Basic Research in Computer Science

BRICS RS-94-22

T. Braüner: A General Adequacy Result for a Linear Functional Language

A General Adequacy Result for a Linear Functional Language

Torben Braüner

BRICS Report Series **RS-94-22**

ISSN 0909-0878 **August 1994**

See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through WWW and anonymous FTP:

`http://www.brics.dk/`
`ftp ftp.brics.dk (cd pub/BRICS)`

A General Adequacy Result for a Linear Functional Language

Torben Braüner*

BRICS†
Computer Science Department
Aarhus University
Ny Munkegade
DK-8000 Aarhus C, Denmark

Abstract

A main concern of the paper will be a Curry-Howard interpretation of Intuitionistic Linear Logic. It will be extended with recursion, and the resulting functional programming language will be given operational as well as categorical semantics. The two semantics will be related by soundness and adequacy results. The main features of the categorical semantics are that convergence/divergence behaviour is modelled by a strong monad, and that recursion is modelled by "linear fixpoints" induced by CPO structure on the hom-sets. The "linear fixpoints" correspond to ordinary fixpoints in the category of free coalgebras w.r.t. the comonad used to interpret the "of course" modality. Concrete categories from (stable) domain theory satisfying the axioms of the categorical model are given, and thus adequacy follows in these instances from the general result.

*Internet: tor@daimi.aau.dk

†**B**asic **R**esearch **i**n **C**omputer **S**cience, Centre of the Danish National Research Foundation.

Contents

1 Introduction

Linear logic was discovered by J.-Y. Girard in 1987 and published in a now famous paper [Gir87]. In the abstract of this paper, it is stated that "a completely new approach to the whole area between constructive logics and computer science is initiated". Since then, a lot of work has been done to corroborate this claim. The present paper will deal with a computational interpretation of Intuitionistic Linear Logic (ILL).

In [Abr90] the first Curry-Howard interpretation of ILL is given. The resulting system is essentially a refinement of the usual λ-calculus where the copying and discarding of values is written explicitly in the terms. One of the rules of this system has a deficiency that force ! to be isomorphic to !! in any reasonable categorical interpretation. It was in 1992 repaired by the authors of [BBdPH92] (and by the author of this paper) by changing the system in an appropriate way, and by discovering a Natural Deduction style presentation equivalent to the hitherto known Gentzen style presentation of ILL. This work settled the question about how to interpret ILL via the Curry-Howard isomorphism. The Natural Deduction style proof-rules will in the present paper be considered as type assignment rules for a programming language cf. Curry-Howard. Moreover, the system will extended with recursion (every decent programming language has recursion!), and given operational as well as categorical semantics such that the two semantics are related by soundness and adequacy results. Our categorical model is able to model convergence/divergence behaviour, and moreover, it deals with fixpoints in a linear context. We have devoted a section to show some results on fixpoints in a linear contexts and their relations to ordinary fixpoints.

Now, Girard worked with coherence spaces and stable maps and observed that the stable function space $A \Rightarrow B$ can be decomposed into more basic operations, namely $!A \multimap B$, where ! is an operation on coherence spaces, and $\multimap$ is the operation corresponding to formation of linear stable function space. To be more precise: The functor that forgets the linearity of linear stable maps has a left adjoint !. This fundamental observation gave rise to the discovery of Linear Logic, and the corresponding coherence space interpretation has since been considered canonical. Now, it turns out that the same phenomenon is present if we consider the category of pre dI domains and stable functions, $predI_s$, and the category of predI domains and affine stable functions, $predI_a$. The functor from $predI_a$ to $predI_s$ that forgets the affine nature has a left adjoint !. This induces a comonad on the symmetric monoidal closed category $predI_a$ in the same way as we have a comonad on the symmetric monoidal closed category of coherence spaces and linear stable maps. Moreover, the forgetful functor from the category of dI domains and linear stable functions, dI_l, to $predI_a$ has a left adjoint which induces a monad on $predI_a$, namely what in similar contexts is called a lift monad. Thus, we have a model of ILL with additional structure which enables us to model convergence/divergence behaviour. This model satisfy all the axioms of our categorical model, and we therefore have a sound and adequate denotational semantics where types are interpreted as pre dI domains, and terms as affine stable functions.

2 A linear functional language, LTS$_{+Rec}$

2.1 The Curry-Howard isomorphism

The classical Curry-Howard isomorphism relates the λ-calculus to Intuitionistic Logic. It says that types can be viewed as formulas and typable terms as proofs and vice versa. The point is that proof-rules for Intuitionistic Logic can be "decorated" with terms such that the term induced by a proof encodes the proof. An appropriate term language for this purpose is the λ-calculus. It turns out that we then get the rules for assigning types to terms, [GLT89]. The present paper deals with an analogous correspondence between ILL and the Linear Term System (LTS). Historically, LTS was discovered as a term language to decorate proof-rules for ILL, but it can be considered as a programming language independently of its historical roots. The proof-rules will then appear as typing rules. We then get the Curry-Howard isomorphism as follows: given a proof of $A_1, ..., A_n \vdash A$ in ILL, that is, a proof of the formula A, one can inductively construct a derivation of a sequent $x_1 : A_1, ..., x_n : A_n \vdash t : A$ in LTS, that is, a term t of type A. Conversely, if one has a derivable sequent $x_1 : A_1, ..., x_n : A_n \vdash t : A$ in LTS, there is an easy way to get a proof of $A_1, ..., A_n \vdash A$ in ILL: erase all variables and terms in the derivation of the type assignment. The two processes are each others inverses modulo renaming of variables. Proof-rules for ILL in Natural Deduction style are given in Appendix A and LTS is introduced formally below.

2.2 Definition of LTS$_{+Rec}$

Types are given by the grammar $s \ ::= \ I \mid s \otimes s \mid s \multimap s \mid s \& s \mid s \oplus s \mid !s$, and terms by the grammar

$$
\begin{array}{rcl}
t & ::= & x \mid \\
 & & * \mid \text{let } t \text{ be } * \text{ in } t \mid \\
 & & t \otimes t \mid \text{let } t \text{ be } x \otimes y \text{ in } t \mid \\
 & & \lambda x.t \mid tt \mid \\
 & & (t,t) \mid \text{fst}(t) \mid \text{snd}(t) \mid \\
 & & \text{inl}(t) \mid \text{inr}(t) \mid \text{case } t \text{ of inl}(x) => t \mid \text{inr}(y) => t \mid \\
 & & \text{let } t, ..., t \text{ be } x_1, ..., x_n \text{ in } !t \mid \text{derelict}(t) \mid \\
 & & \text{discard } t \text{ in } t \mid \text{ copy } t \text{ as } x, y \text{ in } t \mid \\
 & & \text{let } t, ..., t \text{ be } x_1, ..., x_n \text{ in rec} z.t
\end{array}
$$

where $t, ..., t$ means a sequence of n occurrences of t. In what follows, A, B, C, D will range over types, and u, v, w, f will range over terms. Some terms can be assigned a type in a way analogous to the typed λ-calculus. The type assignments will have the form of sequents $x_1 : A_1, ..., x_n : A_n \vdash u : A$ where $x_1, ..., x_n$ are pairwise distinct variables and $\{x_1, ..., x_n\}$ is the set of free variables of the term u. We will frequently write Γ instead of $x_1 : A_1, ..., x_n : A_n$ or $A_1, ..., A_n$, and $!\Gamma$ instead of $x_1 :!A_1, ..., x_n :!A_n$ or $!A_1, ..., !A_n$. The type assignments are derived according to the rules in Appendix C (which also contains the rules for assignment of categorical semantics). The notation will be abused when necessary in the following way: the expression "$\Gamma \vdash u : A$" can mean either the sequent itself or a certain derivation of the sequent. The name of a rule, for example (Id), can

mean either the rule itself or a certain instance of the rule. The actual interpretation is to be decided by the context. The terms together with the typing rules for the fragment corresponding to ILL will be called LTS, and the extension with recursion will be called LTS_{+Rec}. Note that if we remove the terms from the typing rules for LTS, we get the proof-rules for ILL given in Appendix A. From now on, we will consider only typable terms.

Note that the definition of sequents implicitly restricts use of the rules. It is for example not possible to use the $(\otimes - I)$ rule if Γ and Δ have common variables.

2.3 Properties of LTS_{+Rec}

The derivation of a type assignment is essentially unique (which actually is the essence of the Curry-Howard isomorphism):

Proposition 2.1 *If the sequent $\Gamma \vdash u : A$ is derived by a given derivation, then the rule corresponding to the first rule instance above the sequent $\Gamma \vdash u : A$ which is different from an instance of $(Exchange)$, is uniquely determined by the term u.*

Proof. Induction in the derivation of $\Gamma \vdash u : A$. □

Lemma 2.2 *(Substitution Property) If $\Gamma \vdash u : A$ and $\Delta, x : A, \Lambda \vdash v : B$ both are derivable s.t. the variables in Γ and Δ, Λ are pairwise distinct, then $\Delta, \Gamma, \Lambda \vdash v[u/x] : B$ is derivable too.*

Proof. Induction in the derivation of $\Delta, x : A, \Lambda \vdash v : B$. □

The expression $v[u/x]$ denotes the term v where u has been substituted for every free occurence of x, and where bound variables of v have been renamed to avoid capture of free variables in u.

2.4 The choice of rule for ! introduction

Seen from a historical point if view, the term corresponding to the rule for introduction of ! has caused problems. In [Abr90], the first Curry-Howard interpretation of ILL was published. Here the rules are given in Gentzen style, named after the discoverer of a similar system of proof-rules for classical logic. The Natural Deduction formulation was not discovered at this time. In Gentzen style, we only have introduction rules. A connective can be introduced on both sides of the sequent, in opposition to Natural Deduction style, where we can either eliminate a connective, or introduce it on the right hand side. A Gentzen style formulation of ILL can be found in Appendix B. The $(! - I)$ rule of the Natural Deduction formulation corresponds to the $(! - R)$ rule of the Gentzen formulation. In the above mentioned article the $(! - R)$ rule is decorated with the following terms:

$$\frac{x_1 :! A_1, ..., x_n :! A_n \vdash u : A}{x_1 :! A_1, ..., x_n :! A_n \vdash ! u :! A} \, (! - R)$$

The Gentzen style system enjoys the substitution property simply because it is a rule of the system, namely the (Cut) rule. The problem, as pointed out in [Wad91], is as follows: The (Cut) rule together with $(! - R)$ (decorated with terms as above) forces a collapse in the categorical model corresponding to the system. The ! modality is interpreted as a functor, and the two rules together would force ! to be isomorphic to !!. The problem is basically that a given sequent can have several derivations, and they all ought to give rise to the same categorical interpretation. The presence of (Cut) gives us two different interpretations of the same sequent (unless $! \cong !!$ in a canonical way).

In 1992 a new way to decorate the $(! - R)$ rule with terms, together with a Natural Deduction formulation of ILL, was discovered by the authors of [BBdPH92] (and by the author of this paper). The new decoration of $(! - R)$ is as follows:

$$\frac{x_1 :!A_1, ..., x_n :!A_n \vdash u : A}{z_1 :!A_1, ..., z_n :!A_n \vdash \text{let } z_1, ..., z_n \text{ be } x_1, ..., x_n \text{ in } !u :!A} (! - R)$$

The new rule can coexist with (Cut) without collapsing the model, and the derivations that with the old term decoration concluded with identical sequents, now concludes with different sequents (because the induced terms are different). We get a system equivalent to LTS if we take the Gentzen style formulation of ILL and decorate it with terms as originally done in [Abr90] *except* that we pick the correct decoration of the $(! - R)$ rule, cf. the discussion above. The Natural Deduction style formulation of ILL is given in Appendix A. The $(! - I)$ rule of the Natural Deduction formulation, corresponding to the above mentioned $(! - R)$ rule of the Gentzen formulation, is as follows:

$$\frac{\Gamma_1 \vdash w_1 :!A_1 \quad , ..., \quad \Gamma_n \vdash w_n :!A_n \qquad x_1 :!A_1, ..., x_n :!A_n \vdash u : A}{\Gamma_1, ..., \Gamma_n \vdash \text{let } w_1, ..., w_n \text{ be } x_1, ..., x_n \text{ in } !u :!A} (! - I)$$

2.5 The choice of rule for recursion

We want to extend LTS with a rule for parameterised recursion. Values corresponding to the parameters are copied in any reasonable semantics, so the corresponding types should be of ! type because we are in a linear context. Hence, a natural first choice would be:

$$\frac{x_1 :!A_1, ..., x_n :!A_n, z : B \vdash u : B}{x_1 :!A_1, ..., x_n :!A_n \vdash \text{rec} z.u : B} (Recursion)$$

An argument against this solution is that the function which assigns a fixpoint $f^\dagger \in B$ to a function $f : B \to B$ in our canonical model dI_a is not affine, that is, it can not be internalised as a map (a fixpoint operator) in the category. Thus, since we want give a denotational semantics to the rule in the category dI_a (via a categorical semantics), we should look for another definition of recursion. The next suggestion is:

$$\frac{x_1 :!A_1, ..., x_n :!A_n, z :!B \vdash u : B}{x_1 :!A_1, ..., x_n :!A_n \vdash \text{rec} z.u : B} (Recursion)$$

This rule is actually the one used in [Mac92]. We can now give a denotational semantics with our canonical model dI_a, but LTS extended with this rule does not enjoy the Substitution Property. Now, we are trying to extend LTS, where the underlying proof-rules

for ILL are in Natural Deduction style, with a rule for recursion. LTS would loose the Substitution Property in a similar way if we replaced the $(! - I)$ rule with the first of the above mentioned $(! - R)$ rules induced by the Gentzen formulation of ILL. We know how to deal with this problem, so we solve the problem with the rule for recursion in a similar way:

$$\frac{\Gamma_1 \vdash w_1 :!A_1 \quad ,..., \quad \Gamma_n \vdash w_n :!A_n \qquad x_1 :!A_1, ..., x_n :!A_n, z :!B \vdash u : B}{\Gamma_1, ..., \Gamma_n \vdash \text{let } w_1, ..., w_n \text{ be } x_1, ..., x_n \text{ in rec} z.u : B} (Recursion)$$

LTS extended with this rule, (that is, LTS_{+Rec}) enjoys the Substitution Property, and it allows definition of operational as well as denotational semantics in natural ways.

3 Operational semantics of LTS_{+Rec}

3.1 Definition of the operational semantics

We will now give an operational semantics for LTS in Natural Semantics style. We will consider free variables as placeholders for canonical terms, which corresponds to a call-by-value parameter passing strategy where we only substitute canonical terms for variables.

Definition 3.1 *A* canonical term *is a closed typable terms of one of the following shapes:*

$$* \qquad d \otimes e \qquad \lambda x.u \qquad (u, v) \qquad inl(c) \qquad inr(d) \qquad let\ c_1, ..., c_n\ be\ x_1, ..., x_n\ in\ !u$$

where $c, d, e, c_1, ..., c_n$ *are canonical terms.*

Let T be the set of closed typable terms, and C the set of canonical terms. The evaluation rules in Appendix E induces a relation, the evaluation relation: $\longrightarrow \subset T \times C$. Note how the choice of call-by-value parameter passing strategy is reflected in the evaluation rule for application: we evaluate a parameter to a canonical term before plugging it in.

Definition 3.2 *Given a term* u*, we will write* $u \downarrow$ *iff there exists a term* c *s.t.* $u \longrightarrow c$*. We will say that* u converges*.*

3.2 Properties of the operational semantics

If one considers the Gentzen style formulation of LTS, then cut-elimination gives rise to certain reductions on terms in the same way as cut-elimination in ordinary Intuitionistic Logic gives rise to reductions on terms in the λ-calculus. It turns out that the reductions induced by our operational semantics without recursion all are reductions induced by cut-elimination. It should be noted that our operational semantics only evaluates closed terms, and not every redex corresponding to cut-elimination is reduced. For example, every closed term $\lambda x.u$ is canonical, whatever is "inside" the abstraction. The operational semantics enjoys the following properties:

Proposition 3.3 *(Subject Reduction) If* $\vdash u : A$ *and* $u \longrightarrow c$*, then* $\vdash c : A$*.*

Proof. Induction in the derivation of $u \longrightarrow c$, where we use the Substitution Property. □

Proposition 3.4 *(Determinacy) If $u \longrightarrow c$ and $u \longrightarrow d$, then $c = d$.*

Proof. Induction in the derivation of $u \longrightarrow c$. □

Proposition 3.5 *(Convergence) If the term u is without recursion, then $u \downarrow$.*

Proof. A modified version of the proof in [Abr90]. □

4 General considerations about the categorical semantics

4.1 The connection to denotational semantics

In what follows, $[\![A]\!]$ will mean the interpretation of A and $[\![u]\!]$ the interpretation of u. When appropriate, we will abuse notation and omit the brackets. We have stated a reference where the categorical notions used can not be found in [Lan71].

In denotational semantics, a type A is normally interpreted as a set $[\![A]\!]$ with a certain structure. A term u of type A with free variables x_1, ..., x_n of type A_1, ..., A_n is then interpreted as a function $[\![u]\!]$ from $[\![A_1]\!] \times ... \times [\![A_n]\!]$ to $[\![A]\!]$. In particular, if u is closed, then the interpretation is a point in $[\![A]\!]$. For example in [Win93], computer programs are interpreted as continuous functions between appropriate domains. One wants the denotational semantics to have certain properties w.r.t. the operational semantics. Firstly, it has to be sound, that is, evaluation has to preserve the denotation. Secondly, the denotational and operational semantics has to agree w.r.t. relevant observations. This is called adequacy. For example in [Win93], a term may diverge because of recursion, so one wants this to be reflected in the denotational semantics: a term ought to converge if and only if it is interpreted as a non-bottom element in the relevant domain.

I will not give an explicit concrete denotational semantics of LTS_{+Rec} here, but instead give a categorical semantics, defined with the above mentioned goals in mind. It then follows that any concrete category satisfying the axioms of the categorical model induce a sound and adequate denotational semantics. The categorical semantics adheres to the following fundamental ideas of the categorical treatment of proof theory:

- Formulas are interpreted as objects
- Proofs are interpreted as maps
- Proof-rules correspond to natural transformations between appropriate hom-functors.

In [BBdPH92] a categorical semantics is given to a Gentzen style formulation of LTS without additives, but it is not equivalent to the relevant parts of our categorical semantics. Neither is their reductions on terms the same as the reductions on terms induced by the

relevant part of our operational semantics. It is important to notice how the differences in choice of rules for reductions on terms is reflected in differences in the choices of categorical semantics. For example, if we for a moment restrict our attention to LTS without additives, then the reductions induced by our operational semantics is a strict subset of the reductions induced by cut-elimination, as remarked earlier. Therefore the equations imposed on our model to get soundness w.r.t. our Natural Semantics style operational semantics are weaker than the equations imposed on their model to get soundness w.r.t. the reductions on terms induced by cut-elimination. If we compare the full system consisting of LTS_{+Rec} equipped with the operational semantics given in the previous section to the system given in [BBdPH92], there is an important difference: the first system has diverging terms, the second system does not. We therefore need additional categorical machinery to model convergence/divergence behaviour as we are interested in an adequacy result.

4.2 Initial assumptions about the interpretation

Types will be interpreted as objects, and we want to interpret sequents as arrows between appropriate objects. So we need an operation on objects to "put together" the interpretations of hypotheses into one object. To this end, we will assume that we are dealing with a monoidal category $(\mathcal{C}, I, \otimes)$. Good reasons for this choice can be found in [BBdPH92]. The interpretation of a sequent will be defined by induction in its derivation, so we have to be sure that the definition is independent of the derivation. The derivation of a sequent is unique up to applications of the $(Exchange)$ rule, which suggests that our category should be *symmetric* monoidal.

4.3 Modelling convergence/divergence behaviour

Following [Mog89] , we assume that we have a monad (T, η, μ) on $\mathcal{C}$ to model convergence/divergence behaviour. This induces the usual kleisli operator $kleisli$ as follows:

$$kleisli_{A,B} : hom(A, TB) \longrightarrow hom(TA, TB)$$
$$kleisli_{A,B}(f) = [TA \xrightarrow{Tf} TTB \xrightarrow{\mu_B} TB]$$

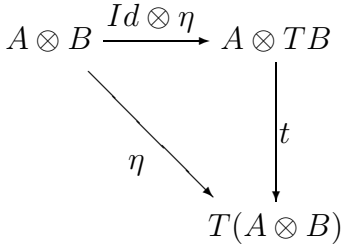
The idea is to distinguish between A, the object of values of type A, and TA, the object of computations of type A. Intuitively, η is the *inclusion* of values into computations, and $kleisli(f)$ is the *extension* of a function f from values to computations, to a function from computations to computations, which first evaluates a computation, and then applies f to the resulting value. In this context, canonical terms are to be thought of as values and arbitrary closed terms as computations. A sequent $x_1 : A_1, ..., x_n : A_n \vdash u : A$ will then be interpreted as a map $[\![u]\!] : [\![A_1]\!] \otimes ... \otimes [\![A_n]\!] \to T[\![A]\!]$ because we consider free variables as placeholders for canonical terms. Now, a program (a closed term) $\vdash u : A$ will be interpreted as a point $[\![u]\!] : I \to T[\![A]\!]$ (a point is a map with domain I). According to the intuition, it should be considered as a value iff it has η as factor. In a concrete case where the objects are CPOs of some kind, this corresponds to $[\![u]\!]$ being non-bottom. This motivates the following definition:

Definition 4.1 *For any point $f : I \to TB$, we will write $f \Downarrow$ iff there exists a map $h : I \to B$ s.t. $f = h; \eta_B$. We will say that f* converges semantically*.*

As in the case with [Mog89] we will use a strength natural transformation

$$t : (-) \otimes T(+) \to T(- \otimes +)$$

to transform a pair consisting of a value and a computation into the computation of a pair of values. The intuition is that t evaluates the second component, and returns the pair of relevant values if the computation converges. Four diagrams have to commute, but the most important one is the following:



It says that if we transform a value-computation pair, where the computation is a value, into the computation of a pair of values, then we get a value.

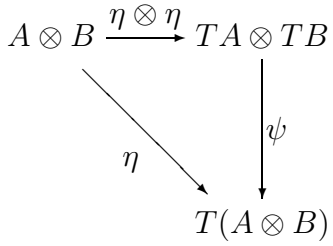
We also need to be able to transform a pair of computations into a computation of a pair of values. Following [Mog89], a natural transformation

$$\psi : T(-) \otimes T(+) \to T(- \otimes +)$$

is induced by the strength t as follows:

$$\psi_{A,B} = (\cong; t_{TB,A}; T(\cong); T(t_{A,B}); \mu_{A \otimes B})$$

Where $\cong$ is the "symmetric" natural isomorphism. The intuition is that ψ first evaluates the first component, if the first computation converges, it evaluates the second component, and if the second computation also converge then it returns the pair of relevant values. It makes the following diagram commute:



It says that if both computations are values, then the resulting computation is a value too.

4.4 Two examples, the $(\otimes - I)$ and $(\otimes - E)$ rules

According to previous assumptions, we are dealing with a symmetric monoidal category $(\mathcal{C}, I, \otimes)$ with a monad (T, η, μ) and a strength t on it. This is enough machinery to give

an example of an interpretation of a rule. We will interpret the logical connective $\otimes$ as the tensor product given by the monoidal structure, that is, $[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$.

Warning: In what follows, the symbol $\otimes$ can be interpreted in three different ways: as a functor, as a logical connective, and as part of the syntax for terms.

Now, the rule for introduction of $\otimes$ looks like this:

$$\frac{\Gamma \vdash u : A \qquad \Delta \vdash v : B}{\Gamma, \Delta \vdash u \otimes v : A \otimes B}\,(\otimes - I)$$

and should give rise to an operation on arrows:

$$\frac{[\![\Gamma]\!] \xrightarrow{[\![u]\!]} T[\![A]\!] \qquad [\![\Delta]\!] \xrightarrow{[\![v]\!]} T[\![B]\!]}{[\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{[\![u \otimes v]\!]} T[\![A \otimes B]\!]}$$

Hence, we define:

$$[\![u \otimes v]\!] = [[\![\Gamma]\!] \otimes [\![\Delta]\!] \xrightarrow{[\![u]\!] \otimes [\![v]\!]} T[\![A]\!] \otimes T[\![B]\!] \xrightarrow{\psi} T([\![A]\!] \otimes [\![B]\!])]$$

It is easy to see that this corresponds to the appropriate rule of the operational semantics:

$$\frac{u \longrightarrow c \qquad v \longrightarrow d}{u \otimes v \longrightarrow c \otimes d}$$

If we want to evaluate the two computations, u and v, then we evaluate both terms, and if they both converge, the result is the two results paired together. This strategy is used in the categorical as well as in the operational semantics. Now, the rule for elimination of $\otimes$:

$$\frac{\Lambda \vdash w : A \otimes B \qquad \Gamma, x : A, y : B \vdash u : C}{\Gamma, \Lambda \vdash \text{let } w \text{ be } x \otimes y \text{ in } u : C}$$

induces the following operation on arrows:

$$\frac{[\![\Lambda]\!] \xrightarrow{[\![w]\!]} T[\![A \otimes B]\!] \qquad [\![\Gamma]\!] \otimes [\![A]\!] \otimes [\![B]\!] \xrightarrow{[\![u]\!]} T[\![C]\!]}{[\![\Gamma]\!] \otimes [\![\Lambda]\!] \xrightarrow{Id \otimes [\![w]\!]} [\![\Gamma]\!] \otimes T([\![A]\!] \otimes [\![B]\!]) \xrightarrow{t} T([\![\Gamma]\!] \otimes [\![A]\!] \otimes [\![B]\!]) \xrightarrow{kleisli([\![u]\!])} T[\![C]\!]}$$

Again, it is easy to see that this corresponds to the appropriate rule of the operational semantics:

$$\frac{w \longrightarrow d \otimes e \qquad u[d/x, e/y] \longrightarrow c}{\text{let } w \text{ be } x \otimes y \text{ in } u \longrightarrow c}$$

We first evaluate w, and if it converges, it gives a pair of values. We then run u with these values as the input. Again, we see that the same strategy is used in the categorical as well as in the operational semantics. Recall that $kleisli([\![u]\!])$ is the extension of $[\![u]\!]$ which first evaluates the computation given as input, and then applies $[\![u]\!]$ to the resulting value. Note that we need the natural transformation t to be able to "move" a tuple of parameters (that is: values) "inside" a computation.

5 Categorical semantics of LTS$_{+Rec}$

5.1 Introduction to the category $\mathcal{O}$

Before stating the definition of a categorical model for LTS$_{+Rec}$, we need to know $\mathcal{O}$, the category of CPOs and continuous functions. A CPO is a partial order where every increasing chain $\{f_n\}_{n\in\omega}$ has a join $\bigsqcup_{n\in\omega} f_n$. Note that we do not assume the existence of a bottom element. A continuous function between CPOs is a monotone function that preserves joins of increasing chains. The category $\mathcal{O}$ is cartesian closed; the finite products $(1, \times)$ are induced by the usual structure on partial orders, and the exponential object $\Rightarrow$ is the set of continuous functions equipped with the pointwise order. We also have finite sums $(0, +)$ induced by the usual structure on partial orders. The usual lift construction on partial orders induces a strong monad $((-)_\perp, lift, down, u)$ on $(\mathcal{O}, \times, 1)$, see Appendix G. Note that a cartesian category also is a monoidal category.

Definition 5.1 *If f is a function between partial orders, we will say that it* reflects the order *iff $f(x) \leq f(y)$ implies $x \leq y$ whenever x and y are elements in the domain of f.*

5.2 Definition of the categorical semantics

In this part we will state the necessary machinery to interpret LTS$_{+Rec}$. To give a categorical semantics to the rule for recursion, we will assume that some of our constructions are $\mathcal{O}$-enriched. An $\mathcal{O}$-category is a category where each hom-set has CPO structure such that composition is continuous. An $\mathcal{O}$-functor between $\mathcal{O}$-categories is a functor between the underlying categories which is continuous on each hom-set. Other notions from category theory can be defined similarly in an $\mathcal{O}$-enriched setting, but we shall not need it here. See [Poi92] for an introduction to enriched category theory.

Proposition 5.2 *An $\mathcal{O}$-category $\mathcal{C}$ induce the functor $hom(I, -) : \mathcal{C} \to \mathcal{O}$. If $\mathcal{C}$ moreover is equipped with a monoidal structure $(I, \otimes)$ where $\otimes$ is an $\mathcal{O}$-functor, then a monoidal structure on the functor $hom(I, -)$ is induced by the map $n_1 : 1 \to hom(I, I)$ and the natural transformation $n : hom(I, -) \times hom(I, +) \longrightarrow hom(I, - \otimes +)$ defined as $n_1(*) = Id_I$ and $n_{A,B}(f, g) = (\cong; (f \otimes g))$ respectively.*

Remark: When assuming the functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ to be an $\mathcal{O}$-functor, we are implicitly assuming $\mathcal{C} \times \mathcal{C}$ to have the obvious $\mathcal{O}$-enrichment induced by the $\mathcal{O}$-enrichment of $\mathcal{C}$.

Remark: The functor $hom(I, -)$ should be thought of as a functor that forgets all structure on an object except the CPO structure on its points. It is easy to see that $hom(I, -)$ is monotone on hom-sets when the maps in $\mathcal{O}$ are ordered pointwise; this corresponds to the ordering of maps in $\mathcal{C}$ being included in the pointwise ordering w.r.t. the partial order on points. Note that $hom(I, -)$ does not necessarily reflect the order on hom-sets. This is for example not the case with the $predI_a$ model where the ordering on hom-sets is the so-called stable order. If $hom(I, -)$ *does* reflect the order, then the ordering of maps in $\mathcal{C}$ is the pointwise ordering.

Definition 5.3 *A categorical model for LTS$_{+Rec}$ is an $\mathcal{O}$-category $\mathcal{C}$ equipped with:*

1. *a symmetric monoidal closed structure* $(I, \otimes, \multimap)$ *where* $\otimes$ *is an* $\mathcal{O}$*-functor s.t. each* $n_{A,B}$ *induced by the monoidal structure reflects the order*

2. *a symmetric monoidal comonad* $(!, \varepsilon, \delta, m_I, m)$ *where* $!$ *is an* $\mathcal{O}$*-functor*

3. *monoidal natural transformations* $e : !(-) \to I$ *and* $d : !(-) \to !(-) \otimes !(-)$

4. *binary product* $\times$ *and sum* $+$ *which both are preserved by* $hom(I, -)$

5. *a strong monad* (T, η, μ, t) *and an isomorphism* $\sigma : hom(I, -)_\perp \to hom(I, T(-))$ *making* $(hom(I, -), \sigma)$ *a functor of strong monads*

6. *a bottom element* $\perp_{A,B}$ *in every* $hom(A, TB)$ *s.t.* $h; \perp_{A,B} = \perp_{I,B}$ *for every* $h : I \to A$

Remark: The assumption that the comonad is symmetric monoidal means that $!$ is a symmetric monoidal functor and ε and δ are monoidal natural transformations, see Appendix G. When assuming the natural transformations e and d to be monoidal, we are implicitly assuming the functors I and $!(-) \otimes !(-)$ to have the obvious monoidal structure induced by the monoidal structure on $!$.

Remark: Since $hom(I, -)$ should be thought of as a functor that forgets all structure on an object except the CPO structure on its points, then a property saying that $hom(I, -)$ respect some structure present on both $\mathcal{C}$ and $\mathcal{O}$ should be thought of as a property saying that the structure on $\mathcal{C}$ behaves as the corresponding structure on $\mathcal{O}$ when only CPO structure on points is considered. The way in which the structure on $\mathcal{C}$ behaves as the structure on $\mathcal{O}$ is determined by the property of $hom(I, -)$. The requirement that $hom(I, -)$ preserves products says that the points $hom(I, A \times B)$ of the product of two objects is isomorphic to the product $hom(I, A) \times hom(I, B)$ of the points $hom(I, A)$ and $hom(I, B)$ of the two objects. Similarly for sum. Condition 6 can also be stated in terms of a preservation property of the $hom(I, -)$ functor; it is equivalent to commutativity of the following diagram:

$$\begin{array}{ccc} hom(I, A) & & \\ \downarrow{\scriptstyle \perp} & \searrow{\scriptstyle hom(I, \perp_{A,B})} & \\ hom(I, B)_\perp & \xrightarrow{\sigma_B} & hom(I, TB) \end{array}$$

where $\perp$: $hom(I, A) \longrightarrow hom(I, B)_\perp$ is the bottom element of $hom_{\mathcal{O}}(hom(I, A), hom(I, B)_\perp)$ that sends every element in $hom(I, A)$ to the bottom element of $hom(I, B)_\perp$. The requirement of an isomorphism σ making $(hom(I, -), \sigma)$ a functor of strong monads from the strong monad (T, η, μ, t) on $(\mathcal{C}, I, \otimes)$ to the strong monad $((-)_\perp, lift, down, u)$ on $(\mathcal{O}, \times, 1)$ say that the strong monad T behaves like the strong monad $(-)_\perp$.

Definition 5.4 *We can now define a generalised cokleisli operator* γ*:*

$$\gamma : hom(!A_1 \otimes ... \otimes !A_n, B) \longrightarrow hom(!A_1 \otimes ... \otimes !A_n, !B)$$
$$\gamma(f) = [!A_1 \otimes ... \otimes !A_n \xrightarrow{\delta_{A_1} \otimes ... \otimes \delta_{A_n}} !!A_1 \otimes ... \otimes !!A_n \xrightarrow{m_{A_1,...,A_n}} !(!A_1 \otimes ... \otimes !A_n) \xrightarrow{!f} !B]$$

Note that in case $n = 0$ then $\gamma(f) = m_I; !f$. In case $n = 1$ this definition is consistent with the usual cokleisli operator. It should be mentioned that the definition of γ is due to [BBdPH92]. Note that the definition of γ is unrelated to the product structure. In [See89], another generalised cokleisli operator is used which is related to the product structure.

Proposition 5.5 *If $f : !A_1 \otimes ... \otimes !A_n \to B$, then $\gamma(f); \varepsilon_B = f$.*

Proof. Straightforward calculation. □

Proposition 5.6 *If $f_i : I \to A_i$ for $i \in \{1, ..., n\}$, and $h : !A_1 \otimes ... \otimes !A_n \to B$, then*

$$[I \cong I \otimes ... \otimes I \xrightarrow{\gamma(f_1)\otimes ...\otimes\gamma(f_n)} !A_1 \otimes ... \otimes !A_n \xrightarrow{\gamma(h)} !B] =$$
$$\gamma([I \cong I \otimes ... \otimes I \xrightarrow{\gamma(f_1)\otimes ...\otimes\gamma(f_n)} !A_1 \otimes ... \otimes !A_n \xrightarrow{h} B])$$

Proof. Induction in n. □

Proposition 5.7 *If $f : I \to A$, then the following diagram commutes:*

$$\begin{array}{ccc} I & \xrightarrow{\gamma(f)} & !A \\ \cong \downarrow & & \downarrow d_A \\ I \otimes I & \xrightarrow{\gamma(f)\otimes\gamma(f)} & !A \otimes !A \end{array}$$

Proof. Straightforward calculation, where we use monoidality of d. □

Proposition 5.8 *$(A + B) \otimes C$ is naturally isomorphic to $(A \otimes C) + (B \otimes C)$.*

Proof. The functor $(-) \otimes C$ has a right adjoint, wherefore it preserve sums. The isomorphism can be shown to be natural in A, B and C. □

Definition 5.9 *Given $f : !A_1 \otimes ... \otimes !A_n \otimes !TB \to TB$, define $f^\sharp : !A_1 \otimes ... \otimes !A_n \to TB$ as*

$$f^\sharp = \bigsqcup_{n\in\omega} \Phi_f^n(\bot)$$

where the continuous function $\Phi_f : hom(!A_1 \otimes ... \otimes !A_n, TB) \longrightarrow hom(!A_1 \otimes ... \otimes !A_n, TB)$ is defined as follows:

$$\Phi_f(h) =$$
$$[!A_1 \otimes ... \otimes !A_n \xrightarrow{D} !A_1 \otimes ... \otimes !A_n \otimes !A_1 \otimes ... \otimes !A_n \xrightarrow{Id\otimes\gamma(h)} !A_1 \otimes ... \otimes !A_n \otimes !TB \xrightarrow{f} TB]$$

where D is a generalisation of the natural transformation d, that is:

$$D_{A_1,...,A_n} =$$
$$[!A_1 \otimes ... \otimes !A_n \xrightarrow{d_{A_1}\otimes ...\otimes d_{A_n}} !A_1 \otimes !A_1 \otimes ... \otimes !A_n \otimes !A_n \xrightarrow{\cong} !A_1 \otimes ... \otimes !A_n \otimes !A_1 \otimes ... \otimes !A_n]$$

The operator $(-)^\sharp$ will be used to interpret the rule for recursion. Note that $\{\Phi_f^n(\bot)\}_{n\in\omega}$ is an increasing chain in $hom(!A_1 \otimes ... \otimes !A_n, TB)$. It follows from the usual fixpoint theorem for CPOs that $f^\sharp$ is well defined and equal to a uniquely determined least solution to the equation $x = \Phi_f(x)$. A solution to this equation is actually what in a later section of this paper will be called a linear fixpoint of the map f.

Definition 5.10 *The interpretation of formulas are defined by induction as follows:*

- $[\![1]\!] = I$
- $[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$
- $[\![A \multimap B]\!] = [\![A]\!] \multimap T[\![B]\!]$
- $[\![A \& B]\!] = T[\![A]\!] \times T[\![B]\!]$
- $[\![A \oplus B]\!] = [\![A]\!] + [\![B]\!]$
- $[\![!A]\!] = !T[\![A]\!]$

Note how the interpretation of $\multimap$ reflects the call-by-value parameter passing of the operational semantics. A term of type $A \multimap B$ expects a value of type A and *computes* a value of type B. Therefore the interpretation of the type is $[\![A]\!] \multimap T[\![B]\!]$.

Definition 5.11 *Given a derivation of the sequent*

$$x_1 : A_1, ..., x_n : A_n \vdash u : A$$

we inductively define a map

$$[\![A_1]\!] \otimes ... \otimes [\![A_n]\!] \xrightarrow{[\![u]\!]} T[\![A]\!]$$

cf. the operations on arrows (corresponding to the typing rules) given in Appendix C.

As shown earlier, the proof is unique up to applications of the (*Exchange*) rule. Given a derivable sequent, the interpretation is therefore uniquely determined.

5.3 Properties of the categorical semantics

I will now sum up what we can obtain with the machinery defined. We will consider naturality of the operation on arrows corresponding to recursion as naturality in the interpretation of $\Gamma_1, ..., \Gamma_n$.

Proposition 5.12 *The typing rules induce operations on arrows which are natural in the interpretation of the unchanged components of the sequents.*

Proof. Check each rule. □

This result gives us an extension of the Substitution Property essentially saying that substitution corresponds to composition. This is necessary to deal with substitutions in the operational semantics:

Lemma 5.13 *(Substitution Lemma) If* $\Gamma \vdash u : A$ *and* $\Delta, x : A, \Lambda \vdash v : B$ *both are derivable s.t. the variables in* Γ *and* Δ, Λ *are pairwise distinct, and there exists a map* $h : [\![\Gamma]\!] \to [\![A]\!]$ *with the property that* $[\![u]\!] = h; \eta$*, then* $\Delta, \Gamma, \Lambda \vdash v[u/x] : B$ *is derivable too with the interpretation*

$$[[\![\Delta]\!] \otimes [\![\Gamma]\!] \otimes [\![\Lambda]\!] \xrightarrow{Id \otimes h \otimes Id} [\![\Delta]\!] \otimes [\![A]\!] \otimes [\![\Lambda]\!] \xrightarrow{[\![v]\!]} T[\![B]\!]]$$

Proof. The theorem is proved by induction in the derivation of $\Delta, x : A, \Lambda \vdash v : B$. All cases except (Id) are covered by the naturality property given by the previous theorem. The (Id) case is trivial. □

Lemma 5.14 *Given a canonical term* $\vdash c : A$*, then* $[\![c]\!] \Downarrow$*.*

Proof. Induction in the derivation of $\vdash c : A$. □

Lemma 5.15 *Given a canonical term* $\vdash \text{let } c_1, ..., c_n \text{ be } x_1, ..., x_n \text{ in } !u$*, then*

$$[\![\text{let } c_1, ..., c_n \text{ be } x_1, ..., x_n \text{ in } !u]\!] = \gamma(h); \eta$$

for some $h : I \to TA$*.*

Proof. Induction in the derivation of $\vdash \text{let } c_1, ..., c_n \text{ be } x_1, ..., x_n \text{ in } !u : !A$. □

Theorem 5.16 *(Soundness Theorem) Given a closed term* u *s.t.* $u \longrightarrow c$*, then* $[\![u]\!] = [\![c]\!]$*.*

Proof. See Appendix E.

To prove the "if" part of the Adequacy Theorem, we will use the fact that a fixpoint is calculated in a certain way, namely as the join of a certain increasing chain in a hom-set. We can then use the technique of *Logical Relations* to prove a result, which has the "if" part of the Adequacy Theorem as a special case. Logical Relations relate the categorical interpretation of a term to its operational behaviour.

Definition 5.17 *(Logical Relations) Let* T_A *be the set of closed typable terms of type* A*, and* C_A *the set of canonical terms of type* A*. We define the relations*

$$\lesssim^{\circ}_A \subseteq hom(I, [\![A]\!]) \times C_A \qquad\qquad \lesssim_A \subseteq hom(I, T[\![A]\!]) \times T_A$$

by induction in the type. The relation $\lesssim^{\circ}_A$ *is defined in terms of relations corresponding to smaller types as follows:*

- $Id_I \lesssim^{\circ}_1 *$
- $f \lesssim^{\circ}_{A \otimes B} d \otimes e$ *iff*
  $\exists g \in hom(I, [\![A]\!]). \exists h \in hom(I, [\![B]\!]).$
  $f = [I \cong I \otimes I \xrightarrow{g \otimes h} [\![A]\!] \otimes [\![B]\!]] \quad \& \quad g \lesssim^{\circ}_A d \quad \& \quad h \lesssim^{\circ}_B e$
- $f \lesssim^{\circ}_{A \multimap B} \lambda x.u$ *iff*
  $\forall g \in hom(I, [\![A]\!]). \forall c \in C_A.$
  $g \lesssim^{\circ}_A c \quad \Rightarrow \quad [I \cong I \otimes I \xrightarrow{f \otimes g} ([\![A]\!] \multimap T[\![B]\!]) \otimes [\![A]\!] \xrightarrow{eval} T[\![B]\!]] \lesssim_B u[c/x]$

- $f \lesssim^{\circ}_{A\&B} (u,v)$ *iff*
  $\exists g \in hom(I, T[\![A]\!]).\exists h \in hom(I, T[\![B]\!]).\quad f =< g,h > \quad \& \quad g \lesssim_A u \quad \& \quad h \lesssim_B v$
- $f \lesssim^{\circ}_{A\oplus B} inl(c)$ *iff*
  $\exists g \in hom(I, [\![A]\!]).\quad f = g; inj_1 \quad \& \quad g \lesssim^{\circ}_A c$
- $f \lesssim^{\circ}_{A\oplus B} inr(d)$ *iff*
  $\exists h \in hom(I, [\![B]\!]).\quad f = h; inj_2 \quad \& \quad h \lesssim^{\circ}_B d$
- $f \lesssim^{\circ}_{!A}$ *let* $c_1, ..., c_n$ *be* $x_1, ..., x_n$ *in* $!u$ *iff*
  $\exists g \in hom(I, T[\![A]\!]).\quad f = \gamma(g) \quad \& \quad g \lesssim_A u[c_1/x_1, ..., c_n/x_n]$

The relation $\lesssim_A$ *is defined in terms of* $\lesssim^{\circ}_A$ *as follows:*

- $f \lesssim_A u$ *iff*
  $\forall g \in hom(I, [\![A]\!]).\quad f = g; \eta \quad \Rightarrow \quad \exists c \in C_A.\quad u \longrightarrow c \quad \& \quad g \lesssim^{\circ}_A c$

We are now in a position to prove that the predicate $(-) \lesssim_A u$ is *inclusive* for any term $\vdash u : A$. This amounts to the following lemma:

Lemma 5.18 *(Inclusiveness Lemma) Let a term* $\vdash u : A$ *be given. If* $\{f_n\}_{n\in\omega}$ *is an increasing chain in* $hom(I, TA)$ *s.t.* $f_n \lesssim_A u$ *for every* n*, then* $\bigsqcup_{n\in\omega} f_n \lesssim_A u$*.*

Proof. Induction in the type. □

Lemma 5.19 *(Approximation Lemma) Let* $x_1 : A_1, ..., x_n : A_n \vdash u : C$ *be given. Let* J *be a finite set s.t. for all* $j \in J$ *we are given* $\vdash v_j : B_j$ *and* $f_j \in hom(I, [\![B_j]\!])$ *with the property that* $f_j; \eta \lesssim_{B_j} v_j$*. Moreover, assume that for each* j *we have a variable* z_j*, s.t. for all* $i, k \in J$ *we have* $i \neq k \Rightarrow z_i \neq z_k$*, s.t.* $\{x_1, ..., x_n\} \subseteq \{z_j |\, j \in J\}$*, and s.t. for all* $r \in \{1, ..., n\}$ *and* $j \in J$ *we have* $x_r = z_j \Rightarrow A_r = B_j$*. We then have:*

$$[I \cong I \otimes ... \otimes I \xrightarrow{f_{j_1} \otimes ... \otimes f_{j_n}} [\![A_1]\!] \otimes ... \otimes [\![A_n]\!] \xrightarrow{[\![u]\!]} T[\![C]\!] \lesssim_C u[..., v_j/z_j, ...]$$

where for all r *in* $\{1, ..., n\}$ *we have chosen* j_r *s.t.* $x_r = z_{j_r}$*.*

Proof. See Appendix F.

Theorem 5.20 *(Adequacy Theorem) Given a closed term* u*, then* $u \downarrow$ *iff* $[\![u]\!] \Downarrow$*.*

Proof. The interpretation of a canonical term converges semantically, which together with Soundness gives the "only if" part. The "if" part is a special case of the Approximation Lemma. □

6 Fixpoints in a linear context

In the categorical semantics of LTS_{+Rec} we used the operator $(-)^\sharp$ to interpret the rule for recursion. The map $f^\sharp$ was defined to be a certain solution to the equation $x = \Phi_f(x)$. The previous section showed that this was an appropriate interpretation of the rule for recursion in the sense that we get soundnes and adequacy results. Solutions to the equation $x = \Phi_f(x)$ are what we in this section will call linear fixpoints, which we will characterise in terms of ordinary fixpoints.

6.1 Fixpoints as usual

The definitions and results concerning fixpoints and fixpoint operators in this subsection can also be found in [Poi92]. To start things off, we will state a definition of fixpoints in a category with finite products. In what follows, $\Delta_A : A \to A \times A$ is the diagonal map.

Definition 6.1 *A category $\mathcal{C}$ with finite products* has fixpoints *iff for every map $f : A \times B \to B$ there exists a specified* fixpoint $f^\dagger : A \to B$ *with the property that*

$$f^\dagger = [A \xrightarrow{\Delta_A} A \times A \xrightarrow{Id \times f^\dagger} A \times B \xrightarrow{f} B]$$

Note how the diagonal map is used to copy parameters. We can deal with fixpoint operators if the category is closed w.r.t. the product structure:

Definition 6.2 *A cartesian closed category $\mathcal{C}$* has fixpoint operators *iff for every object B there is an arrow $Y_B : [B \Rightarrow B] \to B$ with the property that for every $f : A \times B \to B$ the map $curry(f); Y_B$ is a fixpoint of f.*

Fixpoints and fixpoint operators are related according to the following result:

Proposition 6.3 *A cartesian closed category $\mathcal{C}$ has fixpoints iff it has fixpoint operators.*

Proof. Define $Y_B = eval^\dagger$, and conversely, given $f : A \times B \to B$, define $f^\dagger = curry(f); Y_B$. □

6.2 Linear fixpoints

I will now consider fixpoints in a linear context. The paper [Bra] deals with this topic from a proof theoretic point of view. We can not use the previous definition of fixpoints because it assumes the presence of finite products.

Definition 6.4 *Let $(\mathcal{C}, I, \otimes)$ be a monoidal category equipped with a comonad $(!, \varepsilon, \delta)$, and with a natural transformation $d : !(-) \to !(-) \otimes !(-)$. We say that $\mathcal{C}$* has linear fixpoints *iff for every map $f : !A \otimes !B \to B$ there exists a specified* linear fixpoint $f^\sharp : !A \to B$ *with the property that*

$$f^\sharp = [!A \xrightarrow{d_A} !A \otimes !A \xrightarrow{Id \otimes \gamma(f^\sharp)} !A \otimes !B \xrightarrow{f} B]$$

It is simply an extension of the definition of fixpoints in a category with finite products to a linear context, where we have only a "diagonal map" d_A for objects of the shape $!A$. We can deal with linear fixpoint operators if our category is closed w.r.t. the monoidal structure:

Definition 6.5 *Let $(\mathcal{C}, I, \otimes, \multimap)$ be a monoidal closed category equipped with a comonad $(!, \varepsilon, \delta)$, and with a natural transformation $d : !(-) \to !(-) \otimes !(-)$. We say that $\mathcal{C}$* has linear fixpoint operators *iff for every object B there is an arrow $Y_B^{lin} : !(!B \multimap B) \to B$ with the property that for every $f : !A \otimes !B \to B$ the map $\gamma(curry(f)); Y_B^{lin}$ is a linear fixpoint of f.*

Linear fixpoints and linear fixpoint operators are under appropriate circumstances related in a way analogous to the way fixpoints are related to fixpoint operators. In what follows, we need the notion of a categorical model of multiplicative ILL as defined in [BBdPH92]:

Definition 6.6 *A* linear category *is a symmetric monoidal closed category* $(\mathcal{C}, I, \otimes, \multimap)$ *equipped with:*

- *A symmetric monoidal comonad* $(!, \varepsilon, \delta, m_I, m)$.
- *Monoidal natural transformations* $e : !(-) \to I$ *and* $d : !(-) \to !(-) \otimes !(-)$ *such that*
  1. e_A *and* d_A *are maps of coalgebras,*
  2. e_A *and* d_A *give the free coalgebra* $(!A, \delta)$ *structure of a cocommutative comonoid,*
  3. *maps between free coalgebras are maps between cocommutative comonoids.*

Remark: The assumption that the comonad is symmetric monoidal means that ! is a symmetric monoidal functor and ε and δ are monoidal natural transformations. When assuming the natural transformations e and d to be monoidal, we are assuming the functors I and $!(-)\otimes!(-)$ to have the obvious monoidal structure induced by the monoidal structure on !. It can be shown that (I, m_I) and $(!A \otimes !A, (\delta_A \otimes \delta_A); m_{!A,!A})$ are coalgebras. The assumption that e_A is a map of coalgebras amounts to e_A being a map from $(!A, \delta_A)$ to (I, m_I), and the assumption that d_A is a map of coalgebras amounts to d_A being a map from $(!A, \delta_A)$ to $(!A \otimes !A, (\delta_A \otimes \delta_A); m_{!A,!A})$.

If we impose this extra structure on our category we get the following result:

Proposition 6.7 *A linear category has linear fixpoints iff it has linear fixpoint operators.*

Proof. Define $Y_B^{lin} = ((\varepsilon_{!B \multimap B} \otimes Id); eval)^\sharp$, and conversely, given $f : !A \otimes !B \to B$, define $f^\sharp = \gamma(curry(f)); Y_B^{lin}$. $\square$

Now, the definition of linear fixpoints can be explained in terms of fixpoints in the category of free coalgebras. Given a category $\mathcal{C}$ equipped with a comonad $(!, \varepsilon, \delta)$, the coEilenberg-Moore category, $\mathcal{C}^!$ is the category of coalgebras, and the category of free coalgebras is the full subcategory of $\mathcal{C}^!$, whose objects are free coalgebras, that is, coalgebras of the type $(!A, \delta)$. Recall that we have an adjunction $U^! \dashv F^!$ between $\mathcal{C}^!$ and $\mathcal{C}$. The forgetful functor $U^! : \mathcal{C}^! \to \mathcal{C}$ simply forgets the coalgebra structure, while the free functor $F^! : \mathcal{C} \to \mathcal{C}^!$ takes an object A to the free coalgebra $(!A, \delta)$. The adjunction induces the following natural bijection between maps:

$$\phi_{(C,h),A} : hom_{\mathcal{C}^!}((C, h), (!A, \delta)) \cong hom_{\mathcal{C}}(C, A)$$

where (C, h) is a coalgebra, and A is an object of $\mathcal{C}$. The bijection is given by $\phi(f) = f; \varepsilon_A : C \to A$ and $\phi^{-1}(g) = h; !g : (C, h) \to (!A, \delta)$.

In [Bie93] it is shown that $\mathcal{C}^!$ w.r.t. a symmetric monoidal category $(\mathcal{C}, I, \otimes)$ equipped with a symmetric monoidal comonad $(!, \varepsilon, \delta, m_I, m)$ has an induced symmetric monoidal structure: the unit of the tensor product is given by (I, m_I), and given two coalgebras (A, k) and (B, h), their tensor product $(A, k)\Box(B, h)$ is the coalgebra $(A \otimes B, (k \otimes h); m_{A,B})$. If moreover the category is a linear category (not necessarily with $\multimap$, that is, $(-) \otimes A$ does not necessarily have a right adjoint $A \multimap (-)$), then the symmetric monoidal structure

on $\mathcal{C}^!$ is a finite product structure, that is, (I, m_I) is a terminal object, and $\Box$ is a binary product.

Theorem 6.8 *Let $\mathcal{C}$ be a linear category (not necessarily with $\multimap$), then $h : (!A, \delta) \to (!B, \delta)$ is a fixpoint of $f : (!A, \delta)\Box(!B, \delta) \to (!B, \delta)$ iff $\phi(h) : !A \to B$ is a linear fixpoint of $\phi(f) : !A \otimes !B \to B$.*

Proof. Calculation. □

It is obvious that if the the category of free coalgebras is closed under finite products in $\mathcal{C}^!$, that is, the terminal object (I, m_I) is isomorphic to a free coalgebra, and given two free coalgebras $(!A, \delta)$ and $(!B, \delta)$, their tensor product $(A, k)\Box(B, h)$ is isomorphic to a free coalgebra, then it inherits the finite products from the ambient category. This leads to the following result:

Corollary 6.9 *If $\mathcal{C}$ is a linear category (not necessarily with $\multimap$) s.t. the category of free coalgebras is closed under finite products in $\mathcal{C}^!$, then the category of free coalgebras has fixpoints iff $\mathcal{C}$ has linear fixpoints.*

Proof. A straightforward consequence of the theorem. □

If the category of free coalgebras w.r.t. a linear category is closed under finite products in $\mathcal{C}^!$ then it has finite products, as mentioned above. In [Bie93] it is shown that the category of free coalgebras moreover is cartesian closed; given two free coalgebras $(!A, \delta)$ and $(!B, \delta)$, their exponential object $(!A, \delta) \Rightarrow (!B, \delta)$ is given by the free coalgebra $(!(!A \multimap B), \delta)$. This leads to the following result:

Theorem 6.10 *If $\mathcal{C}$ is a linear category s.t. the category of free coalgebras is closed under finite products in $\mathcal{C}^!$, then the category of free coalgebras has fixpoint operators iff $\mathcal{C}$ has linear fixpoint operators.*

Proof. Follows from the previous results. □

This result can also be derived more explicitly, namely as a straightforward consequence of the following theorem:

Theorem 6.11 *If $\mathcal{C}$ is a linear category s.t. the category of free coalgebras is closed under finite products in $\mathcal{C}^!$, then $Y_{(!B,\delta)} : (!B, \delta) \Rightarrow (!B, \delta) \to (!B, \delta)$ is a fixpoint operator in the category of free coalgebras iff $\phi(Y_{(!B,\delta)}) : !(!B \multimap B) \to B$ is a linear fixpoint operator in $\mathcal{C}$.*

Proof. Calculation. □

Now, under which circumstances is the category of free coalgebras closed under finite products? The following observation induces a sufficient condition:

Proposition 6.12 *Let $\mathcal{C}$ be a category equipped with a comonad $(!, \varepsilon, \delta)$. If $\mathcal{C}$ has terminal object 1 then $(!1, \delta)$ is a terminal object in $\mathcal{C}^!$, and if $\mathcal{C}$ has binary product $\times$ then $(!(A \times B), \delta)$ is a binary product of $(!A, \delta)$ and $(!B, \delta)$ in $\mathcal{C}^!$.*

Proof. The free functor $F^! : \mathcal{C} \to \mathcal{C}^!$ is right adjoint to $U^! : \mathcal{C}^! \to \mathcal{C}$. Right adjoints preserve finite products, so if $\mathcal{C}$ has terminal object 1 then 1 is sent into a terminal object $(!1, \delta)$ in $\mathcal{C}^!$, and if $\mathcal{C}$ has binary product $\times$, then a product diagram $A \leftarrow A \times B \rightarrow B$ in $\mathcal{C}$ is sent into a product diagram $(!A, \delta) \leftarrow (!(A \times B), \delta) \rightarrow (!B, \delta)$ in $\mathcal{C}^!$. $\Box$

This has the consequence that if $\mathcal{C}$ is a linear category with finite products, then the category of free coalgebras is closed under finite products.

Moreover, since both (I, m_I) and $(!1, \delta)$ are terminal objects in $\mathcal{C}^!$, I is isomorphic to $!1$, and analogously, since both $(!A \otimes !B, (\delta \otimes \delta); m_{!A,!B})$ and $(!(A \times B), \delta)$ are products of $(!A, \delta)$ and $(!B, \delta)$ in $\mathcal{C}^!$, $!A \otimes !B$ is isomorphic to $!(A \times B)$ such that the isomorphism is natural in A and B. Thus we can define a model of ILL as described in [See89]. Calculations show that the way the isomorphisms are defined and the universal property of (I, m_I) and $(!A \otimes !B, (\delta \otimes \delta); m_{!A,!B})$ forces ! to take the cocommutative comonoid structure w.r.t. the finite products to the cocommutative comonoid structure w.r.t. the symmetric monoidal structure, that is:

$$e_A = [!A \xrightarrow{!<>_A} !1 \cong I] \qquad\qquad d_A = [!A \xrightarrow{!\Delta_A} !(A \times A) \cong !A \otimes !A]$$

Note that the category of free coalgebras is equivalent to the coKleisli category: It is straightforward to check that the comparison functor from $\mathcal{C}_!$ to $\mathcal{C}^!$ is an equivalence of categories when considered as a functor from $\mathcal{C}_!$ to the category of free coalgebras.

6.3 Generalisation of linear fixpoints

The definition of linear fixpoints can be generalised to an arbitrary number of parameters such that it "fits" the definition of recursion in a linear context:

Definition 6.13 *(Generalisation) Let $(\mathcal{C}, I, \otimes)$ be a symmetric monoidal category equipped with a symmetric monoidal comonad $(!, \varepsilon, \delta, m_I, m)$ and a natural transformation $d : !(-) \rightarrow !(-) \otimes !(-)$. We say that $\mathcal{C}$* has linear fixpoints *iff for every map $f : !A_1 \otimes ... \otimes !A_n \otimes !B \rightarrow B$ there exists a specified* linear fixpoint $f^\sharp : !A_1 \otimes ... \otimes !A_n \rightarrow B$ *with the property that:*

$$f^\sharp = [!A_1 \otimes ... \otimes !A_n \xrightarrow{D} !A_1 \otimes ... \otimes !A_n \otimes !A_1 \otimes ... \otimes !A_n \xrightarrow{Id \otimes \gamma(f^\sharp)} !A_1 \otimes ... \otimes !A_n \otimes !B \xrightarrow{f} B]$$

In linear categories, this generalisation is equivalent to the original definition of linear fixpoints. Note that in case $n = 1$ we get the original definition. Similarly, the definition of linear fixpoint operators can be generalised as follows:

Definition 6.14 *(Generalisation) Let $(\mathcal{C}, I, \otimes, \multimap)$ be a symmetric monoidal closed category equipped with a symmetric monoidal comonad $(!, \varepsilon, \delta, m_I, m)$ and a natural transformation $d : !(-) \rightarrow !(-) \otimes !(-)$. We say that $\mathcal{C}$* has linear fixpoint operators *iff for every object B there is an arrow $Y_B^{lin} : !(!B \multimap B) \rightarrow B$ with the property that for every $f : !A_1 \otimes ... \otimes !A_n \otimes !B \rightarrow B$ the map $\gamma(curry(f)); Y_B^{lin}$ is a linear fixpoint of f.*

In linear categories, this generalisation is equivalent to the original definition of linear fixpoint operators. Note that in case $n = 1$ we get the original definition.

7 Concrete models

It should be mentioned that the category of coherence spaces and linear stable functions is not a model of LTS_{+Rec} as defined above. This is also the case with the category of coherence spaces and affine stable functions, where an affine function is defined as below. The problem is that a model of LTS_{+Rec} assumes the existence of an isomorphism $hom(I,A)_\perp \cong hom(I,TA)$, which amounts to the points of TA being isomorphic (as CPO's) to the usual lift construction applied to the points of A. But the lift construction applied to a coherence space is not necessarily a coherence space, thus, we can not use the mentioned categories as models of LTS_{+Rec}.

7.1 The category $predI_a$

In what follows, we will present a concrete category which is a model of LTS_{+Rec}. It is the category of pre dI domains and affine stable functions, $predI_a$. The details can be found in the paper [Bra94], so we will only give a sketch of the constructions here. Pre dI domains are defined as follows:

Let $(D,\sqsubseteq)$ be a (possible empty) poset, and assume that all non-empty finitely bounded subsets X have joins $\sqcup X$ and meets $\sqcap X$. A subset X is finitely bounded if and only if every finite subset of X has an upper bound. Note that D does not necessarily have a bottom element.

An *affine* element of D is an element d s.t. $d \sqsubseteq \sqcup X \Rightarrow \exists x \in X.d \sqsubseteq x$ for any non-empty finitely bounded subset X. We will denote the set of affine elements of D by D_a. D is called *prime algebraic* iff

$$\forall d \in D.(\{d' \in D_a | d' \sqsubseteq d\} \neq \emptyset \;\&\; d = \sqcup\{d' \in D_a | d' \sqsubseteq d)\}$$

A *finite* element of D is an element d s.t. $d \sqsubseteq \sqcup X \Rightarrow \exists x \in X.d \sqsubseteq x$ for any directed subset X. We will denote the set of finite elements of D by D_o. D is called *finitary* iff

$$\forall d \in D_o.|\{d' \in D | d' \sqsubseteq d\}| < \infty$$

A *pre dI domain* is a finitary prime algebraic domain. A *dI domain* is a pre dI domain with a bottom element.

A monotone function f is called *stable* iff $f(\sqcap X) = \sqcap\{f(x) |\, x \in X\}$ for any non-empty finitely bounded subset X. A monotone function f is called *affine* iff $f(\sqcup X) = \sqcup\{f(x) |\, x \in X\}$ for any non-empty finitely bounded subset X. An affine function f between dI domains is called *linear* iff $f(\perp) = \perp$.

The trace $Tr(f)$ of an affine stable function $f : D \to E$ is a subset of $D_a \times E_a$ defined analogously to the trace of a linear stable function between ordinary dI domains.

In what follows, $X\uparrow$ means that X has an upper bound. First of all, we need a symmetric monoidal closed structure on $predI_a$. Let D and E be pre dI domains. The tensor product of D and E is defined as follows:

$$D \otimes E = (\{t \subseteq D_a \times E_a | (\pi_1(t))\uparrow \;\&\; (\pi_2(t))\uparrow \;\&\; t \neq \emptyset \;\&\; t \text{ is down-closed } \}, \subseteq)$$

The unit I is defined to be $I = (\{\bot\}, =)$. Moreover, we define the internal-hom of D and E as follows:

$$D \multimap E = (\{Tr(f) |\ f : D \to E \text{ in } predI_a\}, \subseteq)$$

Now, we want to have a symmetric monoidal comonad on $predI_a$. We will just state how the functor ! is defined on objects. Given a pre dI domain D, we define $!D$ as follows:

$$!D = (\{t \subseteq D_o |\ t \uparrow\ \&\ t \neq \emptyset\ \&\ t \text{ is down-closed }\}, \subseteq)$$

The CPO structure on homsets is the *stable* order, that is, it is the order induced by the inclusion order on traces, and it is easy to check that the appropriate constructions enrich w.r.t. this structure. We also have to define monoidal natural transformations e and d. Given $X \subseteq D$ we define $\ulcorner d \urcorner^X = \{d' \in X | d' \sqsubseteq d\}$, and let D_m denote the set of minimal elements of D. The traces of the components are defined as follows:

$$e_D = \{(\{d\}, \bot) | d \in D_m\} \qquad d_D = \{(\ulcorner d \sqcup d' \urcorner, \ulcorner (\ulcorner d \urcorner, \ulcorner d' \urcorner) \urcorner) | d, d' \in D_o\ \&\ d \uparrow d'\}$$

Binary products and sums are defined as usual for CPOs, and the functor part of the strong monad on $predI_a$ is the usual lift functor.

Note that $predI_a$ is actually a model of Intuitionistic Affine Logic since I is a terminal object.

In the categorical semantics we used the operator $(-)^\sharp$ to interpret the rule for recursion. Given a map f is defined to be a certain solution to the equation $x = \Phi_f(x)$. The solutions to $x = \Phi_f(x)$ are exactly the linear fixpoints of f, cf. the definitions given in previous sections. Linear fixpoints is the same as fixpoints in the category of free coalgebras under the assumption that we are dealing with a linear category. $predI_a$ is actually a linear category, and the category of free coalgebras is equivalent to $coK(predI_a)$, which is isomorphic to $predI_s$, the category of pre dI domains and continuous stable functions. This category has finite sums, but according to [HP90], a cartesian closed category with fixpoints and finite sums is equivalent to the category with one object and one arrow. Thus, $predI_s$ cannot have fixpoints of arbitrary maps which entails that $predI_a$ cannot have linear fixpoints of arbitrary maps. But we do only need linear fixpoints of maps with codomain in the image T, and we do have linear fixpoints of such maps, cf. the categorical results given above since $predI_a$ is a model of LTS_{+Rec}.

Now, if we for a moment make the simplifying assumption that we only apply $(-)^\sharp$ to maps without parameters, it is easy to see that linear fixpoints in $predI_a$ are the same as fixpoints in dI_s, the category of dI domains and continuous stable functions. Notice that $f :!TB \to TB$ as well as $f^\sharp : I \to TB$ are maps in the subcategory dI_a. This category can also be shown to be a linear category with structure inherited from $predI_a$. Moreover, it has finite products, so the category of free coalgebras is closed under finite products. Again we have that the category of free coalgebras is equivalent to $coK(dI_a)$, which is isomorphic to dI_s, the category of dI domains and continuous stable functions. Hence, to calculate a linear fixpoint with the operator $(-)^\sharp$ in $predI_a$, is the same as calculating a fixpoint in dI_s.

7.2 The category $\mathcal{O}$

The category $\mathcal{O}$ is itself a model of LTS_{+Rec}. We take as the required symmetric monoidal closed structure the cartesian closed structure, and as symmetric monoidal comonad we take the identity functor equipped with appropriate identity maps as components of the required natural transformations. The ordering on arrows is the pointwise ordering, and it is easy to check that the appropriate constructions enrich w.r.t. this structure. We also have to define monoidal natural transformations e and d with components $e_A : A \to 1$ and $d_A : A \to A \times A$. As e_A we take the canonical map to the terminal object, and as d_A we take the diagonal map. Binary products and sums are defined as usual for CPOs. As strong monad we take the one induced by the lift construction on partial orders. It is obvious that $hom(I, -)$ has the wanted properties.

Note that $\mathcal{O}$ is actually a model of Intuitionistic Logic since it is cartesian closed.

Acknowledgments. I am grateful to my supervisor, Glynn Winskel, for his guidance and encouragement. Thanks to Gavin Bierman and Valeria de Paiva for comments on a draft of this paper. The diagrams and proof-rules are produced using Paul Taylor's macros.

References

[Abr90] S. Abramsky. Computational interpretations of linear logic. Technical Report 90/20, Department of Computing, Imperial College, 1990.

[BBdPH92] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. Term assignment for intuitionistic linear logic. Technical Report 262, Computer Laboratory, University of Cambridge, 1992.

[Bie93] Gavin Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Computer Laboratory, University of Cambridge, 1993.

[Bra] T. Braüner. The girard translation extended with recursion. Accepted to CSL '94, Warsaw, Poland.

[Bra94] T. Braüner. A model of intuitionistic affine logic from stable domain theory. In *ICALP '94, LNCS 820*, 1994.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[HP90] H. Huwig and A. Poigne. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73, 1990.

[Lan71] S. Mac Lane. *Categories for the Working Matematician*. Springer-Verlag, 1971.

[Mac92] I. Mackie. *Lilac : A Functional Programming Language Based on Linear Logic*. M.Sc. dissertation, Imperial College, 1992.

[Mog89] E. Moggi. Computational lambda-calculus and monads. In *4th LICS Conference*. IEEE, 1989.

[Poi92] A. Poigne. Basic category theory. In S. Abramsky et al, editor, *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[See89] R. A. G. Seely. Linear logic, $*$-autonomous categories, and cofree coalgebras. In *Contemporary Mathematics, Categories in Computer Science and Logic*, volume 92. American Mathematical Society, 89.

[Wad91] Philip Wadler. There's no substitute for linear logic. Manuscript, 1991.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages*. The MIT Press, 1993.

A Appendix, ILL in Natural Deduction style

Axiom

$$\frac{}{A \vdash A}\,(Id)$$

Structural rule

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\,(Exchange)$$

Logical rules, the $I, \otimes, \multimap$ fragment

$$\frac{}{\vdash 1}\,(1-I) \qquad \frac{\Lambda \vdash 1 \quad \Gamma \vdash A}{\Gamma, \Lambda \vdash A}\,(1-E)$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,(\otimes - I) \qquad \frac{\Lambda \vdash A \otimes B \quad \Gamma, A, B \vdash C}{\Gamma, \Lambda \vdash C}\,(\otimes - E)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,(\multimap - I) \qquad \frac{\Lambda \vdash A \multimap B \quad \Delta \vdash A}{\Lambda, \Delta \vdash B}\,(\multimap - E)$$

Logical rules, the $\&, \oplus$ fragment

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\& - I) \qquad \frac{\Lambda \vdash A \& B}{\Lambda \vdash A}\,(\& - E_1) \qquad \frac{\Lambda \vdash A \& B}{\Lambda \vdash B}\,(\& - E_2)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\,(\oplus - I_1) \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\,(\oplus - I_2) \quad \frac{\Lambda \vdash A \oplus B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, \Lambda \vdash C}\,(\oplus - E)$$

Logical rules, the ! fragment

$$\frac{\Gamma_1 \vdash !A_1 \quad ,..., \quad \Gamma_n \vdash !A_n \qquad !A_1, ..., !A_n \vdash A}{\Gamma_1, ..., \Gamma_n \vdash !A}\,(! - I) \qquad \frac{\Lambda \vdash !A}{\Lambda \vdash A}\,(Dereliction)$$

$$\frac{\Lambda \vdash !A \qquad \Gamma, !A, !A \vdash B}{\Gamma, \Lambda \vdash B}\,(Contraction) \qquad \frac{\Lambda \vdash !A \qquad \Gamma \vdash B}{\Gamma, \Lambda \vdash B}\,(Weakening)$$

B Appendix, ILL in Gentzen style

Axiom

$$\frac{}{A \vdash A}\,(Id)$$

Structural rule

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\,(Exchange)$$

Cut rule

$$\frac{\Lambda \vdash A \qquad \Delta, A \vdash C}{\Delta, \Lambda \vdash C}\,(Cut)$$

Logical rules, the $I, \otimes, \multimap$ fragment

$$\frac{}{\vdash 1}\,(1-R) \qquad \frac{\Gamma \vdash C}{\Gamma, 1 \vdash C}\,(1-L)$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,(\otimes - R) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,(\otimes - L)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,(\multimap - R) \qquad \frac{\Lambda \vdash A \qquad \Delta, B \vdash C}{\Delta, \Lambda, A \multimap B \vdash C}\,(\multimap - L)$$

Logical rules, the $\&, \oplus$ fragment

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \& B}\,(\& - R) \qquad \frac{\Lambda, A \vdash C}{\Lambda, A \& B \vdash C}\,(\& - L_1) \qquad \frac{\Lambda, B \vdash C}{\Lambda, A \& B \vdash C}\,(\& - L_2)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\,(\oplus - R_1) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\,(\oplus - R_2) \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\,(\oplus - L)$$

Logical rules, the ! fragment

$$\frac{!A_1, ..., !A_n \vdash A}{!A_1, ..., !A_n \vdash !A}\,(! - R) \qquad \frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}\,(Dereliction)$$

$$\frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}\,(Contraction) \qquad \frac{\Gamma \vdash C}{\Gamma, !A \vdash C}\,(Weakening)$$

C Appendix, LTS$_{+Rec}$ with categorical semantics

Axiom

(Id)

$$\frac{}{x : A \vdash x : A} \qquad \frac{}{A \xrightarrow{\eta} TA}$$

Structural rule

$(Exchange)$

$$\frac{\Gamma, x : A, y : B, \Delta \vdash u : C}{\Gamma, y : B, x : A, \Delta \vdash u : C} \qquad \frac{\Gamma \otimes A \otimes B \otimes \Delta \xrightarrow{u} TC}{\Gamma \otimes B \otimes A \otimes \Delta \xrightarrow{\Gamma \otimes \cong \otimes \Delta} \Gamma \otimes A \otimes B \otimes \Delta \xrightarrow{u} TC}$$

Logical rules, the $I, \otimes, \multimap$ fragment

$(1 - I)$

$$\frac{}{\vdash * : 1} \qquad \frac{}{I \xrightarrow{\eta} TI}$$

$(1 - E)$

$$\frac{\Lambda \vdash w : 1 \qquad \Gamma \vdash u : A}{\Gamma, \Lambda \vdash \text{let } w \text{ be } * \text{ in } u : A} \qquad \frac{\Lambda \xrightarrow{w} TI \qquad \Gamma \xrightarrow{u} TA}{\Gamma \otimes \Lambda \xrightarrow{\Gamma \otimes w} \Gamma \otimes TI \xrightarrow{t} T(\Gamma \otimes I) \xrightarrow{kleisli(\cong; u)} TA}$$

$(\otimes - I)$

$$\frac{\Gamma \vdash u : A \qquad \Delta \vdash v : B}{\Gamma, \Delta \vdash u \otimes v : A \otimes B} \qquad \frac{\Gamma \xrightarrow{u} TA \qquad \Delta \xrightarrow{v} TB}{\Gamma \otimes \Delta \xrightarrow{u \otimes v} TA \otimes TB \xrightarrow{\psi} T(A \otimes B)}$$

$(\otimes - E)$

$$\frac{\Lambda \vdash w : A \otimes B \qquad \Gamma, x : A, y : B \vdash u : C}{\Gamma, \Lambda \vdash \text{let } w \text{ be } x \otimes y \text{ in } u : C}$$

$$\frac{\Lambda \xrightarrow{w} T(A \otimes B) \qquad \Gamma \otimes A \otimes B \xrightarrow{u} TC}{\Gamma \otimes \Lambda \xrightarrow{\Gamma \otimes w} \Gamma \otimes T(A \otimes B) \xrightarrow{t} T(\Gamma \otimes A \otimes B) \xrightarrow{kleisli(u)} TC}$$

$(\multimap - I)$

$$\frac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x.u : A \multimap B} \qquad \frac{\Gamma \otimes A \xrightarrow{u} TB}{\Gamma \xrightarrow{curry(u)} A \multimap TB \xrightarrow{\eta} T(A \multimap TB)}$$

$(\multimap - E)$

$$\frac{\Lambda \vdash f : A \multimap B \qquad \Delta \vdash u : A}{\Lambda, \Delta \vdash fu : B}$$

$$\frac{\Lambda \xrightarrow{f} T(A \multimap TB) \qquad \Delta \xrightarrow{u} TA}{\Lambda \otimes \Delta \xrightarrow{f \otimes u} T(A \multimap TB) \otimes TA \xrightarrow{\psi} T((A \multimap TB) \otimes A) \xrightarrow{kleisli(eval)} TB}$$

Logical rules, the $\&, \oplus$ fragment

$(\& - I)$

$$\frac{\Gamma \vdash u : A \qquad \Gamma \vdash v : B}{\Gamma \vdash (u, v) : A \& B} \qquad \frac{\Gamma \xrightarrow{u} TA \qquad \Gamma \xrightarrow{v} TB}{\Gamma \xrightarrow{<u,v>} TA \times TB \xrightarrow{\eta} T(TA \times TB)}$$

$(\& - E_1)$

$$\frac{\Lambda \vdash w : A \& B}{\Lambda \vdash \mathrm{fst}(w) : A} \qquad \frac{\Lambda \xrightarrow{w} T(TA \times TB)}{\Lambda \xrightarrow{w} T(TA \times TB) \xrightarrow{kleisli(\pi_1)} TA}$$

$(\& - E_2)$ Analogous.

$(\oplus - I_1)$

$$\frac{\Gamma \vdash w : A}{\Gamma \vdash \mathrm{inl}(w) : A \oplus B} \qquad \frac{\Gamma \xrightarrow{w} TA}{\Gamma \xrightarrow{w} TA \xrightarrow{T(inj_1)} T(A + B)}$$

$(\oplus - I_2)$ Analogous.

$(\oplus - E)$

$$\frac{\Lambda \vdash w : A \oplus B \qquad \Gamma, x : A \vdash u : C \qquad \Gamma, y : B \vdash v : C}{\Gamma, \Lambda \vdash \mathrm{case}\ w\ \mathrm{of}\ \mathrm{inl}(x) => u \mid \mathrm{inr}(y) => v : C}$$

$$\frac{\Lambda \xrightarrow{w} T(A + B) \qquad \Gamma \otimes A \xrightarrow{u} TC \qquad \Gamma \otimes B \xrightarrow{v} TC}{\Gamma \otimes \Lambda \xrightarrow{\Gamma \otimes w} \Gamma \otimes T(A + B) \xrightarrow{t} T(\Gamma \otimes (A + B)) \xrightarrow{kleisli(\cong; <u;v>)} TC}$$

Logical rules, the ! fragment

$(! - I)$

$$\frac{\Gamma_1 \vdash w_1 :!A_1 \quad , ..., \quad \Gamma_n \vdash w_n :!A_n \qquad x_1 :!A_1, ..., x_n :!A_n \vdash u : A}{\Gamma_1, ..., \Gamma_n \vdash \mathrm{let}\ w_1, ..., w_n\ \mathrm{be}\ x_1, ..., x_n\ \mathrm{in}\ !u :!A}$$

$$\frac{\Gamma_1 \xrightarrow{w_1} T!TA_1 \quad , ..., \quad \Gamma_n \xrightarrow{w_n} T!TA_n \qquad !TA_1 \otimes ... \otimes !TA_n \xrightarrow{u} TA}{\Gamma_1 \otimes ... \otimes \Gamma_n \xrightarrow{w_1 \otimes ... \otimes w_n} T!TA_1 \otimes ... \otimes T!TA_n \xrightarrow{\psi} T(!TA_1 \otimes ... \otimes !TA_n) \xrightarrow{T(\gamma(u))} T!TA}$$

(*Dereliction*)

$$\frac{\Lambda \vdash u :!A}{\Lambda \vdash \mathrm{derelict}(u) : A} \qquad \frac{\Lambda \xrightarrow{u} T!TA}{\Lambda \xrightarrow{u} T!TA \xrightarrow{kleisli(\epsilon)} TA}$$

(*Weakening*)

$$\frac{\Lambda \vdash w :!A \qquad \Gamma \vdash u : B}{\Gamma, \Lambda \vdash \mathrm{discard}\ w\ \mathrm{in}\ u : B} \qquad \frac{\Lambda \xrightarrow{w} T!TA \qquad \Gamma \xrightarrow{u} TB}{\Gamma \otimes \Lambda \xrightarrow{\Gamma \otimes w} \Gamma \otimes T!TA \xrightarrow{t} T(\Gamma \otimes !TA) \xrightarrow{kleisli((\Gamma \otimes e); \cong; u)} TB}$$

(*Contraction*)

$$\frac{\Lambda \vdash w : !A \qquad \Gamma, x : !A, y : !A \vdash u : B}{\Gamma, \Lambda \vdash \text{copy } w \text{ as } x, y \text{ in } u : B}$$

$$\frac{\Lambda \xrightarrow{w} T!TA \qquad \Gamma \otimes !TA \otimes !TA \xrightarrow{u} TB}{\Gamma \otimes \Lambda \xrightarrow{\Gamma \otimes w} \Gamma \otimes T!TA \xrightarrow{t} T(\Gamma \otimes !TA) \xrightarrow{kleisli((\Gamma \otimes d);u)} TB}$$

Other rules

(*Recursion*)

$$\frac{\Gamma_1 \vdash w_1 : !A_1 \quad , ..., \quad \Gamma_n \vdash w_n : !A_n \qquad x_1 : !A_1, ..., x_n : !A_n, z : !B \vdash u : B}{\Gamma_1, ..., \Gamma_n \vdash \text{let } w_1, ..., w_n \text{ be } x_1, ..., x_n \text{ in rec} z.u : B}$$

$$\frac{\Gamma_1 \xrightarrow{w_1} T!TA_1 \quad , ..., \quad \Gamma_n \xrightarrow{w_n} T!TA_n \qquad !TA_1 \otimes ... \otimes !TA_n \otimes !TB \xrightarrow{u} TB}{\Gamma_1 \otimes ... \otimes \Gamma_n \xrightarrow{w_1 \otimes ... \otimes w_n} T!TA_1 \otimes ... \otimes T!TA_n \xrightarrow{\psi} T(!TA_1 \otimes ... \otimes !TA_n) \xrightarrow{kleisli(u^\sharp)} TB}$$

D Appendix, operational semantics for LTS$_{+Rec}$

The $I, \otimes, \multimap$ fragment

$$\frac{}{* \longrightarrow *} \qquad \frac{w \longrightarrow * \quad u \longrightarrow c}{\text{let } w \text{ be } * \text{ in } u \longrightarrow c}$$

$$\frac{u \longrightarrow c \quad v \longrightarrow d}{u \otimes v \longrightarrow c \otimes d} \qquad \frac{w \longrightarrow d \otimes e \quad u[d/x, e/y] \longrightarrow c}{\text{let } w \text{ be } x \otimes y \text{ in } u \longrightarrow c}$$

$$\frac{}{\lambda x.u \longrightarrow \lambda x.u} \qquad \frac{f \longrightarrow \lambda x.v \quad u \longrightarrow d \quad v[d/x] \longrightarrow c}{fu \longrightarrow c}$$

The $\&, \oplus$ fragment

$$\frac{}{(u,v) \longrightarrow (u,v)} \qquad \frac{w \longrightarrow (u,v) \quad u \longrightarrow c}{\text{fst}(w) \longrightarrow c} \qquad \frac{w \longrightarrow (u,v) \quad v \longrightarrow c}{\text{snd}(w) \longrightarrow c}$$

$$\frac{w \longrightarrow d}{\text{inl}(w) \longrightarrow \text{inl}(d)} \qquad \frac{w \longrightarrow e}{\text{inr}(w) \longrightarrow \text{inr}(e)}$$

$$\frac{w \longrightarrow \text{inl}(d) \quad u[d/x] \longrightarrow c}{\text{case } w \text{ of inl}(x) => u \mid \text{inr}(y) => v \longrightarrow c} \qquad \frac{w \longrightarrow \text{inr}(e) \quad u[e/y] \longrightarrow c}{\text{case } w \text{ of inl}(x) => u \mid \text{inr}(y) => v \longrightarrow c}$$

The ! fragment

$$\frac{w_1 \longrightarrow c_1 \quad ,..., \quad w_n \longrightarrow c_n}{\text{let } w_1,...,w_n \text{ be } x_1,...,x_n \text{ in } !u \longrightarrow \text{let } c_1,...,c_n \text{ be } x_1,...,x_n \text{ in } !u}$$

$$\frac{u \longrightarrow \text{let } c_1,...,c_n \text{ be } x_1,...,x_n \text{ in } !v \quad v[c_1/x_1,...,c_n/x_n] \longrightarrow c}{\text{derelict}(u) \longrightarrow c}$$

$$\frac{w \longrightarrow d \quad u[d/x, d/y] \longrightarrow c}{\text{copy } w \text{ as } x, y \text{ in } u \longrightarrow c} \qquad \frac{w \longrightarrow d \quad u \longrightarrow c}{\text{discard } w \text{ in } u \longrightarrow c}$$

Other rules

$$\frac{w_1 \longrightarrow c_1 \,, ..., \; w_n \longrightarrow c_n \quad u[c_1/x_1,...,c_n/x_n][!(\text{let } c_1,...,c_n \text{ be } x_1,...,x_n \text{ in rec}z.u)/z] \longrightarrow c}{\text{let } w_1,...,w_n \text{ be } x_1,...,x_n \text{ in rec}z.u \longrightarrow c}$$

E Appendix, proof of the Soundness Theorem

Theorem E.1 *(Soundness Theorem) Given a closed term u s.t. $u \longrightarrow c$, then $[\![u]\!] = [\![c]\!]$.*

Proof. Induction in the derivation of $u \longrightarrow c$.

I will only cover the most interesting case, namely the case where the last used rule is:

$$\frac{w_1 \longrightarrow c_1\ , ...,\ w_n \longrightarrow c_n \quad u[c_1/x_1, ..., c_n/x_n][!(\text{let } c_1, ..., c_n \text{ be } x_1, ..., x_n \text{ in rec}z.u)/z] \longrightarrow c}{\text{let } w_1, ..., w_n \text{ be } x_1, ..., x_n \text{ in rec}z.u \longrightarrow c}$$

Due to previous results, we know that for every $r \in \{1, ..., n\}$ there exists a $h_r : I \to TA_r$ such that $[\![c_r]\!] = \gamma(h_r); \eta$. We therefore have:

$$\begin{aligned}
&[\![\text{let } c_1, ..., c_n \text{ be } x_1, ..., x_n \text{ in rec}z.u]\!] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r [\![c_r]\!]} \otimes_r T!TA_r \xrightarrow{\psi} T(\otimes_r !TA_r) \xrightarrow{kleisli([\![u]\!]^\sharp)} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r (\gamma(h_r);\eta)} \otimes_r T!TA_r \xrightarrow{\psi} T(\otimes_r !TA_r) \xrightarrow{kleisli([\![u]\!]^\sharp)} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r \gamma(h_r)} \otimes_r !TA_r \xrightarrow{\otimes_r \eta} \otimes_r T!TA_r \xrightarrow{\psi} T(\otimes_r !TA_r) \xrightarrow{kleisli([\![u]\!]^\sharp)} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r \gamma(h_r)} \otimes_r !TA_r \xrightarrow{\eta} T(\otimes_r !TA_r) \xrightarrow{kleisli([\![u]\!]^\sharp)} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r \gamma(h_r)} \otimes_r !TA_r \xrightarrow{[\![u]\!]^\sharp} TA]
\end{aligned}$$

Now, to prove the result is a matter of direct computation:

$$\begin{aligned}
&[\![c]\!] =^* \\
&[\![u[c_1/x_1, ..., c_n/x_n][!(\text{let } c_1, ..., c_n \text{ be } x_1, ..., x_n \text{ in rec}z.u)/z]]\!] = \\
&[I \cong (\otimes_r I) \otimes I \xrightarrow{(\otimes_r \gamma(h_r))\otimes\gamma(\cong;(\otimes_r\gamma(h_r));[\![u]\!]^\sharp)} (\otimes_r !TA_r)\otimes !TA \xrightarrow{[\![u]\!]} TA] = \\
&[I \cong (\otimes_r I) \otimes I \xrightarrow{(\otimes_r \gamma(h_r))\otimes(\cong;(\otimes_r\gamma(h_r));\gamma([\![u]\!]^\sharp))} (\otimes_r !TA_r)\otimes !TA \xrightarrow{[\![u]\!]} TA] = \\
&[I \cong (\otimes_r I) \otimes I \xrightarrow{(\otimes_r \gamma(h_r))\otimes(\cong;(\otimes_r\gamma(h_r))} (\otimes_r !TA_r) \otimes (\otimes_r !TA_r) \xrightarrow{(Id\otimes\gamma([\![u]\!]^\sharp));[\![u]\!]} TA] = \\
&[I \cong (\otimes_r I) \otimes (\otimes_r I) \xrightarrow{(\otimes_r \gamma(h_r))\otimes(\otimes_r\gamma(h_r))} (\otimes_r !TA_r) \otimes (\otimes_r !TA_r) \xrightarrow{(Id\otimes\gamma([\![u]\!]^\sharp));[\![u]\!]} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r(\cong;(\gamma(h_r)\otimes\gamma(h_r)))} \otimes_r (!TA_r\otimes !TA_r) \cong (\otimes_r !TA_r) \otimes (\otimes_r !TA_r) \xrightarrow{(Id\otimes\gamma([\![u]\!]^\sharp));[\![u]\!]} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r(\gamma(h_r);d)} \otimes_r (!TA_r\otimes !TA_r) \cong (\otimes_r !TA_r) \otimes (\otimes_r !TA_r) \xrightarrow{(Id\otimes\gamma([\![u]\!]^\sharp));[\![u]\!]} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r \gamma(h_r)} \otimes_r !TA_r \xrightarrow{D} (\otimes_r !TA_r) \otimes (\otimes_r !TA_r) \xrightarrow{(Id\otimes\gamma([\![u]\!]^\sharp));[\![u]\!]} TA] =^{**} \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r \gamma(h_r)} \otimes_r !TA_r \xrightarrow{[\![u]\!]^\sharp} TA] = \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r [\![c_r]\!]} \otimes_r T!TA_r \xrightarrow{\psi} T(\otimes_r !TA_r) \xrightarrow{kleisli([\![u]\!]^\sharp)} TA] =^* \\
&[I \cong \otimes_r I \xrightarrow{\otimes_r [\![w_r]\!]} \otimes_r T!TA_r \xrightarrow{\psi} T(\otimes_r !TA_r) \xrightarrow{kleisli([\![u]\!]^\sharp)} TA] = \\
&[\![\text{let } w_1, ..., w_n \text{ be } x_1, ..., x_n \text{ in rec}z.u]\!]
\end{aligned}$$

* Cf. the induction hypothesis. ** We are here using the fact that $[\![u]\!]^\sharp$ is a linear fixpoint of $[\![u]\!]$. □

F Appendix, proof of the Approximation Lemma

Lemma F.1 *(Approximation Lemma) Let $x_1 : A_1, ..., x_n : A_n \vdash u : C$ be given. Let J be a finite set s.t. for all $j \in J$ we are given $\vdash v_j : B_j$ and $f_j \in hom(I, [\![B_j]\!])$ with the property that $f_j ; \eta \lesssim_{B_j} v_j$. Moreover, assume that for each j we have a variable z_j, s.t. for all $i, k \in J$ we have $i \neq k \Rightarrow z_i \neq z_k$, s.t. $\{x_1, ..., x_n\} \subseteq \{z_j | \ j \in J\}$, and s.t. for all $r \in \{1, ..., n\}$ and $j \in J$ we have $x_r = z_j \Rightarrow A_r = B_j$. We then have:*

$$[I \cong I \otimes ... \otimes I \xrightarrow{f_{j_1} \otimes ... \otimes f_{j_n}} [\![A_1]\!] \otimes ... \otimes [\![A_n]\!] \xrightarrow{[\![u]\!]} T[\![C]\!]] \lesssim_C u[..., v_j/z_j, ...]$$

where for all r in $\{1, ..., n\}$ we have chosen j_r s.t. $x_r = z_{j_r}$.

Proof. If $1 \leq p \leq q \leq n$ and $\Gamma = x_p : A_p, ..., x_q : A_q$, that is, it is a subsequence of $x_1 : A_1, ..., x_n : A_n$, we will then define $f_\Gamma : I \to \Gamma$ to be the following morphism:

$$[I \cong I \otimes ... \otimes I \xrightarrow{f_{j_p} \otimes ... \otimes f_{j_q}} A_p \otimes ... \otimes A_q]$$

We proceed by induction in the derivation of $x_1 : A_1, ..., x_n : A_n \vdash u : C$. We will without loss of generality assume that none of the variables $\{z_j | \ j \in J\}$ are bound in u.

I will only cover the most interesting case, namely the case where the last used rule is:

$$\frac{\Gamma_1 \vdash w_1 :!A_1 \quad , ..., \quad \Gamma_m \vdash w_m :!A_m \qquad x_1 :!A_1, ..., x_m :!A_m, z :!B \vdash v : B}{\Gamma_1, ..., \Gamma_m \vdash \text{let } w_1, ..., w_m \text{ be } x_1, ..., x_m \text{ in rec} z.v : B}$$

We want to prove that $\star_1 \lesssim_B (\text{let } w_1, ..., w_m \text{ be } x_1, ..., x_m \text{ in rec} z.v)[..., v_j/z_j, ...]$ where $\star_1$ is the following morphism:

$$[I \cong \otimes_l I \xrightarrow{\otimes_l f_{\Gamma_l}} \otimes_l \Gamma_l \xrightarrow{\otimes_l [\![w_l]\!]} \otimes_l T!TA_l \xrightarrow{\psi} T(\otimes_l !TA_l) \xrightarrow{kleisli([\![v]\!]^\sharp)} TB]$$

where l is in $\{1, ..., m\}$. Now, assume that $\star_1 \neq \bot$. We then have for every l:

$$[I \xrightarrow{f_{\Gamma_l}} \Gamma_l \xrightarrow{[\![w_l]\!]} T!TA_l] = [I \xrightarrow{k_l} !TA_l \xrightarrow{\eta} T!TA_l]$$

for some $k_l : I \to !TA_l$. We now apply the induction hypothesis on $\Gamma_l \vdash w_l :!A_l$ and get for every l:

$$w_l[..., v_j/z_j, ...] \longrightarrow c_l \qquad \text{and} \qquad k_l \lesssim^\circ_{!A_l} c_l$$

for some $c_l \in C_{!A_l}$. This entails that for every l we have $k_l = \gamma(h_l)$ for some $h_l : I \to TA_l$. Thus:

$$\begin{aligned}
&\star_1 = \\
&[I \cong \otimes_l I \xrightarrow{\otimes_l \gamma(h_l)} \otimes_l !TA_l \xrightarrow{\otimes_l \eta} \otimes_l T!TA_l \xrightarrow{\psi} T(\otimes_l !TA_l) \xrightarrow{kleisli([\![v]\!]^\sharp)} TB] = \\
&[I \cong \otimes_l I \xrightarrow{\otimes_l \gamma(h_l)} \otimes_l !TA_l \xrightarrow{[\![v]\!]^\sharp} TB] = \\
&[I \cong \otimes_l I \xrightarrow{\otimes_l \gamma(h_l)} \otimes_l !TA_l \xrightarrow{\bigsqcup_{n \in \omega} \Phi^n(\bot)} TB] = \\
&[I \xrightarrow{\bigsqcup_{n \in \omega} (\cong ; (\otimes_l \gamma(h_l)) ; \Phi^n(\bot))} TB]
\end{aligned}$$

where Φ is the continous function from $hom(!TA_1 \otimes ... \otimes !TA_m, TB)$ to itself, induced by the morphism $[\![u]\!]$. Now, due to the Approximation Lemma, it is enough to show that for all n in ω:

$$[I \xrightarrow{\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot)} TB] \lesssim_B (\text{let } w_1, ..., w_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)[..., v_j/z_j, ...]$$

We will prove this by induction in n. The assertion is true in the case $n = 0$ because any point followed by a bottom element is a bottom element. Let a number n be given and assume that the assertion is true in case of this n. We then want to show that the assertion is true in case of $n + 1$.

First a small initial calculation:

$$[I \xrightarrow{\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot)} TB] \neq \bot$$

entails due to the inner IH that there exists a d in C_B such that:

$$(\text{let } w_1, ..., w_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)[..., v_j/z_j, ...] \longrightarrow d$$

But the rules for the evaluation relation then gives us:

$$\text{let } c_1, ..., c_m \text{ be } x_1, ..., x_m \text{ in rec}z.v \longrightarrow d$$

The conclusion of the initial calculation is that:

$$[I \xrightarrow{\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot)} TB] \lesssim_B \text{let } c_1, ..., c_m \text{ be } x_1, ..., x_m \text{ in rec}z.v$$

But then the definition of the $\lesssim^\circ_{!B}$ relation entails that:

$$[I \xrightarrow{\gamma(\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot))} !TB] \lesssim^\circ_{!B} !(\text{let } c_1, ..., c_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)$$

We can now use the outer IH on $x_1 :!A_1, ..., x_m :!A_m, z :!B \vdash v : B$ to obtaim:

$$\star_2 \lesssim_B v[c_1/x_1, ..., c_m/x_m][!(\text{let } c_1, ..., c_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)/z]$$

where $\star_2$ is defined to be the morphism:

$$[I \cong I \otimes ... \otimes I \otimes I \xrightarrow{\gamma(h_1)\otimes ... \otimes \gamma(h_m) \otimes \gamma(\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot))} !TA_1 \otimes ... \otimes !TA_m \otimes !TB \xrightarrow{[\![v]\!]} TB]$$

Moreover:

$$[I \xrightarrow{\cong;(\otimes_l \gamma(h_l));\Phi^{n+1}(\bot)} TB] =$$
$$[I \cong \otimes_l I \xrightarrow{\otimes_l \gamma(h_l)} \otimes_l !TA_l \xrightarrow{\otimes_l d} \otimes_l (!TA_l \otimes !TA_l) \cong (\otimes_l !TA_l) \otimes (\otimes_l !TA_l) \xrightarrow{(Id \otimes \gamma(\Phi^n(\bot)));[\![v]\!]} TB] =$$
$$[I \cong \otimes_l I \xrightarrow{\otimes_l (\cong;(\gamma(h_l) \otimes \gamma(h_l)))} \otimes_l (!TA_l \otimes !TA_l) \cong (\otimes_l !TA_l) \otimes (\otimes_l !TA_l) \xrightarrow{(Id \otimes \gamma(\Phi^n(\bot)));[\![v]\!]} TB] =$$
$$[I \cong I \otimes I \xrightarrow{\cong \otimes \cong} (\otimes_l I) \otimes (\otimes_l I) \xrightarrow{(\otimes_l \gamma(h_l)) \otimes (\otimes_l \gamma(h_l))} (\otimes_l !TA_l) \otimes (\otimes_l !TA_l) \xrightarrow{(Id \otimes \gamma(\Phi^n(\bot)));[\![v]\!]} TB] =$$
$$[I \cong I \otimes I \xrightarrow{\cong \otimes Id} (\otimes_l I) \otimes I \xrightarrow{(\otimes_l \gamma(h_l)) \otimes Id} (\otimes_l !TA_l) \otimes I \xrightarrow{(Id \otimes \gamma(\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot)));[\![v]\!]} TB] =$$
$$[I \cong (\otimes_l I) \otimes I \xrightarrow{(\otimes_l \gamma(h_l)) \otimes Id} (\otimes_l !TA_l) \otimes I \xrightarrow{(Id \otimes \gamma(\cong;(\otimes_l \gamma(h_l));\Phi^n(\bot)));[\![v]\!]} TB] =$$
$$[I \xrightarrow{\star_2} TB]$$

Now, our final calculation:

$$[I \xrightarrow{\cong;(\otimes_l \gamma(h_l));\Phi^{n+1}(\bot)} TB] \neq \bot$$

entails that:

$$v[c_1/x_1, ..., c_m/x_m][!(\text{let } c_1, ..., c_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)/z] \longrightarrow c$$

for some c. But the rules for the evaluation relation then gives us:

$$(\text{let } w_1, ..., w_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)[..., v_j/z_j, ...] \longrightarrow c$$

Hence:

$$[I \xrightarrow{\cong;(\otimes_l \gamma(h_l));\Phi^{n+1}(\bot)} TB] \lesssim_B (\text{let } w_1, ..., w_m \text{ be } x_1, ..., x_m \text{ in rec}z.v)[..., v_j/z_j, ...]$$

which finishes the proof that the above mentioned assertion is true in case of $n+1$. $\Box$

G Appendix, categorical prerequisites

G.1 Monoidal categories

Definition G.1 *A* monoidal category *is a 6-tuple* $(\mathcal{C}, I, \otimes, \alpha, \lambda, \rho)$ *where* $\mathcal{C}$ *is a category containing a neutral element* I *for a bifunctor* $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ *equipped with natural isomorphisms* α, λ, ρ *having components:*

$$\alpha_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C \qquad \lambda_A : I \otimes A \to A \qquad \rho_A : A \otimes I \to A$$

These are required to satisfy the socalled Kelly-Mac Lane equations, which are: the pentagon law $(Id \otimes \alpha); \alpha; (\alpha \otimes Id) = \alpha; \alpha$*, the triangle law* $\alpha; (\rho \otimes Id) = (Id \otimes \lambda)$*, and* $\lambda_I = \rho_I$*. Such a monoidal category is* symmetric *if there is an additional "symmetry" natural isomorphism* γ *with components:*

$$\gamma_{A,B} : A \otimes B \to B \otimes A$$

satisfying $\gamma; \gamma = Id$*,* $\rho = \gamma; \lambda$ *, and* $\alpha; \gamma; \alpha = (Id \otimes \gamma); \alpha; (\gamma \otimes Id)$*.*

A monoidal functor *from* $(\mathcal{C}, I, \otimes, \alpha, \lambda, \rho)$ *to* $(\mathcal{C}', I', \otimes', \alpha', \lambda', \rho')$ *is a functor* $F : \mathcal{C} \to \mathcal{C}'$ *equipped with a map* $m_{I'} : I' \to F(I)$ *and a natural transformation* $m : F(-) \otimes' F(+) \to F(- \otimes +)$ *which match the involved structure, that is,* $(Id \otimes' m); m; F(\alpha) = \alpha'; (m \otimes' Id); m$*,* $(m_{I'} \otimes' Id); m; F(\lambda) = \lambda'$*, and* $(Id \otimes' m_{I'}); m; F(\rho) = \rho'$*. It is a* symmetric *monoidal functor if additionally* $\gamma'; m = m; F(\gamma)$*.* F preserves *the (symmetric) monoidal structure (or:* F *is a morphism of (symmetric) monoidal categories) iff* $m_{I'}$ *and* m *are isomorphisms.*

A monoidal natural transformation *between monoidal functors* $F, F' : \mathcal{C} \to \mathcal{C}'$ *is a natural transformation* $\sigma : F \to F'$ *satisfying* $m; \sigma = (\sigma \otimes' \sigma); m'$ *and* $m_{I'}; \sigma_I = m'_{I'}$*.*

*A (*symmetric*)* monoidal closed category *is a (symmetric) monoidal category where each functor* $(-) \otimes A$ *has a right adjoint; it will be denoted by* $A \multimap (-)$*.*

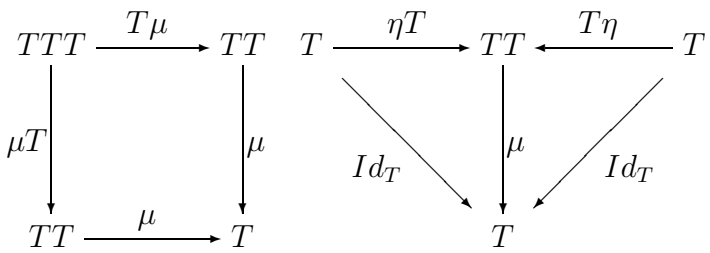
Note that one obtains 2-categories of (symmetric) monoidal categories.

G.2 Monads

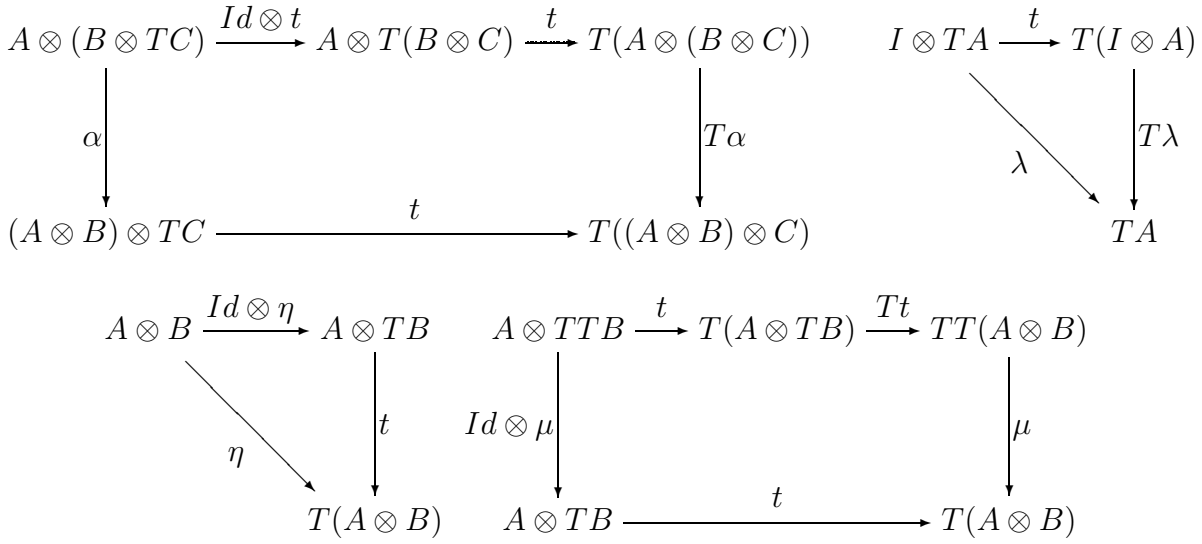
Definition G.2 *A* monad (T, η, μ) *on* $\mathcal{C}$ *is a functor* $T : \mathcal{C} \to \mathcal{C}$*, and two natural transformations:*

$$\eta : Id_{\mathcal{C}} \to T \qquad \mu : TT \to T$$

such that the following diagrams commute:

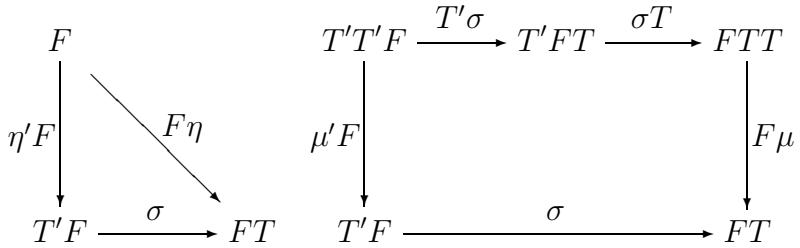


Definition G.3 *Let $(\mathcal{C}, I, \otimes, \alpha, \lambda, \rho)$ be a monoidal category and (T, η, μ) a monad on $\mathcal{C}$. The monad is called* strong *iff there is a "strength" natural transformation $t : (-) \otimes T(+) \to T(- \otimes +)$ s.t. the following diagrams commute:*

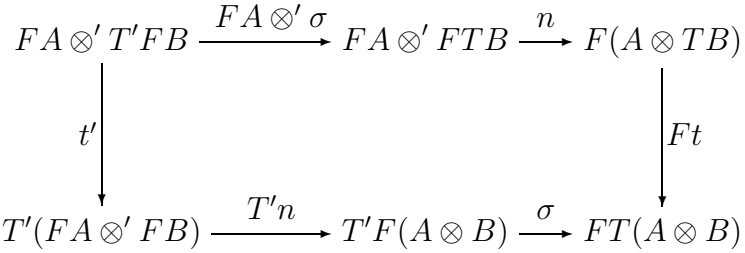


The usual definition of a map between monads assume that the domain and codomain monads are on the same category. The following is a generalisation without this assumption.

Definition G.4 *A* functor of monads *from a monad (T, η, μ) on $\mathcal{C}$ to a monad (T', η', μ') on $\mathcal{C}'$ is a functor $F : \mathcal{C} \to \mathcal{C}'$ and a natural transformation $\sigma : T'F \to FT$ s.t. the following diagrams commute:*



Definition G.5 *Let (T, η, μ, t) be a strong monad on a monoidal category $(\mathcal{C}, I, \otimes)$ and (T', η', μ', t') a strong monad on a monoidal category $(\mathcal{C}', I', \otimes')$. A functor of monads (F, σ) from (T, η, μ) to (T', η', μ') is a* functor of strong monads *iff F is monoidal and the following diagram commutes:*



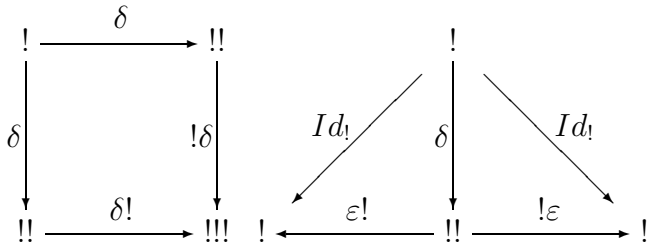
where n is the natural transformation giving monoidal structure to F.

G.3 Comonads

Definition G.6 *A* comonad $(!, \varepsilon, \delta)$ *on* $\mathcal{C}$ *is a functor* $! : \mathcal{C} \to \mathcal{C}$*, and two natural transformations:*

$$\varepsilon :! \to Id_{\mathcal{C}} \qquad \delta :! \to !!$$

such that the following diagrams commute:



Definition G.7 *Given a comonad* $(!, \varepsilon, \delta)$ *on* $\mathcal{C}$*, one can define the* coKleisli category*,* $\mathcal{C}_!$ *as follows: the objects are the same as in* $\mathcal{C}$*, the morphisms are given by* $hom_{\mathcal{C}_!}(A, B) = hom_C(!A, B)$*. If* $f : A \to B$ *is an arrow in* $\mathcal{C}_!$*, then the corresponding arrow in* $\mathcal{C}$ *is denoted by* $f^* :!A \to B$*. Now, given* $f : A \to B$ *and* $g : B \to C$*, arrows in* $\mathcal{C}_!$*, their composition is defined to be* $(f; g)^* = \delta_A; !(f^*); g^* :!A \to C$*. Given an object* A*, the unit is defined to be* $(Id_A)^* = \varepsilon_A :!A \to A$*.*

Recent Publications in the BRICS Report Series

RS-94-22 **Torben Braüner.** ***A General Adequacy Result for a Linear Functional Language*****. August 1994, 39 pp. Presented at MFPS '94.**

RS-94-21 **Søren Riis.** ***Count(q) does not imply Count(p)*****. July 1994, 55 pp.**

RS-94-20 **Peter D. Mosses and Mart´n Musicante.** ***An Action Semantics for ML Concurrency Primitives*****. July 1994, 21 pp. To appear in Proc. FME '94 (Formal Methods Europe, Symposium on Industrial Benefit of Formal Methods), LNCS, 1994.**

RS-94-19 **Jens Chr. Godskesen, Kim G. Larsen, and Arne Skou.** ***Automatic Verification of Real–Timed Systems Using*** Epsilon**. June 1994, 8 pp. Appears in: Protocols, Specification, Testing and Verification PSTV '94.**

RS-94-18 **Sten Agerholm.** ***LCF Examples in HOL*****. June 1994, 16 pp. To appear in:** ***Proceedings of the 7th International Workshop on Higher Order Logic Theorem Proving and its Applications*****, LNCS, 1994.**

RS-94-17 **Allan Cheng.** ***Local Model Checking and Traces*****. June 1994, 30 pp.**

RS-94-16 **Lars Arge.** ***External-Storage Data Structures for Plane-Sweep Algorithms*****. June 1994, 37 pp.**

RS-94-15 **Mogens Nielsen and Glynn Winskel.** ***Petri Nets and Bisimulations*****. May 1994, 36 pp.**

RS-94-14 **Nils Klarlund.** ***The Limit View of Infinite Computations*****. May 1994, 16 pp. To appear in the LNCS proceedings of Concur '94, LNCS, 1994.**

RS-94-13 **Glynn Winskel.** ***Stable Bistructure Models of PCF*****. May 1994, 26 pp.** ***Preliminary draft.*** **Invited lecture for MFCS '94. To appear in the proceedings of MFCS '94, LNCS, 1994.**