Lax Logical Relations

Gordon Plotkin[1⋆], John Power[1⋆⋆],
Donald Sannella[1⋆⋆⋆], and Robert Tennent[1,2†]

[1] Laboratory for Foundations of Computer Science
University of Edinburgh, Edinburgh, U.K. EH9 3JZ
{gdp,ajp,dts}@dcs.ed.ac.uk
[2] Department of Computing and Information Science
Queen's University, Kingston, Canada K7L 3N6
rdt@cs.queensu.ca

Abstract. Lax logical relations are a categorical generalisation of logical relations; though they preserve product types, they need not preserve exponential types. But, like logical relations, they are preserved by the meanings of all lambda-calculus terms. We show that lax logical relations coincide with the correspondences of Schoett, the algebraic relations of Mitchell and the pre-logical relations of Honsell and Sannella on Henkin models, but also generalise naturally to models in cartesian closed categories and to richer languages.

1 Introduction

Logical relations and various generalisations are used extensively in the study of typed lambda calculi, and have many applications, including

- characterising lambda definability [Pl73, Pl80, JT93, Al95];
- relating denotational semantic definitions [Re74, MS76];
- characterising parametric polymorphism [Re83];
- modelling abstract interpretation [Ab90];
- verifying data representations [Mi91];
- defining fully abstract semantics [OR95]; and
- modelling local state in higher-order languages [OT95, St96].

The two key properties of logical relations are

1. the so-called Basic Lemma: a logical relation is preserved by the meaning of every lambda term; and
2. inductive definition: the type-indexed family of relations is determined by the base-type components.

⋆ This author was supported by EPSRC grants GR/J84205 and GR/M56333.
⋆⋆ This author was supported by EPSRC grants GR/J84205 and GR/M56333, and by a grant from the British Council.
⋆⋆⋆ This author was supported by EPSRC grant GR/K63795.
† This author was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

It has long been known that there are type-indexed families of conventional relations that satisfy the Basic Lemma but are *not* determined inductively in a straightforward way. Schoett [Sc87] uses families of relations that are preserved by algebraic operations to treat behavioural inclusion and equivalence of algebraic data types; he terms them "correspondences," but they have also been called "simulations" [Mi71] and "weak homomorphisms" [Gi68]. Furthermore, Schoett conjectures (pages 280–81) that the Basic Lemma will hold when appropriate correspondences are used between models of lambda calculi, and that such relations compose. Mitchell [Mi90, Sect. 3.6.2] terms them "algebraic relations," attributing the suggestion to Gordon Plotkin[1] and Samson Abramsky, independently, and asserts that the Basic Lemma is easily proved and (binary) algebraic relations compose. But Mitchell concludes that, because logical relations are easily constructed by induction on types, they "seem to be the important special case for proving properties of typed lambda calculi."

Recently, Honsell and Sannella [HS99] have shown that such relation families, which they term "pre-logical relations," are both the *largest* class of conventional relations on Henkin models that satisfy the Basic Lemma, and the smallest class that both includes logical relations and is closed under composition. They give a number of examples and applications, and study their closure properties.

We briefly sketch two of their applications.

- The composite of (binary) logical relations need not be logical. It is an easy exercise to construct a counter-example; see, for instance, [HS99]. But the composite of binary pre-logical relations *is* a pre-logical relation.
- Mitchell [Mi91] showed that the use of logical relations to verify data representations in typed lambda calculi is complete, provided that all of the primitive functions are first-order. In [HS99], this is strengthened to allow for higher-order primitives by generalising to *pre*-logical relations. Honsell, Longley et al. [HL$^+$] give an example in which a pre-logical relation is used to justify the correctness of a data representation that cannot be justified using a conventional logical relation.

In this work, we give a *categorical* characterisation of algebraic relations (simulations, correspondences) between Henkin models of typed lambda calculi. The key advantage of this characterisation is its generality. By using it, one can immediately generalise from Henkin models to models in categories very different from *Set*, and to languages very different from the simply typed lambda calculus, for example to languages with co-products or tensor products, or to imperative languages without higher-order constructs.

The paper is organised as follows. In Sect. 2, we recall the definition of logical relation and a category theoretic formulation. In Sect. 3, we give our categorical notion of lax logical relation, proving a Basic Lemma, with a converse. In Sect. 4, we explain the relationship with pre-logical relations and in Sect. 5 give

[1] Plotkin recalls that the suggestion was made to him by Eugenio Moggi in a conversation.

another syntax-based characterisation. In Sect. 6 we consider models in cartesian closed categories. In Sect. 7, we generalise our analysis to richer languages.

2 Logical Relations

Let Σ be a signature of basic types and constants for the simply typed λ-calculus with products [Mi90], generating a language L. We use σ and τ to range over types in L. We denote the set of functions from a set X to a set Y by $[X \Rightarrow Y]$.

Definition 2.1. *A* model *M of L in Set consists of*

- *for each σ, a set M_σ, such that $M_{\sigma\to\tau} = [M_\sigma \Rightarrow M_\tau]$, $M_{\sigma\times\tau} = M_\sigma \times M_\tau$, and $M_1 = \{*\}$;*
- *for each constant c of Σ of type σ, an element $M(c)$ of M_σ.*

A model extends inductively to send every judgement $\Gamma \vdash t\colon \sigma$ of L to a function $M(\Gamma \vdash t\colon \sigma)$ from M_Γ to M_σ, where M_Γ is the evident finite product in *Set*. These are "full" type hierarchies; larger classes of models, such as Henkin models and cartesian closed categories, will be discussed later.

Definition 2.2. *Given a signature Σ and two models, M and N, of the language L generated by Σ, a* (binary) logical relation from M to N *consists of, for each type σ of L, a relation $R_\sigma \subseteq M_\sigma \times N_\sigma$ such that*

- *for all $f \in M_{\sigma\to\tau}$ and $g \in N_{\sigma\to\tau}$, we have $f\,R_{\sigma\to\tau}\,g$ if and only if for all $x \in M_\sigma$ and $y \in N_\sigma$, if $x\,R_\sigma\,y$ then $f(x)\,R_\tau\,g(y)$;*
- *for all $(x_0,x_1) \in M_{\sigma\times\tau}$ and $(y_0,y_1) \in N_{\sigma\times\tau}$, we have $(x_0,x_1)\,R_{\sigma\times\tau}\,(y_0,y_1)$ if and only if $x_0\,R_\sigma\,y_0$ and $x_1\,R_\tau\,y_1$;*
- *$*\,R_1\,*$;*
- *$M(c)\,R_\sigma\,N(c)$ for every constant c in Σ of type σ.*

The data for a binary logical relation are therefore completely determined by its behaviour on base types. The fundamental result about logical relations underlying all of their applications is the following.

Lemma 2.3 (Basic Lemma for Logical Relations). *Let R be a binary logical relation from M to N; for any term $t\colon\sigma$ of L in context Γ, if $x\,R_\Gamma\,y$, then $M(\Gamma \vdash t\colon\sigma)x\,R_\sigma\,N(\Gamma \vdash t\colon\sigma)y$,*

where $x\,R_\Gamma\,y$ is an abbreviation for $x_i\,R_{\sigma_i}\,y_i$ for all i where $\sigma_1,\ldots,\sigma_n$ is the sequence of types in Γ. It is routine to define n-ary logical relations for an arbitrary natural number n, in the spirit of Definition 2.2. The corresponding formulation of the Basic Lemma holds for arbitrary n too.

We now outline a *categorical* formulation of logical relations [MR91, MS92]; this will be relaxed slightly to yield our semantic characterisation of algebraic relations for typed lambda calculi with products.

The language L determines a cartesian closed term category, which we also denote by L, such that a model M of the language L in any cartesian closed category such as *Set* extends uniquely to a cartesian closed functor from L to *Set* [Mi96, Sect. 7.2.6]; i.e., a functor that preserves products and exponentials *strictly* (not just up to isomorphism). We may therefore identify the notion of model of the language L in *Set* with that of a cartesian closed functor from L to *Set*.

Definition 2.4. *The category Rel_2 is defined as follows: an object (X, R, Y) consists of a pair of sets X and Y, and a binary relation $R \subseteq X \times Y$; a map from (X, R, Y) to (X', R', Y') is a pair of functions $(f\colon X \to X', g\colon Y \to Y')$ such that $x\,R\,y$ implies $f(x)\,R'\,g(y)$:*

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, leftrightarrow, "R"'] & X' \arrow[d, leftrightarrow, "R'"] \\ Y \arrow[r, "g"] & Y' \end{tikzcd}$$

Composition is given by ordinary composition of functions.

We denote the forgetful functors from Rel_2 to *Set* sending (X, R, Y) to X or to Y by δ_0 and δ_1, respectively.

Proposition 2.5. *Rel_2 is cartesian closed, and the cartesian closed structure is strictly preserved by the functor $(\delta_0, \delta_1)\colon Rel_2 \longrightarrow Set \times Set$.*

For example, $(X_0, R, Y_0) \Rightarrow (X_1, S, Y_1)$ is $([X_0 \Rightarrow X_1], (R \Rightarrow S), [Y_0 \Rightarrow Y_1])$ where $f(R \Rightarrow S)g$ iff, for all $x \in X_0$ and $y \in Y_0$, xRy implies $(fx)S(gy)$.

These properties of Rel_2, combined with the fact that L is freely generated by a signature for cartesian closed categories (i.e., is the generic model on a suitable sketch [KO+97]), are the key to understanding logical relations categorically, as shown by the following.

Proposition 2.6. *To give a binary logical relation from M to N is equivalent to giving a cartesian closed functor $R\colon L \to Rel_2$ such that $(\delta_0, \delta_1)R = (M, N)$:*

$$\begin{tikzcd} & Rel_2 \arrow[d, "{(\delta_0, \delta_1)}"] \\ L \arrow[ur, "R"] \arrow[r, "{(M, N)}"'] & Set \times Set \end{tikzcd}$$

Proof. Given a binary logical relation, one immediately has the object function of $R\colon L \to Rel_2$. The equation $(\delta_0, \delta_1)R = (M, N)$ determines the behaviour of R on arrows. The fact that, for any term t of type σ in context Γ, the pair $\big(M(\Gamma \vdash t\colon \sigma), N(\Gamma \vdash t\colon \sigma)\big)$ satisfies the condition making it an arrow in Rel_2 from R_Γ to R_σ follows from (and is equivalent to) the Basic Lemma.

The converse construction is given by taking the object part of a cartesian closed functor $R\colon L \to Rel_2$. It is routine to verify that the two constructions are mutually inverse. ∎

This situation generalises to categories other than *Set* and Rel_2, the central point being that both categories are cartesian closed and (δ_0, δ_1) is a cartesian closed functor. We outline the following important example which arises in domain theory to deal with logical relations in the context of least fixed points. Let C be the category of ω-cpos with $\perp$ and continuous maps, and M be the class of admissible monos; then there is an evident cartesian closed functor from $Sub_2(C, M)$, the category of admissible binary relations between cpos, to $C \times C$. A logical relation in this framework is then a cartesian closed functor from L to $Sub_2(C, M)$ (coherent with appropriate models of L in C).

Obviously, one can define a category Rel_n of n-ary relations for an arbitrary natural number n; Propositions 2.5 and 2.6 generalise routinely to arbitrary n.

3 Lax Logical Relations

In this section, we generalise the categorical notion of logical relation to what we call a *lax* logical relation.

Definition 3.1. *Given a signature Σ and the language L generated by Σ, and two models M and N of L in Set, a* (binary) lax logical relation from M to N *is a functor $R: L \to Rel_2$ that strictly preserves finite products and satisfies $(\delta_0, \delta_1)R = (M, N)$.*

Note that exponentials are not necessarily preserved. Evidently, one can adapt this definition to one for n-ary lax logical relations for arbitrary n.

The origin of our terminology is as follows. Any finite-product preserving functor $R: C \to D$ between cartesian closed categories induces a family of *lax* maps $App_{\sigma,\tau}: R_{\sigma\to\tau} \longrightarrow [R_\sigma \Rightarrow R_\tau]$, obtained by taking the Currying in D of the composites

$$R_{\sigma\to\tau} \times R_\sigma \longrightarrow R_{(\sigma\to\tau)\times\sigma} \longrightarrow R_\tau$$

where the first map is determined by preservation of finite products, and the second map is obtained by applying R to the evaluation map in C. This is an instance of (op)lax preservation of structure, specifically, exponential structure.

The notion of Henkin model is closely related to this definition. A Henkin model of the simply typed λ-calculus is a finite-product preserving functor from L to *Set* such that the induced lax maps are injective. This is a kind of lax model, but is not quite the same as giving a unary lax logical relation; nevertheless, it is a natural and useful generalisation of the notion of model we have used, and one to which our results routinely extend.

The Basic Lemma for logical relations extends to lax logical relations; in fact, the lax logical relations can be characterised in terms of a Basic Lemma.

Lemma 3.2 (Basic Lemma for Lax Logical Relations). *Let M and N be models of L in Set. A family of relations $R_\sigma \subseteq M_\sigma \times N_\sigma$ for every type σ of L determines a lax logical relation from M to N if and only if, for every term $t: \sigma$ of L in context Γ, if $x\, R_\Gamma\, y$, then $M(\Gamma \vdash t: \sigma)x\, R_\sigma\, N(\Gamma \vdash t: \sigma)y$,*

where $x\, R_\Gamma\, y$ is an abbreviation for $x_i\, R_{\sigma_i}\, y_i$ for all i when $\sigma_1, \ldots, \sigma_n$ is the sequence of types in Γ.

Proof. For the forward (only-if) direction, suppose Γ has sequence of types $\sigma_1, \ldots, \sigma_n$. The expression $\Gamma \vdash t\colon \sigma$ is a map in L from $\sigma_1 \times \cdots \times \sigma_n$ to σ, so R sends it to the unique map from $R_{\sigma_1 \times \cdots \times \sigma_n}$ to R_σ in Rel_2 that lifts the pair $\big(M(\Gamma \vdash t\colon \sigma), N(\Gamma \vdash t\colon \sigma)\big)$:

$$\begin{tikzcd}
M(\sigma_1 \times \cdots \times \sigma_n) \arrow[r, "M(\Gamma \vdash t\colon \sigma)"] \arrow[d, leftrightarrow, "R_{\sigma_1 \times \cdots \times \sigma_n}"'] & M(\sigma) \arrow[d, leftrightarrow, "R_\sigma"] \\
N(\sigma_1 \times \cdots \times \sigma_n) \arrow[r, "N(\Gamma \vdash t\colon \sigma)"] & N(\sigma)
\end{tikzcd}$$

If $x_i \, R_{\sigma_i} \, y_i$ for all i then $(x_1, \ldots, x_n) \, R_{\sigma_1 \times \cdots \times \sigma_n} \, (y_1, \ldots, y_n)$ because R preserves finite products, and so the result is now immediate, as $x \in M(\sigma_1 \times \cdots \times \sigma_n) = (x_1, \ldots, x_n) \in M(\sigma_1) \times \cdots \times M(\sigma_n)$ and similarly for y.

For the converse, first taking Γ to be a singleton, the condition uniquely determines maps $R(\Gamma \vdash t\colon \sigma)\colon R(\Gamma) \to R(\sigma)$ in Rel_2, giving a graph morphism from L to *Set* such that $(\delta_0, \delta_1)R = (M, N)$. Such a graph morphism is trivially necessarily a functor. Taking $\Gamma \vdash t\colon \sigma$ to be $\phi \vdash *\colon 1$, where $*$ is the unique constant of type 1, the condition yields $* \, R_1 \, *$, so R preserves the terminal object. Taking $\Gamma \vdash t\colon \sigma$ to be $a\colon \sigma_0, b\colon \sigma_1 \vdash (a, b)\colon \sigma_0 \times \sigma_1$ yields that if $x_0 \, R_{\sigma_0} \, y_0$ and $x_1 \, R_{\sigma_1} \, y_1$, then $(x_0, x_1) \, R_{\sigma_0 \times \sigma_1} \, (y_0, y_1)$. And taking $\Gamma \vdash t\colon \sigma$ to be $a\colon \sigma_0 \times \sigma_1 \vdash \pi_i \, a\colon \sigma_i$ for $i = 0, 1$ give the converse. So R preserves finite products. ■

We conclude this section by showing how lax logical relations can be used for the two applications of [HS99] previously discussed.

Definition 3.3. *If R is a type-indexed family of binary relations from M to N and S is a type-indexed family of binary relations from N to P, their* composite $R \,;\, S$ *is defined component-wise; i.e.,* $(R \,;\, S)_\sigma = R_\sigma \,;\, S_\sigma$

where ; on the right-hand side denotes the conventional composition of binary relations.

Proposition 3.4. *If R is a binary lax logical relation from M to N and S is a binary lax logical relation from N to P, then $R \,;\, S$ is a lax logical relation from M to P.*

Proof. We must show that if $R\colon L \to Rel_2$ and $S\colon L \to Rel_2$ strictly preserve finite products, then so does $R \,;\, S$. But $(x_0, x_1) \, (R \,;\, S)_{\sigma \times \tau} \, (y_0, y_1)$ if and only if there exists (z_0, z_1) such that $(x_0, x_1) \, R_{\sigma \times \tau} \, (z_0, z_1)$ and $(z_0, z_1) \, S_{\sigma \times \tau} \, (y_0, y_1)$, and that is so if and only if $x_0 \, (R \,;\, S)_\sigma \, y_0$ and $x_1 \, (R \,;\, S)_\tau \, y_1$. The proof for a terminal object is trivial. ■

Various other closure properties (such as closure with respect to conjunction and universal and existential quantification) have been proved in [HS99] for pre-logical relations; the results in the following section show that lax logical relations also have these closure properties.

Definition 3.5. *Let M and N be models of L in Set, and OBS be a set of types; then M and N are said to be* observationally equivalent with respect to *OBS (written $M \equiv_{OBS} N$) when, for all $\sigma \in OBS$ and all closed $t, t'\colon \sigma$, $M(t) = M(t')$ if and only if $N(t) = N(t')$.*

Proposition 3.6. *$M \equiv_{OBS} N$ if and only if there exists a lax logical relation from M to N which is one-to-one for every $\sigma \in OBS$.*

Proof. For the forward direction, consider the family of relations $R_\sigma \subseteq M_\sigma \times N_\sigma$ defined by $aR_\sigma b$ if and only if there exists a closed term $t\colon \sigma$ such that $M(t) = a$ and $N(t) = b$. This is one-to-one on observable types because of observational equivalence and a lax logical relation because $R_{\sigma\times\tau} = R_\sigma \times R_\tau$.

For the converse, suppose $R_\sigma \subseteq M_\sigma \times N_\sigma$ determine a lax logical relation; if $\sigma \in OBS$ then, for all closed $t\colon \sigma$, $M(t)\ R_\sigma\ N(t)$ by the Basic Lemma and $M(t) = M(t')$ if and only if $N(t) = N(t')$ because R_σ is one-to-one. ∎

4 Pre-logical Relations

We can use the Basic Lemma of Sect. 3 and the corresponding result of [HS99] to see immediately that, for models as we defined them in Sect. 2, the notions of lax logical relation and pre-logical relation coincide. However, in this section we give a more direct exposition of the connection for a larger class of models. In [HS99], the analysis is primarily in terms of the simply typed λ-calculus without product types. But they mention the case of λ-calculi with products and models that satisfy surjective pairing. Hence, consider models now to be functors $M\colon L \longrightarrow \mathit{Set}$ that strictly preserve finite products (but not necessarily exponentials); these include Henkin models. Everything we have said about lax logical relations extends routinely to this class of models.

Definition 4.1. *A* pre-logical relation from M to N *consists of, for each type σ, a relation $R_\sigma \subseteq M_\sigma \times N_\sigma$ such that*

1. *if $x\ R_\sigma\ y$ and $f\ R_{\sigma\to\tau}\ g$, then $App_{\sigma,\tau} f x\ R_\tau\ App_{\sigma,\tau} g y$, where maps $App_{\sigma,\tau}$ are determined by finite-product preservation of M and N, respectively, as discussed in Sect. 3;*
2. *$M(c)\ R_\sigma\ N(c)$ for every constant c of type σ, where the constants are deemed to include*
   - *all constants in Σ,*
   - *$*\colon 1$,*
   - *$(-,-)\colon \sigma \to \tau \to (\sigma \times \tau)$,*
   - *$\pi_0 : \sigma \times \tau \to \sigma$ and $\pi_1 : \sigma \times \tau \to \tau$, and*
   - *all instances of combinators $S_{\rho,\sigma,\tau}\colon (\rho \to \sigma \to \tau) \to (\rho \to \sigma) \to \rho \to \tau$ and $K_{\sigma,\tau}\colon \sigma \to \tau \to \sigma$.*

Theorem 4.2. *A type-indexed family of relations $R_\sigma \subseteq M_\sigma \times N_\sigma$ determines a lax logical relation from M to N if and only if it is a pre-logical relation from M to N.*

Proof. For the second clause in the forward direction, treat all constants as maps in L with domain 1. For the first clause, note that $R_\sigma \times R_{\sigma\to\tau} = R_{\sigma\times(\sigma\to\tau)}$, so applying functoriality of R to the evaluation map $ev\colon \sigma \times (\sigma \to \tau) \longrightarrow \tau$ in L, the result follows immediately.

For the converse, the second condition implies that, for all closed terms t, $M(t)\,R\,N(t)$; that fact, combined with the fact that every map in L is an un-Currying of a closed term, plus the first condition, imply that R is a graph morphism making $(\delta_0, \delta_1)R = (M, N)$, hence trivially a functor. Since $*\colon 1$ is a constant and $M(*) = * = N(*)$, we have $M(*)\,R_1\,N(*)$; so R preserves the terminal object. Since $(-,-)$ is a constant, it follows that if $x_0\,R_{\sigma_0}\,y_0$ and $x_1\,R_{\sigma_1}\,y_1$, then $(x_0, x_1)\,R_{\sigma_0\times\sigma_1}\,(y_0, y_1)$. The inverse holds because π_0 and π_1 are maps in L. So R preserves finite products. ■

5 Another Syntax-Based Characterisation

The key point in the pre-logical characterisation above is that every map in the category L is generated by the constants. In this section, we give an alternative syntax-based characterisation that generalizes more directly to other languages. For simplicity of exposition, we assume, as previously, that models preserve exponentials as well as products.

Theorem 5.1. *To give a lax logical relation from M to N is equivalent to giving, for each type σ of L, a relation $R_\sigma \subseteq M_\sigma \times N_\sigma$ such that*

1. *if $f\,R_{(\sigma\times\tau)\to\rho}\,g$, then $Curry(f)\,R_{\sigma\to\tau\to\rho}\,Curry(g)$*
2. *$App\,R_{((\sigma\to\tau\times\sigma)\to\tau}\,App$*
3. *if $f_0\,R_{\sigma\to\tau}\,g_0$ and $f_1\,R_{\sigma\to\rho}\,g_1$, then $(f_0, f_1)\,R_{\sigma\to(\tau\times\rho)}\,(g_0, g_1)$*
4. *$\pi_0\,R_{\sigma\times\tau\to\sigma}\pi_0$ and $\pi_1\,R_{\sigma\times\tau\to\tau}\,\pi_1$*
5. *if $f\,R_{\sigma\to\tau}\,g$ and $f'\,R_{\tau\to\rho}\,g'$, then $(f'\cdot f)R_{\sigma\to\rho}(g'\cdot g)$*
6. *$id\,R_{\sigma\to\sigma}\,id$*
7. *$x\,R_\sigma\,y$ if and only if $x\,R_{1\to\sigma}\,y$*
8. *$M(c)\,R_\sigma\,N(c)$ for every base term c in Σ of type σ.*

We chose the conditions above because the first four conditions seem particularly natural from the perspective of the λ-calculus, the following two, which are about substitution, are natural category theoretic conditions, the seventh is mundane, and the last evident; *cf.* the “categorical combinators” of [Cu93].

Proof. For the forward direction, the relations R_σ are given by the object part of the functor. The conditions follow immediately from the fact of R being a functor, thereby having an action on all maps, and from the fact that it strictly preserves finite products. For instance, there is a map in L from $(\sigma\to\tau)\times(\sigma\to\rho)$ to $\sigma\to(\tau\times\rho)$, so that map is sent by R to a map in Rel_2, and R strictly preserves finite products, yielding the third condition. So using the definition of a map in Rel_2, and the facts that $(\delta_0, \delta_1)R = (M, N)$ and that M and N are strict structure preserving functors, we have the result.

For the converse, the family of relations gives the object part of the functor R. Observe that the axioms imply

- $(x_0, x_1)\,R_{\sigma\times\tau}\,(y_0, y_1)$ if and only if $x_0\,R_\sigma\,y_0$ and $x_1\,R_\tau\,y_1$
- $*\,R_1\,*$, where $*$ is the unique element of $M_1 = N_1 = 1$

So, R strictly preserves finite products providing it forms a functor. The data for M and N and the desired coherence condition $(\delta_0, \delta_1)R = (M, N)$ on the putative functor determine its behaviour on maps. It remains to check that the image of every map in L actually lies in Rel_2. But the conditions inductively define the Currying of every map in L, so unCurrying by the fifth and seventh conditions, the result follows. It is routine to verify that these constructions are mutually inverse. ∎

The result holds for the more general class of models we have discussed, but an exposition would be encumbered by numerous occurrences of $App_{\sigma,\tau}$. It is routine to generalise Theorems 4.2 and 5.1 to n-ary relations for arbitrary n.

6 Models in Cartesian Closed Categories

Cartesian closed categories are a more general class of models for typed lambda calculi. In this section, we consider a model to be a functor from L to a cartesian closed category, strictly preserving finite products and exponentials.

To discuss "relations" in this context, we adopt the sub-scone approach described in [La88, MR91, MS92, Al95]. Let C be a cartesian closed category, S be a finitely complete cartesian closed category, and $G: C \to S$ be a functor that preserves products (up to isomorphism). A typical example of a suitable functor G is $\hom(1, -): C \to Set$, the global-elements functor; other examples may be found in the references given above. Then these data determine a category G-Rel_2 of *categorical (binary) relations on* C as follows.

Let $Rel_2(S)$ be the category of binary relations on S with evident forgetful functor $Rel_2(S) \longrightarrow S \times S$; then pulling back along $G \times G$ determines a category G-Rel_2 and a forgetful functor to $C \times C$. In detail, the objects of G-Rel_2 are triples (a_0, s, a_1) where a_0 and a_1 are objects of C and s is a sub-object of $G(a_0) \times G(a_1)$; the morphisms from (a_0, s, a_1) to (b_0, t, b_1) are triples (f_0, q, f_1) such that $f_i: a_i \to b_i$ in C, $q: \mathrm{dom}\, s \to \mathrm{dom}\, t$ in S, and the following diagram commutes:

$$\begin{tikzcd} \cdot \arrow[r, tail, "s"] \arrow[d, "q"'] & G(a_0) \times G(a_1) \arrow[d, "G(f_0) \times G(f_1)"] \\ \cdot \arrow[r, tail, "t"] & G(b_0) \times G(b_1) \end{tikzcd}$$

Composition and identities are evident. The forgetful functors $\delta_i: G\text{-}Rel_2 \to C$ for $i = 0, 1$ are defined by $\delta_i(a_0, s, a_1) = a_i$ and similarly for morphisms.

Proposition 6.1. *G-Rel_2 is a cartesian closed category and the cartesian closed structure is strictly preserved by $(\delta_0, \delta_1): G\text{-}Rel_2 \longrightarrow C \times C$; furthermore, this functor is faithful.*

Definition 6.2. *Given a signature Σ and the language L generated by Σ, two models M and N of L in a cartesian closed category C, and a category G-Rel_2 of binary categorical relations on C, a* (binary) lax logical relation from M to N *is a functor $R: L \longrightarrow G\text{-}Rel_2$ that satisfies $(\delta_0, \delta_1)R = (M, N)$ and strictly preserves finite products.*

Lemma 6.3 (Basic Lemma for Categorical Lax Logical Relations). *Let M and N be models of L in a cartesian closed category C, S be a finitely complete cartesian closed category, and $G\colon C \to S$ preserve finite products up to isomorphism; then a family of sub-objects $R_\sigma\colon \cdot \rightarrowtail G(M_\sigma) \times G(N_\sigma)$ for every type σ of L determines a lax logical relation from M to N if and only if, for every term t of L of type σ in context Γ, there exists a unique map q that makes the following diagram commute:*

$$\begin{tikzcd} \cdot \arrow[r, tail, "\Pi_i R_{\sigma_i}"] \arrow[d, dotted, "q"'] & G(\Pi_i M_{\sigma_i}) \times G(\Pi_i N_{\sigma_i}) \arrow[d, "G(M(t)) \times G(N(t))"] \\ \cdot \arrow[r, tail, "R_\sigma"] & G(M_\sigma) \times G(N_\sigma) \end{tikzcd}$$

where $\sigma_1, \ldots, \sigma_n$ is the sequence of types in Γ.

Proof. For the forward direction, R maps $\Gamma \vdash t\colon \sigma$ to a map

$$\begin{tikzcd} \cdot \arrow[r, tail, "R_{\Pi_i \sigma_i}"] \arrow[d, "q"'] & G(M_{\Pi_i \sigma_i}) \times G(M_{\Pi_i \sigma_i}) \arrow[d, "G(M(t)) \times G(N(t))"] \\ \cdot \arrow[r, tail, "R_\sigma"] & G(M_\sigma) \times G(N_\sigma) \end{tikzcd}$$

The result follows because R, M and N preserve products.

In the converse direction, the morphism part of the functor is determined by the assumed maps. Taking $\Gamma \vdash t\colon \sigma$ to be $a\colon \sigma_0, b\colon \sigma_1 \vdash (a, b)\colon \sigma_0 \times \sigma_1$ shows that $R_{\sigma_0} \times R_{\sigma_1} \leq R_{\sigma_0 \times \sigma_1}$, using the fact that G, M and N all preserve products, and taking $\Gamma \vdash t\colon \sigma$ to be $p\colon \sigma_0 \times \sigma_1 \vdash \pi_i p\colon \sigma_i$ for $i = 0, 1$ shows the converse. Finally, taking $\Gamma \vdash t\colon \sigma$ to be $\emptyset \vdash *\colon 1$ shows that R_1 is the "true" sub-object of $G(M_1) \times G(N_1)$. So R preserves products. ∎

This result can be generalised: replace G-Rel_2 and (δ_0, δ_1) by any category D with finite products and a faithful finite-product preserving functor to $C \times C$. This would amount to a lax version of Peter Freyd's suggestion [Mi90, Section 3.6.4] of studying logical relations as subcategories that respect cartesian-closed (here, cartesian) structure, except generalised from subcategory to faithful functor. But many applications require entailments to, or from, the "relations," and so a lax version of Hermida's [He93] fibrations with structure to support a $(\top, \wedge, \Rightarrow, \forall)$ logic might be a more appropriate level of generality.

To consider composition of (binary) lax logical relations in this context, assume first that S is the usual category of sets and functions; then the objects of G-Rel_2 are subsets of sets of the form $G(a) \times G(b)$.

Proposition 6.4. *Composition of (binary) categorical lax logical relations can be defined component-wise.*

To allow recursion in L, consider again the category $Sub_2(C, M)$ discussed at the end of Section 2, with C being the category of ω-cpos with $\perp$ and M being the admissible monos. Using the sconing functor $G = C(1, -)\colon C \to Set$ gives us a category G-Rel_2 as above; because this is constructed as a pullback, there exists a strict finite-product preserving functor F from $Sub_2(C, M)$ to G-Rel_2.

Given any logical relation functor $R: L \to Sub_2(C, M)$, composing with F gives a strict finite-product preserving functor from L to G-Rel_2 (i.e., a lax logical relation) between the original models. This shows how composition is supported in the context of relations on ω-cpos.

More generally, if S is assumed to be a *regular* category [Bo94], a relational composition can be defined. Any pre-sheaf category Set^W, or indeed any topos, is a regular category, so this is a mild assumption. An *axiomatic* treatment of composition of generalized logical relations, including lax logical relations as discussed here, can be found in [KO$^+$97], which emerged from category theoretic treatments of data refinement in which composition of refinements is crucial [JH90, KP96, KP].

7 Generalising from the λ-Calculus

In Sect. 3, the fundamental facts that gave rise to our definition of lax logical relation were the correspondence between the simply typed λ-calculus and cartesian closed categories, and the fact that a signature Σ gave rise to a cartesian closed category L such that a model of Σ could be seen as a functor from L into *Set* (or, more generally, any cartesian closed category) that strictly preserved cartesian closed structure. So in generalising from the simply typed λ-calculus, we generalise the latter fact. This may be done in terms of algebraic structure, or equivalently (finitary) monads, on *Cat*. The central paper about that is Blackwell, Kelly and Power's [BKP89]. We can avoid much of the subtlety here by restricting our attention to maps that preserve structure strictly.

We shall first describe the situation for an arbitrary (finitary) monad T on *Cat* extending finite-product structure. One requires *Set* (or, more generally, any small category C) to have T-structure, L to be the free T-algebra generated by a signature, and define a model M of L to be a strict T-algebra map, *cf.* [KO$^+$97].

A natural general setting in which to define the notion of lax logical relation involves assuming the existence of a small category E (with finite products) of relations, and a strict finite-product preserving forgetful functor (δ_0, δ_1) from E to $C \times C$. One then adds to these data further categorical structure inside the category of small categories and functors that strictly preserve finite products to generalise the composition of binary relations. These definitions and related results appear in [KO$^+$97]. Here, we aim to state a Basic Lemma in familiar terms, and so restrict attention to the special case that $C = Set$ and $E = Rel_2$.

A *lax logical relation* is a strict finite-product preserving functor from L into Rel_2 such that composition with (δ_0, δ_1) yields (M, N).

We can generalise the Basic Lemma to this level of generality as follows.

Lemma 7.1 (Basic Lemma for Lax Logical Relations with Algebraic Structure). *A family of relations $R_\sigma \subseteq M_\sigma \times N_\sigma$ for every type σ of L determines a lax logical relation from M to N if and only if, for every term t of L of type σ in context Γ, if $x\, R_\Gamma\, y$, then $M(\Gamma \vdash t\colon \sigma)x\, R_\sigma\, N(\Gamma \vdash t\colon \sigma)y$*

The proof is exactly as in Sect. 3; similarly,

Proposition 7.2. *Binary lax logical relations (at the current level of generality) compose component-wise.*

In the above we have tacitly assumed that contexts are modelled by finite products. In general, there is no need to make this assumption: contexts could be modelled by a symmetric monoidal structure or, more generally, by a symmetric pre-monoidal structure, or Freyd structure. It would be straightforward to generalise our analysis to include such possibilities, but it may be simpler to deal with them case by case. For an analysis of the notion of lax logical relations where contexts are modelled by a Freyd structure, see [KP99].

We next want to generalise Theorem 5.1. In order to do that, we need to consider the formulation of finitary monads in terms of algebraic structure on *Cat*, and we need to restrict to a particular class of such structures. The general notion of algebraic structure, and the relevant results, appear in [KP93] and [Po97], and we have included it in the Appendix. Using the notation of the Appendix, we consider a special class of algebraic structure.

Definition 7.3. *Algebraic structure* (S, E) *on Cat is* discrete *if* $S(c) = 0$ *whenever* c *is not a discrete category, i.e., whenever* c *is not the discrete category on a finite set.*

It follows from the definition that any discrete algebraic structure may be presented by two families of operations: *object* operations, which have algebras given by functors of the form $C^k \longrightarrow C$, and *arrow* operations, which are given by natural transformations between object operations. One may put equations between these to obtain all operations of any discrete algebraic structure, which are given by functors $C^k \longrightarrow C^{Sk}$, where Sk is a small category.

Assuming *Set* has (S, E)-structure for some given discrete algebraic structure (S, E), a *model* of an (S, E)-algebra in *Set* is a functor that strictly preserves (S, E)-structure.

Examples of discrete algebraic structure have models given by small categories with finite products, with finite coproducts, with monoidal structure, symmetric monoidal structure, a monad [Mo91], an endofunctor, a natural transformation between endofunctors, or any combination of the above.

In order to extend Theorem 5.1, rather than give an analogue, we must include exponentials, although they are not instances of discrete algebraic structure as we have defined it. So we henceforth assume that we are given discrete algebraic structure on *Cat* extending finite-product structure; that L is generated by the simply typed λ-calculus, a signature, and the discrete algebraic structure; that M and N are models of L in *Set* strictly preserving the algebraic structure, and, restricting our definition above, that a *lax logical relation* from M to N is a finite-product preserving functor from L to Rel_2 such that composition with (δ_0, δ_1) yields (M, N).

A methodology for extending Theorem 5.1 is as follows. Algebraic structure on *Cat* is given by an equational presentation. That equational presentation has operations defining objects and arrows. For each operation defining an arrow, one adds an axiom to the list in the statement of Theorem 5.1 in the same spirit. For instance, to define a monoidal structure, one has operations that assign to each pair of maps (f, g), a map $f \otimes g$, and gives associative maps and their inverses, and left and right unit maps and their inverses. So one would add axioms

- if $f_0 \, R_{\sigma_0 \to \tau_0} \, g_0$ and $f_1 \, R_{\sigma_1 \to \tau_1} \, g_1$ then $(f_0 \otimes f_1) \, R_{(\sigma_0 \otimes \sigma_1) \to (\tau_0 \otimes \tau_1)} \, (g_0 \otimes g_1)$;
- $a \, R_{(\sigma \otimes \tau) \otimes \rho \to \sigma \otimes (\tau \otimes \rho)} \, a$, $l \, R_{(\sigma \otimes I) \to \sigma} \, l$ and $r \, R_{(I \otimes \sigma) \to \sigma} \, r$;
- $a^{-1} \, R_{\sigma \otimes (\tau \otimes \rho) \to (\sigma \otimes \tau) \otimes \rho} \, a^{-1}$, $l^{-1} \, R_{\sigma \to (\sigma \otimes I)} \, l^{-1}$ and $r^{-1} \, R_{\sigma \to (I \otimes \sigma)} \, r^{-1}$.

Theorem 7.4. *For any discrete algebraic structure on Cat extending finite-product structure, to give a lax logical relation from M to N is equivalent to giving a family of relations $R_\sigma \subseteq M_\sigma \times N_\sigma$ satisfying the conditions of Theorem 5.1 and also*

- *for any k-ary object operation O, if $f_i \, R_{\sigma_i \to \tau_i} \, g_i$ for all $1 \leq i \leq k$, then*

$$O(f_1, \ldots, f_k) \, R_{O(\sigma_1, \ldots, \sigma_k) \to O(\tau_1, \ldots, \tau_k)} \, O(g_1, \ldots, g_k)$$

- *for any k-ary arrow operation O, we have*

$$O(M\sigma_1, \ldots, M\sigma_k) \, R_\gamma \, O(N\sigma_1, \ldots, N\sigma_k)$$

  where $\gamma = \mathrm{dom}\, O(\sigma_1, \ldots, \sigma_k) \longrightarrow \mathrm{cod}\, O(\sigma_1, \ldots, \sigma_k)$
- *for any k-ary arrow operation O, if $f_i \, R_{\sigma_i \to \tau_i} \, g_i$ for all $1 \leq i \leq k$, then*

$$\mathrm{dom}\, O(f_1, \ldots, f_k) \, R_\beta \, \mathrm{dom}\, O(g_1, \ldots, g_k)$$

  where $\beta = \mathrm{dom}\, O(\sigma_1, \ldots, \sigma_k) \longrightarrow \mathrm{dom}\, O(\tau_1, \ldots, \tau_k)$, *and similarly with* dom *systematically replaced by* cod.

The final two rules here may seem unfamiliar at first sight. An arrow operation takes a k-ary family of objects to an arrow, so syntactically, takes a k-ary family of types to an equivalence class of terms. That leads to our penultimate rule. That a k-ary arrow operation is functorial means that every k-ary family of arrows is sent to a commutative square. So we need rules to the effect that every arrow in that square behaves as required. The penultimate rule above accounts for two arrows of the square, and the final rule accounts for the other two, where the domain of the commutative square is the arrow of the square uniquely determined by the definitions.

Proof. Follow the proof of Theorem 5.1. The conditions show inductively that for every arrow of the category freely generated by the given discrete algebraic structure applied to the signature, one obtains an arrow in *Rel*. ∎

Note that this allows for dropping exponentials, as well as adding various kinds of structure such as finite co-products and tensor products. We hope this generality will lead to interesting new applications.

References

[Ab90] S. Abramsky. Abstract interpretation, logical relations and Kan extensions. *J. of Logic and Computation*, 1:5–40, 1990.

[Al95] M. Alimohamed. A characterization of lambda definability in categorical models of implicit polymorphism. *Theoretical Computer Science*, 146:5–23, 1995.

[BKP89] R. Blackwell, H. M. Kelly, and A. J. Power. Two dimensional monad theory. *J. of Pure and Applied Algebra*, 59:1–41, 1989.

[Bo94] Francis Borceux. *Handbook of Categorical Algebra 2*, volume 51 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.

[Cu93] P.-L. Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming*. Birkhauser, Boston, 1993.

[FRA99] J. Flum and M. Rodriguez-Artalejo, editors. *Computer Science Logic, 13th International Workshop, CSL'99*, volume 1683 of *Lecture Notes in Computer Science*, Madrid, Spain, September 1999. Springer-Verlag, Berlin (1999).

[Gi68] A. Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968.

[He93] Claudio A. Hermida. *Fibrations, logical predicates, and indeterminates*. Ph.D. thesis, The University of Edinburgh, 1993. Available as Computer Science Report CST–103–93 or ECS–LFCS–93–277.

[HL^+] F. Honsell, J. Longley, D. Sannella, and A. Tarlecki. Constructive data refinement in typed lambda calculus. To appear in the Proceedings of FOSSACS 2000, Springer-Verlag *Lecture Notes in Computer Science*.

[HS99] F. Honsell and D. Sannella. Pre-logical relations. In Flum and Rodriguez-Artalejo [FRA99], pages 546–561.

[JH90] He Jifeng and C. A. R. Hoare. Data refinement in a categorical setting. Technical monograph PRG-90, Oxford University Computing Laboratory, Programming Research Group, Oxford, November 1990.

[JT93] A. Jung and J. Tiuryn. A new characterization of lambda definability. In M. Bezen and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 245–257, Utrecht, The Netherlands, March 1993. Springer-Verlag, Berlin.

[KO^+97] Y. Kinoshita, P. O'Hearn, A. J. Power, M. Takeyama, and R. D. Tennent. An axiomatic approach to binary logical relations with applications to data refinement. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, pages 191–212, Sendai, Japan, 1997. Springer-Verlag, Berlin.

[KP] Y. Kinoshita and A. J. Power. Data refinement by enrichment of algebraic structure. To appear in *Acta Informatica*.

[KP93] G. M. Kelly and A. J. Power. Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads. *Journal of Pure and Applied Algebra*, 89:163–179, 1993.

[KP96] Y. Kinoshita and A. J. Power. Lax naturality through enrichment. *J. Pure and Applied Algebra*, 112:53–72, 1996.

[KP99] Y. Kinoshita and J. Power. Data refinement for call-by-value programming languages. In Flum and Rodriguez-Artalejo [FRA99], pages 562–576.

[La88] Y. Lafont. *Logiques, Categories et Machines*. Thèse de Doctorat, Université de Paris VII, 1988.

[Mi71] R. Milner. An algebraic definition of simulation between programs. In *Proceedings of the Second International Joint Conference on Artificial Intelligence*, pages 481–489. The British Computer Society, London, 1971. Also Technical

Report CS-205, Computer Science Department, Stanford University, February 1971.

[Mi90] J. C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 365–458. Elsevier, Amsterdam, and The MIT Press, Cambridge, Mass., 1990.

[Mi91] J. C. Mitchell. On the equivalence of data representations. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 305–330. Academic Press, 1991.

[Mi96] J. C. Mitchell. *Foundations for Programming Languages*. The MIT Press, 1996.

[Mo91] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, July 1991.

[MR91] QingMing Ma and J. C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics, Proceedings of the 7th International Conference*, volume 598 of *Lecture Notes in Computer Science*, pages 1–40, Pittsburgh, PA, March 1991. Springer-Verlag, Berlin (1992).

[MS76] R. E. Milne and C. Strachey. *A Theory of Programming Language Semantics*. Chapman and Hall, London, and Wiley, New York, 1976.

[MS92] J. C. Mitchell and A. Scedrov. Notes on sconing and relators. In E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and M. M. Richter, editors, *Computer Science Logic: 6th Workshop, CSL '92: Selected Papers*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378, San Miniato, Italy, 1992. Springer-Verlag, Berlin (1993).

[OR95] P. O'Hearn and J. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120(1):107–116, 1995.

[OT95] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *J. ACM*, 42(3):658–709, May 1995.

[Pl73] G. D. Plotkin. Lambda-definability and logical relations. Memorandum SAI-RM-4, School of Artificial Intelligence, University of Edinburgh, October 1973.

[Pl80] G. D. Plotkin. Lambda-definability in the full type hierarchy. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 363–373. Academic Press, 1980.

[Po97] A. J. Power. Categories with algebraic structure. In M. Nielsen and W. Thomas, editors, *Computer Science Logic, 11th International Workshop, CSL'99*, volume 1414 of *Lecture Notes in Computer Science*, pages 389–405, Aarhus, Denmark, August 1997. Springer-Verlag, Berlin (1998).

[Re74] J. C. Reynolds. On the relation between direct and continuation semantics. In J. Loeckx, editor, *Proc. 2nd Int. Colloq. on Automata, Languages and Programming*, volume 14 of *Lecture Notes in Computer Science*, pages 141–156. Springer-Verlag, Berlin, 1974.

[Re83] J. C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83*, pages 513–523, Paris, France, 1983. North-Holland, Amsterdam.

[Sc87] O. Schoett. *Data abstraction and the correctness of modular programming*. Ph.D. thesis, University of Edinburgh, February 1987. Report CST-42-87.

[St96] I. Stark. Categorical models for local names. LISP *and Symbolic Computation*, 9(1):77–107, February 1996.

Appendix: Algebraic Structure on Categories

In ordinary universal algebra, an algebra is a set X together with a family of basic operations $\sigma\colon X^n \to X$, subject to equations between derived operations. In order to define algebraic structure on categories, one must replace the set X by a category A. One also replaces the finite number n by a finitely presentable category c. All finite categories are finitely presentable, and finite categories are the only finitely presentable categories we need in this paper. One also allows not only functions from the set $Cat(c, A)$ into the set of objects of A, but also functions from the set $Cat(c, A)$ into the set of arrows in A. These are subject to equations between derived operations. It follows that the category of small such categories with structure and functors that strictly preserve the structure is equivalent to the category of algebras, T-Alg, for a finitary monad T on Cat.

All structures relevant to this paper are instances of a slightly more restricted situation: that of Cat-enriched algebraic structure. So we shall restrict to Cat-enriched structures here. Let C denote the 2-category Cat of small categories. So $C(A, B)$ denotes the category of functors from A to B. Let C_f denote the full sub-2-category of C given by (isomorphism classes of) finitely presentable categories.

Definition A.1. *A* signature *on C is a 2-functor $S\colon \mathrm{ob}\, C_f \longrightarrow C$, regarding* $\mathrm{ob}\, C_f$ *as a discrete 2-category.*

For each $c \in \mathrm{ob}\, C_f$, $S(c)$ is called the category of *basic operations of arity c*. Using S, we construct $S_\omega\colon C_f \longrightarrow C$ as follows: set

$S_0 = J$, the inclusion of C_f into C, and
$S_{n+1} = J + \sum_{d \in \mathrm{ob}\, C_f} C\big(d, S_n(-)\big) \times S(d)$;

and define

$\sigma_0\colon S_0 \to S_1$ to be $inj\colon J \longrightarrow J + \sum_{d \in \mathrm{ob}\, C_f} C\big(d, S_0(-)\big) \times S(d)$; and
$\sigma_{n+1}\colon S_{n+1} \to S_{n+2}$ to be $J + \sum_{d \in \mathrm{ob}\, C_f} C\big(d, \sigma_n(-)\big) \times S(d)$.

Then $S_\omega = \mathrm{colim}_{n<\omega} S_n$, where the colimit exists because C is cocomplete, and it is a colimit in a functor category with base C. In many cases of interest, each σ_n is a monomorphism, so S_ω is the union of $\{S_n\}_{n<\omega}$. For each c, we call $S_\omega(c)$ the category of *derived c-ary operations*.

A signature is typically accompanied by equations between derived operations. So we say

Definition A.2. *The* equations *of an algebraic theory with signature S are given by a* 2*-functor $E\colon \mathrm{ob}\, C_f \longrightarrow C$ together with 2-natural transformations $\tau_1, \tau_2\colon E \longrightarrow S_\omega\big(K(-)\big)$, where $K\colon \mathrm{ob}\, C_f \longrightarrow C_f$ is the inclusion.*

Definition A.3. Algebraic structure *on C consists of a signature S, together with equations (E, τ_1, τ_2).*

We generally denote algebraic structure by (S, E), suppressing τ_1 and τ_2.

We now define the algebras for a given algebraic structure.

Definition A.4. *Given a signature S, an S-*algebra *consists of a small category A together with a functor $\nu_c\colon C(c, A) \longrightarrow C\big(S(c), A\big)$ for each c.*

So, an S-algebra consists of a carrier A and an interpretation of the basic operations of the signature. This interpretation extends canonically to the derived operations, giving an $S_\omega\big(K(-)\big)$-algebra, as follows.

- $\nu_0\colon C(c,A) \longrightarrow C\big(S_0(c),A\big)$ is the identity;
- using the fact that $C(-,A)$ preserves colimits, to give a functor ν_{n+1} from $C(c,A)$ to $C\big(S_{n+1}(c),A\big)$ is equivalent to giving a functor from $C(c,A)$ to $C(c,A)$, which we will make the identity, and, for each d in $\mathrm{ob}\, C_f$, a functor from $C(c,A)$ to $C\big(C(d,S_n(c)),C(S(d),A)\big)$ or, equivalently, a functor from $C(c,A)\times C\big(d,S_n(c)\big)$ to $C\big(S(d),A\big)$ which can be inductively defined by

$$\begin{tikzcd}
C(c,A)\times C\big(d,S_n(c)\big) \arrow[d, "\nu_n\times \mathrm{id}"] \\
C\big(S_n(c),A\big)\times C\big(d,S_n(c)\big) \arrow[d, "\mathrm{comp}"] \\
C(d,A) \arrow[d, "\nu_d"] \\
C\big(S(d),A\big)
\end{tikzcd}$$

Definition A.5. *Given algebraic structure (S,E), an (S,E)-*algebra *is an S-algebra that satisfies the equations, i.e., an S-algebra (A,ν) such that both legs of*

$$\begin{tikzcd}
C(c,A) \arrow[r, "\nu_\omega"] & C\big(S_\omega(Kc),A\big) \arrow[r, shift left, "C(\tau_{1c},A)"] \arrow[r, shift right, "C(\tau_{2c},A)"'] & C\big(E(c),A\big)
\end{tikzcd}$$

agree.

Given (S,E)–algebras (A,ν) and (B,δ), we define the hom-category

$$(S,E)\text{-}Alg\big((A,\nu),(B,\delta)\big)$$

to be the equaliser in C of

$$\begin{tikzcd}
C(A,B) \arrow[rr, "\{C(S(c),-)\}_{c\in \mathrm{ob}\, C_f}"] \arrow[d, "\{C(S(c),-)\}_{c\in \mathrm{ob}\, C_f}"] & & \prod_c C(C(c,A),C(c,B)) \arrow[d, "\prod_c C(C(c,A),\delta_c)"'] \\
\prod_c C(C(S(c),A),C(S(c),B)) \arrow[rr, "\prod_c C(\nu_c, C(S(c),B))"'] & & \prod_c C(C(c,A),C(S(c),B))
\end{tikzcd}$$

This agrees with our usual universal-algebraic understanding of the notion of homomorphism of algebras, internalising it to C. (S,E)-*Alg* can then be made into a 2-category in which composition is induced by that in C. An arrow in (S,E)-*Alg* is a functor $F\colon A \to B$ such that, for all finitely presentable c,

$$F\nu_c(-) = \delta_c(F-)\colon C(c,A) \longrightarrow C\big(S(c),B\big)$$

i.e., a functor that commutes with all basic c-ary operations for all c.

A special case of the main result of [KP93] says

Theorem A.6. *A 2-category is equivalent to (S,E)-Alg for algebraic structure (S,E) on C if and only if there is a finitary 2-monad T on C such that the 2-category is equivalent to T-Alg.*

See [Po97] for an account directed towards a computer science readership.