ISSN 1361 - 6161

THE UNIVERSITY of MANCHESTER

Computer Science
University of Manchester

A Theory of Classes: Proofs and Models

Barnaby P. Hilken and David E. Rydeheard

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-96-2-1

A Theory of Classes: Proofs and Models*

Barnaby P. Hilken and David E. Rydeheard

Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.
der@cs.man.ac.uk

13th February 1996

*To appear in *Mathematical Structures in Computer Science*, C.U.P. 1996.

Abstract

We investigate the proof structure and models of theories of classes, where classes are 'collections' of entities. The theories are weaker than set theories and arise from a study of type classes in programming languages, as well as comprehension schemata in categories. We introduce two languages of proofs, one a simple type theory and the other involving proof environments for storing and retrieving proofs. The relationship between these languages is defined in terms of a normalisation result for proofs. We use this result to define a categorical semantics for classes and establish its coherence. Finally, we show how the formal systems relate to type classes in programming languages.

1 Introduction

The *classes* of the title are collections of entities. We choose this term as more neutral than *sets*, as we have a precise idea of the rather primitive theory we wish to explore which is quite distant from a set theory. The theory focuses on the *extent* (or *extension* or *abstraction*) or, as we shall say, the *comprehension* of a predicate ϕ, by which we mean the *class* of entities of class K which satisfy ϕ, a class which we write, using a fairly standard notation, as $\{x : K \mid \phi\}$ where x is a variable possibly occurring in formula ϕ.

The formal systems we present arise from two sources. One source is the notion of *type classes* in programming languages. Type classes were introduced in [Wadler, Blott 89] and provide a systematic treatment of implicit conversion between structures using operator overloading. Type classes have been incorporated in functional languages such as Haskell [Hudak, Wadler 1990] and Gofer [Jones 1992]. The reader need not know of type classes to understand this paper, but should be aware that the theory we present is a reformulation of (and extension of) the theory of type classes described in [Hilken, Rydeheard 1992]. There we described a primitive functional language with type classes and gave a semantics in category-theoretic terms. A major part of the work lay in showing that this semantics was properly defined, needing a so-called *coherence* result. The present paper attempts a reconstruction of a language of type classes as a language of proofs of a simple logic. In this way, we extend the propositions-as-types correspondence (see, for example, [Girard *et al.* 1989]) to cover type classes. However, in doing so we deviate from standard methods of presenting logics and the formulation here is closer to a programming language in that it includes 'environments' as well as contexts.

The other source for this theory of classes is the notion of a *comprehension schema* introduced into category theory in [Lawvere 1970]. Lawvere formulates a comprehension schema as an adjoint in an indexed category and shows how such schemata capture the 'extent' of a predicate. We describe comprehension schemata later in the paper and use this link to categories to provide models for the formal systems of classes.

There is a superficial resemblance between the formal systems presented in this paper and set theories, especially those in Natural Deduction form as in [Fitch 1952], [Prawitz 1965] and [Hallnäs 1988]. However, what we present differs from these set theories in that, in the systems in this paper, (1) there are no variables ranging over classes, and (2) there is no membership predicate, so that in the expression $\{x : K \mid \phi\}$, the proposition ϕ does not include membership. Instead of membership there are judgements of the form $\Gamma \vdash t : K$, in which a term t is assigned to a class K in context Γ. As a result, classes as described here are quite different from sets. For instance, we cannot formulate paradoxes such as Russell's paradox. Set theories in Natural Deduction form may admit proofs of paradoxes but these proofs fail to normalise under a proof rewriting system. For the simpler formal systems in this paper, proof rewriting is strongly normalising.

1.1 Overview

To analyse classes we introduce three formal systems and examine the relationship between them. We begin with a simple system $\mathcal{C}$ which deals with classes, terms classified by classes, and formulae. The class comprehension rules are in such a form that the usual correspondence between terms and proofs of their well-formedness cannot be obtained directly. However, in a version of the system where terms are explicitly enriched with some proof information, we may reconstruct a proof of the well-formedness of a term from its syntax. For the judgements of the resulting system $\mathcal{C}^{\mathsf{Der}}$, equality is introduced and axiomatised. A decision procedure for this equality is given in terms of a normal form for judgements (the 'long $\beta\eta$-normal form').

As an alternative, to cope with proofs in the original system $\mathcal{C}$, another formal system, $\mathcal{C}^{\mathsf{Env}}$, is introduced, where the extra proof components are stored, not in the terms, but in the context, in a 'proof environment'. It is this system that we consider to be a language of 'type classes' in a way that is explained towards the end of the paper (Section 6). The systems $\mathcal{C}^{\mathsf{Env}}$ and $\mathcal{C}^{\mathsf{Der}}$ are then related by providing translations from derivations and judgements of $\mathcal{C}^{\mathsf{Env}}$ to judgements of $\mathcal{C}^{\mathsf{Der}}$. It is shown that the two translations are consistent with each other, and so the translation of the resulting judgement given as the translation of its derivation is actually independent of the derivation. This in turn is used to prove a coherence result, which can be used to define maps on derivable judgements of $\mathcal{C}^{\mathsf{Env}}$ by induction on their derivation. These results rely on the decision procedure for equality in $\mathcal{C}^{\mathsf{Der}}$.

As an application, we consider a model theory for the systems. For the first order system $\mathcal{C}^{\mathsf{Der}}$, models in strict indexed categories with comprehension schemata are formalised (based upon [Lawvere 1970]), a semantics of judgements in such models is given, and a soundness theorem is proved (completeness holds as well). This provides a basis for a semantics of $\mathcal{C}^{\mathsf{Env}}$, given straightforwardly using the coherence result. Finally, we show how this relates to type classes in programming languages.

2 A formal system of classes

We begin by describing a first order system $\mathcal{C}$ which incorporates a notion of comprehension. The system is presented as a sequent calculus in a Natural Deduction style [Prawitz 1965]. It consists of terms t which are classified by what we call classes K, and we write $t : K$. Properties of terms are described by propositional formulae ϕ which may have free variables for which terms may be substituted.

A note about terminology: Type theory and proof theory each come with their own terminology, which, in the systems we present can lead to confusion. In $t : K$, we would normally call K the *type* of t. But as a type theory, t itself is a type and K, which classifies types, is often called a *kind*. To prevent confusion, we avoid speaking of 'types' and call t a term and K its class.

We first describe the system and then explain why it is as it is. We begin with the contexts and judgements of $\mathcal{C}$. Contexts are of two forms:

Class contexts	$\Gamma ::= x_1 : K_1, \ldots, x_n : K_n$
Formula contexts	$\Delta ::= \phi_1, \ldots, \phi_n$

for $n \geq 0$. In class contexts $x_1 : K_1, \ldots, x_n : K_n$ the variables x_i are all distinct. Empty contexts may be omitted in judgements.

There are four forms of judgement:

$\vdash K\,\mathsf{class}$ means that K is a well-formed class. For context $\Gamma = x_1 : K_1, \ldots, x_n : K_n$, we write $\vdash \Gamma\,\mathsf{class}$ for the list of judgements $\vdash K_1\,\mathsf{class}, \ldots, \vdash K_n\,\mathsf{class}$.

$\Gamma \vdash \phi\,\mathsf{prop}$ means that ϕ is a well-formed propositional formula with free variables declared in context Γ. We write $\Gamma \vdash \phi_1, \ldots, \phi_n\,\mathsf{prop}$ to abbreviate the list of judgements $\Gamma \vdash \phi_1\,\mathsf{prop}, \ldots, \Gamma \vdash \phi_n\,\mathsf{prop}$.

$\Gamma \vdash t : K$ means that t is a well-formed term of class K with free variables declared in Γ.

$\Gamma; \Delta \vdash \phi$ means that the formulae Δ entail ϕ in a context with free variables declared in Γ.

As a vocabulary for $\mathcal{C}$ we introduce a signature, Σ, of symbols and axioms (strictly speaking we should write $\mathcal{C}_\Sigma$ and will do so when required):

1. a collection of class symbols (metavariable A ranging over them);

2. a collection of predicate symbols, each with a given (possibly empty) class of arguments, thus if $P_{K_1,\ldots,K_n}$ is a predicate symbol of (well-formed) classes $K_1, \ldots, K_n$ and t_i is a term of class K_i (for $i = 1, \ldots, n$), then $P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)$ is a well-formed formula;

3. a collection of symbols for forming terms, each with a result class and possibly argument classes, thus if $k_{K_1,\ldots,K_n \to K}$ is such a symbol with argument classes $K_1, \ldots, K_n$ and result class K and t_i is a term of class K_i (for $i = 1, \ldots, n$) then $k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n)$ is a well-formed term of class K (if there are no argument classes, we write $k_K : K$ and k_K itself is a well-formed term of class K); and finally,

4. a collection of axioms of form $\Gamma; \Delta \vdash \phi$ for given Γ, Δ (possibly empty) and ϕ. The inference rule for axioms ensures that axioms are derivable (assertable) only when $\vdash \Gamma\,\mathsf{class}$, and $\Gamma \vdash \Delta, \phi\,\mathsf{prop}$ are derivable.

For forming classes, we introduce products of classes as well as class comprehension. At the level of propositions there are only atomic propositions constructed by applying predicates to terms. Later, we show how to extend this system to include propositional logic (Section 3.3). The syntax of $\mathcal{C}$ is therefore:

$$\begin{array}{ll}
\text{Classes} & K ::= A \mid K_1 \times K_2 \mid \{x : K \mid \phi\} \\
\text{Formulae} & \phi ::= P_{K_1,\ldots,K_n}(t_1, \ldots, t_n) \\
\text{Terms} & t ::= x \mid k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n) \mid \langle t_1, t_2 \rangle \mid \pi^1_{K_1,K_2}(t) \mid \pi^2_{K_1,K_2}(t)
\end{array}$$

Remarks. We explain why system $\mathcal{C}$ is as it is. Firstly, it is a type system without *dependent* types. Equivalently, it is a simple first order system corresponding to the basic structure of an indexed category. This determines the forms of judgement, in particular excluding the judgement form $\Gamma; \Delta \vdash t : K$ where well-formedness of term t may depend upon the variables in Δ. The rules for comprehension in Figure 1 require comment. These rules are an obvious choice for a restricted form of class comprehension, yet their structure is fairly intricate. Notice that the

$$\frac{\Gamma \vdash \phi_1, \ldots, \phi_n \; \mathsf{prop}}{\Gamma; \phi_1, \ldots, \phi_n \vdash \phi_j} \, (\, \mathsf{prop\ Ass}, \quad n \geq 1, 1 \leq j \leq n)$$

$$\frac{\vdash K_1 \, \mathsf{class} \; \cdots \vdash K_n \, \mathsf{class}}{x_1 : K_1, \ldots, x_n : K_n \vdash x_j : K_j} \, (\, \mathsf{class\ Ass}, \quad n \geq 1, 1 \leq j \leq n)$$

$$\frac{\vdash K_1 \, \mathsf{class} \quad \vdash K_2 \, \mathsf{class}}{\vdash K_1 \times K_2 \, \mathsf{class}} \, (\times \; \mathsf{Form})$$

$$\frac{\Gamma \vdash t_1 : K_1 \quad \Gamma \vdash t_2 : K_2}{\Gamma \vdash \langle t_1, t_2 \rangle : K_1 \times K_2} \, (\times \; \mathsf{Intro}) \qquad \frac{\Gamma \vdash t : K_1 \times K_2}{\Gamma \vdash \pi^j_{K_1, K_2}(t) : K_j} \, (\times \; \mathsf{Elim}, \quad j = 1, 2)$$

$$\frac{x : K \vdash \phi \; \mathsf{prop}}{\vdash \{x : K \mid \phi\} \, \mathsf{class}} \, (\{\} \; \mathsf{Form})$$

$$\frac{\Gamma \vdash t : K \quad x : K \vdash \phi \; \mathsf{prop} \quad \Gamma; \vdash \phi[t/x]}{\Gamma \vdash t : \{x : K \mid \phi\}} \, (\{\} \; \mathsf{Intro})$$

$$\frac{\Gamma \vdash t : \{x : K \mid \phi\}}{\Gamma \vdash t : K} \, (\{\} \; \mathsf{Elim1}) \qquad \frac{\Gamma \vdash t : \{x : K \mid \phi\} \quad \Gamma \vdash \Delta \; \mathsf{prop}}{\Gamma; \Delta \vdash \phi[t/x]} \, (\{\} \; \mathsf{Elim2})$$

Figure 1: The rules of system $\mathcal{C}$.

$$\vdash A \, \mathsf{class} \qquad \frac{\vdash \Gamma \, \mathsf{class} \quad \Gamma \vdash t_1 : K_1 \quad \cdots \quad \Gamma \vdash t_n : K_n}{\Gamma \vdash P_{K_1, \ldots, K_n}(t_1, \ldots, t_n) \; \mathsf{prop}} \, (n \geq 0)$$

$$\frac{\vdash \Gamma \, \mathsf{class} \quad \Gamma \vdash t_1 : K_1 \quad \cdots \quad \Gamma \vdash t_n : K_n \quad \vdash K \, \mathsf{class}}{\Gamma \vdash k_{K_1, \ldots, K_n \to K}(t_1, \ldots, t_n) : K} \, (n \geq 0)$$

For an axiom $x_1 : K_1, \ldots, x_m : K_m; \phi_1, \ldots, \phi_n \vdash \phi$, $(m, n \geq 0)$, in Σ, we have rule:

$$\frac{\begin{array}{c} \vdash \Gamma \, \mathsf{class} \quad \Gamma \vdash t_1 : K_1 \quad \cdots \quad \Gamma \vdash t_m : K_m \\ \Gamma \vdash \Delta, \phi \; \mathsf{prop} \quad \Gamma; \Delta \vdash \phi_1[t_1/x_1, \ldots, t_m/x_m] \quad \cdots \quad \Gamma; \vdash \phi_n[t_1/x_1, \ldots, t_m/x_m] \end{array}}{\Gamma; \Delta \vdash \phi[t_1/x_1, \ldots, t_m/x_m]}$$

Figure 2: The Σ rules of $\mathcal{C}$.

form of the Elim1 rule means that terms can belong to more than one class. In type theory, there are good reasons for avoiding this multiple classification of terms, whereas, for classes viewed as sets and for functional programming, it is a natural requirement. The well-formedness of formula ϕ is necessary in the Intro rule, as the only other occurrence of ϕ is a substitution instance of it. Strictly speaking, the Intro rule should be viewed as schematic over the well-formedness condition, that is, for each ϕ such that $x : K \vdash \phi\,\mathsf{prop}$ there is an Intro rule without the well-formedness condition. In presenting formal systems, this distinction is not usually drawn so that well-formedness conditions become hypotheses of rules. However, in a categorical treatment (Section 5) this distinction becomes clear. This explains the absence of an elimination rule to produce the conclusion $x : K \vdash \phi\,\mathsf{prop}$. In the expression $\{x : K \mid \phi\}$, x is bound and there are no other free variables in ϕ. More generally, we may wish to consider a higher-order class structure in which classes are parameterised. This would extend the present notion of type classes in programming and, when we come to consider models, necessitate a further level of indexing of categories. Logical systems which include parameterised comprehension forms have been investigated by Hallnäs [Hallnäs 1988] amongst others. The additional formula context Δ in the second elimination rule makes weakening admissible in the system.

Example 2.1. Here is a simple example of derivations in $\mathcal{C}$. We shown that we may derive $z : \{x : A \mid P_A(x)\}$ from $z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}$. We give two derivations of this result. The derivations differ in substance in a way we make precise in the next section.

Let Γ be the context $z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}$. The following is a derivation tree of $z : \{x : A \mid P_A(x)\}$ from Γ.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\ D\\ \Gamma \vdash z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}\end{matrix}}{\Gamma \vdash z : \{x : A \mid P_A(x)\}}}{\Gamma \vdash z : A} \qquad \dfrac{\vdash A\,\mathsf{class} \qquad \dfrac{\vdash A\,\mathsf{class}}{x : A \vdash x : A}}{x : A \vdash P_A(x)\,\mathsf{prop}} \qquad \dfrac{\begin{matrix}\vdots\ D\\ \Gamma \vdash z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}\end{matrix}}{\Gamma ; \vdash P_A(z)}}{\Gamma \vdash z : \{x : A \mid P_A(x)\}}$$

where the missing derivation D is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash A\,\mathsf{class} \qquad \dfrac{\vdash A\,\mathsf{class}}{x : A \vdash x : A}}{x : A \vdash P_A(x)\,\mathsf{prop}}}{\vdash \{x : A \mid P_A(x)\}\,\mathsf{class}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash A\,\mathsf{class} \qquad \dfrac{\vdash A\,\mathsf{class}}{x : A \vdash x : A}}{x : A \vdash P_A(x)\,\mathsf{prop}}}{\vdash \{x : A \mid P_A(x)\}\,\mathsf{class}}}{y : \{x : A \mid P_A(x)\} \vdash y : \{x : A \mid P_A(x)\}}}{y : \{x : A \mid P_A(x)\} \vdash y : A}}{y : \{x : A \mid P_A(x)\} \vdash P_A(y)\,\mathsf{prop}}}{\vdash \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}\,\mathsf{class}}}{\Gamma \vdash z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}}$$

The following is a simpler derivation of the same result.

$$\dfrac{\begin{array}{c}\vdots\ D\\ \Gamma \vdash z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}\end{array}}{\Gamma \vdash z : \{x : A \mid P_A(x)\}}$$

where D is as above. The key to the difference between these two derivations lies in the two occurrences of P_A in the context, each of which may be used to construct the result (see Section 3.2.1 for an explanation of this).

The remainder of the paper is devoted to an analysis of system $\mathcal{C}$, beginning with languages for the derivations in $\mathcal{C}$ and then models for these languages and, finally, relating them to languages with type classes.

3 A calculus of proofs (I)

We begin the analysis of system $\mathcal{C}$ by examining the structure of its derivations. In doing so, not only do we provide proof-theoretic results, but also we set up machinery for defining semantics later (Section 5).

Our starting point is fairly standard: we introduce a language $\mathcal{C}^{\mathsf{Der}}$ of derivations in $\mathcal{C}$. This language is a type theory whose judgements are in bijective correspondence to the derivations in $\mathcal{C}$. In this type theory, we view class comprehension as a form of strong sum type.

For each inference rule in $\mathcal{C}$, we introduce a form of expression building proofs of the conclusion of the rule from proofs of the hypotheses. Here are some of the points involved:

1. Some of the rules in $\mathcal{C}$ already encode their proofs. This applies to all formation rules for classes and most for propositional formulae. However, there are no expressions for derivations of propositional consequence $\Gamma; \Delta \vdash \phi$. To remedy this, we follow standard procedure and include variables in the context Δ and terms α coding the proofs. In this case, proofs are constructed from axioms for which we introduce axiom symbols (ranged over by a).

2. The class comprehension rules require special treatment. The introduction rule,

$$\frac{\Gamma \vdash t : K \quad x : K \vdash \phi\ \mathsf{prop} \quad \Gamma; \vdash \phi[t/x]}{\Gamma \vdash t : \{x : K \mid \phi\}}$$

   does not record in its conclusion the proofs of the second and third hypotheses. We, therefore, modify the language of terms to incorporate the additional proofs. That is, we replace term t in the conclusion of the above rule with a term $[t, \alpha]_{(x:K)\phi}$ where α is a proof of $\Gamma; \vdash \phi[t/x]$. The subscript $(x : K)\phi$ captures the content of the second hypothesis of the rule. In this expression x may occur freely in ϕ but is bound in the overall expression. The two elimination rules correspond to extracting the two components of this term using projections ν^1 and ν^2. This is, therefore, a class indexed sum of propositional formulae yielding, not a formula, but a class.

$$\frac{\Gamma \vdash \phi_1, \ldots, \phi_n \ \mathsf{prop}}{\Gamma; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash v_j : \phi_j} \ (\ \mathsf{prop\ Ass}, \quad n \geq 1, 1 \leq j \leq n)$$

$$\frac{\vdash K_1 \ \mathsf{class} \ \cdots \vdash K_n \ \mathsf{class}}{x_1 : K_1, \ldots, x_n : K_n \vdash x_j : K_j} \ (\ \mathsf{class\ Ass}, \quad n \geq 1, 1 \leq j \leq n)$$

$$\frac{\vdash K_1 \ \mathsf{class} \quad \vdash K_2 \ \mathsf{class}}{\vdash K_1 \times K_2 \ \mathsf{class}} \ (\times \ \mathsf{Form})$$

$$\frac{\Gamma \vdash t_1 : K_1 \quad \Gamma \vdash t_2 : K_2}{\Gamma \vdash \langle t_1, t_2 \rangle : K_1 \times K_2} \ (\times \ \mathsf{Intro}) \quad \frac{\Gamma \vdash t : K_1 \times K_2}{\Gamma \vdash \pi^j_{K_1,K_2}(t) : K_j} \ (\times \ \mathsf{Elim}, \quad j = 1, 2)$$

$$\frac{x : K \vdash \phi \ \mathsf{prop}}{\vdash \{x : K \mid \phi\} \ \mathsf{class}} \ (\{\} \ \mathsf{Form})$$

$$\frac{\Gamma \vdash t : K \quad x : K \vdash \phi \ \mathsf{prop} \quad \Gamma; \vdash \alpha : \phi[t/x]}{\Gamma \vdash [t, \alpha]_{(x:K)\phi} : \{x : K \mid \phi\}} \ (\{\} \ \mathsf{Intro})$$

$$\frac{\Gamma \vdash t : \{x : K \mid \phi\}}{\Gamma \vdash \nu^1_{(x:K)\phi}(t) : K} \ (\{\} \ \mathsf{Elim1}) \quad \frac{\Gamma \vdash t : \{x : K \mid \phi\} \quad \Gamma \vdash \Delta \ \mathsf{prop}}{\Gamma; \Delta \vdash \nu^2_{(x:K)\phi}(t) : \phi[\nu^1_{(x:K)\phi}(t)/x]} \ (\{\} \ \mathsf{Elim2})$$

Figure 3: The rules of system $\mathcal{C}^{\mathsf{Der}}$.

The syntax of $\mathcal{C}^{\mathsf{Der}}$ is:

Classes	$K ::= A \mid K_1 \times K_2 \ \mid \{x : K \mid \phi\}$
Formulae	$\phi ::= P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)$
Terms	$t ::= x \mid k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n) \mid \langle t_1, t_2 \rangle \mid \pi^1_{K_1,K_2}(t) \mid \pi^2_{K_1,K_2}(t)$
	$\mid [t, \alpha]_{(x:K)\phi} \mid \nu^1_{(x:K)\phi}(t)$
Proofs	$\alpha ::= v \mid a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n \to \phi}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n) \mid \nu^2_{(x:K)\phi}(t)$

The contexts Γ are as previous. The formula contexts Δ now include variables v ranging over proofs, so are of the form $v_1 : \phi_1, \ldots v_n : \phi_n$ with the v's distinct. The only form of judgement to change is that for propositional entailment which becomes $\Gamma; \Delta \vdash \alpha : \phi$ and means that proof α establishes that Δ entails ϕ in a context with free (term) variables declared in Γ and free propositional variables in Δ.

The inference rules (Figures 3 and 4) are unchanged from those of $\mathcal{C}$ except those for propositional consequence and those for class comprehension.

Example 3.1. As an example, we give an expression in $\mathcal{C}^{\mathsf{Der}}$ for the two derivations in $\mathcal{C}$ in Example 2.1 in which, from $z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}$, we derived $z : \{x : A \mid P_A(x)\}$. Corresponding to the first derivation is the rather unwieldy expression:

$$\Gamma \vdash [\nu^1_{(x:A)P_A(x)}(\nu^1_\gamma(z)), \nu^2_\gamma(z)]_{(x:A)P_A(x)} : \{x : A \mid P_A(x)\}$$

$$\vdash A\,\mathsf{class} \quad \frac{\vdash \Gamma\,\mathsf{class} \quad \Gamma \vdash t_j : K_j \quad (j = 1, \ldots, n)}{\Gamma \vdash P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)\,\mathsf{prop}}\,(n \geq 0)$$

$$\frac{\vdash \Gamma\,\mathsf{class} \quad \vdash K\,\mathsf{class} \quad \Gamma \vdash t_j : K_j \quad (j = 1, \ldots, n)}{\Gamma \vdash k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n) : K}\,(n \geq 0)$$

$$\frac{\begin{array}{c}\vdash \Gamma\,\mathsf{class} \quad \Gamma \vdash t_i : K_i \quad (i = 1, \ldots, m) \\ \Gamma \vdash \Delta, \phi\,\mathsf{prop} \quad \Gamma; \Delta \vdash \alpha_j : \phi_j[\vec{t}/\vec{x}] \quad (j = 1, \ldots, n)\end{array}}{\Gamma; \Delta \vdash a_{K_1,\ldots,K_m;\phi_1[\vec{t}/\vec{x}],\ldots,\phi_n[\vec{t}/\vec{x}] \to \phi[\vec{t}/\vec{x}]}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n) : \phi[\vec{t}/\vec{x}]}\,(m, n \geq 0)$$

where $[\vec{t}/\vec{x}]$ is $[t_1/x_1, \ldots, t_m/x_m]$ and the following is an axiom:
$x_1 : K_1, \ldots, x_m : K_m; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n \to \phi}(x_1, \ldots, x_m; v_1, \ldots, v_n) : \phi$

Figure 4: The Σ rules of $\mathcal{C}^{\mathsf{Der}}$.

where Γ is $z : \{y : \{x : A \mid P_A(x)\} \mid P_A(\nu^1_{(x:A)P_A(x)}(y))\}$ and the subscript γ is $(y : \{x : A \mid P_A(x)\})P_A(\nu^1_{(x:A)P_A(x)}(y))$. This directly encodes the Intro and Elim structure of the original derivation. For example, it ends with a $\{\}$-Intro of a term built from a judgement $\Gamma \vdash \nu^1_{(x:A)P_A(x)}(\nu^1_\gamma(z)) : A$, a judgement $x : A \vdash P_A(x)\,\mathsf{prop}$ and a proof of satisfaction $\nu^2_\gamma(z)$.

The second derivation of $z : \{x : A \mid P_A(x)\}$ from $z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\}$ corresponds to the simple judgement:

$$\Gamma \vdash \nu^1_\gamma(z) : \{x : A \mid P_A(x)\}$$

where Γ and γ are as above.

We complete the description of $\mathcal{C}^{\mathsf{Der}}$ by introducing a relationship of equality between judgements. This equality allows us to:

1. link derivations and judgements in systems of classes in such a way that we can define maps on derivable judgements by defining maps on derivations, in particular defining semantics of systems of classes, and

2. examine the notion of 'essentially distinct' derivations and count them.

Results of this form are called 'coherence' results and have been extensively studied in category theory (see [Mac Lane 1982] for example) and in proof theory (see [Curien 1990] and [Curien, Ghelli 1990] for example).

The appropriate equality (Figure 5) arises as a description of equivalence of proofs under 'inversion principles' [Prawitz 1965]. It includes both the usual β-rules, where an introduction is followed by an elimination, and also η-rules, where an elimination is followed by an introduction. Equality is at the level of *judgements* rather than at the level of constituents of judgements (such as terms). This is forced by the dependency between the syntax of classes, propositional formulae, terms and proofs, so that an equality on syntax of one of these induces equality on

$$
\begin{array}{rcl}
\Gamma \vdash \pi^j_{K_1,K_2}(\langle t_1, t_2\rangle) : K_j & = & \Gamma \vdash t_j : K_j \quad (j = 1, 2) \\
\Gamma \vdash \langle \pi^1_{K_1,K_2}(t), \pi^2_{K_1,K_2}(t)\rangle : K_1 \times K_2 & = & \Gamma \vdash t : K_1 \times K_2 \\
\Gamma \vdash \nu^1_{(x:K)\phi}([t, \alpha]_{(x:K)\phi}) : K & = & \Gamma \vdash t : K \\
\Gamma; \Delta \vdash \nu^2_{(x:K)\phi}([t, \alpha]_{(x:K)\phi}) : \psi & = & \Gamma; \Delta \vdash \alpha : \psi \\
\Gamma \vdash [\nu^1_{(x:K)\phi}(t), \nu^2_{(x:K)\phi}(t)]_{(x:K)\phi} : \{x : K \mid \phi\} & = & \Gamma \vdash t : \{x : K \mid \phi\}
\end{array}
$$

Figure 5: The equations of system $\mathcal{C}^{\mathsf{Der}}$.

the others. The equations for the symbols in the signature Σ (Figure 6) assert that if a symbol occurs in Σ with a given 'type', then it also occurs at any equal 'type'. The equational theory is generated from the equations in Figures 5 and 6 using the rules of symmetry and transitivity, together with the rules of congruence, i.e. the rules of $\mathcal{C}^{\mathsf{Der}}$ adapted for equations. For example, the class formation rule for equations is:

$$
\frac{\vdash K\,\mathsf{class} \;=\; \vdash K'\,\mathsf{class} \qquad x : K \vdash \phi\,\mathsf{prop} \;=\; x : K' \vdash \phi'\,\mathsf{prop}}{\vdash \{x : K \mid \phi\}\,\mathsf{class} \;=\; \vdash \{x : K' \mid \phi'\}\,\mathsf{class}}
$$

As in λ-calculus, it is possible (though rather unintuitive) to present the system entirely in terms of equations. Derivability of judgement J is then the derivability of $J = J$.

We give, in the next two propositions, some standard properties of formal systems as they apply to $\mathcal{C}^{\mathsf{Der}}$. The first deals with the four forms of substitution in $\mathcal{C}^{\mathsf{Der}}$.

Proposition 3.1

1. *If* $x_1 : K_1, \ldots, x_n : K_n \vdash \phi\,\mathsf{prop}$ *and* $\Gamma \vdash t_1 : K_1 \;\cdots\; \Gamma \vdash t_n : K_n$ *are derivable in* $\mathcal{C}^{\mathsf{Der}}$ *then so is* $\Gamma \vdash \phi[t_1/x_1, \ldots, t_n/x_n]\,\mathsf{prop}$*,*

2. *If* $x_1 : K_1, \ldots, x_n : K_n \vdash t : K$ *and* $\Gamma \vdash t_1 : K_1 \;\cdots\; \Gamma \vdash t_n : K_n$ *are derivable in* $\mathcal{C}^{\mathsf{Der}}$ *then so is* $\Gamma \vdash t[t_1/x_1, \ldots, t_n/x_n] : K$*,*

3. *If* $x_1 : K_1, \ldots, x_n : K_n; \Delta \vdash \alpha : \phi$ *and* $\Gamma \vdash t_1 : K_1 \;\cdots\; \Gamma \vdash t_n : K_n$ *are derivable in* $\mathcal{C}^{\mathsf{Der}}$ *then so is* $\Gamma; \Delta[t_1/x_1, \ldots, t_n/x_n] \vdash \alpha[t_1/x_1, \ldots, t_n/x_n] : \phi[t_1/x_1, \ldots, t_n/x_n]$*,*

4. *If* $\Gamma; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash \alpha : \phi$ *and* $\Gamma; \Delta \vdash \alpha_1 : \phi_1 \;\cdots\; \Gamma; \Delta \vdash \alpha_n : \phi_n$ *are derivable in* $\mathcal{C}^{\mathsf{Der}}$ *then so is* $\Gamma; \Delta \vdash \alpha[\alpha_1/v_1, \ldots, \alpha_n/v_n] : \phi$*.*

$$\vdash A\,\mathsf{class} \;=\; \vdash A\,\mathsf{class}$$

$$\frac{\vdash \Gamma\,\mathsf{class} \;= \vdash \Gamma'\,\mathsf{class} \quad \Gamma \vdash t_j : K_j \;=\; \Gamma' \vdash t'_j : K'_j \quad (j = 1, \ldots, n)}{\Gamma \vdash P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)\,\mathsf{prop} \;=\; \Gamma' \vdash P_{K'_1,\ldots,K'_n}(t'_1, \ldots, t'_n)\,\mathsf{prop}}\,(n \geq 0)$$

$$\frac{\begin{array}{c}\vdash \Gamma\,\mathsf{class} \;= \vdash \Gamma'\,\mathsf{class} \quad \vdash K\,\mathsf{class} \;=\; \vdash K'\,\mathsf{class}\\ \Gamma \vdash t_j : K_j \;=\; \Gamma' \vdash t'_j : K'_j \quad (j = 1, \ldots, n)\end{array}}{\Gamma \vdash k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n) : K \;=\; \Gamma' \vdash k_{K'_1,\ldots,K'_n \to K'}(t'_1, \ldots, t'_n) : K'}\,(n \geq 0)$$

$$\frac{\begin{array}{c}\vdash \Gamma\,\mathsf{class} \;= \vdash \Gamma'\,\mathsf{class} \quad \Gamma \vdash t_i : K_i \;=\; \Gamma' \vdash t'_i : K'_i \quad (i = 1, \ldots, m)\\ \Gamma \vdash \Delta\,\mathsf{prop} \;=\; \Gamma' \vdash \Delta'\,\mathsf{prop}\\ \Gamma; \Delta \vdash \alpha_j : \phi_j[\vec{t}/\vec{x}] \;=\; \Gamma'; \Delta' \vdash \alpha'_j : \phi'_j[\vec{t'}/\vec{x}] \quad (j = 1, \ldots, n)\end{array}}{\begin{array}{c}\Gamma; \Delta \vdash a_{K_1,\ldots,K_m;\phi_1[\vec{t}/\vec{x}],\ldots,\phi_n[\vec{t}/\vec{x}] \to \phi[\vec{t}/\vec{x}]}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n) : \phi[\vec{t}/\vec{x}] \;=\\ \Gamma'; \Delta' \vdash a_{K'_1,\ldots,K'_m;\phi'_1[\vec{t'}/\vec{x}],\ldots,\phi'_n[\vec{t'}/\vec{x}] \to \phi'[\vec{t'}/\vec{x}]}(t'_1, \ldots, t'_m; \alpha'_1, \ldots, \alpha'_n) : \phi'[\vec{t'}/\vec{x}]\end{array}}\,(m, n \geq 0)$$

where $[\vec{t}/\vec{x}]$ is $[t_1/x_1, \ldots, t_m/x_m]$, and the following is an axiom:

$$x_1 : K_1, \ldots, x_m : K_m; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n \to \phi}(x_1, ..., x_m; v_1, ..., v_n) : \phi$$

Figure 6: The Σ equations of $\mathcal{C}^{\mathsf{Der}}$.

Proposition 3.2

1. **Well-formedness.** *If* $\Gamma \vdash t : K \;=\; \Gamma' \vdash t' : K'$ *in* $\mathcal{C}^{\mathsf{Der}}$*, then* $\vdash \Gamma\, \mathsf{class} \;=\; \vdash \Gamma'\, \mathsf{class}$ *and* $\vdash K\, \mathsf{class} \;=\; \vdash K'\, \mathsf{class}$ *in* $\mathcal{C}^{\mathsf{Der}}$*. Likewise for the components of other forms of judgement.*

2. **Substitution.** *If* $x_1 : K_1, \ldots, x_n : K_n \vdash t : K \;=\; x_1 : K_1', \ldots, x_n : K_n' \vdash t' : K'$ *and* $\Gamma \vdash t_j : K_j \;=\; \Gamma' \vdash t_j' : K_j'$ *for* $j = 1, \ldots, n$ *in* $\mathcal{C}^{\mathsf{Der}}$ *then*

$$\Gamma \vdash t[t_1/x_1, \ldots, t_n/x_n] : K \;=\; \Gamma' \vdash t'[t_1'/x_1, \ldots, t_n'/x_n] : K'$$

in $\mathcal{C}^{\mathsf{Der}}$*. Likewise for the other forms of substitution.*

3. **Generation.** *If* $\vdash \{x : K \mid \phi\}\, \mathsf{class} \;=\; \vdash \{x : K' \mid \phi'\}\, \mathsf{class}$ *then* $\vdash K\, \mathsf{class} \;=\; \vdash K'\, \mathsf{class}$ *and* $x : K \vdash \phi\, \mathsf{prop} \;=\; x : K' \vdash \phi'\, \mathsf{prop}$ *in* $\mathcal{C}^{\mathsf{Der}}$*. Likewise for other judgements.*

4. **Replacement.** *If* $\Gamma \vdash \phi\, \mathsf{prop}$ *is derivable and* $\vdash \Gamma\, \mathsf{class} \;=\; \vdash \Gamma'\, \mathsf{class}$ *in* $\mathcal{C}^{\mathsf{Der}}$*, then there is a* ϕ' *such that* $\Gamma' \vdash \phi'\, \mathsf{prop}$ *is derivable and* $\Gamma \vdash \phi\, \mathsf{prop} \;=\; \Gamma' \vdash \phi'\, \mathsf{prop}$*. Analogous results hold for the other forms of judgement.*

Proof. Each case of Proposition 3.1 is established by structural induction on the first judgement. For Proposition 3.2, parts (1), (2) and (3) are established by induction on the derivation of equality. (4) is by structural induction on the judgement and the derivation of equality.

3.1 Solving a word problem for $\mathcal{C}^{\mathsf{Der}}$

This section contains the analysis on which much of the remainder of the paper relies. We solve a 'word problem' for $\mathcal{C}^{\mathsf{Der}}$ by presenting a *decision procedure* for equality in $\mathcal{C}^{\mathsf{Der}}$, a procedure which, given two derivable judgements J_1 and J_2 in $\mathcal{C}^{\mathsf{Der}}$, will decide whether or not they are equal under the equations of $\mathcal{C}^{\mathsf{Der}}$. The link between decision procedures and 'coherence' results is well-known (see for example [Curien, Ghelli 1990], [Curien 1990]) and will be illustrated later in the paper (Section 4).

The decision procedure operates by converting expressions to a 'canonical form', such that expressions are equal in $\mathcal{C}^{\mathsf{Der}}$ iff their canonical forms are syntactically identical. Because of the presence of both β- and η-rules, canonical forms cannot be computed by rewriting terms to a normal form under a simple rewrite system. A more delicate notion of canonical form is necessary in which the class of a term has a role as well as the form of the term itself. The definition of canonical form that we give is an adaptation of the 'long $\beta\eta$' normal forms of λ-calculus (see [Huet 1976] and [Jay 1991]).

We first define the appropriate notion of canonical form and then describe a procedure for converting judgements into this form. For judgements $\Gamma \vdash t : K$ canonical forms are η-expanded versions of t according to the form of class K. In the remaining cases, canonical forms are β-reduced (hence we introduce *reduced* forms), except at predicate, term and axiom symbols, when further η-expansion is necessary.

Definition 3.1 *We define when a well-formed judgement is* canonical.

- $\Gamma \vdash t : K$ *is canonical when* Γ *and* K *are* reduced *and* $t : K$ *is canonical, as defined below,*

- $\vdash K\, \mathsf{class}$ *is canonical when* K *is* reduced,

- $\Gamma \vdash \phi\,\mathsf{prop}$ *is canonical when* Γ *and* ϕ *are* reduced,
- $\Gamma; \Delta \vdash \alpha : \phi\,\mathsf{prop}$ *is canonical when* Γ, Δ, α *and* ϕ *are all* reduced.

Here $t : K$ *is defined to be* canonical *by structural induction on* K *as follows:*

- $\langle t_1, t_2\rangle : K_1 \times K_2$ *is canonical when* $t_i : K_i$ *are canonical,*
- $[t, \alpha]_{(x:K)\phi} : \{x : K \mid \phi\}$ *is canonical when* $t : K$ *is canonical and* α, K *and* ϕ *are* reduced
- $t : A$ *is canonical when* t *is* reduced *(where* A *is a class symbol in* Σ*).*

We define when an expression is reduced *as follows:*

- *variable* x *is reduced,*
- $\langle t_1, t_2\rangle$ *is reduced when* t_1 *and* t_2 *are reduced,*
- $\pi^j_{K_1,K_2}(t)$ *is reduced when* t *is reduced and not of the form* $\langle t_1, t_2\rangle$*,*
- $[t, \alpha]_{(x:K)\phi}$ *is reduced when* Γ, t, K *and* ϕ *are reduced,*
- $\nu^1_{(x:K)\phi}(t)$ *is reduced when* t *is not of form* $[t', \alpha']_\gamma$ *and* Γ, t, K, *and* ϕ *are reduced,*
- $k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n)$ *is reduced when* $t_j : K_j$ $(j = 1, \ldots, n)$ *is* canonical *and* K *is reduced,*
- *class* A *is reduced,*
- $K_1 \times K_2$ *is reduced when* K_1 *and* K_2 *are reduced,*
- $\{x : K \mid \phi\}$ *is reduced when* K *and* ϕ *are reduced,*
- $P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)$ *is reduced when* $t_1 : K_1$ *through to* $t_n : K_n$ *are* canonical,
- *variable* v *is reduced,*
- $a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n \to \phi}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n)$ *is reduced when* $t_1 : K_1$ *through to* $t_m : K_m$ *are* canonical *and the* ϕ*'s and* α*'s are reduced,*
- $\nu^2_{(x:K)\phi}(t)$ *is reduced when* K, ϕ *and* t *are reduced and* t *is not of the form* $[t', \alpha']_\gamma$

Proposition 3.3

1. *Canonical forms are equal in* $\mathcal{C}^{\mathsf{Der}}$ *iff they are syntactically identical,*
2. *For each derivable judgement* J *of* $\mathcal{C}^{\mathsf{Der}}$ *there is a (derivable) canonical form* $\mathsf{C}(J)$ *with* $\mathsf{C}(J) = J$ *and such that* $J_1 = J_2$ *in* $\mathcal{C}^{\mathsf{Der}}$ *iff* $\mathsf{C}(J_1) \equiv \mathsf{C}(J_2)$*, where* $\equiv$ *is syntactic equality.*

Proof. We use induction on the derivation of equality to prove (1). Suppose the derivation is one of the generating equations. It cannot be a β-rule since then the judgements would not be in canonical form. If it is a product or comprehension η-rule, then the components are canonical and so we use structural induction. If the derivation of equality is via inference rules then we use the induction hypothesis immediately for Intro-rules and further structural induction for Elim-rules.

For (2), we describe a procedure C to convert judgements to canonical form. For judgements $\Gamma \vdash t : K$ we η-expand t according to the form of K and then, when no further expansion is possible, we β-reduce the result using conversion R. This is the content of the following definition.

$\mathsf{C}(\Gamma \vdash t : K) = \mathsf{R}(\Gamma) \vdash \mathsf{C}(t : K)$
  where $\mathsf{C}(t : K)$ is defined by structural induction on K as follows:
$\mathsf{C}(t : K_1 \times K_2) = \langle t'_1, t'_2 \rangle : K'_1 \times K'_2$ where $t'_j : K'_j = \mathsf{C}(\pi^j_{K_1,K_2}(t) : K_j)$
$\mathsf{C}(t : \{x : K \mid \phi\}) = [t', \mathsf{R}(\nu^2_{(x:K)\phi}(t))]_{(x:\mathsf{R}(K))\mathsf{R}(\phi)} : \{x : \mathsf{R}(K) \mid \mathsf{R}(\phi)\}$
  where $t' : K' = \mathsf{C}(\nu^1_{(x:K)\phi}(t) : K)$
$\mathsf{C}(t : A) = \mathsf{R}(t) : A$

On other forms of judgement C is defined directly in terms of R:

$\mathsf{C}(\vdash K\,\mathsf{class}) = \;\vdash \mathsf{R}(K)\,\mathsf{class}$
$\mathsf{C}(\Gamma \vdash \phi\,\mathsf{prop}) = \mathsf{R}(\Gamma) \vdash \mathsf{R}(\phi)\,\mathsf{prop}$
$\mathsf{C}(\Gamma; \Delta \vdash \alpha : \phi) = \mathsf{R}(\Gamma); \mathsf{R}(\Delta) \vdash \mathsf{R}(\alpha) : \mathsf{R}(\phi)$

The conversion R is defined on terms by:

$\mathsf{R}(x) = x$
$\mathsf{R}(\langle t_1, t_2 \rangle) = \langle \mathsf{R}(t_1), \mathsf{R}(t_2) \rangle$
$\mathsf{R}(\pi^j_{K_1,K_2}(t)) = \;$ let $t' = \mathsf{R}(t)$ in if $t' \equiv \langle t_1, t_2 \rangle$ then t_j else $\pi^j_{\mathsf{R}(K_1),\mathsf{R}(K_2)}(t')$
$\mathsf{R}([t, \alpha]_{(x:K)\phi}) = [\mathsf{R}(t), \mathsf{R}(\alpha)]_{(x:\mathsf{R}(K))\mathsf{R}(\phi)}$
$\mathsf{R}(\nu^1_{(x:K)\phi}(t)) = \;$ let $t' = \mathsf{R}(t)$ in if $t' \equiv [t'', \alpha'']_\gamma$ then t'' else $\nu^1_{(x:\mathsf{R}(K))\mathsf{R}(\phi)}(t')$
$\mathsf{R}(k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n)) = k_{K'_1,\ldots,K'_n \to \mathsf{R}(K)}(t'_1, \ldots, t'_n)$ where $t'_j : K'_j = \mathsf{C}(t_j : K_j)$

Here $\equiv$ is syntactic identity used in pattern matching. On other expressions, R is defined simply in terms of their structure, except again using η-expansion when symbols of Σ are encountered and β-reduction for the projection ν^2:

$\mathsf{R}(A) = A$
$\mathsf{R}(K_1 \times K_2) = \mathsf{R}(K_1) \times \mathsf{R}(K_2)$
$\mathsf{R}(\{x : K \mid \phi\}) = \{x : \mathsf{R}(K) \mid \mathsf{R}(\phi)\}$
$\mathsf{R}(P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)) = P_{K'_1,\ldots,K'_n}(t'_1, \ldots, t'_n)$ where $t'_j : K'_j = \mathsf{C}(t_j : K_j)$
$\mathsf{R}(v) = v$
$\mathsf{R}(a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n \to \phi}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n)) =$
    $a_{\mathsf{R}(K_1),\ldots,\mathsf{R}(K_m);\mathsf{R}(\phi_1),\ldots,\mathsf{R}(\phi_n) \to \mathsf{R}(\phi')}(t'_1, \ldots, t'_m; \mathsf{R}(\alpha_1), \ldots, \mathsf{R}(\alpha_n))$
  where $t'_j : K'_j = \mathsf{C}(t_j : K_j)$
$\mathsf{R}(\nu^2_{(x:K)\phi}(t)) = \;$ let $t' = \mathsf{R}(t)$ in if $t' \equiv [t'', \alpha'']_\gamma$ then α'' else $\nu^2_{(x:\mathsf{R}(K))\mathsf{R}(\phi)}(t')$

Despite the interplay of η-expansion and β-reduction, the termination of this on derivable judgements is not difficult to establish. The β-reduction is that of projections for products and comprehension and so does not involve substitution (as it does in λ-calculus). β-reduction by itself therefore terminates. For uniformity, let us change notation and write $\mathsf{C}(e, \bullet)$ for $\mathsf{R}(e)$, and view the above as a rewriting system. We order $\mathsf{C}(e, T)$ as the lexical product of the following orders:

1. $\mathsf{C}(e, T) < \mathsf{C}(e', T')$, when T is a (strict) subterm of T', or $T \equiv \bullet$ (and T' does not),

2. $\mathsf{C}(e,T) < \mathsf{C}(e',T')$ when e is a β-reduced form of e',
3. $\mathsf{C}(e,T) < \mathsf{C}(e',T')$ when e and e' are the same except that some occurrences of C in the term structure of e' are lower in the term structure of e or absent.

To handle the duplication arising in the rule for $\mathsf{C}(t : K_1 \times K_2)$, for example, we make the termination order $\sqsubset$ the *multiset order* of the above lexical product. Then $\sqsubset$ is wellfounded and, for each of the above rules, RHS $\sqsubset$ LHS.

Now, to complete the proof, we have $\mathsf{C}(J) = J$ in $\mathcal{C}^{\mathsf{Der}}$ since conversion steps arise from the equations of $\mathcal{C}^{\mathsf{Der}}$. That $\mathsf{C}(J)$ is canonical is by construction (by induction on the termination order). Finally, if $J_1 = J_2$ then $\mathsf{C}(J_1) = J_1 = J_2 = \mathsf{C}(J_2)$ and thus (using (1)) $\mathsf{C}(J_1) \equiv \mathsf{C}(J_2)$.

Remark. The canonical forms described here are canonical for both the product and comprehension structure. There is a looser notion of canonical form which is canonical for the comprehension structure alone. A $\{\}$-*canonical form* is defined as above (including the η-expansion at product type) but $\pi^j_{K_1,K_2}(t)$ is defined to be reduced simply when t is reduced. The conversion to canonical form is likewise modified: $\mathsf{R}(\pi^j_{K_1,K_2}(t)) = \pi^j_{\mathsf{R}(K_1),\mathsf{R}(K_2)}(\mathsf{R}(t))$. Then judgements are equal in $\mathcal{C}^{\mathsf{Der}}$ iff their $\{\}$-canonical forms are equal in $\mathcal{E}(\times)$, where $\mathcal{E}(\times)$ is the equational theory generated by the β and η rules for products:

$$\begin{array}{rcl} \Gamma \vdash \pi^j_{K_1,K_2}(\langle t_1, t_2\rangle) : K_j & = & \Gamma \vdash t_j : K_j \quad j = 1,2 \\ \Gamma \vdash \langle \pi^1_{K_1,K_2}(t), \pi^2_{K_1,K_2}(t)\rangle : K_1 \times K_2 & = & \Gamma \vdash t : K_1 \times K_2 \end{array}$$

3.2 Relating $\mathcal{C}$ and $\mathcal{C}^{\mathsf{Der}}$

Because of the way that the system $\mathcal{C}^{\mathsf{Der}}$ is set up, (1) each derivable judgement determines its unique derivation, and (2) derivations in $\mathcal{C}$ and judgements in $\mathcal{C}^{\mathsf{Der}}$ are notational variants: there is a map D from judgements in $\mathcal{C}^{\mathsf{Der}}$ to derivations in $\mathcal{C}$ and an inverse D^-.

On judgements, there is a map J taking derivable judgements in $\mathcal{C}^{\mathsf{Der}}$ to judgements in $\mathcal{C}$ which 'drops proof information':

$\mathsf{J}(\vdash K\ \mathsf{class}) = \vdash \overline{K}\ \mathsf{class}$
$\mathsf{J}(\Gamma \vdash \phi\ \mathsf{prop}) = \overline{\Gamma} \vdash \overline{\phi}\ \mathsf{prop}$
$\mathsf{J}(\Gamma \vdash t : K) = \overline{\Gamma} \vdash \overline{t} : \overline{K}$
$\mathsf{J}(\Gamma; \Delta \vdash \alpha : \phi) = \overline{\Gamma}; \overline{\Delta} \vdash \overline{\phi}$
where $\overline{v_1 : \phi_1, \ldots, v_n : \phi_n} = \overline{\phi_1}, \ldots, \overline{\phi_n}$

where $\overline{e}$ is defined on classes, propositional formulae and terms by:

$\overline{x} = x$
$\overline{[t,a]_{(x:K)\phi}} = \overline{t}$
$\overline{\nu^1_{(x:K)\phi}(t)} = \overline{t}$
$\overline{\langle t_1, t_2\rangle} = \langle \overline{t_1}, \overline{t_2}\rangle$
$\overline{\pi^j_{K_1,K_2}(t)} = \pi^j_{\overline{K_1},\overline{K_2}}(\overline{t})$
$\overline{k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n)} = k_{\overline{K_1},\ldots,\overline{K_n} \to \overline{K}}(\overline{t_1}, \ldots, \overline{t_n})$
$\overline{A} = A$
$\overline{K_1 \times K_2} = \overline{K_1} \times \overline{K_2}$
$\overline{\{x : K \mid \phi\}} = \{x : \overline{K} \mid \overline{\phi}\}$
$\overline{P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)} = P_{\overline{K_1},\ldots,\overline{K_n}}(\overline{t_1}, \ldots, \overline{t_n})$

These maps interact properly: A derivation D in $\mathcal{C}$ is a derivation of $\mathsf{J}(\mathsf{D}^-(D))$ (proof: simple structural induction on D) and hence, for judgement J in $\mathcal{C}^{\mathsf{Der}}$, $\mathsf{D}(J)$ is a derivation of $\mathsf{J}(J)$ in $\mathcal{C}$ and so, for all derivable J in $\mathcal{C}^{\mathsf{Der}}$, $\mathsf{J}(J)$ is derivable.

If we introduce an equational theory of judgements in $\mathcal{C}$, we can further establish: For derivations D_1 and D_2 in $\mathcal{C}$, if $\mathsf{D}^-(D_1) = \mathsf{D}^-(D_2)$ in $\mathcal{C}^{\mathsf{Der}}$, then $\mathsf{J}(\mathsf{D}^-(D_1)) = \mathsf{J}(\mathsf{D}^-(D_2))$ in $\mathcal{C}$. Here the equational theory of $\mathcal{C}$ is that generated from the β and η rules for product classes:

$$\begin{array}{rcl} \Gamma \vdash \pi^j_{K_1,K_2}(\langle t_1, t_2\rangle) : K_j & = & \Gamma \vdash t_j : K_j \quad (j = 1,2) \\ \Gamma \vdash \langle \pi^1_{K_1,K_2}(t), \pi^2_{K_1,K_2}(t)\rangle : K_1 \times K_2 & = & \Gamma \vdash t : K_1 \times K_2 \end{array}$$

3.2.1 Coherence: Counting derivations

This section is an aside from the main development of the paper and may be omitted by the reader. We describe the relationship between equality in $\mathcal{C}^{\mathsf{Der}}$ and the derivations in $\mathcal{C}$, giving a 'coherence' result which allows us to count the number of distinct derivations of a judgement in $\mathcal{C}$. We give a brief outline of results, motivating them with examples.

The example to bear in mind here is the two derivations of the same judgement given in Example 2.1. They differ because the predicate symbols on the right of the turnstile are derived from different occurrences in the context. To capture this, we follow standard treatment of such coherence results (see [Mac Lane 1982]) and introduce an appropriate notion of a *graph* which links occurrences of symbols.

For simplicity, we begin with a much reduced form of system $\mathcal{C}$. We drop product classes and, amongst the symbols in Σ, admit only class symbols A and predicate symbols $P_{K_1,\ldots,K_n}$ (no term formation symbols or axioms).

For each class expression in this reduced system, we extract the tree of its predicate symbols as follows:

$$\begin{array}{l} \|\{x_3 : \{x_2 : \{x_1 : A \mid P_A(x_1)\} \mid Q_{\{x_1:A \,\mid\, P_A(x_1)\}}(x_2)\} \mid R_{\{x_1:A \,\mid\, P_A(x_1)\}}(x_3)\}\| \\ \quad = [P_{\|A\|}, Q_{\|\{x_1:A \,\mid\, P_A(x_1)\}\|}, R_{\|\{x_1:A \,\mid\, P_A(x_1)\}\|}] \\ \quad = [P_{[]}, Q_{[P_{\|A\|}]}, R_{[P_{\|A\|}]}] \\ \quad = [P_{[]}, Q_{[P_{[]}]}, R_{[P_{[]}]}] \end{array}$$

We are interested in which trees correspond to well-formed classes. Such trees have the property that, at each node, the predicate symbols of the children of the node are a selection (including repetition) of those predicate symbols that occur as siblings to the left of the node. Thus, in the above example, the P_A which occurs as a child of R occurs also to the left of R at the same level as R in the tree. In general, child nodes may occur several times to the left of the node, in which case each choice corresponds to a distinct derivation. We thus label occurrences of predicate symbols with their "justifying" occurrence. In the above example, the P_A which occurs as a child of R is labelled with 1 to indicate that it derives from the first sibling of R.

Definition 3.2 *A* graph *(for this reduced system) is a tree of predicate symbols, each predicate symbol P not at the root being labelled with a natural number n such that, if (P, n) is a child of a node, then P occurs as the n-th sibling of the node (from left to right) and this sibling is to the left of the node. Moreover, for each predicate symbol P, the subtrees at all occurrences of P are identical.*

The last condition says that predicate symbols come with fixed classes of argument. From a tree which supports a graph structure, we may construct a unique (to within α-conversion and choice of class symbols) well-formed class expression: the tree $T = [P^1_{T_1}, P^2_{T_2}, \ldots, P^m_{T_m}]$ giving the class

$$\widetilde{T} = \{x_m : \{\cdots\{x_2 : \{x_1 : A \mid P^1_{\widetilde{T_1}}(x_1)\} \mid P^2_{\widetilde{T_2}}(x_2)\}\cdots\} \mid P^m_{\widetilde{T_m}}(x_m)\}.$$

For a judgement $J = x_1 : K_1, \ldots, x_n : K_n \vdash x_j : K, (1 \leq j \leq n)$ to be well-formed (i.e. derivable) the predicate symbols in K must be drawn from those in K_j in exactly the same way that predicate symbols are linked in a graph. Thus if we introduce a special predicate symbol Φ to stand from the turnstile, then a graph for J is a graph on the tree $\|\{x_j : K_j \mid \Phi_K(x_j)\}\|$.

That such a graph exists for derivable judgements, is a consequence of the following.

Proposition 3.4 *For this reduced form of system $\mathcal{C}$, there is a bijective correspondence between graphs for a derivable judgement J in $\mathcal{C}$ and $\{\}$-canonical forms C such that $\mathsf{J}(C) \equiv J$.*

Note that this says that the number of distinct derivations of judgement J in $\mathcal{C}$ (equational classes of judgements J' in $\mathcal{C}^{\mathsf{Der}}$ such that $\mathsf{D}(J') = J$) is the number of graphs for J.

The construction of a canonical form proceeds by structural induction on judgement J in $\mathcal{C}$. Conversely, the construction of a graph from a canonical form proceeds by structural induction on the canonical form. We illustrate here the exact correspondence between graphs and canonical forms. In Example 2.1, the first derivation of

$$z : \{y : \{x : A \mid P_A(x)\} \mid P_A(y)\} \vdash z : \{x : A \mid P_A(x)\} \qquad -(*)$$

in $\mathcal{C}$ corresponds to the canonical form:

$$\Gamma \vdash [\nu^1_{(x:A)P_A(x)}(\nu^1_\gamma(z)), \nu^2_\gamma(z)]_{(x:A)P_A(x)} : \{x : A \mid P_A(x)\} \qquad -(1)$$

where Γ is $z : \{y : \{x : A \mid P_A(x)\} \mid P_A(\nu^1_{(x:A)P_A(x)}(y))\}$ and the subscript γ is $(y : \{x : A \mid P_A(x)\})P_A(\nu^1_{(x:A)P_A(x)}(y))$.

The other derivation of $(*)$ is not in canonical form (it ends with a projection but is of comprehension class) but equals the following canonical form:

$$\Gamma \vdash [\nu^1_{(x:A)P_A(x)}(\nu^1_\gamma(z)), \nu^2_{(x:A)P_A(x)}(\nu^1_\gamma(z))]_{(x:A)P_A(x)} : \{x : A \mid P_A(x)\} \qquad -(2)$$

The two graphs corresponding to these canonical forms are, for (1)

$$[P_{[]}, P_{[]}, \Phi_{[(P_{[]},2)]}]$$

and for (2),

$$[P_{[]}, P_{[]}, \Phi_{[(P_{[]},1)]}]$$

differing only in the index of the third occurrence of P indicating which of the two previous occurrences is the justifying occurrence.

The link between canonical forms and graphs lies in the projections ν which indicate which occurrences of predicate symbols are being used. In more detail, each occurrence of ν^1 under a ν^2 moves the justifying occurrence to the left one place. Thus no ν^1 means that the justifying occurrence is immediately to the left of the node, one occurrence means that the justifying occurrence is two places to the left of the node, etc. This observation can be turned into a description of a bijective correspondence.

To extend this result to include products we consider multiple graphs with shared substructure and Proposition 3.4 is modified so that $\mathsf{J}(C)$ is not identical to J but is equal in the equational theory of products $\mathcal{E}(\times)$. Term constructor symbols and axioms each set up a correspondence between predicate symbols which fixes part of the graph structure. We then count only graphs varying over this fixed substructure. A full account of this would be out of place here but will appear elsewhere.

$$\frac{\vdash \Gamma\ \mathsf{class}}{\Gamma \vdash \top\ \mathsf{prop}}\ (\top\ \mathsf{Form}) \qquad \frac{\Gamma \vdash \phi_1\ \mathsf{prop} \quad \Gamma \vdash \phi_2\ \mathsf{prop}}{\Gamma \vdash \phi_1 \wedge \phi_2\ \mathsf{prop}}\ (\wedge\ \mathsf{Form})$$

$$\frac{\Gamma \vdash \Delta\ \mathsf{prop}}{\Gamma; \Delta \vdash * : \top}\ (\top\ \mathsf{Intro})$$

$$\frac{\Gamma; \Delta \vdash \alpha_1 : \phi_1 \quad \Gamma; \Delta \vdash \alpha_2 : \phi_2}{\Gamma; \Delta \vdash \langle \alpha_1, \alpha_2 \rangle : \phi_1 \wedge \phi_2}\ (\wedge\ \mathsf{Intro}) \qquad \frac{\Gamma; \Delta \vdash \alpha : \phi_1 \wedge \phi_2}{\Gamma; \Delta \vdash \pi^j_{\phi_1,\phi_2}(\alpha) : \phi_j}\ (\wedge\ \mathsf{Elim},\ \ j = 1,2)$$

Figure 7: The rules for $\top$ and $\wedge$.

3.3 Propositional logic

We now explain how to extend the above formal systems to include propositional logic. We introduce logical constants, inference rules and an equality on proofs of propositions based on β- and η-rules. The key idea is that we do not change the decision procedure to accommodate these new equations. Thus the procedure yields canonical forms which are no longer unique for each equational class, but are unique only to with the equational theory of propositional proofs. A decision procedure for this equational theory, which may well be available, would then provide a full decision procedure. However, this is not what is required for coherence: it introduces too fine a distinction. Perhaps we labour the point here but such structural independence properties ('extensibility' and 'modularity' results) are important in the incremental design of programming languages.

To illustrate these ideas we we choose a fairly minimal propositional logic containing only constants $\top$ (true) and $\wedge$ (conjunction). We add the standard inference rules for $\top$ and $\wedge$ (Figure 7) to $\mathcal{C}^{\mathsf{Der}}$ to define $\mathcal{C}^{\mathsf{Der}}(\top, \wedge)$.

The β- and η-rules for $\top$ and $\wedge$ are:

$$\begin{array}{rcl}
\Gamma; \Delta \vdash \alpha : \top & = & \Gamma; \Delta \vdash * : \top \\
\Gamma; \Delta \vdash \pi^j_{\phi_1,\phi_2}(\langle \alpha_1, \alpha_2 \rangle) : \phi_j & = & \Gamma; \Delta \vdash \alpha_j : \phi_j \quad j = 1,2 \\
\Gamma; \Delta \vdash \langle \pi^1_{\phi_1,\phi_2}(\alpha), \pi^2_{\phi_1,\phi_2}(\alpha) \rangle : \phi_1 \wedge \phi_2 & = & \Gamma; \Delta \vdash \alpha : \phi_1 \wedge \phi_2
\end{array}$$

We add these equations to those of $\mathcal{C}^{\mathsf{Der}}$ for the equational theory of $\mathcal{C}^{\mathsf{Der}}(\top, \wedge)$. The equational theory generated from the above equations alone we denote by $\mathcal{E}(\top, \wedge)$. We use essentially the same definition of canonical form, only including the clauses:

$\top$ is reduced, and $\phi_1 \wedge \phi_2$ is reduced when ϕ_1 and ϕ_2 are reduced,
$*$ is reduced,
$\langle \alpha_1, \alpha_2 \rangle$ is reduced when α_1 and α_2 are reduced,
$\pi^j_{\phi_1,\phi_2}(\alpha)$ is reduced when ϕ_1, ϕ_2 and α are reduced.

Then Proposition 3.3 becomes:

Proposition 3.5

1. *Canonical forms are equal in* $\mathcal{C}^{\mathsf{Der}}(\top, \wedge)$ *iff they are equal in* $\mathcal{E}(\top, \wedge)$,

2. *For each derivable judgement* J *in* $\mathcal{C}^{\mathsf{Der}}(\top, \wedge)$ *there is a canonical form* $\mathsf{C}(J)$ *with* $\mathsf{C}(J) = J$ *in* $\mathcal{C}^{\mathsf{Der}}(\top, \wedge)$ *and such that* $J_1 = J_2$ *in* $\mathcal{C}^{\mathsf{Der}}(\top, \wedge)$ *iff* $\mathsf{C}(J_1) = \mathsf{C}(J_2)$ *in* $\mathcal{E}(\top, \wedge)$.

The conversion to canonical form C is defined as previously but incorporating the above notion of reduced form for the new expressions. Notice that there is no η-expansion for expressions of type $\top$ or $\phi_1 \wedge \phi_2$.

Following this example, other logical constants such as $\supset$ (implication), $\vee$ (or) and $\perp$ (false) as well as quantification may be added. Implication introduces a λ-calculus of proofs $\lambda v : \phi.\alpha$. We may also consider adding recursion to this system. Of course, we will need to modify the notion of model to cope with these additional constructs. For $\supset$, $\vee$, $\perp$ and quantification, the relevant categorical structure is well-known (see [Lawvere 1970], for example).

4 A calculus of proofs (II)

We now present an alternative language, which we call $\mathcal{C}^{\mathsf{Env}}$, of derivations in the system of classes $\mathcal{C}$. In the previous system $\mathcal{C}^{\mathsf{Der}}$, the judgements are judgements of $\mathcal{C}$ modified with sufficient proof information to enable us to reconstruct a unique derivation. This additional proof information resides in the expressions of the modified language. As an alternative, we keep the language as close as possible to $\mathcal{C}$ and store the required proof information not in the expressions but in the context, in what we call *proof environments*. This is a rather radical departure from a logical viewpoint but is a very natural idea for programming languages. Indeed, it is this system $\mathcal{C}^{\mathsf{Env}}$ that is a 'language of type classes' and a major purpose of the paper is to analyse type classes by reference to the underlying logical system $\mathcal{C}$ (see Section 6).

Let us begin by looking again at the introduction rule for class comprehension in $\mathcal{C}$:

$$\frac{\Gamma \vdash t : K \quad x : K \vdash \phi \,\mathsf{prop} \quad \Gamma; \vdash \phi[t/x]}{\Gamma \vdash t : \{x : K \mid \phi\}}$$

What is missing from the conclusion is the proof α of satisfaction $\Gamma; \vdash \phi[t/x]$. Instead of modifying the term t to include α, we extend the context to include proofs of satisfaction together with a means of accessing the correct proof for any assertion that a term belongs to a class.

This idea is familiar from programming where we bind names to expressions for later use. Sequences of named expressions are called *environments* (sometimes *definitions*, *stores*, *states* or *bindings*). For example, in a λ-calculus we might have an environment

$$x = 3, f = (\lambda x : \mathsf{int}.x + 1).$$

We call the defined constants x and f *operators* (a terminology reflecting their use in type classes). In this environment, we may form judgements such as

$$x = 3, f = (\lambda x : \mathsf{int}.x + 1) \vdash f(3) = x + 1.$$

Environments differ from contexts which declare variables. In the above, x and f are not variables standing for hypothetical values which may be substituted with actual terms. However, we may substitute like for like, so that x may be replaced with 3 in $f(3) = x + 1$. Neither are environments general equational theories: We need access to unique occurrences of the operators. This uniqueness does not forbid some re-use of operators. For example,

$$f[\mathsf{int}] = (\lambda x : \mathsf{int}.x + 1), f[\mathsf{bool}] = (\lambda x : \mathsf{bool}.not(x))$$

$$\vdash \ (\mathsf{if}\ f[\mathsf{bool}](true)\ \mathsf{then}\ f[\mathsf{int}](0)\ \mathsf{else}\ f[\mathsf{int}](1)) = 2$$

or, with implicit typing,

$$f[\mathsf{int}] = (\lambda x : \mathsf{int}.x + 1), f[\mathsf{bool}] = (\lambda x : \mathsf{bool}.not(x))$$

$$\vdash \ (\mathsf{if}\ f(true)\ \mathsf{then}\ f(0)\ \mathsf{else}\ f(1)) = 2$$

In the latter case, we can disambiguate the operator f at each occurrence using the type information.

We now apply these ideas to the description of proofs. We first introduce proof environments, Θ. A proof environment is a sequence of entries of the form $o[t] = \alpha$ where o is an operator (simply a name), t a term and α a proof of a propositional formula $\phi[t/x]$. Thus, α is the proof that t satisfies ϕ. It is this proof that is required for the Intro rule for class comprehension and it may be accessed in a proof environment by operator o. This proof α has no free proof variables (there is no formula context Δ in the third hypothesis in the Intro rule).

Previously, for a term to belong to a class we wrote

$$t : \{x : K \mid \phi\}$$

This will no longer suffice. We modify the assertion so that we can access the proof of satisfaction $\phi[t/x]$ in a given proof environment. Just as the formulae ϕ ranges over possible terms t in the class, so does the proof of satisfaction. We thus write,

$$t : \{x : K \mid o : \phi\}$$

where o is an operator, and the proof of satisfaction is found as $o[t]$ in a given environment.

The syntax of this formal system, which we call $\mathcal{C}^{\mathsf{Env}}$, is as follows:

Classes	$K ::= A \mid K_1 \times K_2 \mid \{x : K \mid o : \phi\}$
Formulae	$\phi ::= P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)$
Terms	$t ::= x \mid k_{K_1,\ldots,K_n \to K}(t_1, \ldots, t_n) \mid \langle t_1, t_2\rangle \mid \pi^1_{K_1,K_2}(t) \mid \pi^2_{K_1,K_2}(t)$
Proofs	$\alpha ::= v \mid a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n \to \phi}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n) \mid o[t]$

The contexts Γ are as previous. The formula contexts Δ again include variables v ranging over proofs, so are of the form $v_1 : \phi_1, \ldots v_n : \phi_n$. Proof environments Θ are sequences of entries of the form $o[t] = \alpha$. What is meant by the unique occurrence of entries $o[t]$ in a context $\Gamma; \Theta$? We must ensure that there is unique occurrence of entries in Θ but, in addition, we must take account of entries in Γ. For instance, in $x : \{y : K \mid o : \phi\}$ there is an implied presence of $o[x]$. We thus define: $o[t]$ *occurs in* $\Gamma; \Theta$ according to the structure of t and $\Gamma; \Theta$ as follows.

$o[t]$ occurs in $o[t] = \alpha$,
$o[x]$ occurs in $x : \{y : K \mid o : \phi\}$,
$o[x]$ occurs in $x : \{y : K \mid o' : \phi\}$ with $o \neq o'$, if $o[x]$ occurs in $x : K$, and finally,
$o[\pi^j_{K_1,K_2}(x)]$ occurs in $x : K_1 \times K_2$ if $o[y]$ occurs in $y : K_j$, where y is a new variable ($j = 1, 2$).

Neither terms t nor proofs of satisfaction α in $o[t] = \alpha$ contain free proof variables v, hence it makes sense to order the contexts and environments as $\Gamma; \Theta; \Delta$. Thus the forms of judgement, modified from those of $\mathcal{C}^{\mathsf{Der}}$, are:

$\vdash K\,\mathsf{class}$ means that K is a well-formed class,

$\Gamma; \Theta \vdash \phi\,\mathsf{prop}$ means that ϕ is a well-formed propositional formula with free variables declared in context Γ, and operators defined in the proof environment Θ,

$\Gamma; \Theta \vdash t : K$ means that t is a well-formed term of class K with free variables declared in Γ and and operators defined in the proof environment Θ,

$\Gamma; \Theta; \Delta \vdash \alpha : \phi$ means that α is a proof that Δ entails ϕ in a context with free variables declared in Γ and operators defined in the proof environment Θ.

In the inference rules for $\mathcal{C}^{\mathsf{Env}}$, we avoid introducing well-formedness condition for contexts $\Gamma; \Theta$. Instead, we ensure that whenever an item in Θ is exhibited, the well-formedness of its constituents is ensured. A general assumption rule, involving a well-formed $\Gamma; \Theta$, is replaced by rules of exchange, weakening and a special assumption rule. These structural rules (Figure 8) are routine – they are included here for completeness of the description. The remainder of the rules (Figure 9) are those of $\mathcal{C}^{\mathsf{Der}}$ (modified to include proof environments) except those for comprehension where the operators o give access to proofs of satisfaction and the first elimination rule reverts to an analogue of that in $\mathcal{C}$. The rules for the symbols in Σ are those of $\mathcal{C}^{\mathsf{Der}}$ except that proof environments occur in the contexts.

Example 4.1. Let us look at a derivation in this system to see how proof environments work. We present a derivation of a judgement that corresponds to the first derivation in $\mathcal{C}$ in Example 2.1. Let Γ be the context

$$z : \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}$$

We show that from Γ we may derive $z : \{x : A \mid o : P_A(x)\}$ in a proof environment which includes the definition $o[z] = o_2[z]$ (there is a derivation of the same result which uses $o[z] = o_1[z]$).

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\ D\\ \Gamma; \vdash z : \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}\end{array}}{\Gamma; \vdash z : \{x : A \mid o_1 : P_A(x)\}}}{\Gamma; \vdash z : A}}{\Gamma; o[z] = o_2[z] \vdash z : A} \quad \dfrac{\vdash A\,\mathsf{class} \quad \dfrac{\vdash A\,\mathsf{class}}{x : A; \vdash x : A}}{x : A; \vdash P_A(x)\,\mathsf{prop}} \quad \begin{array}{c}\vdots\ D'\\ \Gamma; o[z] = o_2[z] \vdash o[z] : P_A(z)\end{array}}{\Gamma; o[z] = o_2[z] \vdash z : \{x : A \mid o : P_A(x)\}}$$

where derivation D is:

$$\frac{\Gamma;\Theta \vdash \phi\ \mathsf{prop}}{\Gamma;\tilde{\Theta} \vdash \phi\ \mathsf{prop}}\,(\mathsf{Exchange}_1) \quad \frac{\Gamma;\Theta \vdash t : K}{\Gamma;\tilde{\Theta} \vdash t : K}\,(\mathsf{Exchange}_2) \quad \frac{\Gamma;\Theta;\Delta \vdash \alpha : \phi}{\Gamma;\tilde{\Theta};\Delta \vdash \alpha : \phi}\,(\mathsf{Exchange}_3)$$

where, in each case, $\tilde{\Theta}$ is a permutation of Θ.

$$\frac{\Gamma;\Theta \vdash \phi'\ \mathsf{prop} \quad \Gamma;\Theta \vdash t : K \quad \Gamma;\Theta;\vdash \alpha : \phi}{\Gamma;\Theta, o[t] = \alpha \vdash \phi'\ \mathsf{prop}}\,(\mathsf{Weakening}_1)$$

$$\frac{\Gamma;\Theta \vdash t' : K' \quad \Gamma;\Theta \vdash t : K \quad \Gamma;\Theta;\vdash \alpha : \phi}{\Gamma;\Theta, o[t] = \alpha \vdash t' : K'}\,(\mathsf{Weakening}_2)$$

$$\frac{\Gamma;\Theta;\Delta \vdash \alpha' : \phi' \quad \Gamma;\Theta \vdash t : K \quad \Gamma;\Theta;\vdash \alpha : \phi}{\Gamma;\Theta, o[t] = \alpha;\Delta \vdash \alpha' : \phi'}\,(\mathsf{Weakening}_3)$$

$$\frac{\Gamma;\Theta \vdash t : K \quad \Gamma;\Theta;\vdash \alpha : \phi}{\Gamma;\Theta, o[t] = \alpha;\vdash o[t] : \phi}\,(\Theta\ \ \mathsf{Ass})$$

where, in the four rules above, $o[t]$ does not occur in $\Gamma;\Theta$.

Figure 8: The proof environment rules of system $\mathcal{C}^{\mathsf{Env}}$.

$$\frac{\Gamma;\Theta \vdash \phi_1\ \mathsf{prop} \cdots \Gamma;\Theta \vdash \phi_n\ \mathsf{prop}}{\Gamma;\Theta; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash v_j : \phi_j}\ (\,\mathsf{prop\ Ass},\quad n \geq 1, 1 \leq j \leq n)$$

$$\frac{\vdash K_1\ \mathsf{class}\ \cdots \vdash K_n\ \mathsf{class}}{x_1 : K_1, \ldots, x_n : K_n; \vdash x_j : K_j}\ (\,\mathsf{class\ Ass},\quad n \geq 1, 1 \leq j \leq n)$$

$$\frac{\vdash K_1\ \mathsf{class} \quad \vdash K_2\ \mathsf{class}}{\vdash K_1 \times K_2\ \mathsf{class}}\ (\times\ \mathsf{Form})$$

$$\frac{\Gamma;\Theta \vdash t_1 : K_1 \quad \Gamma;\Theta \vdash t_2 : K_2}{\Gamma;\Theta \vdash \langle t_1, t_2 \rangle : K_1 \times K_2}\ (\times\ \mathsf{Intro}) \qquad \frac{\Gamma;\Theta \vdash t : K_1 \times K_2}{\Gamma;\Theta \vdash \pi^j_{K_1,K_2}(t) : K_j}\ (\times\ \mathsf{Elim},\quad j = 1, 2)$$

$$\frac{x : K; \vdash \phi\ \mathsf{prop}}{\vdash \{x : K \mid o : \phi\}\ \mathsf{class}}\ (\{\}\ \mathsf{Form},\ \text{where } o \text{ does not appear in } K \text{ or } \phi.)$$

$$\frac{\Gamma;\Theta \vdash t : K \quad x : K; \vdash \phi\ \mathsf{prop} \quad \Gamma;\Theta; \vdash o[t] : \phi[t/x]}{\Gamma;\Theta \vdash t : \{x : K \mid o : \phi\}}\ (\{\}\ \mathsf{Intro})$$

$$\frac{\Gamma;\Theta \vdash t : \{x : K \mid o : \phi\}}{\Gamma;\Theta \vdash t : K}\ (\{\}\ \mathsf{Elim1}) \qquad \frac{\Gamma;\Theta \vdash t : \{x : K \mid o : \phi\} \quad \Gamma;\Theta \vdash \Delta\ \mathsf{prop}}{\Gamma;\Theta;\Delta \vdash o[t] : \phi[t/x]}\ (\{\}\ \mathsf{Elim2})$$

Figure 9: The proposition and class rules of system $\mathcal{C}^{\mathsf{Env}}$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash A\ \mathsf{class} \quad \dfrac{\vdash A\ \mathsf{class}}{x : A; \vdash x : A}}{x : A; \vdash P_A(x)\ \mathsf{prop}}}{\vdash \{x : A \mid o_1 : P_A(x)\}\ \mathsf{class}} \quad \dfrac{\dfrac{\dfrac{\dfrac{\vdash A\ \mathsf{class} \quad \dfrac{\vdash A\ \mathsf{class}}{x : A; \vdash x : A}}{x : A; \vdash P_A(x)\ \mathsf{prop}}}{\vdash \{x : A \mid o_1 : P_A(x)\}\ \mathsf{class}}}{y : \{x : A \mid o_1 : P_A(x)\}; \vdash y : \{x : A \mid o_1 : P_A(x)\}}}{y : \{x : A \mid o_1 : P_A(x)\}; \vdash y : A}}{y : \{x : A \mid o_1 : P_A(x)\}; \vdash P_A(y)\ \mathsf{prop}}}{\vdash \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}\ \mathsf{class}}}{\Gamma; \vdash z : \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}}
$$

and D' is:

$$
\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\ D\\ \Gamma; \vdash z : \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}\end{array}}{\Gamma; \vdash z : \{x : A \mid o_1 : P_A(x)\}}}{\Gamma; \vdash z : A} \quad \dfrac{\dfrac{\begin{array}{c}\vdots\ D\\ \Gamma; \vdash z : \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}\end{array}}{\Gamma; ; \vdash o_2[z] : P_A(z)}}{\Gamma; o[z] = o_2[z]; \vdash o_2[z] : P_A(z)}}{\Gamma; o[z] = o_2[z] \vdash o[z] : P_A(z)}
$$

4.1 Relating the formal systems

We now consider the relationship between the system of proof environments $\mathcal{C}^{\mathsf{Env}}$ and that of explicit proofs $\mathcal{C}^{\mathsf{Der}}$. In what sense is the formal system of proof environments $\mathcal{C}^{\mathsf{Env}}$ a description of derivations in $\mathcal{C}$? It may be imagined that we can reconstruct a derivation (a judgement in $\mathcal{C}^{\mathsf{Der}}$) from a judgement in $\mathcal{C}^{\mathsf{Env}}$ by simply replacing operators with the proofs to which they are associated in the environment. Each derivable judgement in $\mathcal{C}^{\mathsf{Env}}$ has sufficient proofs in the proof environment for such a reconstruction, but how these proofs are to be assembled into an actual proof is not uniquely determined by the judgement.

In this section, we give a construction of judgements in $\mathcal{C}^{\mathsf{Der}}$ from judgements in $\mathcal{C}^{\mathsf{Env}}$. This construction K is defined on derivable judgements in $\mathcal{C}^{\mathsf{Env}}$ and yields *canonical* judgements in $\mathcal{C}^{\mathsf{Der}}$ (as defined in Section 3.1). To show that the construction is defined on derivable judgements we introduce a further correspondence, E, this time from *derivations* in $\mathcal{C}^{\mathsf{Env}}$ to judgements in $\mathcal{C}^{\mathsf{Der}}$. The canonical nature of the results of the first correspondence allows us to show that the two correspondences 'cohere' in a way we make precise below.

We define the map K from judgements in $\mathcal{C}^{\mathsf{Env}}$ to judgements in $\mathcal{C}^{\mathsf{Der}}$. The key idea is as follows: For judgements $\Gamma; \Theta \vdash t : \{x : K \mid o : \phi\}$, we construct a judgement in $\mathcal{C}^{\mathsf{Der}}$ whose term is of the form $[t', \alpha']_{(x:K')\phi'}$ where α' is derived from a proof named as $o[t]$ in the context $\Gamma; \Theta$. The definition follows closely that of the conversion to canonical form (Section 3.1). Note that $o[t]$ may be defined in Θ or, in the case where t is a variable y, may occur in an item $y : \{x : K \mid o : \phi\}$ in the context Γ. The map K is a *partial* function, there may not be sufficient information in a judgement of $\mathcal{C}^{\mathsf{Env}}$ to construct a judgement in $\mathcal{C}^{\mathsf{Der}}$, in particular, a look-up via an operator may fail. We will show that K is defined on *derivable* judgements.

Definition 4.1 *On judgements of the form $\Gamma;\Theta\vdash t:K$, the translation* K *is defined by:*

$\mathsf{K}(\Gamma;\Theta\vdash t:K_1\times K_2)=\Gamma'\vdash\langle t'_1,t'_2\rangle:K'_1\times K'_2$
where $\Gamma'\vdash t'_j:K'_j=\mathsf{K}(\Gamma;\Theta\vdash\pi^j_{K_1,K_2}(t):K_j)$

$\mathsf{K}(\Gamma;\Theta\vdash t:\{x:K\mid o:\phi\})=\Gamma'\vdash[t',\alpha']_{(x:K')\phi'}:\{x:K'\mid\phi'\}$
where $\Gamma'\vdash t':K'=\mathsf{K}(\Gamma;\Theta\vdash t:K)$
and $x:K'\vdash\phi'\,\mathsf{prop}=\mathsf{K}(x:K;\vdash\phi\,\mathsf{prop})$
and $\Gamma';\vdash\alpha':\phi''=\mathsf{K}(\Gamma;\Theta;\vdash o[t]:\phi[t/x])$

$\mathsf{K}(\Gamma;\Theta\vdash t:A)=\mathsf{S}(\Gamma;\Theta\vdash t:A)$

where $\mathsf{S}(\Gamma;\Theta\vdash t:K)$ *is defined by:*

$\mathsf{S}(\Gamma;\Theta\vdash x:K)=\mathsf{K}(\Gamma)\vdash x:\mathsf{K}(K)$

$\mathsf{S}(\Gamma;\Theta\vdash\langle t_1,t_2\rangle:K_1\times K_2)=\Gamma'\vdash\langle t'_1,t'_2\rangle:K'_1\times K'_2$
where $\Gamma'\vdash t'_j:K'_j=\mathsf{S}(\Gamma;\Theta\vdash t_j:K_j)$

$\mathsf{S}(\Gamma;\Theta\vdash\pi^j_{K_1,K_2}(t):K_j)=$
if $\mathsf{S}(\Gamma;\Theta\vdash t:K_1\times K_2)\equiv\Gamma'\vdash\langle t_1,t_2\rangle:K'_1\times K'_2$
then $\Gamma'\vdash t_j:K'_j$ *else* $\Gamma'\vdash\pi^j_{K'_1,K'_2}(t'):K'_j$
where $\Gamma'\vdash t':K'_1\times K'_2=\mathsf{S}(\Gamma;\Theta\vdash t:K_1\times K_2)$

$\mathsf{S}(\Gamma;\Theta\vdash k_{K_1,\ldots,K_n\to K}(t_1,\ldots,t_n):K)=\Gamma'\vdash k_{K'_1,\ldots,K'_n\to K'}(t'_1,\ldots,t'_n):K'$
where $\Gamma'\vdash t'_j:K'_j=\mathsf{K}(\Gamma;\Theta\vdash t_j:K_j)$
and $\vdash\Gamma',K'\,\mathsf{class}=\mathsf{K}(\vdash\Gamma,K\,\mathsf{class})$

On other forms of judgement K *is defined componentwise, as follows:*

$\mathsf{K}(\vdash A\,\mathsf{class})=\vdash A\,\mathsf{class}$

$\mathsf{K}(\vdash K_1\times K_2\,\mathsf{class})=\vdash K'_1\times K'_2\,\mathsf{class}$
where $\vdash K'_j\,\mathsf{class}=\mathsf{K}(\vdash K_j\,\mathsf{class})$

$\mathsf{K}(\vdash\{x:K\mid o:\phi\}\,\mathsf{class})=\vdash\{x:K'\mid\phi'\}\,\mathsf{class}$
where $x:K'\vdash\phi'\,\mathsf{prop}=\mathsf{K}(x:K;\vdash\phi\,\mathsf{prop})$

$\mathsf{K}(\Gamma;\Theta\vdash P_{K_1,\ldots,K_n}(t_1,\ldots,t_n)\,\mathsf{prop})=\Gamma'\vdash P_{K'_1,\ldots,K'_n}(t'_1,\ldots,t'_n)\,\mathsf{prop}$
where $\Gamma'\vdash t'_j:K'_j=\mathsf{K}(\Gamma;\Theta\vdash t_j:K_j)$
and $\vdash\Gamma'\,\mathsf{class}=\mathsf{K}(\vdash\Gamma\,\mathsf{class})$

$\mathsf{K}(\Gamma;\Theta;\Delta\vdash v:\phi)=\Gamma';\Delta'\vdash v:\phi'$
where $\Gamma'\vdash\Delta',\phi'\,\mathsf{prop}=\mathsf{K}(\Gamma;\Theta\vdash\Delta,\phi\,\mathsf{prop})$

$\mathsf{K}(\Gamma;\Theta;\Delta\vdash a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n\to\phi}(t_1,\ldots,t_m;\alpha_1,\ldots,\alpha_n):\phi)$
$=\Gamma';\Delta'\vdash a_{K'_1,\ldots,K'_m;\phi'_1,\ldots,\phi'_n\to\phi'}(t'_1,\ldots,t'_m;\alpha'_1,\ldots,\alpha'_n):\phi'$
where $\Gamma'\vdash t'_j:K'_j=\mathsf{K}(\Gamma;\Theta\vdash t_j:K_j)$
and $\Gamma';\Delta'\vdash\alpha'_j:\phi'_j=\mathsf{K}(\Gamma;\Theta;\Delta\vdash\alpha_j:\phi_j)$
and $\Gamma'\vdash\Delta',\phi'\,\mathsf{prop}=\mathsf{K}(\Gamma;\Theta\vdash\Delta,\phi\,\mathsf{prop})$

Judgements of the form $\Gamma;\Theta;\Delta\vdash o[t]:\phi$ are a special case, for which the conversion K *is defined by induction on the form of the contexts. This describes how hypothetical and stored* *proofs are retrieved, and follows the above definition of an occurrence of $o[t]$ in $\Gamma;\Theta$.*

$\mathsf{K}(\Gamma;\Theta_1,o[t]=\alpha,\Theta_2;\Delta\vdash o[t]:\phi)=\mathsf{K}(\Gamma;\Theta_1,o[t]=\alpha,\Theta_2;\Delta\vdash \alpha:\phi)$

$\mathsf{K}(\Gamma_1,x:\{y:K\mid o:\phi\},\Gamma_2;\Theta;\Delta\vdash o[x]:\phi)=\Gamma';\Delta'\vdash \nu^2_{(y:K')\phi'}(x):\phi'[\nu^1_{(y:K')\phi'}(x)/y]$

where $\Gamma'\vdash x:\{y:K'\mid\phi'\}=\mathsf{S}(\Gamma_1,x:\{y:K\mid o:\phi\},\Gamma_2;\Theta\vdash x:\{y:K\mid o:\phi\})$

and $\Gamma'\vdash\Delta'\,\mathsf{prop}=\mathsf{K}(\Gamma_1,x:\{y:K\mid o:\phi\},\Gamma_2;\Theta\vdash\Delta\,\mathsf{prop})$

$\mathsf{K}(\Gamma_1,x:\{y:K\mid o':\phi\},\Gamma_2;\Theta;\Delta\vdash o[x]:\phi)=$

$\mathsf{K}(\Gamma_1,x:\{y:K\mid o':\phi\},z:K,\Gamma_2;\Theta;\Delta\vdash o[z]:\phi)$

$\mathsf{K}(\Gamma_1,x:K_1\times K_2,\Gamma_2;\Theta;\Delta\vdash o[\pi^j_{K_1,K_2}(x)]:\phi)=$

$\mathsf{K}(\Gamma_1,x:K_1\times K_2,z:K_j,\Gamma_2;\Theta;\Delta\vdash o[z]:\phi)$

Here z is a new variable and $o\neq o'$. We show below that K is defined on derivable *judgements in $\mathcal{C}^{\mathsf{Env}}$.*

Finally, we introduce a correspondence E between *derivations* in $\mathcal{C}^{\mathsf{Env}}$ and *judgements* in $\mathcal{C}^{\mathsf{Der}}$.

Definition 4.2 *Define E by structural induction on the derivations in $\mathcal{C}^{\mathsf{Env}}$. Most of the rules for $\mathcal{C}^{\mathsf{Env}}$ are those of $\mathcal{C}^{\mathsf{Der}}$ modified with proof environments and the definition of E is immediate for these rules. Only the rules for proof environments and class comprehension need special attention. We define E on these rules as follows. For the assumption rule for proof environments,*

$$\frac{\begin{array}{c}D_1\\ \Gamma;\Theta\vdash t:K\end{array}\qquad\begin{array}{c}D_2\\ \Gamma;\Theta;\vdash\alpha:\phi\end{array}}{\Gamma;\Theta,o[t]=\alpha;\vdash o[t]:\phi}$$

where $o[t]$ does not occur in context $\Gamma;\Theta$, and D_1 and D_2 are derivations of $\Gamma;\Theta\vdash t:K$ and $\Gamma;\Theta;\vdash\alpha:\phi$ respectively, the corresponding judgement in $\mathcal{C}^{\mathsf{Der}}$ for this derivation is simply $\mathsf{E}(D_2)$.

For the class introduction rule:

$$\frac{\begin{array}{c}D_1\\ \Gamma;\Theta\vdash t:K\end{array}\qquad\begin{array}{c}D_2\\ x:K;\vdash\phi\,\mathsf{prop}\end{array}\qquad\begin{array}{c}D_3\\ \Gamma;\Theta;\vdash o[t]:\phi[t/x]\end{array}}{\Gamma;\Theta\vdash t:\{x:K\mid o:\phi\}}\ (\{\}\ \mathsf{Intro})$$

the corresponding judgement in $\mathcal{C}^{\mathsf{Der}}$ is

$$\Gamma'\vdash[t',\alpha']_{(x:K')\phi'}:\{x:K'\mid\phi'\}$$

where $\Gamma'\vdash t':K'=\mathsf{E}(D_1)$, $x:K'\vdash\phi'\,\mathsf{prop}=\mathsf{E}(D_2)$ *and* $\Gamma';\vdash\alpha':\phi''=\mathsf{E}(D_3)$.

Similarly, for the two elimination rules,

$$\frac{\begin{array}{c}D_1\\ \Gamma;\Theta\vdash t:\{x:K\mid o:\phi\}\end{array}}{\Gamma;\Theta\vdash t:K}\ (\{\}\ \mathsf{Elim1})$$

$$\frac{\begin{array}{c}D_1\\ \Gamma;\Theta\vdash t:\{x:K\mid o:\phi\}\end{array}\qquad\begin{array}{c}D_2\\ \Gamma;\Theta\vdash\Delta\,\mathsf{prop}\end{array}}{\Gamma;\Theta;\Delta\vdash o[t]:\phi[t/x]}\ (\{\}\ \mathsf{Elim2})$$

the corresponding judgements in $\mathcal{C}^{\mathsf{Der}}$ *are, for the first rule:*

$$\Gamma' \vdash \nu^1_{(x:K')\phi'}(t') : K'$$

where $\Gamma' \vdash t' : \{x : K' \mid \phi'\} = \mathsf{E}(D_1)$*, and for the second rule:*

$$\Gamma'; \Delta' \vdash \nu^2_{(x:K')\phi'}(t') : \phi'[\nu^1_{(x:K')\phi'}(t')/x]$$

where $\Gamma' \vdash t' : \{x : K' \mid \phi'\} = \mathsf{E}(D_1)$ *and* $\Gamma' \vdash \Delta' \,\mathsf{prop} = \mathsf{E}(D_2)$*.*

The following proposition describes the requisite properties of the conversions K from judgements and E from derivations.

Proposition 4.1 *If derivation* D *establishes judgement* J *in* $\mathcal{C}^{\mathsf{Env}}$*, then judgement* $\mathsf{K}(J)$ *is defined and* $\mathsf{C}(\mathsf{E}(D)) \equiv \mathsf{K}(J)$ *(syntactically identical), where* C *is the conversion to canonical form (Section 3.1).*

Proof. Note that the proposition says that $\mathsf{K}(J)$ is in canonical form and is equal to $\mathsf{E}(J)$ in the equational theory of $\mathcal{C}^{\mathsf{Der}}$.

The proof proceeds by structural induction on the derivation D. The key ideas can be illustrated with an example. Suppose that D concludes with an $\mathsf{Elim1}$-rule for comprehension, so that D looks like:

$$\frac{\begin{array}{c} D_1 \\ \Gamma; \Theta \vdash t : \{x : K \mid o : \phi\} \end{array}}{\Gamma; \Theta \vdash t : K}$$

where D_1 is a derivation of $\Gamma; \Theta \vdash t : \{x : K \mid o : \phi\}$.

By the definition of E,

$$\mathsf{E}(D) = \Gamma' \vdash \nu^1_{(x:K')\phi'}(t') : K'$$

where $\Gamma' \vdash t' : \{x : K' \mid \phi'\} = \mathsf{E}(D_1)$.

The induction hypothesis gives that $\mathsf{K}(\Gamma; \Theta \vdash t : \{x : K \mid o : \phi\})$ is defined and identical to $\mathsf{C}(\mathsf{E}(D_1))$. By the definition of K, this is

$$\begin{array}{l} \Gamma'' \vdash [t'', \alpha'']_{(x:K'')\phi''} : \{x : K'' \mid \phi''\} \\ \quad \text{where } \Gamma'' \vdash t'' : K'' = \mathsf{K}(\Gamma; \Theta \vdash t : K) \\ \quad \text{and } x : K'; \vdash \phi'' \,\mathsf{prop} = \mathsf{K}(x : K; \vdash \phi \,\mathsf{prop}) \\ \quad \text{and } \Gamma''; \vdash \alpha'' : \phi^0 \,\mathsf{prop} = \mathsf{K}(\Gamma; \Theta; \vdash o[t] : \phi[t/x]) \end{array}$$

But $\mathsf{C}(\mathsf{E}(D_1)) = \mathsf{E}(D_1)$ in $\mathcal{C}^{\mathsf{Der}}$. The rules for equality now give that

$$\Gamma' \vdash \nu^1_{(x:K')\phi'}(t') : K' \;=\; \Gamma'' \vdash \nu^1_{(x:K'')\phi''}([t'', \alpha'']_{(x:K'')\phi''}) : K''$$

The RHS is equal in $\mathcal{C}^{\mathsf{Der}}$ to $\Gamma'' \vdash t'' : K''$ which is $\mathsf{K}(\Gamma; \Theta \vdash t : K)$. Moreover, by Definition 3.1 (the definition of canonical form), because $\Gamma'' \vdash [t'', \alpha'']_{(x:K'')\phi''} : \{x : K'' \mid \phi''\}$ is canonical, then so is $\Gamma'' \vdash t'' : K''$. Hence,

$$\mathsf{C}(\mathsf{E}(D)) \equiv \mathsf{K}(\Gamma; \Theta \vdash t : K)$$

as required.

We are now in a position to give a general coherence result for $\mathcal{C}^{\mathsf{Env}}$. We phrase this in the form of a principle for the well-definedness of functions on $\mathcal{C}^{\mathsf{Env}}$.

Corollary 4.1 *If μ is a map from derivations in $\mathcal{C}^{\mathsf{Env}}$ which factors as*

$$\begin{tikzcd} Der(\mathcal{C}^{\mathsf{Env}}) \arrow[rr, "\mathsf{E}"] \arrow[dr, "\mu"'] & & \mathcal{C}^{\mathsf{Der}} \arrow[dl, "\mu'"] \\ & S & \end{tikzcd}$$

where $\mathsf{E} : Der(\mathcal{C}^{\mathsf{Env}}) \to \mathcal{C}^{\mathsf{Der}}$ is defined in Definition 4.2 above, and μ' respects equality (i.e. $J_1 = J_2$ in $\mathcal{C}^{\mathsf{Der}}$ implies $\mu'(J_1) = \mu'(J_2)$ in S), then the map μ is independent of derivation in the sense that, if D_1 and D_2 are both derivations of judgement J in $\mathcal{C}^{\mathsf{Env}}$, then $\mu(D_1) = \mu(D_2)$ in S. In fact, $\mu(D) = \mu'(\mathsf{K}(J))$, where K is the map from judgements in $\mathcal{C}^{\mathsf{Env}}$ to those in $\mathcal{C}^{\mathsf{Der}}$ defined above.

Proof. Let derivation D establish judgement J in $\mathcal{C}^{\mathsf{Env}}$. Now, by Proposition 4.1, $\mathsf{E}(D) = \mathsf{K}(J)$ in $\mathcal{C}^{\mathsf{Der}}$ and so $\mu'(\mathsf{E}(D)) = \mu'(\mathsf{K}(J))$ in S (as μ' respects equality). Thus $\mu(D) = \mu'(\mathsf{K}(J))$ in S.

We now use this method for defining maps to give a semantics of the system $\mathcal{C}^{\mathsf{Env}}$ via a semantics of $\mathcal{C}^{\mathsf{Der}}$.

5 Models

We begin by defining categorical models for the system $\mathcal{C}^{\mathsf{Der}}$. There is a standard interpretation of first order logics in indexed categories and fibrations (see, for example, [Pitts 1987] or [Jacobs 1990]), where terms denote arrows in the base category and proofs (or entailment relations) denote arrows in the fibres. We extend this to include class comprehension using comprehension schemata in indexed categories. We present here a simple form of the semantics using strict indexed categories (i.e. functors $p : \mathbb{C}^{\mathrm{op}} \to \mathbf{Cat}$). It is possible to remove the strictness requirement so that the indexed category is functorial only up to isomorphism (as in [Lawvere 1970]), or consider comprehension schemata in fibrations as in [Jacobs 1990], [Jacobs 1991]. The language $\mathcal{C}^{\mathsf{Der}}$ may be interpreted in these models, some of the equations being interpreted as isomorphisms.

5.1 Indexed categories and comprehension

A strict indexed category is a functor $p : \mathbb{C}^{\mathrm{op}} \to \mathbf{Cat}$. For indexed category $p : \mathbb{C}^{\mathrm{op}} \to \mathbf{Cat}$, $\mathbb{C}$ is the *base category* and $p(K)$ the *fibre* over K. For arrow s in the base, we write s^* for the functor $p(s)$ when p is understood. The composition of arrows in categories is written as ';' in diagrammatic order.

We introduce a simple form of comprehension schema:

Definition 5.1 *Let $p : \mathbb{C}^{\mathrm{op}} \to \mathbf{Cat}$ be an indexed category such that each fibre $p(K)$ has a terminal object $\top_K$ preserved under the functors s^*. A* **comprehension schema** *in p consists of*

1. *for each object L of $\mathbb{C}$ a functor taking objects ψ in $p(L)$ to objects $\{L \mid \psi\}$ in $\mathbb{C}$ and taking arrows $\alpha : \psi \to \psi'$ to arrows $\{L \mid \alpha\} : \{L \mid \psi\} \to \{L \mid \psi'\}$. For each L and ψ there is an arrow $\gamma_{L,\psi} : \{L \mid \psi\} \to L$ such that $\{L \mid \alpha\}; \gamma_{L,\psi'} = \gamma_{L,\psi}$,*

2. *for each $s : K \to L$, a natural bijection between arrows*

$$\alpha : \top_K \to s^*(\psi)$$

in the fibre $p(K)$ and arrows $h : K \to \{L \mid \psi\}$ such that the following commutes.

$$\begin{tikzcd} K \arrow[rr, "h"] \arrow[dr, "s"'] & & \{L \mid \psi\} \arrow[dl, "\gamma_{L,\psi}"] \\ & L & \end{tikzcd}$$

In brief, a comprehension schema is a functor $\{L \mid \_\} : p(L) \to \mathbb{C}/L$ for each object L of $\mathbb{C}$ and a natural family of bijections,

$$p(K)(\top_K, s^*(\psi)) \simeq (\mathbb{C}/L)(s, \{L \mid \psi\})$$

where $\mathbb{C}/L$ is the slice category over L.

Comprehension schemata in an indexed category are unique to within an isomorphism. For examples and properties of comprehension schemata see [Lawvere 1970].

An arrow $\alpha : \top_K \to s^*(\psi)$ in the fibre $p(K)$, where $s : K \to L$, induces, under the bijection, an arrow $K \to \{L \mid \psi\}$, which we denote by $\theta_\psi(s, \alpha)$, so that $\theta_\psi(s, \alpha); \gamma_{L,\psi} = s$. In the other direction, the image under the bijection of an arrow $h : K \to \{L \mid \psi\}$, such that $s = h; \gamma_{L,\psi}$, is denoted $\bar{\theta}_{L,\psi}(h) : \top_K \to s^*\psi$. Naturality of the bijection in $s : K \to L$ is the following equation.

$$h; \theta_\psi(s, \alpha) \quad = \quad \theta_\psi(h; s, h^*\alpha) \tag{1}$$

Naturality in ψ is the following, for any $\alpha : \psi \to \psi'$ in $p(L)$:

$$\theta_\psi(s, \beta); \{L \mid \alpha\} \quad = \quad \theta_{\psi'}(s, \beta; s^*(\alpha)) \tag{2}$$

The corresponding equations for $\bar{\theta}$ are:

$$\bar{\theta}_{L,\psi}(h; g) \quad = \quad h^*(\bar{\theta}_{L,\psi}(g)) \tag{3}$$
$$\bar{\theta}_{L,\psi'}(g; \{L \mid \alpha\}) \quad = \quad \bar{\theta}_{L,\psi}(g); s^*\alpha \tag{4}$$

which are equivalent to (1) and (2) given that θ and $\bar{\theta}$ are inverse.

5.2 A semantics of $\mathcal{C}^{\mathsf{Der}}$

Models for the interpretation $\mathcal{C}^{\mathsf{Der}}$ are to be strict indexed categories with a comprehension schema and finite products in the base category and in the fibres, those in the fibres being strictly preserved by the functors s^*. The terminal object in the base is denoted 1 with terminal arrows $1_K : K \to 1$ and the binary product in the base is $\times$. In each fibre $p(K)$, the terminal object is denoted $\top_K$ with terminal arrows $\top_\psi : \psi \to \top_K$ and binary product is denotes $\wedge$. In the base category and in the fibres, product pairing is $\langle\, , \,\rangle$ and projections are $\pi^j_{K_1,K_2}$ $(j = 1, 2)$.

For a signature Σ, a Σ-structure in such an indexed category p consists of an interpretation for each symbol in Σ. In detail, (1) each class symbol A denotes an object $[\![A]\!]$ in the base category, (2) each predicate symbol $P_{K_1,\ldots,K_n}$ denotes an object $[\![P]\!]$ in the fibre $p([\![K_1]\!] \times \cdots \times [\![K_n]\!])$, (3) each term constructor $k_{K_1,\ldots,K_n\to K}$ denotes an arrow $[\![k]\!] : [\![K_1]\!] \times \cdots \times [\![K_n]\!] \to [\![K]\!]$ and, finally, (4) each axiom

$$\Gamma; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n\to\phi}(x_1, \ldots, x_m; v_1, \ldots, v_n) : \phi$$

denotes an arrow $[\![a]\!] : [\![\Gamma \vdash \phi_1\,\mathsf{prop}]\!] \wedge \cdots \wedge [\![\Gamma \vdash \phi_n\,\mathsf{prop}]\!] \to [\![\Gamma \vdash \phi\,\mathsf{prop}]\!]$ in the fibre $p([\![K_1]\!] \times \cdots \times [\![K_m]\!])$ where $\Gamma = x_1 : K_1, \ldots, x_m : K_m$. The semantics $[\![J]\!]$ of judgement J is then defined by structural induction as follows.

The semantics of judgement $\vdash K\,\mathsf{class}$ is an object $[\![K]\!]$ of $\mathcal{C}$ defined as:

$$\begin{aligned}
&[\![A]\!] = [\![A]\!]\\
&[\![K_1 \times K_2]\!] = [\![K_1]\!] \times [\![K_2]\!]\\
&[\![\{x : K \mid \phi\}]\!] = \{[\![K]\!] \mid [\![x : K \vdash \phi\,\mathsf{prop}]\!]\}
\end{aligned}$$

The semantics of a context $x_1 : K_1, \ldots, x_n : K_n$ is given by

$$[\![x_1 : K_1, \ldots, x_n : K_n]\!] = [\![K_1]\!] \times \cdots \times [\![K_n]\!].$$

Judgements of the form $\Gamma \vdash \phi\,\mathsf{prop}$ are interpreted as objects in the fibre over $[\![\Gamma]\!]$ as follows:

$$[\![\Gamma \vdash P_{K_1,\ldots,K_n}(t_1, \ldots, t_n)\,\mathsf{prop}]\!] = \langle [\![\Gamma \vdash t_1 : K_1]\!], \ldots, [\![\Gamma \vdash t_n : K_n]\!]\rangle^* [\![P]\!]$$

The semantics of judgement $\Gamma \vdash t : K$ is an arrow from $[\![\Gamma]\!]$ to $[\![K]\!]$ in the category $\mathbb{C}$:

$$\begin{aligned}
&[\![x_1 : K_1, \ldots, x_n : K_n \vdash x_j : K_j]\!] = \pi^j_{[\![K_1]\!],\ldots,[\![K_n]\!]}\\
&[\![\Gamma \vdash k_{K_1,\ldots,K_n\to K}(t_1, \ldots, t_n) : K]\!] = \langle [\![\Gamma \vdash t_1 : K_1]\!], \ldots, [\![\Gamma \vdash t_n : K_n]\!]\rangle; [\![k]\!]\\
&[\![\Gamma \vdash \langle t_1, t_2\rangle : K_1 \times K_2]\!] = \langle [\![\Gamma \vdash t_1 : K_1]\!], [\![\Gamma \vdash t_2 : K_2]\!]\rangle\\
&[\![\Gamma \vdash \pi^j_{K_1,K_2}(t) : K_j]\!] = [\![\Gamma \vdash t : K_1 \times K_2]\!]; \pi^j_{[\![K_1]\!],[\![K_2]\!]}\\
&[\![\Gamma \vdash [t, \alpha]_{(x:K)\phi} : \{x : K \mid \phi\}]\!] = \theta_{[\![x:K\vdash\phi\,\mathsf{prop}]\!]}([\![\Gamma \vdash t : K]\!], [\![\Gamma; \vdash \alpha : \phi[t/x]]\!])\\
&[\![\Gamma \vdash \nu^1_{(x:K)\phi}(t) : K]\!] = [\![\Gamma \vdash t : \{x : K \mid \phi\}]\!]; \gamma_{[\![K]\!],[\![x:K\vdash\phi\,\mathsf{prop}]\!]}
\end{aligned}$$

A context $\Gamma; v_1 : \phi_1, \ldots, v_n : \phi_n$ is interpreted as an object $[\![\Gamma \vdash \phi_1\,\mathsf{prop}]\!] \wedge \cdots \wedge [\![\Gamma \vdash \phi_n\,\mathsf{prop}]\!]$ in the fibre over $[\![\Gamma]\!]$. Finally, a judgement $\Gamma; \Delta \vdash \alpha : \phi$ is interpreted as an arrow from $[\![\Gamma; \Delta]\!]$ to $[\![\Gamma \vdash \phi\,\mathsf{prop}]\!]$ in the fibre over $[\![\Gamma]\!]$ defined as follows.

$$\begin{aligned}
&[\![\Gamma; v_1 : \phi_1, \ldots, v_n : \phi_n \vdash v_j : \phi_j]\!] = \pi^j_{[\![\Gamma\vdash\phi_1\,\mathsf{prop}]\!],\ldots,[\![\Gamma\vdash\phi_n\,\mathsf{prop}]\!]}\\
&[\![\Gamma; \Delta \vdash a_{K_1,\ldots,K_m;\phi_1,\ldots,\phi_n\to\phi}(t_1, \ldots, t_m; \alpha_1, \ldots, \alpha_n) : \phi]\!] = \langle \beta_1, \ldots, \beta_n\rangle; (\langle s_1, \ldots, s_m\rangle^* [\![a]\!])\\
&\quad\text{where } s_i = [\![\Gamma \vdash t_i : K_i]\!]\\
&\quad\text{and } \beta_i = [\![\Gamma; \Delta \vdash \alpha_i : \phi_i[t_1/x_1, \ldots, t_m/x_m]]\!]\\
&[\![\Gamma; \Delta \vdash \nu^2_{(x:K)\phi}(t) : \phi[t/x]]\!] = \top_{[\![\Gamma;\Delta]\!]}; \bar\theta_{[\![K]\!],[\![x:K\vdash\phi\,\mathsf{prop}]\!]}([\![\Gamma \vdash t : \{x : K \mid \phi\}]\!])
\end{aligned}$$

Proposition 5.1

1. *If J is a derivable judgement in $\mathcal{C}^{\mathsf{Der}}$, then $[\![J]\!]$ is defined in all Σ-structures,*

2. *If $J_1 = J_2$ is an equational judgement in the equational theory of $\mathcal{C}^{\mathsf{Der}}$, then $[\![J_1]\!] = [\![J_2]\!]$ in all Σ-structures.*

Example. (1) is by structural induction on the derivation. (2) follows from the fact that the generating equations (1) and (2) hold in Σ-structures, and the following substitution lemmas hold.

$$\begin{array}{l} [\![\Gamma \vdash \phi[t_1/x_1,\ldots,t_n/x_n]\,\mathsf{prop}]\!] = \\ \quad \langle [\![\Gamma \vdash t_1 : K_1]\!],\ldots,[\![\Gamma \vdash t_1 : K_1]\!]\rangle^* [\![x_1 : K_1,\ldots,x_n : K_n \vdash \phi\,\mathsf{prop}]\!] \\ [\![\Gamma \vdash t[t_1/x_1,\ldots,t_n/x_n] : K]\!] = \\ \quad \langle [\![\Gamma \vdash t_1 : K_1]\!],\ldots,[\![\Gamma \vdash t_1 : K_1]\!]\rangle ; [\![x_1 : K_1,\ldots,x_n : K_n \vdash t : K]\!] \\ [\![\Gamma;\Delta[t_1/x_1,\ldots,t_n/x_n] \vdash \alpha[t_1/x_1,\ldots,t_n/x_n] : \phi[t_1/x_1,\ldots,t_n/x_n]\,\mathsf{prop}]\!] = \\ \quad \langle [\![\Gamma \vdash t_1 : K_1]\!],\ldots,[\![\Gamma \vdash t_1 : K_1]\!]\rangle^* [\![x_1 : K_1,\ldots,x_n : K_n;\Delta \vdash \alpha : \phi]\!] \\ [\![\Gamma;\Delta \vdash \alpha[\alpha_1/v_1,\ldots,\alpha_n/v_n]\phi]\!] = \\ \quad \langle [\![\Gamma;\Delta \vdash \alpha_1 : \phi_1]\!],\ldots,[\![\Gamma;\Delta \vdash \alpha_1 : \phi_1]\!]\rangle ; [\![\Gamma; v_1 : \phi_1,\ldots,v_n : \phi_n \vdash \alpha : \phi\,\mathsf{prop}]\!] \end{array}$$

□

This semantics is also complete – we may construct a classifying model $p : \mathbb{C}^{\mathrm{op}} \to \mathbf{Cat}$ from the syntax of $\mathcal{C}^{\mathsf{Der}}$ in a standard way, as long as we add finite products of propositions and a terminal class 1 and associated rules (or change the syntax for classes to be $\{\Gamma \mid \phi\}$ or even $\{\Gamma \mid \Delta\}$). Objects in $\mathbb{C}$ are constructed from derivable judgements $\vdash K\,\mathsf{class}$ under $\mathcal{C}^{\mathsf{Der}}$-equality. Arrows from $\vdash \Gamma\,\mathsf{class}$ to $\vdash K\,\mathsf{class}$ are derivable judgements $\Gamma \vdash t : K$ under $\mathcal{C}^{\mathsf{Der}}$-equality. Objects in the fibre over $\vdash \Gamma\,\mathsf{class}$ are constructed from derivable judgements $\Gamma \vdash \phi\,\mathsf{prop}$ under $\mathcal{C}^{\mathsf{Der}}$-equality. Arrows in the fibre from $\Gamma \vdash \Delta\,\mathsf{prop}$ to $\Gamma \vdash \phi\,\mathsf{prop}$ are derivable judgements $\Gamma;\Delta \vdash \alpha : \phi$ under $\mathcal{C}^{\mathsf{Der}}$-equality. The results in Propositions 3.1 and 3.2 ensure that this is well-defined. Class comprehension in $\mathcal{C}^{\mathsf{Der}}$ defines a comprehension schema in this indexed category.

5.3 A semantics of $\mathcal{C}^{\mathsf{Env}}$

After this rather lengthy exposition, we are now in a position to give a semantics of $\mathcal{C}^{\mathsf{Env}}$. To do so, we use the relationship of $\mathcal{C}^{\mathsf{Der}}$ to $\mathcal{C}^{\mathsf{Env}}$, the coherence result for defining maps from $\mathcal{C}^{\mathsf{Env}}$, and the above semantics of $\mathcal{C}^{\mathsf{Der}}$. Define the semantics of a derivation D of a judgement J in $\mathcal{C}^{\mathsf{Env}}$ to be $[\![\mathsf{E}(D)]\!]$. The soundness of the semantics (Proposition 5.1) and the coherence result (Corollary 4.1) establish that $[\![\mathsf{E}(D)]\!]$ is independent of the choice of derivation D and in fact $[\![\mathsf{E}(D)]\!] = [\![\mathsf{K}(J)]\!]$.

An equational theory for the language $\mathcal{C}^{\mathsf{Env}}$ has no role in the development. We could however, formulate such an equational theory – it would be useful, for example when comparing an operational semantics of $\mathcal{C}^{\mathsf{Env}}$ with this denotational semantics. The equational theory could include β- and η-rules as well as the equations in the proof environments. Soundness for these equations follows from the soundness of the semantics of $\mathcal{C}^{\mathsf{Der}}$ and the treatment of proof environments in $\mathcal{C}^{\mathsf{Env}}$.

One of the starting points for this paper was an attempt to give a semantics for languages of type classes in terms of comprehension schemata in categories. Any such attempt requires formalising a language of type classes and a coherence result to show that the semantics is properly defined. In [Hilken, Rydeheard 1992], we defined a semantics directly, starting with a language with unusual features which were difficult to justify except by comparison with existing languages. The coherence was established through rewriting categorical expressions. Whilst coherence results are often treated categorically, in this case the proof looked opaque. Here, the treatment is more elaborate: we introduce an underlying logical system $\mathcal{C}$ and an intermediate language of derivations $\mathcal{C}^{\mathsf{Der}}$. This enables us to give some justification for the form of the language of type classes and provides a medium, $\mathcal{C}^{\mathsf{Der}}$, in which to state and prove coherence using more familiar normalisation methods.

6 Type classes

We now show how the above analysis applies to languages with type classes, in particular we show that $\mathcal{C}^{\mathsf{Env}}$ is itself such a language. To do so, a shift of perspective is needed: What we have previously called propositions are to be interpreted as *types*, proofs of propositions are now *terms*, classes are indeed *type classes* (or products of them), and what we previously called terms are now *type schemata*. For reference, this is set out in the following table:

	Logical systems	Functional languages
α	Proofs of propositions	Terms (programs)
ϕ	Propositional formulae	Types
K	Classes	Kinds - (products of) type classes
t	Terms	Type schemata

For this correspondence to make sense, we consider systems with a distinguished class Ω (to be thought of as the class of all types) and a bijective rule between judgements $\Gamma \vdash t : \Omega$ and judgements $\Gamma \vdash t : \mathsf{prop}$, or, indeed, drop judgements of the form $\Gamma \vdash \phi : \mathsf{prop}$ altogether as in [Hilken, Rydeheard 1992]. This follows the standard treatment of programming languages as first order systems and so called "hyperdoctrine models" (see [Pitts 1987], for example).

Following this interpretation, $\mathcal{C}^{\mathsf{Env}}$ is a language of type classes with the same basic type structure as introduced in [Wadler, Blott 1989] and implemented in Haskell [Hudak, Wadler 1990] and Gofer [Jones 1992]. To illustrate this, we revisit the running example of the paper (Example 2.1) and give a corresponding fragment of Haskell. Recall (Example 4.1) that we gave a derivation in $\mathcal{C}^{\mathsf{Env}}$ of the judgement:

$$z : \{y : \{x : A \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}; o[z] = o_2[z] \vdash z : \{x : A \mid o : P_A(x)\}$$

Corresponding to this judgement (setting $A = \Omega$ and $P_A(x) = x$) is the Haskell program:

```
class C1 x where o1 :: x

class (C1 y) => (C2 y) where o2 :: y

class C x where o :: x

instance (C2 z) => (C z) where o = o2.
```

The alternative judgement, with the environment $o[z] = o_1[z]$, is the same program only replacing the last line with

```
instance (C2 z) => (C z) where o = o1.
```

The syntax differs from that in a formal system. In the program, we incrementally build classes in terms of previous classes, naming the class built at each stage. Thus `C1` corresponds to the class $\{x : A \mid o_1 : P_A(x)\}$. The class `C2`, which is built from `C1`, corresponds to $\{y : \{x : K \mid o_1 : P_A(x)\} \mid o_2 : P_A(y)\}$ and `C` corresponds to $\{x : A \mid o : P_A(x)\}$. The `instance` declaration in the program shows how any type `z` of class `C2` may be considered to be of class `C` by defining the operator `o` in in terms of `o1`. This correspond to the above judgement in $\mathcal{C}^{\mathsf{Env}}$.

In [Jones 1992], there is a detailed analysis of functional languages with type classes. A proof is given of the correctness of a compiler – a translation from a language with implicit 'evidence' to one with explicit 'evidence'. A coherence result is required and, to this end, an equational theory of judgements is introduced with both β- and η-rules. The analysis is, in overall structure, that of the relationship between $\mathcal{C}^{\mathsf{Env}}$ and $\mathcal{C}^{\mathsf{Der}}$ in this paper, although details differ. [Jones 1992] is somewhat more general, especially in the treatment of parametric polymorphism, but lacks the underlying logical systems of this paper.

Other related work includes [Curien, Ghelli 1990] which examines coherence for languages with explicit subtyping; [Bailey 1996] which introduces implicit type conversions into the type system LEGO and [Nipkow, Snelting 1990] which deals with the overloading mechanism in type classes using order-sorted algebra.

7 Conclusions

This has been a fairly lengthly exercise in understanding a feature of programming languages from a logical viewpoint. The treatment revolved around an underlying logical system $\mathcal{C}$ and a correspondence between its derivations and programs in a (prototype) programming language. Because the language has a form of polymorphism in which types may belong to more than one class, this correspondence involves a 'coherence' result.

There is clearly more to do using this approach. The language $\mathcal{C}^{\mathsf{Env}}$ is rather rudimentary as a programming language. It has neither function abstraction nor recursion (but see Section 3.3). Although it has type variables, it lacks ML-style polymorphism and $\forall$ type quantification. To introduce this polymorphism requires a modified definition of the well-formedness of proof environments as described in [Jones 1992]. Other extensions include adding classifications of type constructors, as in [Jones 1996]. This involves a lambda calculus of types: in the categorical treatment, a cartesian closed base category.

Other interesting avenues to explore include linear versions of classes, and, more speculatively, languages of records and variants, and features of object-oriented languages, viewed from this proof-theoretic perspective.

Acknowledgements

The research was undertaken with support from an EPSRC research grant and funding from the E.U. projects 'Categorical Logic in Computer Science II' and 'Types for Proofs and Programs'. Discussions with Harold Simmons clarified some of the ideas in the paper. Some diagrams rely on Paul Taylor's diagram-drawing package. Useful comments from the refereeing have improved the paper.

References

[Bailey 1996] A. Bailey. LEGO with implicit coercions. Draft Report, Dept. of Computer Science, University of Manchester, Manchester, U.K. (1996).

[Barendregt 1989] H. Barendregt, Introduction to Generalised Type Systems. *Proc. Third Italian Conference on Theoretical Computer Science*, Ed. U. Moscati. Montova, November, 1989. World Scientific Publishing, Singapore.

[Curien 1990] P.-L. Curien, Substitution up to Isomorphism. Report LIENS-90-9, Laboratoire d'Informatique de l'Ecole Normale Supérieure, 45 Rue d'Ulm, 75230 Paris, 1990.

[Curien, Ghelli 1990] P.-L. Curien and G. Ghelli, Coherence of Subsumption. Research Report LIENS-90-10, Laboratoire d'Informatique de l'Ecole Normale Supérieure, 45 Rue d'Ulm, 75230 Paris, 1990.

[Ehrhard 1989] Th. Ehrhard, A categorical semantics of constructions. In Proc. *Logic in Computer Science* I.E.E.E. publ. Computer Society Press, Washington, 1989.

[Fitch 1952] F. Fitch *Symbolic Logic*, New York, 1952.

[Girard *et al.* 1989] J.-Y. Girard, Y. Lafont and P. Taylor, *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1989.

[Hallnäs 1988] L. Hallnäs, On normalisation of proofs in set theory. *Dissertationes Mathematicae*, Polska Akademia Nauk, Instytut Matematyczny, Warszawa, 1988.

[Hilken, Rydeheard 1992] B.P. Hilken and D.E. Rydeheard, Towards a categorical semantics of type classes. *Fundamenta Informaticae*, 16, 2. 1992.

[Hudak, Wadler 1990] P. Hudak and P. Wadler (editors) *Report on the Programming Language, Haskell, A Non-strict Purely Functional Language (Version 1.0).* Yale University, Department of Computer Science, April 1990. Technical Report No. YALEU/DCS/RR777.

[Hyland, Pitts 1987] J.M.E. Hyland and A.M. Pitts, Theory of constructions: categorical semantics and topos-theoretic models. In J.W. Gray and A. Scedrov (editors), *Proc. A.M.S. Conference on Categories in Computer Science and Logic, Boulder, Colorado (1987).* American Mathematical Society, 1989.

[Huet 1976] G. Huet, Résolutions d'équations dans des langages d'ordre $1, 2, \ldots, \omega$. *Thèse d'Etat, Université de Paris VII*, 1976.

[Jacobs 1990] B. Jacobs, Comprehension Categories and The Semantics of Type Dependency. Preprint, Dept. Comp. Sci., Toernooiveld, 6525 ED Nijmegen, Holland, 1990.

[Jacobs 1991] B. Jacobs, *Categorical Type Theory.* PhD Thesis, Catholic University of Nijmegen, The Netherlands. 1991.

[Jacobs et al. 1991] B. Jacobs, E. Moggi and T. Streicher (1991) Relating Models of Impredicative Type Theories, *Proc. Conference on Category Theory and Computer Science*, Paris, September 1991. Springer LNCS 530.

[Jay 1991] C.B. Jay, Long $\beta\eta$ Normal Forms in Confluent Categories. Preprint: LFCS, Department of Computer Science, University of Edinburgh, The King's Buildings, Mayfield Road, Edinburgh, 1991.

[Jones 1992] M.P. Jones, *Qualified Types: Theory and Practice.* PhD thesis, Oxford University Computing Laboratory, Programming Research Group. July 1992. Technical Monograph, PRG–106.

[Jones 1996] M.P. Jones, A system with constructor classes: Overloading and implicit higher order polymorphism. *J. Functional Programming.* To appear.

[Lambek, Scott 1986] J. Lambek and P. Scott, *Introduction to higher order categorical logic.* Cambridge University Press, 1986.

[Lawvere 1970] F.W. Lawvere, Equality in Hyperdoctrines and the Comprehension Schema as an Adjoint Functor. In *Proc. Symp. in Pure Math., XVII: Applications of Categorical Algebra*, Am. Math. Soc. pp. 1–14, 1970.

[Mac Lane 1982] S. Mac Lane, Why commutative diagrams coincide with equivalent proofs (presented in honour of Nathan Jacobson). In *Contemporary Mathematics 13*, American Mathematical Society. pp 387–401, 1982.

[Mac Lane, Paré 1985] S. Mac Lane and R. Paré, Coherence for Bicategories and Indexed Categories. *J. Pure and Applied Algebra*, 37. pp 59–80, 1985.

[Moggi 1991] E. Moggi, A category-theoretic account of program modules. *Math. Structures in Comp. Sci.* 1.1. pp 103–139, 1991.

[Nipkow, Snelting 1990] T. Nipkow and G. Snelting, Type Classes and Overloading Resolution via Order-Sorted Unification. Technical Report No. 200, Computer Laboratory, University of Cambridge, Cambridge, U.K, 1990.

[Peyton Jones, Wadler 1990] S.L. Peyton Jones and P. Wadler, A static semantics for Haskell. Preprint, Dept Computer Science, University of Glasgow, U.K, 1990.

[Pitts 1987] A.M. Pitts, Polymorphism is Set Theoretic, Constructively. *Proc. Summer Conference on Category Theory and Computer Science*, Edinburgh 1987, Springer LNCS 283, 1987.

[Prawitz 1965] Dag Prawitz, *Natural Deduction, A Proof-Theoretical Study*, Stockholm Studies in Philosophy 3. Almqvist and Wiksell, 1965.

[Rydeheard, Burstall 1986] D.E. Rydeheard and R.M. Burstall, *Computational Category Theory.* Prentice-Hall International Series in Computer Science (ed. C.A.R. Hoare), 1986.

[Seely 1984] R.A.G. Seely, Locally Cartesian Closed Categories and Type Theory. Math. *Proc. Camb. Phil. Soc.*, 95, pp. 33–48, 1984.

[Seely 1987] R.A.G. Seely, Categorical semantics for higher order polymorphic lambda calculus. *J. Symbolic Logic*, 52, pp 969–989, 1987.

[Wadler, Blott 1989] P. Wadler and S. Blott, How to make *ad hoc* polymorphism less *ad hoc*. In *Proceedings of 16th ACM Symposium on Principles of Programming Languages*, A.C.M., 1989.