Gamma = (E_1, \ldots, E_{n+1})$, a morphism $(f_1, \ldots, f_{n+1}) \in R(\{1, \ldots, n\})^{n+1}$ from $ft(\Gamma)$ to Γ is a section of p_Γ if an only if $f_i = i$ for $i = 1, \ldots, n$. Therefore, any such section is determined by its last component f_{n+1} and mapping $((E_1, \ldots, E_n), (E_1, \ldots, E_{n+1}), (f_1, \ldots, f_{n+1}))$ to $(E_1, \ldots, E_n, E_{n+1}, f_{n+1})$ we get a bijection

$$\widetilde{Ob}(CC(R, LM)) \cong \coprod_{n \geq 0} LM(\emptyset) \times \ldots \times LM(\{1, \ldots, n\}) \times R(\{1, \ldots, n\}) \quad (2)$$

Using the notations of type theory we can write elements of $Ob(CC(R, LM))$ as

$$\Gamma = (T_1, \ldots, T_n \rhd)$$

where $T_i \in LM(\{1, \ldots, i-1\})$ and the elements of $\widetilde{Ob}(CC(R, LM))$ as

$$\mathcal{J} = (T_1, \ldots, T_n \vdash t : T)$$

where $T_i \in LM(\{1, \ldots, i-1\})$, $T \in LM(\{1, \ldots, n\})$ and $t \in R(\{1, \ldots, n\})$.

In this notation the operations $T, \widetilde{T}, S, \widetilde{S}$ and δ which were introduced in [15] take the form:

1. $T((\Gamma, T_{n+1} \rhd), (\Gamma, \Delta \rhd)) = (\Gamma, T_{n+1}, t_{n+1}(\Delta) \rhd)$ when $l(\Gamma) = n$,
2. $\widetilde{T}((\Gamma, T_{n+1} \rhd), (\Gamma, \Delta \vdash r : R)) = (\Gamma, T_{n+1}, t_{n+1}(\Delta) \vdash t_{n+1}(r : R))$ when $l(\Gamma) = n$,
3. $S((\Gamma \vdash s : S), (\Gamma, S, \Delta \rhd)) = (\Gamma, s_{n+1}(\Delta[s/n+1]) \rhd)$ when $l(\Gamma) = n$,

4. $\widetilde{S}((\Gamma \vdash s : S), (\Gamma, S, \Delta \vdash r : R)) = (\Gamma, s_{n+1}(\Delta[s/n+1]) \vdash s_{n+1}((r : R)[s/n+1])$ when $l(\Gamma) = n$,
5. $\delta(\Gamma, T\rhd) = (\Gamma, T \vdash (n+1) : T)$ when $l(\Gamma) = n$.

Remark 3.4 One can easily construct on the function $(R, LM) \mapsto CC(R, LM)$ the structure of a functor from the "large module category" of [7] to the category of C-systems and their homomorphisms.

4 C-subsystems of $CC(R, LM)$.

Let CC be a C-subsystem of $CC(R, LM)$. By [15] CC is determined by the subsets $C = Ob(CC)$ and $\widetilde{C} = \widetilde{Ob}(CC)$ in $Ob(CC(R, LM))$ and $\widetilde{Ob}(CC(R, LM))$.

For $\Gamma = (E_1, \dots, E_n)$ we write $(\Gamma \rhd_C)$ if $(E_1, \dots, E_n)$ is in C and $(\Gamma \vdash_{\widetilde{C}} t : T)$ if $(E_1, \dots, E_n, T, t)$ is in $\widetilde{C}$.

The following result is an immediate corollary of [15, Proposition 4.3] together with the description of the operations $T, \widetilde{T}, S, \widetilde{S}$ and δ for $CC(R, LM)$ which is given above.

Proposition 4.1 *Let (R, LM) be a monad on Sets and a left module over it with values in Sets. A pair of subsets*

$$C \subset \coprod_{n \geq 0} \prod_{i=0}^{n-1} LM(\{1, \dots, i\})$$

$$\widetilde{C} \subset \coprod_{n \geq 0} (\prod_{i=0}^{n} LM(\{1, \dots, i\})) \times R(\{1, \dots, n\})$$

corresponds to a C-subsystem CC of $CC(R, LM)$ if and only if the following conditions hold:

1. $(\rhd_C)$
2. $(\Gamma, T \rhd_C) \Rightarrow (\Gamma \rhd_C)$
3. $(\Gamma \vdash_{\widetilde{C}} r : R) \Rightarrow (\Gamma, R \rhd_C)$
4. $(\Gamma, T \rhd_C) \wedge (\Gamma, \Delta, \vdash_{\widetilde{C}} r : R) \Rightarrow (\Gamma, T, t_{n+1}(\Delta) \vdash_{\widetilde{C}} t_{n+1}(r : R))$ *where* $n = l(\Gamma_1)$
5. $(\Gamma \vdash_{\widetilde{C}} s : S) \wedge (\Gamma, S, \Delta \vdash_{\widetilde{C}} r : R) \Rightarrow (\Gamma, s_{n+1}(\Delta[s/n+1]) \vdash_{\widetilde{C}} s_{n+1}((r : R)[s/n+1]))$ *where* $n = l(\Gamma_1)$,
6. $(\Gamma, T \rhd_C) \Rightarrow (\Gamma, T \vdash_{\widetilde{C}} n+1 : T)$ *where* $n = l(\Gamma)$.

Note that conditions (4) and (5) together with condition (6) and condition (3) imply the following

4a $(\Gamma, T \rhd_C) \wedge (\Gamma, \Delta \rhd_C) \Rightarrow (\Gamma, T, t_{n+1}(\Delta) \rhd_C)$ where $n = l(\Gamma_1)$,

5a $(\Gamma \vdash_{\widetilde{C}} s : S) \wedge (\Gamma, S, \Delta \rhd_C) \Rightarrow (\Gamma, s_{n+1}(\Delta[s/n+1]) \rhd_C)$ where $n = l(\Gamma_1)$.

Note also that modulo condition (2), condition (1) is equivalent to the condition that $C \neq \emptyset$.

Remark 4.2 If one re-writes the conditions of Proposition 4.1 in the more familiar in type theory form where the variables introduced in the context are named rather than directly numbered one arrives at the following rules:

$$\frac{}{\rhd_C} \qquad \frac{x_1 : T_1, \ldots, x_n : T_n \rhd_C}{x_1 : T_1, \ldots, x_{n-1} : T_{n-1} \rhd_C} \qquad \frac{x_1 : T_1, \ldots, x_n : T_n \vdash_{\widetilde{C}} t : T}{x_1 : T_1, \ldots, x_n : T_n, y : T \rhd_C}$$

$$\frac{x_1 : T_1, \ldots, x_n : T_n, y : T \rhd_C \qquad x_1 : T_1, \ldots, x_n : T_n, \ldots, x_m : T_m \vdash_{\widetilde{C}} r : R}{x_1 : T_1, \ldots, x_n : T_n, y : T, x_{n+1} : T_{n+1}, \ldots, x_m : T_m \vdash_{\widetilde{C}} r : R}$$

$$\frac{x_1 : T_1, \ldots, x_n : T_n \vdash_{\widetilde{C}} s : S \qquad x_1 : T_1, \ldots, x_n : T_n, y : S, x_{n+1} : T_{n+1}, \ldots, x_m : T_m \vdash_{\widetilde{C}} r : R}{x_1 : T_1, \ldots, x_n : T_n, x_{n+1} : T_{n+1}[s/y], \ldots, x_m : T_m[s/y] \vdash_{\widetilde{C}} (r : R)[s/y]}$$

$$\frac{x_1 : E_1, \ldots, x_n : E_n \rhd_C}{x_1 : E_1, \ldots, x_n : E_n \vdash_{\widetilde{C}} x_n : E_n}$$

which are similar (and probably equivalent) to the "basic rules of DTT" given in [9, p.585]. The advantage of the rules given here is that they are precisely the ones which are necessary and sufficient for a given collection of contexts and judgements to define a C-system.

Lemma 4.3 *Let CC be as above and let $(E_1, \ldots, E_m), (T_1, \ldots, T_n) \in Ob(CC)$ and $(f_1, \ldots, f_n) \in R(\{1, \ldots, m\})^n$. Then*

$$(f_1, \ldots, f_n) \in Hom_{CC}((E_1, \ldots, E_m), (T_1, \ldots, T_n))$$

if and only if $(f_1, \ldots, f_{n-1}) \in Hom_{CC}((E_1, \ldots, E_m), (T_1, \ldots, T_{n-1}))$ and

$$E_1, \ldots, E_m \vdash_{\widetilde{C}} f_n : T_n(f_1/1, \ldots, f_{n-1}/n-1)$$

Proof: Straightforward using the fact that the canonical pull-back squares in $CC(R, LM)$ are given by (1).

Example 4.4 The category $CC(R, R)$ for the identity monad is empty. For the monad of the form $R(X) = pt$ the C-system $CC(R, R)$ has only two subsystems - itself and the trivial one for which $C = pt$.

The first non-trivial example is the monad $R(X) = X \amalg \{*\}$. We conjecture that in this case the set of all subsystems of $CC(R, R)$ is *uncountable*.

One can probably show this as follows. Let $\epsilon : \mathbf{N} \to \{0, 1\}$, be a sequence of 0's and 1's. Consider the C-subsystem of CC_ϵ of $CC(R, R)$ which is generated by the set of elements of the form $(*, 1, 2, \ldots, n \rhd) \in Ob(CC(R, R))$ for all $n \geq 0$ and elements $(*, 1, \ldots, n+1 \vdash n+2 : *) \in \widetilde{Ob}(CC(R, R))$ for n such that $\epsilon(n) = 1$.

It should be possible to show that $CC_\epsilon \neq CC_{\epsilon'}$ for $\epsilon \neq \epsilon'$ which would imply the conjecture.

5 Operations σ and $\widetilde{\sigma}$ on $CC(R, LM)$.

C-systems of the form $CC(R, LM)$ have an important additional structure which will play a role in the next section. This structure is given by two operations:

1. for $\Gamma = (T_1, \ldots, T_n, \ldots, T_{n+i})$ and $\Gamma' = (T'_1, \ldots, T'_n)$ we set

$$\sigma(\Gamma, \Gamma') = (T'_1, \ldots, T'_n, T_{n+1}, \ldots, T_{n+i})$$

   This gives us an operation with values in Ob defined on the subset of $Ob \times Ob$ which consists of pairs (Γ, Γ') such that $l(\Gamma) > l(\Gamma')$,

2. for $\mathcal{J} = (T_1, \ldots, T_{n-1}, \ldots, T_{n-1+i} \vdash t : T_{n+i})$, $\Gamma' = (T'_1, \ldots, T'_n)$ we set

$$\widetilde{\sigma}(\mathcal{J}, \Gamma') = \begin{cases} (T'_1, \ldots, T'_n, T_{n+1}, \ldots, T_{n+i-1} : t : T_{n+i}) & \text{for } i > 0 \\ (T'_1, \ldots, T'_{n-1} \vdash t : T'_n) & \text{for } i = 0 \end{cases}$$

   This gives us an operation with values in $\widetilde{Ob}$ defined on the subset of $\widetilde{Ob} \times Ob$ which consists of pairs $(\mathcal{J}, \Gamma')$ such that $l(\partial(\mathcal{J})) \leq l(\Gamma')$.

6 Regular sub-quotients of $CC(R, LM)$.

Let (R, LM) be as above and

$$Ceq \subset \coprod_{n \geq 0} (\prod_{i=0}^{n-1} LM(\{1, \ldots, i\})) \times LM(\{1, \ldots, n\})^2$$

$$\widetilde{Ceq} \subset \coprod_{n \geq 0} (\prod_{i=0}^{n} LM(\{1, \ldots, i\})) \times R(\{1, \ldots, n\})^2$$

be two subsets.

For $\Gamma = (T_1, \ldots, T_n) \in ob(CC(R, LM))$ and $S_1, S_2 \in LM(\{1, \ldots, n\})$ we write $(\Gamma \vdash_{Ceq} S_1 = S_2)$ to signify that $(T_1, \ldots, T_n, S_1, S_2) \in Ceq$. Similarly for $T \in LM(\{1, \ldots, n\})$ and $o, o' \in R(\{1, \ldots, n\})$ we write $(\Gamma \vdash_{\widetilde{Ceq}} o = o' : S)$ to signify that $(T_1, \ldots, T_n, S, o, o') \in \widetilde{Ceq}$. When no confusion is possible we will omit the subscripts Ceq and $\widetilde{Ceq}$ at $\vdash$.

Similarly we will write $\rhd$ instead of $\rhd_C$ and $\vdash$ instead of $\vdash_{\widetilde{C}}$ if the subsets C and $\widetilde{C}$ are unambiguously determined by the context.

Definition 6.1 *Given subsets C, $\widetilde{C}$, Ceq, $\widetilde{Ceq}$ as above define relations $\sim$ on C and $\simeq$ on $\widetilde{C}$ as follows:*

1. *for $\Gamma = (T_1, \ldots, T_n)$, $\Gamma' = (T'_1, \ldots, T'_n)$ in C we set $\Gamma \sim \Gamma'$ iff $ft(\Gamma) \sim ft(\Gamma')$ and*

$$T_1, \ldots, T_{n-1} \vdash T_n = T'_n,$$

2. *for $(\Gamma \vdash o : S)$, $(\Gamma' \vdash o' : S')$ in $\widetilde{C}$ we set $(\Gamma \vdash o : S) \simeq (\Gamma' \vdash o' : S')$ iff $(\Gamma, S) \sim (\Gamma', S')$ and*

$$(\Gamma \vdash o = o' : S).$$

Proposition 6.2 *Let C, $\widetilde{C}$, Ceq, $\widetilde{Ceq}$ be as above and suppose in addition that one has:*

1. *C and $\widetilde{C}$ satisfy conditions (1)-(6) of Proposition 4.1 which are referred to below as conditions (1.1)-(1.6) of the present proposition,*

2. $$\begin{array}{ll} (a) & (\Gamma\vdash T=T')\Rightarrow(\Gamma,T\rhd) \\ (b) & (\Gamma,T\rhd)\Rightarrow(\Gamma\vdash T=T) \\ (c) & (\Gamma\vdash T=T')\Rightarrow(\Gamma\vdash T'=T) \\ (d) & (\Gamma\vdash T=T')\wedge(\Gamma\vdash T'=T'')\Rightarrow(\Gamma\vdash T=T'') \end{array}$$

3. $$\begin{array}{ll} (a) & (\Gamma\vdash o=o':T)\Rightarrow(\Gamma\vdash o:T) \\ (b) & (\Gamma\vdash o:T)\Rightarrow(\Gamma\vdash o=o:T) \\ (c) & (\Gamma\vdash o=o':T)\Rightarrow(\Gamma\vdash o'=o:T) \\ (d) & (\Gamma\vdash o=o':T)\wedge(\Gamma\vdash o'=o'':T)\Rightarrow(\Gamma\vdash o=o'':T) \end{array}$$

4. $$\begin{array}{ll} (a) & (\Gamma_1\vdash T=T')\wedge(\Gamma_1,T,\Gamma_2\vdash S=S')\Rightarrow(\Gamma_1,T',\Gamma_2\vdash S=S') \\ (b) & (\Gamma_1\vdash T=T')\wedge(\Gamma_1,T,\Gamma_2\vdash o=o':S)\Rightarrow(\Gamma_1,T',\Gamma_2'\vdash o=o':S) \\ (c) & (\Gamma\vdash S=S')\wedge(\Gamma\vdash o=o':S)\Rightarrow(\Gamma\vdash o=o':S') \end{array}$$

5. $$\begin{array}{lll} (a) & (\Gamma_1,T\rhd)\wedge(\Gamma_1,\Gamma_2\vdash S=S')\Rightarrow(\Gamma_1,T,t_{i+1}\Gamma_2\vdash t_{i+1}S=t_{i+1}S') & i=l(\Gamma) \\ (b) & (\Gamma_1,T\rhd)\wedge(\Gamma_1,\Gamma_2\vdash o=o':S)\Rightarrow(\Gamma_1,T,t_{i+1}\Gamma_2\vdash t_{i+1}o=t_{i+1}o':t_{i+1}S) & i=l(\Gamma) \end{array}$$

6. $$\begin{array}{ll} (a)\quad (\Gamma_1,T,\Gamma_2\vdash S=S')\wedge(\Gamma_1\vdash r:T)\Rightarrow & \\ (\Gamma_1,s_{i+1}(\Gamma_2[r/i+1])\vdash s_{i+1}(S[r/i+1])=s_{i+1}(S'[r/i+1])) & i=l(\Gamma_1) \\ (b)\quad (\Gamma_1,T,\Gamma_2\vdash o=o':S)\wedge(\Gamma_1\vdash r:T)\Rightarrow & \\ (\Gamma_1,s_{i+1}(\Gamma_2[r/i+1])\vdash s_{i+1}(o[r/i+1])=s_{i+1}(o'[r/i+1]):s_{i+1}(S[r/i+1])) & i=l(\Gamma_1) \end{array}$$

7. $$\begin{array}{ll} (a)\quad (\Gamma_1,T,\Gamma_2,S\rhd)\wedge(\Gamma_1\vdash r=r':T)\Rightarrow & \\ (\Gamma_1,s_{i+1}(\Gamma_2[r/i+1])\vdash s_{i+1}(S[r/i+1])=s_{i+1}(S[r'/i+1])) & i=l(\Gamma_1) \\ (b)\quad (\Gamma_1,T,\Gamma_2\vdash o:S)\wedge(\Gamma_1\vdash r=r':T)\Rightarrow & \\ (\Gamma_1,s_{i+1}(\Gamma_2[r/i+1])\vdash s_{i+1}(o[r/i+1])=s_{i+1}(o[r'/i+1]):s_{i+1}(S[r/i+1])) & i=l(\Gamma_1) \end{array}$$

Then the relations $\sim$ and $\simeq$ are equivalence relations on C and $\widetilde{C}$ which satisfy the conditions of [15, Proposition 5.4] and therefore they correspond to a regular congruence relation on the C-system defined by $(C,\widetilde{C})$.

Lemma 6.3 *One has:*

1. *If conditions (1.2), (4a) of the proposition hold then $(\Gamma\vdash S=S')\wedge(\Gamma\sim\Gamma')\Rightarrow(\Gamma'\vdash S=S')$.*

2. *If conditions (1.2), (1.3), (4a), (4b), (4c) hold then $(\Gamma\vdash o=o':S)\wedge((\Gamma,S)\sim(\Gamma',S'))\Rightarrow(\Gamma'\vdash o=o':S')$.*

Proof: By induction on $n = l(\Gamma) = l(\Gamma')$.

(1) For $n = 0$ the assertion is obvious. Therefore by induction we may assume that $(\Gamma \vdash S = S') \wedge (\Gamma \sim \Gamma') \Rightarrow (\Gamma' \vdash S = S')$ for all $i < n$ and all appropriate Γ,Γ', S and S' and that $(T_1, \ldots, T_n \vdash S = S') \wedge (T_1, \ldots, T_n \sim T'_1, \ldots, T'_n)$ holds and we need to show that $(T'_1, \ldots, T'_n \vdash S = S')$ holds. Let us show by induction on j that $(T'_1, \ldots, T'_j, T_{j+1}, \ldots, T_n \vdash S = S')$ for all $j = 0, \ldots, n$. For $j = 0$ it is a part of our assumptions. By induction we may assume that $(T'_1, \ldots, T'_j, T_{j+1}, \ldots, T_n \vdash S = S')$. By definition of $\sim$ we have $(T_1, \ldots, T_j \vdash T_{j+1} = T'_{j+1})$. By the inductive assumption we have $(T'_1, \ldots, T'_j \vdash T_{j+1} = T'_{j+1})$. Applying (4a) with $\Gamma_1 = (T'_1, \ldots T'_j)$, $T = T_{j+1}$, $T' = T'_{j+1}$ and $\Gamma_2 = (T_{j+2}, \ldots, T_n)$ we conclude that $(T'_1, \ldots, T'_{j+1}, T_{j+2}, \ldots, T_n \vdash S = S')$.

(2) By the first part of the lemma we have $\Gamma' \vdash S = S'$. Therefore by (4c) it is sufficient to show that $(\Gamma \vdash o = o' : S) \wedge (\Gamma \sim \Gamma') \Rightarrow (\Gamma' \vdash o = o' : S)$. The proof of this fact is similar to the proof of the first part of the lemma using (4b) instead of (4a).

Lemma 6.4 *One has:*

1. *Assume that conditions (1.2), (2b), (2c), (2d) and (4a) hold. Then $\sim$ is an equivalence relation.*
2. *Assume that conditions of the previous part of the lemma as well as conditions (1.3), (3b), (3c), (3d), (4b) and (4c) hold. Then $\simeq$ is an equivalence relation.*

Proof: By induction on $n = l(\Gamma) = l(\Gamma')$.

(1) Reflexivity follows directly from (1.2) and (2b). For $n = 0$ the symmetry is obvious. Let $(\Gamma, T) \sim (\Gamma', T')$. By induction we may assume that $\Gamma' \sim \Gamma$. By Lemma 6.3(a) we have $(\Gamma' \vdash T = T')$ and by (2c) we have $(\Gamma' \vdash T' = T)$. We conclude that $(\Gamma', T') \sim (\Gamma, T)$. The proof of transitivity is by a similar induction.

(2) Reflexivity follows directly from reflexivity of $\sim$, (1.3) and (3b). Symmetry and transitivity are also easy using Lemma 6.3.

From this point on we assume that all conditions of Proposition 6.2 hold. Let $C' = C/\sim$ and $\widetilde{C}' = \widetilde{C}/\simeq$. It follows immediately from our definitions that the functions $ft : C \to C$ and $\partial : \widetilde{C} \to C$ define functions $ft' : C' \to C'$ and $\partial' : \widetilde{C}' \to C'$.

Lemma 6.5 *The conditions (3) and (4) of [15, Proposition 5.4] hold for $\sim$ and $\simeq$.*

Proof: 1. We need to show that for $(\Gamma, T\rhd)$, and $\Gamma \sim \Gamma'$ there exists $(\Gamma', T'\rhd)$ such that $(\Gamma, T) \sim (\Gamma', T')$. It is sufficient to take $T = T'$. Indeed by (2b) we have $\Gamma \vdash T = T$, by Lemma 6.3(1) we conclude that $\Gamma' \vdash T = T$ and by (1a) that $\Gamma', T\rhd$.

2. We need to show that for $(\Gamma \vdash o : S)$ and $(\Gamma, S) \sim (\Gamma', S')$ there exists $(\Gamma' \vdash o' : S')$ such that $(\Gamma' \vdash o' : S') \simeq (\Gamma \vdash o : S)$. It is sufficient to take $o' = o$. Indeed, by (3b) we have $(\Gamma \vdash o = o : S)$, by Lemma 6.3(2) we conclude that $(\Gamma' \vdash o = o : S')$ and by (2a) that $(\Gamma' \vdash o : S')$.

Lemma 6.6 *The equivalence relations $\sim$ and $\simeq$ are compatible with the operations $T, \widetilde{T}, S, \widetilde{S}$ and δ.*

Proof: (1) Given $(\Gamma_1, T\triangleright) \sim (\Gamma'_1, T'\triangleright)$ and $(\Gamma_1, \Gamma_2\triangleright) \sim (\Gamma'_1, \Gamma'_2\triangleright)$ we have to show that

$$(\Gamma_1, T, t_{n+1}\Gamma_2) \sim (\Gamma'_1, T', t_{n+1}\Gamma'_2).$$

where $n = l(\Gamma_1) = l(\Gamma'_1)$.

Proceed by induction on $l(\Gamma_2)$. For $l(\Gamma_2) = 0$ the assertion is obvious. Let $(\Gamma_1, T\triangleright) \sim (\Gamma'_1, T'\triangleright)$ and $(\Gamma_1, \Gamma_2, S\triangleright) \sim (\Gamma'_1, \Gamma'_2, S'\triangleright)$. The later condition is equivalent to $(\Gamma_1, \Gamma_2\triangleright) \sim (\Gamma'_1, \Gamma'_2\triangleright)$ and $(\Gamma_1, \Gamma_2 \vdash S = S')$. By the inductive assumption we have $(\Gamma_1, T, t_{n+1}\Gamma_2) \sim (\Gamma'_1, T', t_{n+1}\Gamma'_2)$. By (5a) we conclude that $(\Gamma_1, T, t_{n+1}\Gamma_2 \vdash t_{n+1}S = t_{n+1}S')$. Therefore by definition of $\sim$ we have $(\Gamma_1, T, t_{n+1}\Gamma_2, t_{n+1}S) \sim (\Gamma'_1, T', t_{n+1}\Gamma'_2, t_{n+1}S')$.

(2) Given $(\Gamma_1, T\triangleright) \sim (\Gamma'_1, T'\triangleright)$ and $(\Gamma_1, \Gamma_2 \vdash o : S) \simeq (\Gamma'_1, \Gamma'_2 \vdash o' : S')$ we have to show that $(\Gamma_1, T, t_{n+1}\Gamma_2 \vdash t_{n+1}o : t_{n+1}S) \simeq (\Gamma'_1, T', t_{n+1}\Gamma'_2 \vdash t_{n+1}o' : t_{n+1}S')$ where $n = l(\Gamma_1) = l(\Gamma'_1)$. We have $(\Gamma_1, \Gamma_2, S) \sim (\Gamma'_1, \Gamma'_2, S')$ and $(\Gamma_1, \Gamma_2 \vdash o = o' : S)$. By (5b) we get $(\Gamma_1, T, t_{n+1}\Gamma_2 \vdash t_{n+1}o = t_{n+1}o' : t_{n+1}S)$. By (1) of this lemma we get $(\Gamma_1, T, t_{n+1}\Gamma_2, t_{n+1}S) \sim (\Gamma'_1, T', t_{n+1}\Gamma'_2, t_{n+1}S')$ and therefore by definition of $\simeq$ we get $(\Gamma_1, T, t_{n+1}\Gamma_2 \vdash t_{n+1}o : t_{n+1}S) \simeq (\Gamma'_1, T', t_{n+1}\Gamma'_2 \vdash t_{n+1}o' : t_{n+1}S')$.

(3) Given $(\Gamma_1 \vdash r : T) \simeq (\Gamma'_1 \vdash r' : T')$ and $(\Gamma_1, T, \Gamma_2\triangleright) \sim (\Gamma'_1, T', \Gamma'_2\triangleright)$ we have to show that

$$(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1])) \sim (\Gamma'_1, s_{n+1}(\Gamma'_2[r'/n+1])).$$

where $n = l(\Gamma_1) = l(\Gamma'_1)$. Proceed by induction on $l(\Gamma_2)$. For $l(\Gamma_2) = 0$ the assertion follows directly from the definitions. Let $(\Gamma_1 \vdash r : T) \simeq (\Gamma'_1 \vdash r' : T')$ and $(\Gamma_1, T, \Gamma_2, S\triangleright) \sim (\Gamma'_1, T', \Gamma'_2, S'\triangleright)$. The later condition is equivalent to $(\Gamma_1, T, \Gamma_2\triangleright) \sim (\Gamma'_1, T', \Gamma'_2\triangleright)$ and $(\Gamma_1, T, \Gamma_2 \vdash S = S')$. By the inductive assumption we have $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1])) \sim (\Gamma'_1, s_{n+1}(\Gamma'_2[r'/n+1]))$. It remains to show that $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(S[r/n+1]) = s_{n+1}(S'[r'/n+1]))$. By (2d) it is sufficient to show that $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(S[r/n+1]) = s_{n+1}(S'[r/n+1]))$ and $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(S'[r/n+1]) = s_{n+1}(S'[r'/n+1]))$. The first relation follows directly from (6a). To prove the second one it is sufficient by (7a) to show that $(\Gamma_1, T, \Gamma_2, S'\triangleright)$ which follows from our assumption through (2c) and (2a).

(4) Given $(\Gamma_1 \vdash r : T) \simeq (\Gamma'_1 \vdash r' : T')$ and $(\Gamma_1, T, \Gamma_2 \vdash o : S) \simeq (\Gamma'_1, T', \Gamma'_2 \vdash o' : S')$ we have to show that

$$(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(o[r/n+1]) : s_{n+1}(S[r/n+1])) \simeq$$
$$(\Gamma'_1, s_{n+1}(\Gamma'_2[r'/n+1]) \vdash s_{n+1}(o'[r'/n+1]) : s_{n+1}(S'[r'/n+1])).$$

where $n = l(\Gamma_1) = l(\Gamma'_1)$ or equivalently that

$$(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]), s_{n+1}(S[r/n+1])) \sim (\Gamma'_1, s_{n+1}(\Gamma'_2[r'/n+1]), s_{n+1}(S'[r'/n+1]))$$

and $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(o[r/n+1]) = s_{n+1}(o'[r'/n+1]) : s_{n+1}(S[r/n+1]))$. The first statement follows from part (3) of the lemma. To prove the second statement it is sufficient by (3d) to show that $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(o[r/n+1]) = s_{n+1}(o'[r/n+1]) : s_{n+1}(S[r/n+1]))$ and $(\Gamma_1, s_{n+1}(\Gamma_2[r/n+1]) \vdash s_{n+1}(o'[r/n+1]) = s_{n+1}(o'[r'/n+1]) : s_{n+1}(S[r/n+1]))$. The first assertion follows directly from (6b). To prove the second one it is sufficient in view of (7b) to show that $(\Gamma_1, T, \Gamma_2 \vdash o' : S)$ which follows conditions (3c) and (3a).

(5) Given $(\Gamma, T) \sim (\Gamma', T')$ we need to show that $(\Gamma, T \vdash (n+1) : T) \simeq (\Gamma', T' \vdash (n+1) : T')$ or equivalently that $(\Gamma, T, T) \sim (\Gamma, T', T')$ and $(\Gamma, T \vdash (n+1) = (n+1) : T)$. The second part follows from (3b). To prove the first part we need to show that $(\Gamma, T \vdash T = T')$. This follows from our assumption by (5a).

Lemma 6.7 *Let C be a subset of $Ob(CC(R, LM))$ which is closed under ft. Let $\leq$ be a transitive relation on C such that:*

1. *$\Gamma \leq \Gamma'$ implies $l(\Gamma) = l(\Gamma')$,*
2. *$\Gamma \in C$ and $ft(\Gamma) \leq F$ implies $\sigma(\Gamma, F) \in C$ and $\Gamma \leq \sigma(\Gamma, F)$.*

Then $\Gamma \in C$ and $ft^i(\Gamma) \leq F$ for some $i \geq 1$, implies that $\Gamma \leq \sigma(\Gamma, F)$.

Proof: Simple induction on i.

Lemma 6.8 *Let C and $\leq$ be as in Lemma 6.7. Then one has:*

1. *$(\Gamma, T) \leq (\Gamma, T')$ and $\Gamma \leq \Gamma'$ implies that $(\Gamma, T) \leq (\Gamma', T')$,*
2. *if $\leq$ is ft-monotone (i.e. $\Gamma \leq \Gamma'$ implies $ft(\Gamma) \leq ft(\Gamma')$) and symmetric then $(\Gamma, T) \leq (\Gamma', T')$ implies that $(\Gamma, T) \leq (\Gamma, T')$.*

Proof: The first assertion follows from

$$(\Gamma, T) \leq (\Gamma, T') \leq \sigma((\Gamma, T'), \Gamma') = (\Gamma', T')$$

The second assertion follows from

$$(\Gamma, T) \leq (\Gamma', T') \leq \sigma((\Gamma', T'), \Gamma) = (\Gamma, T')$$

where the second $\leq$ requires $\Gamma' \leq \Gamma$ which follows from ft-monotonicity and symmetry.

Lemma 6.9 *Let $C, \leq$ be as in Lemma 6.7, let $\widetilde{C}$ be a subset of $\widetilde{Ob}(CC(R, LM))$ and $\leq'$ a transitive relation on $\widetilde{C}$ such that:*

1. *$\mathcal{J} \leq' \mathcal{J}'$ implies $\partial(\mathcal{J}) \leq \partial(\mathcal{J}')$,*
2. *$\mathcal{J} \in \widetilde{C}$ and $\partial(\mathcal{J}) \leq F$ implies $\widetilde{\sigma}(\mathcal{J}, F) \in \widetilde{C}$ and $\mathcal{J} \leq' \widetilde{\sigma}(\mathcal{J}, F)$.*

Then $\mathcal{J} \in \widetilde{C}$ and $ft^i(\partial(\mathcal{J})) \leq F$ for some $i \geq 0$ implies $\mathcal{J} \leq \widetilde{\sigma}(\mathcal{J}, F)$.

Proof: Simple induction on i.

Lemma 6.10 *Let $C, \leq$ and $\widetilde{C}, \leq'$ be as in Lemma 6.9. Then one has:*

1. *$(\Gamma \vdash o : T) \leq' (\Gamma \vdash o' : T)$ and $(\Gamma, T) \leq (\Gamma', T')$ implies that $(\Gamma \vdash o : T) \leq' (\Gamma' \vdash o' : T')$,*
2. *if $(\leq, \leq')$ is ∂-monotone (i.e. $\mathcal{J} \leq' \mathcal{J}'$ implies $\partial(\mathcal{J}) \leq \partial(\mathcal{J}')$) and $\leq$ is symmetric then $(\Gamma \vdash o : T) \leq' (\Gamma' \vdash o' : T')$ implies that $(\Gamma \vdash o : T) \leq' (\Gamma \vdash o' : T)$.*

Proof: The first assertion follows from

$$(\Gamma \vdash o : T) \leq' (\Gamma \vdash o' : T) \leq' \widetilde{\sigma}((\Gamma \vdash o' : T), (\Gamma', T')) = (\Gamma' \vdash o' : T')$$

The second assertion follows from

$$\Gamma \vdash o : T) \leq' (\Gamma' \vdash o' : T') \leq' \sigma((\Gamma' \vdash o' : T'), (\Gamma, T)) = (\Gamma \vdash o' : T)$$

where the second $\leq$ requires $\Gamma' \leq \Gamma$ which follows from ∂-monotonicity of $\leq'$ and symmetry of $\leq$.

Proposition 6.11 *Let $(C, \widetilde{C})$ be subsets in $Ob(CC(R, LM))$ and $\widetilde{Ob}(CC(R, LM))$ respectively which correspond to a C-subsystem CC of $CC(R, LM)$. Then the constructions presented above establish a bijection between pairs of subsets $(Ceq, \widetilde{Ceq})$ which together with $(C, \widetilde{C})$ satisfy the conditions of Proposition 6.2 and pairs of equivalence relations $(\sim, \simeq)$ on $(C, \widetilde{C})$ such that:*

1. *$(\sim, \simeq)$ corresponds to a regular congruence relation on CC (i.e., satisfies the conditions of [15, Proposition 5.4]),*
2. *$\Gamma \in C$ and $ft(\Gamma) \sim F$ implies $\Gamma \sim \sigma(\Gamma, F)$,*
3. *$\mathcal{J} \in \widetilde{C}$ and $\partial(\mathcal{J}) \sim F$ implies $\mathcal{J} \simeq \widetilde{\sigma}(\mathcal{J}, F)$.*

Proof: One constructs a pair $(\sim, \simeq)$ from $(Ceq, \widetilde{Ceq})$ as in Definition 6.1. This pair corresponds to a regular congruence relation by Proposition 6.2. Conditions (2),(3) follow from Lemma 6.3.

Let $(\sim, \simeq)$ be equivalence relations satisfying the conditions of the proposition. Define Ceq as the set of sequences (Γ, T, T') such that $(\Gamma, T), (\Gamma, T') \in C$ and $(\Gamma, T) \sim (\Gamma, T')$. Define $\widetilde{Ceq}$ as the set of sequences (Γ, T, o, o') such that $(\Gamma, T, o), (\Gamma, T, o') \in \widetilde{C}$ and $(\Gamma, T, o) \simeq (\Gamma, T, o')$.

Let us show that these subsets satisfy the conditions of Proposition 6.2. Conditions (2.a-2.d) and (3.a-3d) are obvious.

Condition (4a) follows from (2) by Lemma 6.7. Conditions (4b) and (4c) follow from (3) by Lemma 6.9.

Conditions (5a) and (5b) follow from the compatibility of $(\sim, \simeq)$ with T and $\widetilde{T}$.

Conditions (6a),(6b),(7a),(7b) follow from the compatibility of $(\sim, \simeq)$ with S and $\widetilde{S}$.

7 Pairs (R, LM) associated with nominal signatures.

The constructions of this paper produce C-systems from a pair (R, LM) where R is a monad on $Sets$ and LM is a left R-module with values in $Sets$ together with sets C, $\widetilde{C}$, Ceq and $\widetilde{Ceq}$.

One class of such pairs is obtained by taking R to be the monad defined by a signature as in [6, p.228]. For example, the contextual category of the Martin-Lof type theory from 1972, $MLTT72$ defined in [10], is obtained by applying Proposition 6.2 in the case of the pair (R, R) where R is the monad defined by the signature that corresponds to the nominal signature of Example 7.2.

The following construction that covers more examples associates a pair (R, LM) to a quadruple $(\Sigma, Term, P, \mathbf{Type})$ where Σ is a nominal signature with one name-sort Var and a set of data-sorts D, $Term \in \mathbf{D}$ is a data-sort, P is a family of sets parametrized by $\mathbf{D} - \{Term\}$, and $\mathbf{Type} \subset \{\mathbf{D}\}$ is a subset of data-sorts.

In most examples either $\mathbf{D} = \{Term\}$ or $\mathbf{D} = \{Term, Type\}$, $\mathbf{Type} = \{Type\}$ and $P = P_{Type}$ is the set of "type-variables".

The only example which I know of where there are more than two data-sorts is the logic-enriched type theory of [1] where $\mathbf{D} = \{Term, Type, Prop\}$, $\mathbf{Type} = \{Type\}$, P_{Type} is the set of type variables and P_{Prop} is the set of propositional variables.

The construction is as follows. A *nominal signature* (see [13, Section 8.1]) starts with a set of name-sorts $\mathbf{N}$ and the set of data-sorts $\mathbf{D}$. We will be interested in the case when there is only one name-sort Var.

A compound sort S is defined as an expression formed from Var, elements of $\mathbf{D}$, and the unit sort 1 using two operations: one sending S_1 and S_2 to (S_1, S_2) and another one sending S to $Var.S$. For better notations one takes $(\_,\_)$ to associate on the left i.e. (S_1, S_2, S_3) means $((S_1, S_2), S_3)$ and similarly for longer sequences.

Let CS be the set of compound sorts. An arity is a pair (S, D) where $S \in CS$ and $D \in \mathbf{D}$. Let $A(\mathbf{D})$ be the set of arities for the set of data-sorts $\mathbf{D}$.

A nominal signature is defined as a set Op, which is called the set of operations, together with a function $Ar : Op \to A(\mathbf{D})$ which assigns to any operation its "arirty". One writes $O : S \to D$ to denote that operation O has arity (S, D). We let Ar_{CS} and $Ar_{\mathbf{D}}$ denote the two components of the arity.

For example, the nominal signature of the lambda calculus has one data-sort $Term$ and three operations V, L, and A of the form:

$$V : Var \to Term$$

$$L : Var.Term \to Term$$

$$A : Term.Term \to Term$$

The algebraic signature with one sort $Term$, one operation S in one variable and one constant O will, in this language, have *three* operations:

$$V : Var \to Term$$

$$S : Term \to Term$$

$$O : 1 \to Term$$

More generally, an algebraic signature is a nominal signature where

$$Op = Op_0 \coprod \{v_D\}_{D \in \mathbf{D}}$$

with

$$v_D : Var \to D$$

and for $O \in Op_0$

$$O : (D_1, \ldots, D_n) \to D$$

for some $n \geq 0$ and $D_1, \ldots, D_n, D \in \mathbf{D}$ where n and D's may depend on O.

An example of a signature where variables are not terms is given in [13].

A nominal signature can be used to construct terms of all compound sorts in the more or less obvious way. Next one defines the notion free and bound occurrences of variables in these terms and the notion of the α-equivalence. For a nominal signature Σ and a compound sort S one writes $\Sigma_\alpha(S)$ for the set of α-equivalence classes of terms of sort S build using Σ.

In the case when Σ is the λ-calculus signature one gets the usual set of α-equivalence classes of λ-terms considering $\Sigma_\alpha(Term)$.

To any nominal signature Σ one associates, following [13], a functor $T_\Sigma : Nom^{\mathbf{D}} \to Nom^{\mathbf{D}}$ where Nom is the category of nominal sets, as follows.

First one associates a functor $[S] : Nom^{\mathbf{D}} \to Nom$ to any compound sort S by the rule:

$$[Var](X) = \mathbf{A}$$

$$[D](X) = X_D$$

$$[1](X) = 1$$

$$[(S_1, S_2)](X) = X \times X$$

$$[(Var.S)] = [\mathbf{A}](X)$$

where $\mathbf{A}$ is the standard atomic nominal set (the set of names with the canonical action of the permutation group $Perm$) and $[\mathbf{A}]$ is the name-abstraction functor $Nom \to Nom$ which is defined in [13, Section 4].

Let Op_D for $D \in \mathbf{D}$ be the set of operations O with the target sort D, i.e., such that $Ar_{\mathbf{D}}(O) = D$. Then one defines $T_\Sigma(X)$ by the rule

$$T_\Sigma(X)_D = \coprod_{O \in Op_D} [Ar_{CS}(O)].$$

For example, if Σ is the signature of λ-calculus then

$$T_\Sigma(X) = \mathbf{A} \coprod [\mathbf{A}](X) \coprod (X \times X)$$

One of the main results of [13] is that the functor T_Σ has an initial algebra I_Σ for any Σ and $(I_\Sigma)_D = \Sigma_\alpha(D)$.

Let us extend this construction to a monad on $Nom^{\mathbf{D}}$ and then on $Sets^{\mathbf{D}}$. First observe that for any $X \in Nom^{\mathbf{D}}$ the functor $Y \mapsto T_\Sigma(Y) \coprod X$ is finitely presented and therefore it has an initial algebra. Let us denote this algebra by $NR_\Sigma(X)$.

By [2, pp. 243-244], NR_Σ is a monad on $Nom^{\mathbf{D}}$ whose category of algebras is equivalent to the category of T_Σ-algebras (i.e. NR_Σ is the free monad generated by T_Σ).

The functor $Discr : Sets \to Nom$ which takes a set to the corresponding discrete nominal set has a right adjoint $Inv : Nom \to Sets$ which sends a nominal set X to the set of its fixed points X^{Perm}. The functors $Discr^{\mathbf{D}}$ and $Inv^{\mathbf{D}}$ form an adjoint pair between the categories $Nom^{\mathbf{D}}$ and $Sets^{\mathbf{D}}$.

Given a monad R on a category $\mathcal{C}$ and an adjoint pair (LF, RF) where $RF : \mathcal{C} \to \mathcal{C}'$ is the right adjoint, the composition $R' = RF \circ R \circ LF$ is a monad on $\mathcal{C}'$.

Applying this fact to the monad NR_Σ and the pair $(Discr^{\mathbf{D}}, Inv^{\mathbf{D}})$ we conclude that the functor

$$R_\Sigma : X \mapsto NR_\Sigma(Discr^{\mathbf{D}}(X))^{Perm}$$

is a monad on $Sets^{\mathbf{D}}$.

For a family of sets X the functor $T_\Sigma \coprod Discr^{\mathbf{D}}(X)$ is naturally isomorphic to the functor $T_{\Sigma+X}$ where $\Sigma + X$ is the signature with the set of operations $Op \coprod (\coprod_{D\in\mathbf{D}} X_D)$ and the arity function defined on $x \in X_D$ by $Ar(x) = (1, D)$ and

$$R_\Sigma(X) = NR_\Sigma(Discr^{\mathbf{D}}(X))^{Perm} = I^{Perm}_{T_\Sigma \coprod Discr^{\mathbf{D}}(X)} = I^{Perm}_{T_{\Sigma+X}}.$$

Therefore

$$(R_\Sigma(X))_D = (\Sigma + X)_\alpha(D)^{Perm}$$

is the set of invariants in the set of α-equivalence classes of terms of sort D with respect to the signature $\Sigma+X$ i.e. the set of α-equivalence classes of closed terms of sort D with respect to $\Sigma+X$.

If $X_D = \{x_{1,D}, \dots, x_{n_D,D}\}$ are finite sets, then the terms with respect to the signature $\Sigma+X$ can be seen as terms with respect to Σ which depend on additional parameters $x_{i,D}$ of the corresponding sorts and the closed terms as the terms with respect to Σ relative to the name space $\mathbf{A}^{\mathbf{D}} + X$ such that all the occurrences of names from $\mathbf{A}^{\mathbf{D}}$ are bound and all the occurrences of names from X are free.

To obtain from this construction a pair (R, LM) of a monad on $Sets$ and a left module over this monad with values in $Sets$ we will use Lemma 2.5. Let $Term \in \mathbf{D}$ and $\mathbf{Type} \subset \mathbf{D}$. Let P a family of sets parametrized by $\mathbf{D} - Term$. For a set X let (X, P) be the family such that $(X, P)_{Term} = X$ and $(X, P)_D = P_D$ for $D \neq Term$.

Then $X \mapsto (R_\Sigma(X, P))_{Term}$ is a monad $R_{\Sigma,Term,P}$ on $Sets$ by Lemma 2.2 and

$$X \mapsto \coprod_{D\in\mathbf{Type}} (R_\Sigma(X, P))_D$$

is a left module $LM_{\Sigma,Term,P,\mathbf{Type}}$ over $R_{\Sigma,Term,P}$ by Lemmas 2.5 and 2.4(b).

Example 7.1 The C-systems of generalized algebraic theories (GATs) of [3],[4] (see also [5]) are obtained by using algebraic signatures with two data sorts $\mathbf{D} = \{Term, Type\}$, $\mathbf{Type} = \{Type\}$ and $P = \emptyset$. The "symbols" of the GAT are operations of the corresponding algebraic signature. The term symbols of degree n have arity $(Term, \dots, Term) \to Term$ and the type symbols of degree n have arity $(Term, \dots, Term) \to Type$ where in both cases the lentth of the sequence $(Term, \dots, Term)$ is n.

Example 7.2 To define the Martin-Lof Type theory MLTT72 of [10] one needs to consider the case when $\mathbf{D} = \{Term\}$ and the nominal signature is of the form:

$$v : Var \to Term$$

$$\Pi : (Term, Var.Term) \to Term \quad \lambda : (Term, Var.Term) \to Term$$

$$app : (Term, Term) \to Term$$

$$\Sigma : (Term, Var.Term) \to Term \quad pair : (Term, Term) \to Term$$

$$E : (Term, Var.(Var.Term)) \to Term$$

$$+ : (Term, Term) \to Term \quad i : Term \to Term \quad j : Term \to Term$$

$$D : ((Term, Var.Term), Var.Term)) \to Term$$

$$V : 1 \to Term$$

$$N_n : 1 \to Term \quad i_n : 1 \to Term \quad R_n : (Term, \ldots, Term), \ldots) \to Term$$

$$n \geq 0 \quad 1 \leq i \leq n$$

$$N : 1 \to Term \quad 0 : 1 \to Term \quad s : Term \to Term$$

$$R : ((Term, Term), (Var.(Var.Term))) \to Term$$

Note that in fact E, D, R_n, and R should also have the type family C (see [10, 2.3.6, 2.3.8, 2.3.10, 2.3.12]) as an argument which, in our notation, means an additional component of the form $Var.Term$ in their arities.

In fact, the original definition from [10] allows for additional "type constants" (see [10, 2.2.1]) of various algebraic arities which are analogous to the predicate constants in the predicate logic. As such it is a definition of a family of type systems. The signatures underlying all type systems in this family are obtained by extending the signature described above by a set of operations of the form $P : (Term, \ldots, Term) \to Term$.

For the signature of the MLTT78 see [11, p. 158]

Remark 7.3 It is possible to "encode" a nominal signature in typed λ-calculus using the idea that closed terms are objects of a base type *term*, terms with one free variable are objects of the type $term \to term$, terms with two free variables are objects of $term \to term \to term$ etc. This encoding allows one to describe the substitutions of closed terms into terms with free variables as applications in the meta-theory. However, it does not allow to describe the substitution of, e.g., terms with one free variable into terms with one free variable, i.e., the full monadic structure is not recoverable from such a description. This is the reason why the use of typed λ-calculus systems such as the Logical Framework for the description of the syntax of dependent type theories is of limited use.

References

[1] Peter Aczel and Nicola Gambino. Collection principles in dependent type theory. In *Types for proofs and programs (Durham, 2000)*, volume 2277 of *Lecture Notes in Comput. Sci.*, pages 1–23. Springer, Berlin, 2002.

[2] Steve Awodey. *Category theory*, volume 52 of *Oxford Logic Guides*. Oxford University Press, Oxford, second edition, 2010.

[3] John Cartmell. Generalised algebraic theories and contextual categories. *Ph.D. Thesis, Oxford University*, 1978. `https://uf-ias-2012.wikispaces.com/Semantics+of+type+theory`.

[4] John Cartmell. Generalised algebraic theories and contextual categories. *Ann. Pure Appl. Logic*, 32(3):209–243, 1986.

[5] Richard Garner. Combinatorial structure of type dependency. *J. Pure Appl. Algebra*, 219(6):1885–1914, 2015.

[6] André Hirschowitz and Marco Maggesi. Modules over monads and linearity. In *Logic, language, information and computation*, volume 4576 of *Lecture Notes in Comput. Sci.*, pages 218–237. Springer, Berlin, 2007.

[7] André Hirschowitz and Marco Maggesi. Higher order theories. *http://arxiv.org/abs/0704.2900*, 2010.

[8] André Hirschowitz and Marco Maggesi. Modules over monads and initial semantics. *Inform. and Comput.*, 208(5):545–564, 2010.

[9] Bart Jacobs. *Categorical logic and type theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1999.

[10] Per Martin-Löf. An intuitionistic theory of types. *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.926&rep=rep1&type=pdf*, 1972.

[11] Per Martin-Löf. Constructive mathematics and computer programming. In *Logic, methodology and philosophy of science, VI (Hannover, 1979)*, volume 104 of *Stud. Logic Found. Math.*, pages 153–175. North-Holland, Amsterdam, 1982.

[12] Eugenio Moggi. Notions of computation and monads. *Inform. and Comput.*, 93(1):55–92, 1991. Selections from the 1989 IEEE Symposium on Logic in Computer Science.

[13] Andrew M. Pitts. *Nominal sets*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2013. Names and symmetry in computer science.

[14] Vladimir Voevodsky. Notes on type systems. *https://github.com/vladimirias/old_notes_on_type_systems*, 2009-2012.

[15] Vladimir Voevodsky. Subsystems and regular quotients of C-systems. In *Conference on Mathematics and its Applications, (Kuwait City, 2014)*, number to appear, pages 1–11, 2015.