© Copyright by Brian Patrick Dunphy, 2002
PARAMETRICITY AS A NOTION OF UNIFORMITY IN REFLEXIVE GRAPHS

BY

BRIAN PATRICK DUNPHY

B.S., North Carolina State University, 1993
M.S., North Carolina State University, 1994

THESIS

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mathematics in the Graduate College of the University of Illinois at Urbana-Champaign, 2002

Urbana, Illinois
Acknowledgments

I would like to thank Peter O’Hearn both for suggesting the direction of this thesis and for his comments and support during the research. The many discussions I had with him were most helpful.

I would also like to thank John Power, whose comments on a paper of my advisor provided direction to this research.

I also benefited from conversations with Carsten Fuhrmann, Achim Jung, Maria Emilia Maietti, and Eike Ritter during my visit to the University of Birmingham.

I am grateful to Dan Grayson and Ward Henson for their comments on this thesis. Their assistance has improved the presentation of this material.

Most of the diagrams in this thesis were produced using Paul Taylor’s diagram macros for \LaTeX.

Finally, I thank my advisor, Uday Reddy. I could not have completed this dissertation without his feedback and guidance during my research.
# Table of Contents

Chapter 1 Introduction ........................................... 1  
1.1 Contributions of this Thesis ................................. 6

Chapter 2 Background ................................................. 10  
2.1 Polymorphic Lambda Calculi ................................. 10  
2.2 Relational Parametricity ..................................... 20  
2.3 Reflexive Graph Categories ................................. 24

Chapter 3 Reflexive Graph Models of Polymorphism ...... 34  
3.1 Parametric Limits and Colimits ............................. 34  
3.2 Non-Variant Functors ......................................... 47  
3.3 Models of Polymorphism ..................................... 49  
3.4 Modeling Impredicativity .................................... 54  
3.5 Previous Work on Categorical Models ..................... 61

Chapter 4 Parametricity Graphs ................................. 65  
4.1 Relational Reflexive Graph Categories ..................... 66  
4.2 Identity Condition ........................................... 71  
4.3 Subsumption .................................................. 72  
4.4 Fibrations .................................................... 76  
4.5 Parametricity Graphs ........................................ 83

Chapter 5 Representation Results ............................... 88  
5.1 Formal System of Parametricity ............................... 89  
5.2 Relation to Other Systems ................................... 102  
5.3 Consequences of Parametricity .............................. 111  
5.4 Soundness .................................................... 129  
5.5 Semantic Representation Results ......................... 146  
5.6 A Negative Result ........................................... 181

Chapter 6 Imperative Parametricity ............................ 186  
6.1 Polymorphic Idealized Algol ................................. 186  
6.2 Models of Idealized Algol ................................ 190  
6.3 Representation Results in PIA .............................. 201
Chapter 1

Introduction

Uniformity is an important concept in good programming practice. In this introduction, we discuss the uniformity of polymorphic functions and the uniformity underlying data abstraction. We also review some previous work on formalizing uniformity.

While discussing polymorphic programming languages, Strachey [Str67] pointed out that there is a significant difference between polymorphic functions that act the “same” way for arguments of all types and polymorphic functions that act differently for various types. He referred to the former kind of function as parametrically polymorphic, in contrast to the ad hoc polymorphic of the latter. Computer scientists have developed many theories of programming using parametrically polymorphic functions. Since a parametrically polymorphic function is to act the same way at all types, it should not be possible to examine a type variable in order to select different code based on the value of that type variable. The parametricity of polymorphic functions plays a significant role in showing certain program equivalences and similar results [Mil71, Hoa72, Rey83, Wad89, OT95, Pit96, Sta96, Red98].

The functions mentioned above are pieces of code as opposed to functions in the usual set-theoretic sense. Both meanings for the term function are well-established among computer scientists, and we shall freely use both. This ambiguity arises from an understanding of programming where functions in the sense of code intuitively encode functions in the set-theoretic sense. Formal efforts to describe the behavior of programs consist of semantic models that interpret programs as more easily understood entities. Many semantic models formalize the intuition mentioned above by interpreting functions (in the sense of code) as functions (in the set-theoretic sense).

In a polymorphic programming language, a parametrically polymorphic
function encodes a family of functions (one for each type) that all act the same way. There are many examples of such families that are familiar to mathematicians and computer scientists. For instance, consider the family of \textit{list concatenation} functions.

\[ \text{concat}_A : \text{lists}[A] \times \text{lists}[A] \rightarrow \text{lists}[A] \]

This family of functions, indexed by the set \( A \) from which the entries of the lists are drawn, acts the same way for all sets \( A \). The family of \textit{set union} functions is similarly uniform in \( A \).

\[ \text{union}_A : \mathcal{P}(\mathcal{P}(A)) \rightarrow \mathcal{P}(A) \]

(Here \( \mathcal{P}(A) \) denotes the set of all subsets of \( A \).) For another example most mathematicians are familiar with, we suggest the \textit{determinant} of an \( n \times n \) matrix for any natural number \( n \). Considering a commutative ring \( K \), the determinant operation maps the non-singular \( n \times n \) matrices over \( K \) to the group of units of \( K \). This gives rise to a family of functions which act uniformly in \( K \).

\[ \text{det}_K : \text{GL}_n(K) \rightarrow K^* \]

Uniformity is a relevant concept even when higher types are involved. For example, the family of functions that send an endomorphism \( f \) to the composite \( f \circ f \) is uniform.

\[ \text{twice}_A : [A \rightarrow A] \rightarrow [A \rightarrow A] \]

Here, the inner arrow expression \( A \rightarrow A \) refers to the set of functions from \( A \) to \( A \). These functions \( \text{twice}_A \) act the same way for all \( A \).

In the polymorphic programming languages that we consider, families of functions such as \( \text{twice} \) are encoded by polymorphic functions whose types are written with the universal quantifier \( \forall \). All the languages that we consider here are parametrically polymorphic languages which means that all the polymorphic functions are parametric. We use \( T_1 \Rightarrow T_2 \) to for the type of functions from type \( T_1 \) to \( T_2 \), and as usual, a colon separates a value from its type.

\[ \text{Twice} : \forall X : [X \Rightarrow X] \Rightarrow [X \Rightarrow X] \]

Such a polymorphic function can be \textit{instantiated} at a type, which intuitively corresponds to selecting a particular member of the family. In the languages considered here, the instantiation of \( \text{Twice} \) at the type \( T \) is denoted using
square brackets.

\[ \text{Twice}[T]; [T] \Rightarrow [T] \Rightarrow [T] \]

*Abstract data types* embody uniformity in the form of *information hiding*. Information hiding enforces the uniform treatment of those entities that differ only on hidden information.

As a familiar example of an abstract data type, consider the complex numbers. There are two representations of complex numbers, one in terms of real and imaginary parts (the Cartesian representation) and the other in terms of length and argument (the polar representation). For each representation, basic operations, such as addition and conjugation, are defined. The operations on complex numbers are defined in such a way that the correspondence between the two representations is respected. Hence, the entire theory of complex numbers can be built in a way that is insensitive to their representation. In terms of computer programs, we expect any client for a complex numbers package will act the same way no matter which representation is used in the package.

Another common abstract data type is that of *queues*. Consider a package for queues of integers with the following operations.

\[
\begin{array}{ll}
\text{newQueue} & : \text{Queue} \\
\text{append} & : (\text{Queue} \times \text{Int}) \Rightarrow \text{Queue} \\
\text{front} & : \text{Queue} \Rightarrow \text{Int} \\
\text{remove} & : \text{Queue} \Rightarrow \text{Queue} \\
\text{empty?} & : \text{Queue} \Rightarrow \text{Bool}
\end{array}
\]

A queue provides ordered storage. In a queue of integers, only one integer (the one at the front of the queue) is accessible at a time. Once this integer is no longer needed, it can be removed so the next integer in line comes to the front. Integers are appended to the back of the queue, so they are accessible in the order they arrived; queues are a First-In, First-Out data structure. New queues are empty, so the predicate \text{empty?} applied to \text{newQueue} will necessarily return \text{true}.

A straightforward way of representing a queue (in a language with pointers, such as C) is as a *linked list*, with the head of the list pointing to the front of the queue. Using this representation, the \text{front} and \text{remove} operations are performed in constant time, but \text{append} requires time proportional to the length of the queue. If at some later time, it is determined that appending an element to the queue needs to be done as efficiently as possible (and removing an element from the queue can afford to be inefficient), then
one might change the representation of the queue to consider the head of the linked list to be the back of the queue. (It is also possible to choose a representation using a little bit more storage space in which all basic operations take constant time.)

Ideally, any client program would act the same way no matter which representation is used in the package for queues of integers, and therefore a programmer could make the change mentioned above to the queue package without affecting the behavior of client programs.

We consider some polymorphic programming languages that include abstract data types whose types are written with the existential quantifier $\exists$. For instance, the abstract data type for queues of integers has the following type.

$$\exists X. X \times ((X \times \text{Int}) \rightarrow X) \times (X \rightarrow \text{Int}) \times (X \rightarrow X) \times (X \rightarrow \text{Bool})$$

This notation is suggestive, as it indicates that there is some type $X$ implementing queues (called a concrete implementation) along with implementations for the basic operations.

Various good semantic models for parametrically polymorphic languages have been produced using families of functions which seem uniform to interpret parametrically polymorphic functions [JT93, OR94, OT95]. An abstract characterization of uniform families applicable to a wide range of settings beyond that of sets and functions would be useful in producing semantic models for more polymorphic languages.

A characterization of uniform families can be used to characterize the information hiding aspects of abstract data types as well. A client of an abstract data type is equivalent to a polymorphic function, where the type variable holds the place of the concrete representation. One can use the uniformity of polymorphism to characterize abstraction by defining two concrete representations to be representations of the same abstract type if they are indistinguishable by any uniform family.

Historically, the first effort at formalizing the uniformity of polymorphism was realized in the development of category theory. As remarked by MacLane [Mac71], categories were introduced to define functors, and the latter to define natural transformations. Natural transformations capture uniformity by insisting that all the arrows in the source category are respected by the transformations.

It was noticed during the development of automata theory that the preservation of arrows in the category of finite automata is not enough to
characterize which automata are equivalent. Instead, one needed to use structure-preserving relations, variously called “coverings” or “simulations” (see [Gin68], for instance). Milner [Mil71] applied a similar technique, using simulations between programs to show the equivalence of programs. Reynolds [Rey83] derived a notion of uniformity, called relational parametricity, in terms of respecting simulation relations. (We review relational parametricity in section 2.2.)

Reynolds’ treatment is at a concrete level, tied to sets, functions, and relations, as opposed to the abstract level to which the earlier development of category theory points us. As a result, it is not immediate how to generalize Reynolds’ ideas to other settings, where we might have kinds of structures other than sets (such as algebraic structures, partial orders, etc.), and kinds of correspondences other than plain relations. We give some examples of different settings, such as using special families of relations and span-like correspondences, in section 2.3.

O’Hearn and Tennent, in a seminal paper [OT95], used reflexive graph categories to generalize Reynolds’ ideas for their application (which was to the semantics of Algol-like languages). The idea is to use two separate categories, a “vertex category” and an “edge category”, together with a graph structure between them. The vertex category plays the role similar to the category SET, and the edge category plays a role similar to the category of relational correspondences between sets. (This motivating example of a reflexive graph category is defined in section 2.3.) Further, there is an “identity edge” 1a: a ↠ a for each vertex a; this edge plays a role similar to that of the identity arrow in ordinary categories. (This motivates the terminology of reflexive graphs.) O’Hearn and Tennent used these structures to define what it means for a family of arrows to be parametric. Parametric transformations capture the notion of uniformity by insisting that all “edges” (objects of the edge category) between “vertices” (objects of the vertex category) be respected by the transformation. Reflexive graph categories quickly became popular. They were used by Robinson and Rosolini [RR94] for exhibiting models of polymorphic lambda calculus, and by Pitts and Stark [PS93] to model dynamic storage in programming languages.

In this thesis, we further develop the theory of reflexive graphs of categories.
1.1 Contributions of this Thesis

The objective of this thesis is to formulate an axiomatization of parametricity at the categorical level. As a result of having such an axiomatization, parametricity becomes applicable to a wide range of settings including the structures used in programming language semantics as well as structures studied in mathematics.

The basis of our axiomatization is the structure of reflexive graph categories identified by O'Hearn and Tennent [OT95]. We review these in section 2.3. In the same section, we observe that reflexive graph categories have a pleasing structure in that the exponent of two of them can be viewed as a "reflexive graph category of functors" (an internal Hom for the category of reflexive graph categories). The situation is thus similar to that of ordinary categories where functors from one category to another in turn form a category, which is the exponent. (Therefore, we use the standard notation $H^G$ for the reflexive graph category of "functors" from $G$ to $H"). This key observation will prove useful. This property does not hold for graph categories (which were used in [MS93], for instance). The exponent of graph categories is not made up of "functors" that form the appropriate notion of arrows between graph categories.

In section 3.1, we examine the notions of limit and colimit appropriate for reflexive graph categories. For an appropriate functor $F$, we denote the limit as $\forall X F(X)$ since it captures at a categorical level the notion of parametrically polymorphic families [Rey83]. The notion is suggestive because parametrically polymorphic families provide appropriate operations for every type in a uniform manner. Similarly, the colimit is suggestively denoted as $\exists X F(X)$ since it captures the notion of abstract data types [MP88, Red98]. Abstract data types ensure that there exists some type with appropriate operations. Using the functor category $H^G$, these notions can be generalized to limit and colimit functors, defined as adjoints to the diagonal functor in the standard categorical fashion. Thus, it is demonstrated that by using the right setting, namely reflexive graph categories, analogies of quantified types in parametrically polymorphic languages are given by standard categorical constructions. We use the structure of reflexive graph categories in section 3.3 to define categorical models of polymorphic lambda calculi which incorporate parametricity criteria.

While the definitions given in Chapter 3 provide a wide range of models for polymorphic lambda calculi, they can not claim to enforce a good level of uniformity. The problem is that the notion of "relation" in the categorical
setting is an abstract concept and one is free to choose as few or as many relations as one pleases in a particular model. In order for the preservation of edges to ensure a reasonable notion of uniformity, we need to include certain relations in the collection of relations used in the model. Some typical tests used to support claims that the interpretation of polymorphic types contain only uniform families are referred to as representation results. For example, the interpretation of \( \forall X.X \) is expected to be the initial object and the interpretation of \( \forall X.X \Rightarrow X \) is expected to be the terminal object. These results are known to hold in the term models (categories where the objects are types and arrows are equivalence classes of terms) for some parametrically polymorphic languages. Showing these results hold in a proposed model is a form of abstraction result, establishing agreement between what exists in the model and what is possible in the language. In Chapter 4, we present structures called parametricity graphs that incorporate additional axioms concerning relations so that representation results can be proved.

Two fairly general representation results encode initial algebras and final coalgebras. In section 5.3, we prove these two representation results by using a lambda calculus for reasoning about relations in parametricity graphs, called System P (that is defined in section 5.1). This calculus is fashioned after an earlier calculus, called System R, that was proposed by Abadi, Cardelli and Curien [ACC93]. We make some simplifications to their system and add rules corresponding to the axioms of parametricity graphs to obtain System P. We show in section 5.4 that System P can be modeled in any well-pointed parametricity graph (with enough structure). The representation results proved in System P hold in all such models.

The restriction to well-pointed parametricity graphs turns out to be significant in the above results. We give an example of a presheaf-like parametricity graph (which is not well-pointed) where the initial algebra results do not hold (section 5.6). In particular, the interpretation of \( \forall X.X \Rightarrow X \) is not terminal.

However, this does not seem to mean that uniformity has broken down in the non-well-pointed case. As far as we have examined them, the parametric transformations present in these models are still intuitively uniform. Rather, the problem is that the representation results, as traditionally stated, rely on the well-pointedness of the models. We do not know how to state representation results for the general case. We do not think that a satisfactory solution exists.

To give an illustration of what happens in presheaf-like models, we consider models of a particular polymorphic imperative programming language
in Chapter 6. Imperative programming languages are traditionally modeled in presheaf categories [Rey81, Ole82] and parametric presheaf-like reflexive graph categories [OT95]. The later are used to capture the uniformity and information hiding aspects of local variables (even without polymorphic functions). Many imperative programming languages allow variables to be local to some region, such as a specific procedure. In these languages, it is possible for there to be more variables in scope at a procedure call than there are in scope at that procedure's declaration. We expect a procedure to be uniform in the sense that it should not depend on variables in scope at procedure call but not at procedure definition. O'Hearn and Tennent [OT95] used parametric transformations in reflexive graph categories to enforce that uniformity. By adding parametric polymorphism to such an imperative language (section 6.1), we obtain a second layer of parametricity that is best modeled using presheaf-like parametricity graphs. For the polymorphic imperative language considered here, the traditional representation results are intuitively valid. For instance, there is only one closed term of type \( \forall X.X \Rightarrow X \). In section 6.3, we exhibit a presheaf-like model for this language where the representation results hold: the interpretation of \( \forall X.X \Rightarrow X \) is terminal. The axioms of parametricity graphs play a crucial role in the construction of this model. This is the first known study of the semantics of polymorphic imperative programming languages.

Finally, in Chapter 7, we consider the problems posed by recursion, an important component of programming languages. There is tension between recursion and parametric polymorphism. The parametric polymorphism implies the existence of coproducts, but coproducts cannot coexist with recursion (unless the model is trivial) [HP90]. We follow a suggestion of Plotkin [Plo93] that the situation is best studied using a polymorphic linear lambda calculus. In section 7.2, we give domain models of predicative polymorphic calculi with recursion, which are instances of parametricity graphs. Two natural candidates for parametric models are in terms of complete relations [Rey83, RO00] and spans over CPOs (see examples in section 2.3). We present both models. It is apparent that the two models take very different approaches to modeling the linear connectives \( \odot \) and \( \otimes \) (the linear function space and the corresponding tensor product). A question arises to whether this difference in modeling results in a practical difference — are those transformations parametric with respect to complete relations the same as those transformations parametric with respect to spans? We have not been able to settle this question. In section 7.5, we give limited results indicating that for a number of types, the parametric transformations
between the interpretations of those types are the same in both models.

In summary, the contributions and results of this thesis include the following.

- axiomatic definition of parametricity graphs
- definition of parametric models of polymorphic lambda calculi in parametricity graphs
- initial algebra and final coalgebra representation results in well-pointed parametricity graphs
- parametric model of a polymorphic imperative programming language
- parametric domain models for polymorphic linear lambda calculus using complete relations as well as spans
Chapter 2

Background

When Reynolds formalized relational parametricity [Rey83], his goal was to produce a model for a polymorphic lambda calculus. Therefore, we begin our discussion of background material by reviewing some polymorphic lambda calculi. We present System F as well as a predicative polymorphic lambda calculus. Discussion about a set-based model will lead us to Reynolds’ notion of relational parametricity. We present Reynolds’ model, which uses a pair of complementary interpretations for types, one as sets and another as relations. O’Hea and Tennent [OT95] pointed out that a reflexive graph describes the complementary structure of sets and relations used by Reynolds. Therefore reflexive graph categories can be used as an abstract setting to formalize parametricity. We define reflexive graph categories and give some basic facts about them. We end this chapter by giving several examples of reflexive graph categories.

2.1 Polymorphic Lambda Calculi

Alonzo Church’s lambda calculus [Chu27] provides a foundation (or precursor) to many modern day computer programming languages. Various modified versions of the lambda calculus are useful for describing or investigating certain aspects of a programming language. One such modification is to identify terms with similar structure, giving each term a type. Function application can then be restricted to terms of a certain type. Perhaps the first such system was the simply typed lambda calculus proposed by Church [Chu40]. In programming languages where arguments can be passed to procedures, it is often the case that a procedure requires its argument to have a certain structure. Typed lambda calculi provide a mechanism for expressing and enforcing such restrictions.

In a typed programming language (or lambda calculus), there may be
times when one requires functions of different types that do essentially the
same thing, such as returning the first from a pair of inputs. Rather than
write a different function for every type, it would be convenient to write just
one function that could be used at different types. One way to achieve this
is by allowing a new kind of argument, which is used in determining the
type of other arguments. A polymorphic lambda calculus (called System F)
was introduced as a formal language that allows such functions with types
as arguments [Gir72, Rey74, Gir86].

The type system of System F is built up from an infinite collection of
type variables. In this work, we shall use the meta-variables X, Y and Z to
range over type variables. The collection of type expressions is indicated by
the following grammar.

\[ \tau ::= X \mid (\tau_1 \Rightarrow \tau_2) \mid (\forall X. \tau) \]

This standard notation describes the type expressions as follows. The meta-
variable \( \tau \) is used to range over type expressions, which are finite expres-
sions. The collection of type expressions is the least collection that includes
the type variables (indicated here by \( X \)) and is closed under the \( \Rightarrow \)
and \( \forall \) constructions. If \( \tau_1 \) and \( \tau_2 \) are type expressions, then \( (\tau_1 \Rightarrow \tau_2) \) is a type expression. If \( \tau \) is a type expression and \( X \) is a type variable, then \( (\forall X. \tau) \)
is a type expression.

To simplify notation, we will frequently suppress the parentheses when
they can be reconstructed from the following conventions. The \( \Rightarrow \) con-
struction binds more tightly than \( \forall \) does, and arrows associate to the right. For
instance, the expression \( \forall X. X \Rightarrow X \Rightarrow X \) is an abbreviated form of the more
cumbersome \( (\forall X. (X \Rightarrow (X \Rightarrow X))) \) expression.

Types are equivalence classes of type expressions where one equates type
expressions which differ only by the choice of type variables used in quanti-
fiers. We define \( \equiv \) to be the least equivalence relation on type expressions
such that

- if \( \tau_1 \equiv \tau_1 \) and \( \tau_2 \equiv \tau_2 \), then \( \tau_1 \Rightarrow \tau_2 \equiv \tau_1 \Rightarrow \tau_2 \),

- if \( \tau \equiv \tau' \) then \( \forall X. \tau \equiv \forall X. \tau' \), and

- if \( \tau \equiv \tau' \) and \( Y \) is a type variable such that \( Y \) does not occur free in
\( \tau' \) and \( X \) does not occur within the scope of a binding of \( Y \) in \( \tau' \),
then \( \forall X. \tau \equiv \forall Y. \tau' [Y/X] \)

where \( \tau' [Y/X] \) denotes the expression that results from using \( Y \) to replace
all free occurrences of \( X \) in \( \tau' \). We define the types of System F to be
equivalence classes of type expressions under the equivalence relation ≡.
We adopt the usual convention of using a type expression to denote a type
(the equivalence class containing that type expression).

The above substitution is a special case of $E[Y/X]$ where an expression
$F$ is used to replace the free occurrences of the variable $X$ in the expression
$E$. This substitution is defined inductively with the usual convention of
renaming bound variables of $E$ to prevent capturing free variables of $F$. In
the case of type expressions (where the only binding construct is $\forall$), if $Y$ is a
free variable of $F$, then $(\forall Y.\tau)[F/X]$ is formed by selecting a fresh variable $Z$
and uses $F$ to replace all free occurrences of $X$ in $(\forall Z.\tau[Z/Y])$. (Substitution
is defined analogously for other kinds of expressions besides type expressions,
such as term expressions.) A variable is fresh relative to a set of expressions
if that variable does not appear in any of the expressions. Stating a variable
is fresh as one of several hypotheses means that the variable is fresh relative
to the expressions in all other hypotheses.

The terms of System $F$ are built from an infinite collection of term vari-
ables. We use the meta-variables $x$, $y$ and $z$ in this work to range over term
variables. A type will be assigned to each term, dependent on an assignment
of types to free term variables. A type assumption is an expression of the
form $x:\tau$ where $x$ is a term variable and $\tau$ is a type. A context is a finite
sequence of type assumptions where no term variable is repeated. We typi-
cally use $\Gamma$ as a meta-variable to range over contexts. We use a comma both
as a separator between elements in a sequence (such as $x_1:\tau_1, x_2:\tau_2, x_3:\tau_3$)
and as a combinator to append one sequence to the end of another (such as
$\Gamma, \Gamma'$). Since an explicit listing of a sequence is the same as the concatena-
tion of several single-element sequences, no ambiguity will result. We use $\emptyset$
to denote an empty sequence.

It will also be necessary to keep track of type variables. For this, we
define a typing context to be a finite sequence of distinct type variables. The
meta-variable $\eta$ is used to range over typing contexts. We introduce some
notation to express some frequently used assertions about type variables.

A type judgment is an expression of the form $\eta \vdash \alpha$ where $\eta$ is a typing
context and $\alpha$ is either a type $\tau$ or a context $\Gamma$. A type judgment $\eta \vdash \alpha$
holds in System $F$ (or is a type judgment of System $F$) provided the free
type variables of $\alpha$ are contained in $\eta$.

The collection of terms will be defined via term judgments. A term judg-
ment is an expression of the form $\eta ; \Gamma \vdash M : \tau$ where $\eta$ is a typing context,
$\Gamma$ is a context, $\tau$ is a type, and $M$ is some expression. The terms of System
$F$ are those expressions $M$ that appear between the turnstile and the final
\[
\frac{\eta \vdash x_1 : \tau_1, \ldots, x_m : \tau_m \quad 1 \leq j \leq m}{\eta \vdash x_1 : \tau_1, \ldots, x_m : \tau_m \vdash x_j : \tau_j} \quad \text{\{variable\}}
\]
\[
\frac{\eta ; \Gamma, x : \tau_1 \vdash M : \tau_2}{\eta ; \Gamma \vdash \lambda x : \tau_1 . M : \tau_1 \Rightarrow \tau_2} \quad \text{\{fun\}_\text{intro}\}
\]
\[
\frac{\eta ; \Gamma \vdash M_1 : \tau_1 \Rightarrow \tau_2 \quad \eta ; \Gamma \vdash M_2 : \tau_1}{\eta ; \Gamma \vdash M_1 \ M_2 : \tau_2} \quad \text{\{fun\}_\text{elim}\}
\]
\[
\frac{\eta, X ; \Gamma \vdash M : \tau \quad \eta \vdash \Gamma}{\eta ; \Gamma \vdash \Lambda X . M : \forall X . \tau} \quad \text{\{poly\}_\text{intro}\}
\]
\[
\frac{\eta ; \Gamma \vdash M : \forall X . \tau_1 \quad \eta \vdash \tau_2}{\eta ; \Gamma \vdash M[\tau_2] : \tau_1[\tau_2/X]} \quad \text{\{poly\}_\text{elim}\}
\]

Table 2.1: Term Forming rules of System F

colon in a term judgment of System F. We use the meta-variables \( M \) and \( N \) to range over terms. The term judgments of System F are defined inductively using introduction and elimination rules for the type constructors \( \Rightarrow \) and \( \forall \) as well as the obvious rule for variables. The expression \( \eta ; \Gamma \vdash M : \tau \) is a term judgment of System F (or the term judgment is derivable in System F) if and only if there exists a finite derivation of it from the empty set of hypotheses using the rules in table 2.1.

In the \{poly\}_\text{intro} rule, we use a capital \( \Lambda \) to bind a type variable, as opposed to the lower case \( \lambda \) used for binding a term variable in \{fun\}_\text{intro}. The formal syntax \( M[\tau_2] \) in the \{poly\}_\text{elim} rule is typically referred to as the instantiation of the term \( M \) at the type \( \tau_2 \). Note the hypothesis \( \eta \vdash \Gamma \) in the \{poly\}_\text{intro} rule. Since \( \eta, X \) is a typing context, the type variable \( X \) does not appear in \( \eta \). Therefore the hypothesis \( \eta \vdash \Gamma \) implies that no type mentioned in \( \Gamma \) uses the type variable \( X \). The variable \( X \) will be bound in \( \forall X . \tau \) and \( \Lambda X . M \). Since there are no occurrences of \( X \) in \( \Gamma \), \( X \) will no longer be a free variable anywhere in the judgment \( \eta ; \Gamma \vdash (\Lambda X . M) : \forall X . \tau \).

Much as we do for types, we shall frequently suppress parentheses in terms. The convention is that parentheses would extend as far to the right as possible. For instance, the expression \( (\lambda x : \tau_1 . \Lambda Y . \lambda y : \tau_1 \Rightarrow Y . y \ x) \ N \) is a shortened form of \( (\lambda x : \tau_1 . (\Lambda Y . (\lambda y : \tau_1 \Rightarrow Y . y \ x)) \ N \).

An intuitive understanding of terms is that they describe computations. For any term judgment \( \eta ; \Gamma \vdash M : \tau \), the term \( M \) describes a way to compute a value of type \( \tau \) from inputs as specified in the context \( \Gamma \). There are
several terms that intuitively encode the same computation. An equivalence relation on term judgments is defined, using the notation \( \eta ; \Gamma \vdash M = N \) to express that \( \eta ; \Gamma \vdash M : \tau \) and \( \eta ; \Gamma \vdash N : \tau \) are related. The equivalence relation is defined inductively using the apparent rules for term forming operations and having the \( \alpha \), \( \beta \), and \( \eta \)-equivalence for functions and polymorphic types as base cases. The rules for forming the equivalence relation are given in table 2.2 (page 15). Observe that the \( \beta \) and \( \eta \)-equivalences for each type construction assert that the corresponding introduction and elimination constructs are inverses of each other. The \( \alpha \)-equivalences assert that the choice of variable name is irrelevant.

This forms a basic typed lambda calculus. One could extend the type system with additional constants or constructors, such as products \( \tau_1 \times \tau_2 \) or existential types \( \exists X. \tau \). These would typically come with additional term forming constructs and appropriate equational rules. Since the term forming constructs typically come as an introduction/elimination pair, one can usually anticipate the corresponding equational rules (for propagating equivalences through new constructs and corresponding \( \beta \) and \( \eta \)-equivalences). We therefore do not generally mention equational rules for most of the calculi presented in the rest of this work.

Mathematicians often produce models for formal systems by interpreting pieces of syntax as particular mathematical structures to allow knowledge or intuitions about the more concrete representations to be used as an aid in the understanding of the more abstract syntax. For instance, a straightforward model of the simply typed lambda calculus can be given which interprets types as sets. This model uses the set of functions from \( A_1 \) to \( A_2 \) (which we shall denote as \( A_1 \rightarrow A_2 \)) to interpret the function type \( \tau_1 \rightarrow \tau_2 \), where each \( A_i \) is the interpretation of the corresponding \( \tau_i \). This model can be extended to account for type variables, interpreting a type judgment in a typing context \( \eta \) with \( n \) type variables as a function \( [\eta \vdash \alpha] \colon \text{Set}^n \rightarrow \text{Set} \). The interpretations of type judgments are given by the following inductive definition for any \( \bar{A} = (A_1, \cdots, A_n) \) in \( \text{Set}^n \).

\[
\begin{align*}
[X_1, \cdots, X_n \vdash X_j] \bar{A} &= A_j \\
[\eta \vdash \tau_1 \rightarrow \tau_2] \bar{A} &= [\eta \vdash \tau_1] \bar{A} \Rightarrow [\eta \vdash \tau_2] \bar{A} \\
[\eta \vdash x_1 : \tau_1, \cdots, x_m : \tau_m] \bar{A} &= \prod_{i=1}^{m} [\eta \vdash \tau_i] \bar{A}
\end{align*}
\]

A term judgment \( \eta ; \Gamma \vdash M : \tau \) is interpreted as a collection of functions \( [\eta ; \Gamma \vdash M : \tau] \bar{A} ; [\eta \vdash \Gamma] \bar{A} \rightarrow [\eta \vdash \tau] \bar{A} \) indexed by \( n \)-tuples of sets, \( \bar{A} \). The inductive definition is given as follows, where we use \( \Gamma = x_1 : \tau_1, \cdots, x_m : \tau_m \).
\[ \eta \vdash x_1 : \tau_1, \ldots, x_m : \tau_m \quad 1 \leq j \leq m \quad \text{equiv.var} \]

\[ \eta ; \Gamma, x : \tau_1 \vdash M = N \]
\[ \eta ; \Gamma \vdash (\lambda x : \tau_1 . M) = (\lambda y : \tau_1 N) \quad \text{equiv.fun.intro} \]

\[ \eta ; \Gamma \vdash M_1 = N_1 \quad \eta ; \Gamma \vdash M_2 = N_2 \]
\[ \eta ; \Gamma \vdash M_1, M_2 = N_1, N_2 \quad \text{equiv.fun.elim} \]

\[ \eta, X ; \Gamma \vdash M = N \]
\[ \eta ; \Gamma \vdash (\Lambda X. M) = (\Lambda X. N) \quad \text{equiv.poly.intro} \]

\[ \eta ; \Gamma \vdash M = N \]
\[ \eta ; \Gamma \vdash M[x_2] = N[x_2] \quad \text{equiv.poly.elim} \]

\[ \eta ; \Gamma, x : \tau_1 \vdash M : \tau_2 \quad \text{y is fresh} \]
\[ \lambda x : \tau_1 . M = \lambda y : \tau_1 . M[\eta/x] \quad \text{\(\alpha\)-fun} \]

\[ \eta, X ; \Gamma \vdash M : \tau \quad \eta \vdash \Gamma \quad Y \text{ is fresh} \]
\[ \eta ; \Gamma \vdash \Lambda X. M = \Lambda Y. M[\eta/x] \quad \text{\(\alpha\)-poly} \]

\[ \eta ; \Gamma, x : \tau_1 \vdash M : \tau_2 \quad \eta ; \Gamma \vdash N : \tau_1 \]
\[ \eta ; \Gamma \vdash (\lambda x : \tau_1 . M) N = M[N/x] \quad \text{\(\beta\)-fun} \]

\[ \eta ; \Gamma \vdash M : \tau_1 \Rightarrow \tau_2 \quad \text{x is fresh} \]
\[ \eta ; \Gamma \vdash (\lambda x : \tau_1 . M \ x) = M \quad \text{\(\eta\)-fun} \]

\[ \eta, X ; \Gamma \vdash M : \tau_1 \quad \eta \vdash \Gamma \quad \eta \vdash \tau_2 \]
\[ \eta ; \Gamma \vdash (\Lambda X. M)[\tau_2] = M[\eta/x] \quad \text{\(\beta\)-poly} \]

\[ \eta ; \Gamma \vdash M : \forall X. \tau \quad Y \text{ is fresh} \]
\[ \eta ; \Gamma \vdash (\Lambda Y. M[Y]) = M \quad \text{\(\eta\)-poly} \]

\[ \eta ; \Gamma \vdash M = N \]
\[ \eta ; \Gamma \vdash N = M \quad \text{symm} \]

\[ \eta ; \Gamma \vdash M = N \quad \eta ; \Gamma \vdash N = N' \]
\[ \eta ; \Gamma \vdash M = N' \quad \text{trans} \]

Table 2.2: Equivalence rules for System F
and \( \bar{a} = (a_1, \ldots, a_m) \in [\eta \vdash \Gamma] \bar{A} \) in giving the rules.

\[
[\eta ; x_1 : \tau_1, \cdots, x_m : \tau_m \vdash x_j : \tau_j] \bar{A}\bar{a} = a_j
\]

\[
[\eta ; \Gamma \vdash (\lambda x : \tau . M) : \tau' \vdash \tau'] \bar{A}\bar{a} = \text{the function mapping}
\]

\[
b \in [\eta ; \tau'] \bar{A} \quad \text{to} \quad [\eta ; \Gamma, x : \tau \vdash M : \tau'] \bar{A}(a_1, \cdots, a_m, b)
\]

\[
[\eta ; \Gamma \vdash M_1 M_2 : \tau_2] \bar{A}\bar{a} = ([\eta ; \Gamma \vdash M_1 : \tau_1] \bar{A}\bar{a}) b
\]

where \( b = [\eta ; \Gamma \vdash M_2 : \tau_2] \bar{A}\bar{a} \).

When the context and type information can be inferred, we shall sometimes suppress its explicit inclusion in the notation, as in the following abbreviation.

\[
[M] \bar{A} : [\Gamma] \bar{A} \rightarrow [\tau] \bar{A}
\]

In order to extend this to a model of System F, one would need to provide a set \([\eta \vdash \forall X.\tau] \bar{A}\) for type judgements involving polymorphic types. There would need to be a function from \([\eta \vdash \forall X.\tau] \bar{A}\) to \([\eta, X \vdash \tau](A_1, \cdots, A_n, B)\) where \( B = [\eta \vdash \tau_2] \bar{A}\) for any \( \tau_2 \). This is not an easy task. While there are models of System F that have been produced in other settings (GLT89, BTC88, Pit87), attempts for a simple, set-based model run into foundational problems. For instance, an obvious attempt is to interpret a polymorphic type \( \emptyset \vdash \forall X.\tau \) as the indexed product \( \Pi_{B \in \text{Set}}[X \vdash \tau] B \). However this product indexed over all sets is, in general, too big to be a set.

These foundational problems do not arise when the indexing is done over a set \( S \) of sets, rather than over the proper class \( \text{Set} \). The indexed product \( \Pi_{B \in S}[X \vdash \tau] B \) is a set, although, in general, not an element of \( S \). Such an indexed product is appropriate for modeling a predicative version of the polymorphic lambda calculus, which models the type systems of programming languages like standard ML [MTH90] and Haskell [HJW92]. A predicative form of polymorphism is one where the quantification in a type \( \forall X.\tau \) does not range over all types, but rather over a sub-collection that does not contain the type \( \forall X.\tau \) itself. The predicative polymorphic lambda calculus we consider in this work has two kinds of types: simple types \( \tau \) and (polymorphic) types \( \phi \). Again, types are defined as equivalence classes of type expressions, where the equivalence is defined by renaming of bound variables. We shall often refer to this predicative polymorphic lambda calculus as simply the predicative calculus. The type expressions are given by

16
the following grammar.

\[ \tau \ := \ X \mid (\tau_1 \Rightarrow \tau_2) \]
\[ \phi \ := \ \tau \mid (\forall X.\phi) \mid (\exists X.\phi) \]

In this predicative form of lambda calculus, the quantifiers are taken to range over only simple types. The idea is that simple types take values from a set \( S \) of sets and that values for types come from the proper class \textbf{Set}. In addition to the universal quantifier \( \forall \) for binding type variables, there is also the existential quantifier \( \exists \), which is used for abstract data types.

An abstract data type is a data type which could be defined in terms of simpler types. The complex numbers and queues of integers are examples of abstract data types that were mentioned in the introduction. Rather than committing to a particular representation (in terms of more basic types), abstract data types focus on the basic operations one would want for that type. If \( \phi \) is the type of basic operations one desires (using a type variable \( X \) to represent the new type), then the abstract data type will have some concrete representation (that is, some type) \( X \) and a term of type \( \phi \) (which gives the basic operations on that representation). An abstract data type with basic operations of type \( \phi \) is an element of the existential type \( \exists X.\phi \).

A type \( \tau \) (the concrete representation) and a term \( M \) of type \( \phi[\tau/X] \) (the code for the basic operations) are packed together using the notation (pack \( \tau \) with \( M \)) to produce a term of type \( \exists X.\phi \). The type \( \tau \) is called the concrete representation for the abstract data type (pack \( \tau \) with \( M \)). Given an abstract data type (a term of type \( \exists X.\phi \)), it can be opened to provide a concrete representation and code for basic operations. If \( N \) is a term that uses a type variable \( X \) and a term variable \( x \) of type \( \phi \), then the notation (open \( M \) as \((X, x)\) in \( N \)) is used to open a term \( M \) of type \( \exists X.\phi \) for use in \( N \). Any such term \( N \) that uses an abstract data type is called a dient for the abstract data type.

The predicative calculus has three kinds of judgments.

\[ \eta \vdash \alpha \quad \text{type judgment} \]
\[ \eta \vdash \tau \text{ simple} \quad \text{simplicity judgment} \]
\[ \eta ; \Gamma \vdash M : \phi \quad \text{term judgment} \]

The type judgments of the predicative calculus are analogous to type judgments in System F. The judgment \( \eta \vdash \alpha \) holds if the free type variables of \( \alpha \) (either a type or a context) are in \( \eta \) and no variable is repeated in \( \eta \).
\[ \eta \vdash x_1: \phi_1, \ldots, x_m: \phi_m \quad 1 \leq j \leq m \]
\[ \eta \vdash x_i: \phi_1, \ldots, x_m: \phi_m \vdash x_j: \phi_j \quad \text{variable} \]
\[ \eta; \Gamma, x: \tau_1 \vdash M: \tau_2 \quad \eta \vdash \tau_1 \quad \text{simple} \quad \eta \vdash \tau_2 \quad \text{simple} \]
\[ \eta; \Gamma \vdash (\lambda x: \tau_1. M): \tau_1 \Rightarrow \tau_2 \quad \text{fun_intro} \]
\[ \eta; \Gamma \vdash M: \tau_1 \Rightarrow \tau_2 \quad \eta; \Gamma \vdash N: \tau_1 \]
\[ \eta; \Gamma \vdash M N: \tau_2 \quad \text{fun_elim} \]
\[ \eta, X; \Gamma \vdash M: \phi \quad \eta \vdash \Gamma \]
\[ \eta; \Gamma \vdash (\lambda X.M); \forall X. \phi \quad \text{poly_intro} \]
\[ \eta; \Gamma \vdash M: \forall X. \phi \quad \eta \vdash \tau \quad \text{simple} \]
\[ \eta; \Gamma \vdash M[\tau]: \phi[\tau/\chi] \quad \text{poly_elim} \]
\[ \eta \vdash \tau \quad \text{simple} \quad \eta; \Gamma \vdash M[\tau]: \phi[\tau/\chi] \]
\[ \eta; \Gamma \vdash (\text{pack } \tau \text{ with } M): \exists X. \phi \quad \text{abs_intro} \]
\[ \eta; \Gamma \vdash M: \exists X. \phi \quad \eta, X; \Gamma, x: \phi \vdash N: \phi' \quad \eta \vdash \phi' \]
\[ \eta; \Gamma \vdash (\text{open } M \text{ as } (X, x) \text{ in } N): \phi' \quad \text{abs_elim} \]

Table 2.3: Term forming rules for the predicative calculus

Simplicity judgments add an additional criteria.

\[ \eta \vdash \tau \quad \text{simple} \iff \eta \vdash \tau \quad \text{and} \quad \tau \text{ is a simple type} \]

The term judgments of the predicative calculus are those expressions that can be derived from a finite number of applications of the rules in table 2.3. The fact that quantifiers range over simple types is reflected in the rules \{poly_elim\} and \{abs_intro\}. Polymorphic types can only be instantiated at simple types, and only simple types can be used as concrete representations for existential types. The \( \eta \vdash \phi' \) hypothesis in the \{abs_elim\} rule plays a similar to the \( \eta \vdash \Gamma \) hypothesis in \{poly_intro\}. The type variable \( X \) does not appear in \( \phi' \), and the hypothesis \( \eta; \Gamma \vdash M: \exists X. \phi \) implies that \( X \) does not appear in \( \Gamma \). Hence \( X \) is not free in the term judgment \( \eta; \Gamma \vdash (\text{open } M \text{ as } (X, x) \text{ in } N): \phi' \).

A naive model of the predicative calculus can be defined as follows. Let \( S \) be a set of sets that is closed under the function space constructor \( \Rightarrow \). We interpret a type judgment \( \eta \vdash \alpha \) as a function \( \eta \vdash [\alpha]: S^n \rightarrow \text{Set} \) where \( n \) is the number of type variables in \( \eta \). The inductive definition for interpreting
type judgments is given in below.

$$[X_1, \ldots, X_n \vdash X_j](A_1, \ldots, A_n) = A_j$$

$$[\eta \vdash \tau_1 \Rightarrow \tau_2] \bar{A} = [\eta \vdash \tau_1] \bar{A} \Rightarrow [\eta \vdash \tau_2] \bar{A}$$

$$[\eta \vdash \forall X. \phi](A_1, \ldots, A_n) = \Pi_{B \in S}[\eta, X \vdash \phi](A_1, \ldots, A_n, B)$$

$$[\eta \vdash \exists X. \phi](A_1, \ldots, A_n) = \Sigma_{B \in S}[\eta, X \vdash \phi](A_1, \ldots, A_n, B)$$

$$[\eta \vdash x_1: \tau_1, \ldots, x_m: \tau_m] \bar{A} = \Pi_{j=1}^m [\eta \vdash \tau_j] \bar{A}$$

The interpretation of a polymorphic type is given using the indexed product. An element $p \in [\eta \vdash \forall X. \phi]\bar{A}$ is given by a collection of elements $p_B$ (one for each $B \in S$) where $p_B \in [\eta, X \vdash \phi](A_1, \ldots, A_n, B)$ for each $B$. These components $p_B$ give the interpretations for possible instantiations. (The details will appear in the interpretation of term judgments.) Existential types are interpreted as the indexed sum (also called indexed coproduct or disjoint union). Elements of $[\eta \vdash \exists X. \phi]\bar{A}$ are given as $\text{in}_B(b)$, providing a set $B$ (corresponding to the concrete representation) and an element $b$ (corresponding to the basic operations) for an abstract data type.

For any simple type $\tau$ and any $n$-tuple $\bar{A} \in S^n$, the set $[\eta \vdash \tau] \bar{A}$ is an element of $S$. We define the interpretation of a simplicity judgment as the function $[\eta \vdash \tau \text{ simple}]: S^n \to S$ that agrees with the interpretation of the corresponding type judgment.

$$[\eta \vdash \tau \text{ simple}] \bar{A} = [\eta \vdash \tau] \bar{A}$$

A term judgment is interpreted as a family of functions indexed by $n$-tuples $\bar{A} \in S^n$.

$$[\eta ; \Gamma \vdash M : \phi]_{\bar{A}} : [\eta \vdash \Gamma] \bar{A} \to [\eta \vdash \phi] \bar{A}$$

The inductive definition of this interpretation is given in table 2.4 (page 20).

Since the interpretations of simple types map into $S$, the indexing over $S$ in the interpretation of $\eta \vdash \forall X. \phi$ is sufficient. Having a component for each $B \in S$ implies that there is an appropriate component corresponding to any simple type that can be needed for the interpretation corresponding to $\text{poly}_\text{lim}$. There is a direct correspondence between limiting instantiations of polymorphic types to only simple types in the calculus and restricting the indexing for the product to the set $S$ in the model. Similarly, the limitation that concrete representations of abstract data types must be simple types
\[ \eta ; x_1: \tau_1, \ldots, x_m: \tau_m \vdash x_j: \tau \rightarrow a \alpha = a_j \]

\[ \eta ; \Gamma \vdash (\lambda x: \tau.M: \tau \rightarrow \tau')_\alpha \rightarrow a = \text{the function} \]

\[ b \rightarrow [\eta ; \Gamma, x: \tau \vdash M: \tau']_\alpha(a_1, \ldots, a_m, b) \]

\[ [\eta ; \Gamma \vdash M_1 M_2: \tau']_\alpha = ([\eta ; \Gamma \vdash M_1: \tau \rightarrow \tau']_\alpha b) \]

where \[ b = [\eta ; \Gamma \vdash M_2: \tau]_\alpha \]

\[ ([\eta ; \Gamma \vdash (\lambda X.M): \forall X.\phi]_\alpha)_{\beta} = [\eta, X; \Gamma \vdash M: \phi]_{(A_1, \ldots, A_n, \eta)} \]

\[ [\eta ; \Gamma \vdash M[\tau]: \phi[\tau/X]]_\alpha = (\eta, \Gamma \vdash M: \forall X.\phi]_\alpha)_{\beta} \]

where \[ \beta = [\eta ; \Gamma \vdash \tau \text{ simple}]_\alpha \]

\[ \eta ; \Gamma \vdash (\text{pack } \tau \text{ with } M): \exists X.\phi]_\alpha = \text{in}_\beta(b) \text{ where} \]

\[ B = [\eta, \tau \text{ simple}]_\alpha \text{ and } b = [\eta ; \Gamma \vdash M: \phi[\tau/X]]_\alpha \]

\[ [\eta ; \Gamma \vdash (\text{open } M \text{ as } (X, x) \text{ in } N): \phi']_\alpha = \]

\[ ([\eta, X; \Gamma, x: \phi \vdash N: \phi']_{(A_1, \ldots, A_n, \eta)}(a_1, \ldots, a_m, b)) \]

where \[ \text{in}_\beta(b) = [\eta ; \Gamma \vdash \exists X.\phi]_\alpha \]

Table 2.4: Interpretation of term judgments of predicative calculus

corresponds to the indexing for the sum being over \( S \).

### 2.2 Relational Parametricity

Terms of polymorphic lambda calculi seem to exhibit uniformity of the intuitive kind mentioned in the introduction. Any term of polymorphic type encodes essentially the same computation regardless of the type at which the term is instantiated. Hence, the interpretations of terms will be restricted to certain intuitively uniform families from the indexed product \( \Pi_{B \in \mathcal{S}}[\eta, x: \phi]_\alpha A, B \) in the above model. Reynolds [Rey83] proposed that this insight could be used to refine the interpretation of polymorphic types, trimming down the product to only the “uniform” elements. Reynolds formalized this idea by defining an interpretation of type judgments using relations to complement the interpretation using sets. We review Reynolds’ model from [Rey83].

For any type judgment \( \eta \vdash \alpha \) where \( |\eta| = n \), an interpretation is given as a function \( [\eta \vdash \alpha]: \mathcal{S}^n \rightarrow \mathcal{S} \). The corresponding relational interpretation maps an \( n \)-tuple of relations to a relation. We use the notation \( R: A \leftrightarrow A' \) to denote that \( R \) is a (binary) relation between the sets \( A \) and \( A' \). Under the
assumption that \( R \in A_i \leftrightarrow A_i' \) is a relation between sets of \( S \) for each \( i \) from 1 to \( n \), the correspondence between the set-interpretation and relation interpretation is that the \( n \)-tuple \( \vec{R} = (R_1, \ldots, R_n) \) of relations is mapped to a relation \( [\eta \vdash \alpha]\vec{R} : [\eta \vdash \alpha] \vec{A} \leftrightarrow [\eta \vdash \alpha] \vec{A}' \). Since the expressions involved are frequently more complicated than one normally encounters in stating two things are related (such as \( x R y \)), we shall often add brackets around the relation in this notation as an aid in parsing (for instance, \( f x [\eta \vdash \tau] \vec{R} f' x' \)).

Reynolds did not directly consider existential types \( \exists X. \phi \). The interpretation presented here is due to Reddy [Red98]. The interpretation of \( \exists X. \phi \) is given by quotienting the indexed sum by an equivalence relation (which intuitively relates those concrete representations of the same abstract data type). This equivalence relation and the interpretations of type judgments \( \eta \vdash \phi \) involving types are defined by simultaneous induction. The definitions of the interpretations (modulo the definition of the equivalence relation \( \approx \)) is given in table 2.5 (page 22). Here, \( \Delta_A \) is used to denote the diagonal relation on the set \( A \), defined by \( [\Delta_A] b \) if and only if \( a = b \). The equivalence relation \( \approx \) used to define the set interpretation of \( \exists X. \phi \) is defined as follows.

The indexed sum \( \Sigma_{S \in S} [\eta, X \vdash \phi] (A_1, \ldots, A_n, B) \) is given by the set \( \{ \text{in}_B(b) \mid B \in S, b \in [\eta, X \vdash \phi] (A_1, \ldots, A_n, B) \} \). Two elements \( \text{in}_B(b) \) and \( \text{in}_B'(b) \) of \( \Sigma_{S \in S} [\eta, X \vdash \phi] (A_1, \ldots, A_n, B) \) are called similar, denoted \( \text{in}_B(b) \sim \text{in}_B'(b) \), if and only if there is a relation \( R : B \leftrightarrow B' \) such that \( b [\eta, X \vdash \phi] (\Delta_{A_1}, \ldots, \Delta_{A_n}, R) b' \). The equivalence relation \( \approx \) is defined to be the transitive closure of \( \sim \). We use \([x] \) to denote the equivalence class of \( x \) so one could alternative state that the interpretation \( [\eta \vdash \exists X. \phi] \vec{A} \) is the set \( \{ [x] \mid x \in \Sigma_{S \in S} [\eta, X \vdash \phi] (A_1, \ldots, A_n, B) \} \) of equivalence classes of \( \approx \).

The definition of \( [\eta \vdash \tau_1 \Rightarrow \tau_2] \vec{R} \) uses a general construction of a relation for functions. (This construction will give the exponent in the category of relational correspondences to be identified in section 2.3.) For relations \( R : A \leftrightarrow A' \) and \( S : B \leftrightarrow B' \), we use \( R \Rightarrow S \) to denote the relation between \( A \Rightarrow B \) and \( A' \Rightarrow B' \) given as follows.

\[
f [R \Rightarrow S] f' \iff (x [\vec{R}] x' \implies f x [\vec{S}] f' x')
\]

A type judgment \( \eta \vdash \Gamma \) for a context is given a pair of interpretations in an obvious manner as the product of the types declared in \( \Gamma \).

\[
[\eta \vdash x_1 : \phi_1, \ldots, x_m : \phi_m] \vec{A} = [\eta \vdash \phi_1] \vec{A} \times \times [\eta \vdash \phi_m] \vec{A} \] 

\[
\vec{a} [\eta \vdash x_1 : \phi_1, \ldots, x_m : \phi_m] \vec{R} \vec{a}' \iff \text{for all } i, a_i [\eta \vdash \phi_i] \vec{R} a'_i
\]

21
\[ [X_1, \ldots, X_n] A = A_j \]
\[ [\eta \vdash \tau_1 \Rightarrow \tau_2] A = [\eta \vdash \tau_1] A \Rightarrow [\eta \vdash \tau_2] A \]
\[ [\eta \vdash \forall X. \phi] A = \{ p \in \Pi_{B \in S}[\eta, X \vdash \phi](A_1, \ldots, A_n, B) \text{ such that for every } R: B \leftrightarrow B', \]
\[ p_B \left[ [\eta, X \vdash \phi](\Delta_{A_1}, \ldots, \Delta_{A_n}, R) \right] p_{B'} \} \]
\[ [\eta \vdash \exists X. \exists Y. \exists Z. \exists \lambda. \phi] A = \left( \Sigma_{B \in S}[\eta, X \vdash \phi](A_1, \ldots, A_n, B) \right)_\approx \]

A term judgment is given a single interpretation as a family of functions
\[ [\eta ; \Gamma \vdash M: \phi] A : [\eta \vdash \Gamma] A \rightarrow [\eta \vdash \phi] A \text{ indexed by } n \text{-tuples } A \in S^n. \] The interpretation is defined by induction on derivations, as indicated in table 2.6 (page 23).

The motivation behind the interpretation of existential types can be described in terms of abstract data types. Two concrete representations are representations of the same abstract data type if there is a relation between the representations which is preserved by the basic operations. This leads to viewing an abstract data type as a collection of concrete representations which are related. The relation between the two representations provides a way to view elements from the different representations as being the same element of the abstract data type.

A similar observation had previously motivated Reynolds when he first proposed using the complementary relation interpretations. If a computation proceeds the same way at different arguments regardless of the type of those arguments, then that computation should proceed the same way at different arguments no matter how the arguments are related. The interpretation of a polymorphic type can be limited to only the parametric families. A parametric family of the indexed product \( \Pi_{B \in S}[\eta, X \vdash \phi](A_1, \ldots, A_n, B) \) is defined to be one which for every relation \( R: B \leftrightarrow B' \), the \( B \) and \( B' \) com-
ponents are related as follows.

\[ p_B \left[ \eta ; X \vdash \phi \right] (\Delta_{A_1}, \ldots, \Delta_{A_n}, R) \]

The family of functions interpreting a given term judgment \( \eta ; \Gamma \vdash M : \phi \) is itself a uniform family characterized by the preservation of relations. That is, for any \( n \)-tuple \( \vec{R} \vec{\bar{A}} \leftrightarrow \vec{A}' \) of relations, the functions \( [\eta ; \Gamma \vdash M : \phi]_{\vec{A}} \) and \( [\eta ; \Gamma \vdash M : \phi]_{\vec{R}} \) map \( [\eta ; \Gamma] \vec{R} \)-related inputs to \( [\eta \vdash \phi] \vec{A} \)-related outputs. This situation of mapping related inputs to related outputs is typically denoted by the following diagram.

\[
\begin{array}{ccc}
[\eta ; \Gamma] \vec{A} & \xrightarrow{[\eta ; \Gamma \vdash M : \phi]} & [\eta \vdash \phi] \vec{A} \\
| & | & | \\
[\eta ; \Gamma] \vec{R} & \xrightarrow{[\eta ; \Gamma \vdash M : \phi]} & [\eta \vdash \phi] \vec{R} \\
| & | & | \\
[\eta ; \Gamma] \vec{A}' & \xrightarrow{[\eta ; \Gamma \vdash M : \phi]} & [\eta \vdash \phi] \vec{A}'
\end{array}
\]

The fact that terms are interpreted as uniform families allows one to show that for any \( \vec{a} \in [\eta \vdash \Gamma] \vec{A} \), the interpretation \( p = [\eta ; \Gamma \vdash (\Lambda X.M) : \forall X.\phi]_{\vec{A}} \vec{a} \) is in the interpretation \( [\eta ; \vdash \forall X.\phi] \vec{A} \), in particular, in showing that \( p \) is a uniform family. It is also crucial for showing that the interpretation \( [\eta ; \Gamma \vdash (\text{open} \ M \text{ as } (X,x) \text{ in } N) : \phi']_{\vec{A}} \vec{a} \) is well-defined. If \( \text{in}_B(b) \sim \text{in}_{B'}(b') \), the uniformity of \( [\eta, X ; \Gamma, x : \phi \vdash N : \phi'] \) ensures the following.

\[
[\eta, X ; \Gamma, x : \phi \vdash N : \phi']_{(A_1, \ldots, A_n, B)}(a_1, \ldots, a_m, b) = [\eta, X ; \Gamma x : \phi \vdash N : \phi']_{(A_1, \ldots, A_n, B)}(a_1, \ldots, a_m, b')
\]
Thus the value of \( \eta ; \Gamma \vdash (X, x) \in N \vdash \phi' \) \( \tilde{a} \) does not depend on the particular element \( \in_{\tilde{B}} (b) \) of the equivalence class used to define it.

It was previously mentioned that Reynolds’ aim was to trim down the interpretation for polymorphic types given in the naïve model (section 2.1). In some cases, the set given for the interpretation of a polymorphic type has been trimmed down to agree with the set of (equivalence classes of) closed terms with that type in the predicative calculus. These sets frequently admit interesting and intuitive descriptions. Some such characterizations of polymorphic types were given by Reynolds [Rey83]. Even more results have been described, including what Wadler called “Theorems for Free” [ACC93, PA93, Wad89].

As an example, the type \( \forall X.X \Rightarrow X \) contains only a single term (up to equivalence), the polymorphic identity function. This is to be expected, as it agrees with our intuitive understanding of uniform computations of the type \( \forall X.X \Rightarrow X \). Given an element \( x \) of an unknown type \( X \), since the only thing known about \( X \) is that it contains \( x \), the only way to return an element of type \( X \) is to return \( x \). A similar line of reasoning can be formalized to prove the analogous result in Reynolds’ model: the interpretation of \( \emptyset \vdash \forall X.X \Rightarrow X \) (evaluated at the unique element * of \( S^0 \)) is a singleton set.

For any \( p \in [\emptyset \vdash \forall X.X \Rightarrow X] * \) and any set \( B \), we show the function \( p_B : B \rightarrow B \) is the identity function. For any \( b \in B \), consider the relation \( only_b = \{(b, b)\} \). Note that since \( p \) is a parametric family, we know \( p_B \left[ [X \vdash X \Rightarrow X] only_b \right] p_B \) and hence \( p_B \) maps \( only_b \)-related inputs to \( only_b \)-related outputs. Therefore, \( p_B (b) \) is related to itself by \( only_b \), which implies it is \( b \). Since \( p_B (b) = b \) for any \( b \in B \), \( p_B \) is the identity function.

2.3 Reflexive Graph Categories

O’Hearn and Tennent [OT95] pointed out that Reynolds’ relational parametricity can be seen to involve two categories. One is the category \( \text{SET} \) of sets and functions. The second is the category \( \text{Rel} \) of relational correspondences between sets. The objects of \( \text{Rel} \) are triples \( R = \langle A_0, A_1, \tilde{R} \rangle \) where \( \tilde{R} \subseteq A_0 \times A_1 \) (that is, objects are relations). The arrows are pairs \( \langle f_0, f_1 \rangle : \langle A_0, A_1, \tilde{R} \rangle \rightarrow \langle B_0, B_1, \tilde{S} \rangle \) where \( f_i : A_i \rightarrow B_i \) for \( i = 0, 1 \) are functions such that \( f_0 \left[ \tilde{R} \Rightarrow \tilde{S} \right] f_1 \). These arrows compose pointwise:

\[
\langle g_0, g_1 \rangle \circ \langle f_0, f_1 \rangle = \langle g_0 \circ f_0, g_1 \circ f_1 \rangle
\]
The identity arrows are \( \langle \text{id}_{A_0}, \text{id}_{A_1} \rangle : \langle A_0, A_1, R \rangle \to \langle A_0, A_1, R \rangle \). These two categories are bound by a reflexive graph structure, as we shall now describe.

An ordinary reflexive graph is a directed graph with a designated edge, \( I_v : v \leftrightarrow v \) for each vertex \( v \). Thus, a reflexive graph can be described as a structure consisting of two sets and three functions:

\[
\begin{array}{ccc}
V & \xleftarrow{\partial_0} & E \\
\downarrow & & \downarrow \\
\partial_1 & & \\
\end{array}
\]

satisfying the condition \( \partial_i \circ I = \text{id}_V \) (for \( i = 0, 1 \)). The idea is that \( \partial_0 \) and \( \partial_1 \) pick out the source and target for each edge and the function \( I \) assigns to each vertex a designated edge from the vertex to itself. This definition can be generalized to arbitrary categories and, in particular, to the category of categories, \( \text{CAT} \). We use \( \text{ID}_C \) to denote the identity functor on the category \( C \). The subscript \( C \) may be dropped when it can be inferred.

**Definition 2.1**

A reflexive graph category consists of two categories and three functors:

\[
\begin{array}{ccc}
G_v & \xleftarrow{\partial_0} & G_e \\
\downarrow & & \downarrow \\
\partial_1 & & \\
\end{array}
\]

such that \( \partial_0 \circ I = \text{ID}_{G_v} \) and \( \partial_1 \circ I = \text{ID}_{G_v} \).

As a general convention, \( G \) and \( H \) will be used to denote reflexive graph categories. Before proceeding, let us fix some terminology and notation. The category \( G_v \) is called the *vertex category* and \( G_e \) is called the *edge category* of the reflexive graph category \( G \). The objects and arrows of the vertex category are typically referred to as *vertices* and *morphisms*. The objects of the edge category are called *edges*. For any edge \( R \), the notation \( R : A_0 \leftrightarrow A_1 \) is used to indicate that \( \partial_0(R) = A_0 \) and \( \partial_1(R) = A_1 \). The arrows of \( G_e \) are called *squares* of the reflexive graph. For any arrow \( \sigma \) of \( G_e \), we use the following diagram to denote that \( \sigma : R \to S \) as well as \( R : A_0 \leftrightarrow A_1, S : B_0 \leftrightarrow B_1, \partial_0(\sigma) = f_0 : A_0 \to B_0 \) and \( \partial_1(\sigma) = f_1 : A_1 \to B_1 \).

\[
\begin{array}{ccc}
A_0 & \xrightarrow{f_0} & B_0 \\
\downarrow & & \downarrow \\
A_1 & \xrightarrow{f_1} & B_1 \\
\end{array}
\]

\( \sigma : R \)

The above diagram is called the *shape* of the square \( \sigma \). The edge which results from applying \( I \) to a vertex \( A \) is called the *identity edge over* \( A \), and
is typically denoted I₂. The vertex in the subscript may be suppressed when it can be inferred.

As mentioned previously, Reynolds' relational parametricity involves the categories \textbf{SET} and \textbf{Rel} which we regard as the vertex and edge categories of the reflexive graph category \textbf{REL}. The associated functors are as follows. For each \(i \in \{0, 1\}\), the functor \(\partial_i\) sends a relation \(<A_0, A_1, \tilde{R}_i>\) to \(A_i\) and sends an arrow \(<f_0, f_1>: <A_0, A_1, \tilde{R}_i> \to <B_0, B_1, \tilde{S}_i>\) to \(f_i: A_i \to B_i\). The functor \(I\) sends a set \(A\) to the diagonal relation \(<A, A, \Delta_A>\) and a function \(f: A \to B\) to the pair \(<f, f>: <A, A, \Delta_A> \to <B, B, \Delta_B>\). We shall typically use the less cumbersome notation \(R\) for an edge of \textbf{REL} rather than the more explicit \(<A_0, A_1, \tilde{R}_i>\). In situations where it is necessary to specify the source and target of a relation, they can be stated using the \(R: A_0 \leftrightarrow A_1\) notation. Other examples of reflexive graphs are given at the end of this section.

Reflexive graphs can be seen as \textit{indexed categories}, as follows. For a category \(V\), a \(V\)-indexed category (or indexed category with base \(V\)) is a functor\(^1\) from \(V^{\text{op}}\) to \textbf{CAT}. Let \(B\) be the category with two objects and 5 non-identity morphisms, as indicated here:

\[
\begin{array}{c}
\text{B} \\
\text{B}
\end{array} \quad \begin{array}{ccc}
\partial_0 & \overset{\text{v}}{\Rightarrow} & \text{id} \\
\partial_1 & \underset{\text{i}}{\Rightarrow} & \text{id}
\end{array} \quad \begin{array}{c}
\text{B} \\
\text{B}
\end{array}
\]

A reflexive graph \(G\) is simply a \(B^{\text{op}}\)-indexed functor, that is, a functor \(G: B \to \text{CAT}\).

This observation puts us back in the familiar territory of categories. For any small category \(V\), the \(V\)-indexed categories form a 2\textit{-category} (see [Mac71], for instance). An indexed functor \(F: G \to H\) is a natural transformation and an indexed natural transformation \(\eta: F \to G\) is a \textit{modification} (see [Mac71], for instance). Indexed categories, indexed functors and indexed natural transformations form a 2\textit{-category} and, hence, all the usual category theory concepts, such as adjunctions, limits, colimits, monads, comonads, algebras and coalgebras, can be applied to reflexive graph categories in a natural fashion. We use \(R\text{G}\) to denote the 2\textit{-category} of reflexive graph categories.

Another category theoretic term we apply to reflexive graph categories is the adjective \textit{well-pointed}. A category is well-pointed if it has a terminal object \(1\), and for any objects \(A, B\), and any arrows \(f, g: A \to B\), the equality

\(^1\)Here we use the term indexed category as referring to a functor, rather than a pseudo-functor (which would allow identities and composites be preserved up to isomorphism). Some people would prefer this to be called a \textit{strict} indexed category. The reader is welcome to mentally add \textit{strict} everywhere \textit{indexed} is used throughout the text if so desired.
f = g holds if and only if f \circ r = g \circ r \text{ for all arrows } r: 1 \to A. A reflexive graph category is well-pointed if both the vertex category and edge category are well-pointed.

As with any $\mathbb{V}$-indexed categories, $\mathbb{RG}$ is a Cartesian closed category. Products are obtained component-wise. In particular, the terminal reflexive graph category $1$ is the constant functor that returns the one-object, one arrow category (the terminal category). The exponent of $\mathbb{V}$-indexed categories can be described using the Hom-functor (much as the exponent of presheaves can be described). For each object $w$ of $\mathbb{B}$, $\text{Hom}(w, -)$ denotes the reflexive graph category whose $x$ component is the set (that is, discrete category) of arrows $\text{Hom}(w, x)$. For each arrow $f$ of $\mathbb{B}$, the functor $\text{Hom}(w, f)$ is defined by composition in the apparent manner. The $w$ component of the reflexive graph category $\mathbb{H}^G$ can be described as the category of indexed functors $\text{Hom}(w, -) \times G \to H$. Since $\text{Hom}(w, -) \cong 1$, the vertex category of $\mathbb{H}^G$ is made up of indexed functors $G \to H$ and the indexed natural transformations between them. To avoid confusion arising from the proliferation of morphisms of $\mathbb{B}$ in the notation, we use the reflexive graph category $\mathbb{E} \cong \text{Hom}(e, -)$ in describing the edge category of $\mathbb{H}^G$.

There are two objects of $\mathbb{E}$, which we denote $s$ and $t$, and there are three edges, $I_s : s \leftrightarrow s$, $I_t : t \leftrightarrow t$, and $E : s \leftrightarrow t$. The edges of $\mathbb{H}^G$ are indexed functors $\mathbb{E} \times G \to H$, and squares are indexed natural transformations. The functor $\partial_0$ fixes the $\mathbb{E}$ component at $s$, and $\partial_1$ fixes the $\mathbb{E}$ component at $t$. The functor $I$ sends $F : G \to H$ to the apparent functor $\mathbb{E} \times G \to H$ that is constant in $\mathbb{E}$.

A noteworthy fact is that the exponent $\mathbb{H}^G$ has such a simple structure. In particular, its vertices are precisely the indexed functors $G \to H$. For this reason, we shall call $\mathbb{H}^G$ the functor graph of $G$ and $H$. The fact that we use reflexive graphs (rather than merely graphs) is crucial for obtaining the simple structure of $\mathbb{H}^G$.

The reflexive graph category $\mathbb{E}$ basically describes a single edge. For any edge $R : X \leftrightarrow X'$ of $G$, there is an indexed functor $K_R : \mathbb{E} \to G$ given as follows.

\[
\begin{align*}
K_R(s) &= X & K_R(I_s) &= I_X \\
K_R(t) &= X' & K_R(I_t) &= I_{X'} \\
K_R(E) &= R
\end{align*}
\]

All indexed functors from $\mathbb{E}$ to $G$ are of the form $K_R$ for some $R \in G_e$. Similarly, one can equate squares $\sigma$ with indexed natural transformations,
$K_\sigma$ as follows:

$$
(K_\sigma)_s = \partial_0(\sigma) \quad (K_\sigma)_t = \partial_1(\sigma) \\
(K_\sigma)_h = 1_{\partial_0(\sigma)} \quad (K_\sigma)_l = 1_{\partial_1(\sigma)} \\
(K_\sigma)_E = \sigma
$$

In short, the edge category of $G$ is isomorphic to the category of indexed functors $E$ to $G$ via $K$.

A brief analysis of the above discussion reveals that none of the technical development is dependent on there being precisely 2 domain functions. The definition of reflexive graph category can easily be generalized to $n$-ary reflexive graph categories by changing the number of distinct morphisms $e \rightarrow v$ in $B$. Thus, an $n$-ary reflexive graph category consists of two categories $G_v, G_e$, a functor $I: G_v \rightarrow G_e$ and $n$ functors $\partial_i: G_e \rightarrow G_e$ such that $\partial_i \circ I = \text{ID}$ for all $i$.

The settings of $n$-ary reflexive graph categories provide abstract generalizations of $n$-ary relations. Binary reflexive graph categories were chosen as the basic notion as they are very intuitive and suffice to illustrate the relevant properties. While the $n$-ary generalization is worth the mention, the main focus of this paper will remain the binary case.

We close this section by giving several examples of reflexive graph categories to illustrate the richness of the concept.

**Posets** The reflexive graph **Poset** has the category of posets (sets together with partial orders) and monotone functions as the vertex category. The edge category has relations and relation-preserving pairs of functions, as in **REL**. However, the I functor assigns to a poset $\langle A, \sqsubseteq_A \rangle$ the partial order $\sqsubseteq_A$ and to a monotone function $f: \langle A, \sqsubseteq_A \rangle \rightarrow \langle B, \sqsubseteq_B \rangle$ the square $\langle f, f \rangle: \sqsubseteq_A \rightarrow \sqsubseteq_B$. The monotonicity of $f$ guarantees that this is a proper square.

**Complete Partial Orders** The reflexive graph **CPO** has the category of directed complete partial orders (CPOs) and continuous functions as the vertex category. The edge category has directed complete relations (relations closed under least upper bounds of directed subsets) and relation preserving pairs of continuous functions. The I functor sends a CPO to the diagonal relation over it, as in **REL**.

**Pointed Complete Partial Orders** The sub-category $\text{cpo}_\perp$ of **CPO** consists of those CPOs which have a least element $\perp$, and strict continuous functions, that is, continuous functions which map $\perp$ to $\perp$. The re-
flexive graph category $\text{CPO}_\bot$ has $\text{cpo}_\bot$ as its vertex category and similarly restricts edges to only the pointed complete relations (that is, relations that relate $\bot$ to $\bot$).

**Partial Equivalence Relations** A *partial equivalence relation* (or PER) over a set $X$ is a binary relation $A$ that is symmetric and transitive, but not necessarily reflexive. (Symmetry and transitivity ensure that $x A x$ holds for all $x$ which are related to any $y$ by $A$. So $A$ is essentially an equivalence relation on some subset of $X$.) The quotient $X/A$ is defined to be the set of equivalence classes $\{[x]_A \mid x A x\}$.

Consider PERs over the set of natural numbers. (More generally, one could use any partial combinatory algebra). A PER-morphism $f: A \rightarrow B$ is a function $f: \mathbb{N}/A \rightarrow \mathbb{N}/B$ that is realizable in the sense that there is a partial recursive function $\phi_k: \mathbb{N} \rightarrow \mathbb{N}$ that maps $A$-related inputs to $B$-related results and $f([x]_A) = [\phi_k(x)]_B$. Such a partial recursive function is said to realize $f$. The reflexive graph category $\text{PER}$ has the category of PERs over $\mathbb{N}$ and PER-morphisms for the vertex category.

The edge category of $\text{PER}$ is motivated by the relational parametricity identified by Bainbridge et al. [BFSS90] and used in [BAC95, PA93]. An edge $R: A \leftrightarrow B$ is a relation between natural numbers satisfying the following.

$$n' A n \land n R m \land m B n' \implies n' R m'$$

Such a relation $R$ is called a saturated relation. A square from an edge $R: A \leftrightarrow B$ to $R': A' \leftrightarrow B'$ is a pair $\langle f, g \rangle$ where $f: A \rightarrow A'$ and $g: B \rightarrow B'$ are PER-morphisms that have realizers $k$ and $l$, respectively, satisfying the following.

$$n R m \implies \phi_k(n) [R'] \phi_l(m)$$

The $I$ functor sends a PER $A$ to itself regarded as a saturated relation $A: A \leftrightarrow A$, and $f: A \rightarrow B$ to the pair $\langle f, f \rangle$.

**Arrow Graphs** Every category $\text{C}$ can be turned into a reflexive graph category. The vertex category is $\text{C}$ itself. The edge category is the arrow category $\text{C}_\rightarrow$ whose objects are the arrows of $\text{C}$ and arrows $\langle f_0, f_1 \rangle: g \rightarrow h$ are pairs of arrows of $\text{C}$ such that the following is a
commuting diagram.

\[
\begin{array}{ccc}
A_0 & \overset{f_0}{\rightarrow} & B_0 \\
\downarrow g & & \downarrow h \\
A_1 & \overset{f_1}{\rightarrow} & B_1
\end{array}
\]

We denote this reflexive graph category by \(C^{Arr}\). This definition gives (the object portion of) a full and faithful embedding of \(\text{CAT}\) in \(\text{RG}\). The indexed functors \(C^{Arr} \rightarrow D^{Arr}\) are precisely functors \(C \rightarrow D\) and the indexed natural transformations \(F \rightarrow G\) between indexed functors are precisely the natural transformations.

**Trivial Graphs** There are actually several ways that a category can be turned into a reflexive graph category. Another way is the trivial graph \(\text{Tr}(C)\), which has a single edge \(R: A_1 \leftrightarrow A_0\) for each pair of objects of \(C\). There is a square of the following shape if and only if \(f_0: A_0 \rightarrow B_0\) and \(f_1: A_1 \rightarrow B_1\).

\[
\begin{array}{ccc}
A_0 & \overset{f_0}{\rightarrow} & B_0 \\
\downarrow R & & \downarrow S \\
A_1 & \overset{f_1}{\rightarrow} & B_1
\end{array}
\]

Since there is a square for all pairs of morphisms, edges and squares provide no useful information.

**Spans** For any category \(C\), there is another canonical reflexive graph category, \(\text{Span}(C)\), that has \(C\) as its vertex category. The edge category of \(\text{Span}(C)\) is the functor category \(C^V\), where \(V\) is the three-object category indicated here.

\[
\begin{array}{ccc}
0 & \overset{p_0}{\leftarrow} & W \\
\downarrow & & \downarrow p_1 \\
1
\end{array}
\]

The edges are generally referred to as *spans*. The projections \(p_0\) and \(p_1\) are given by evaluation at 0 and 1, respectively, while identity edges are given by constant functors. Evaluation at \(W\) determines a third functor \(W: \text{Span}(C)_e \rightarrow C\). The object \(W(R)\) for a span \(R\) is typically called the *witness* of \(R\).
**Jointly Monic Spans** A variation of \( \text{Span}(C) \) is the reflexive graph category of *jointly monic spans* over \( C \), denoted \( \text{JMS}(C) \). This also has vertex category \( C \). The edge category \( \text{JMS}(C)_e \) is the full subcategory of \( \text{Span}(C)_e \) taking only those spans \( R:X \leftrightarrow Y \) that are jointly monic. A span is jointly monic if, for all pairs \( f,f':A \to R(W) \) (for any \( A \)),

\[
f = f' \iff (R(p_0) \circ f = R(p_0) \circ f') \land (R(p_1) \circ f = R(p_1) \circ f')
\]

(If \( C \) has products, \( R \) is a jointly monic span if and only if the pairing \( \langle R(p_0),R(p_1) \rangle:R(W) \to X \times Y \) is monic.)

Another variation of \( \text{Span}(C) \) cuts down the number of squares, rather than the number of edges. \( \text{Sp}(C) \) has the same vertices, morphisms, and edges as \( \text{Span}(C) \). There is a unique edge of a given shape in \( \text{Sp}(C) \) if and only if there is at least one span of that shape in \( \text{Span}(C) \).

**F-Algebras** Let \( F \) be an indexed functor on \( \text{REL} \). \( F \)-algebras are pairs of the form \( \langle A,\alpha:FA \to A \rangle \) with homomorphisms \( f: \langle A,\alpha \rangle \to \langle B,\beta \rangle \) being functions \( f:A \to B \) that preserve the structure maps in the following sense. \( \beta \circ F(f) = f \circ \alpha \)

The reflexive graph category \( F-\text{Alg} \) has the category of \( F \)-algebras as its vertex category. The edge category is as follows: The edges \( R: \langle A,\alpha \rangle \leftrightarrow \langle A',\alpha' \rangle \) are \( F \)-simulations, that is, relations \( R:A \leftrightarrow A' \) such that \( \alpha[F,R\Rightarrow R'] \alpha' \). A square is a pair \( \langle f,f' \rangle : R \to S \) where \( f \) and \( f' \) are homomorphisms such that \( f[R\Rightarrow S] f' \).

The \( I \) functor sends \( \langle A,\alpha \rangle \) to the diagonal relation \( \Delta_A \) and a homomorphism \( f: \langle A,\alpha \rangle \to \langle B,\beta \rangle \) to \( \langle f,f \rangle : \Delta_A \to \Delta_B \).

Note that this gives a recipe to build reflexive graph categories for familiar algebraic structures, such as monoids, groups, rings, etc. For instance, for monoids, a simulation relation \( R:A \leftrightarrow A' \) would be one that satisfies

\[
x_1Rx_1' \land \cdots \land x_nRx_n' \implies (\Pi_i x_i) (R) (\Pi_i x_i')
\]

**Free Algebras** Let \( T = \langle T,\eta,\mu \rangle \) be an indexed monad on \( \text{REL} \), that is, \( T \) is an indexed functor and \( \eta: \text{ID} \to T \) and \( \mu:T^2 \to T \) are indexed natural transformations satisfying the usual laws of monads. A reflexive
graph of $T$-algebras can be identified in the same way as the previous example. However, for free algebras, we have another possibility.

Recall that the Kleisli category $\text{SET}_T$ has sets as objects and functions of the form $f:A \to TB$ as arrows from $A$ to $B$. We define a reflexive graph category $\text{REL}_T$ with $\text{SET}_T$ as the vertex category by taking edges to be relations and squares from $R:A_0 \leftrightarrow A_1$ to $S:B_0 \leftrightarrow B_1$ to be pairs $(f_0:A_0 \to TB_0, f_1:A_1 \to TB_1)$ such that $f_0 [R \Rightarrow TS] f_1$. The composition of squares is pointwise.

\[
\begin{array}{c}
A_0 & \xrightarrow{f_0} & TB_0 & \xrightarrow{Tg_0} & T^2C_0 & \xrightarrow{\mu_{C_0}} & TC_0 \\
R & \xrightarrow{TS} & \downarrow & & \downarrow & & P \\
A_1 & \xrightarrow{f_1} & TB_1 & \xrightarrow{Tg_1} & T^2C_1 & \xrightarrow{\mu_{C_1}} & TC_1
\end{array}
\]

(The last square is the $P$ component of the indexed natural transformation $\mu$.) The $I$ functor sends $A$ to $A_{\perp}$ and $f$ to $(f, f)$.

This gives a recipe to build reflexive graph categories for familiar Kleisli categories, such as $\text{Pfin}$, $\text{rel}$, and those of “computational” monads considered by Moggi [Mog91].

We look at $\text{Pfin}$ as a concrete example. The monad under question is that of lifting: $TA = A_{\perp} = A \uplus \{\perp\}$. We extend it to an indexed functor by defining $R_{\perp}:A_0_{\perp} \leftrightarrow A_{\perp}$ as follows.

\[x[R_{\perp}]y \iff (x = \perp \land y = \perp) \lor x \land y \]

Note that $f_0 [R \Rightarrow S] f_1$ implies $f_0_{\perp} [R_{\perp} \Rightarrow S_{\perp}] f_1_{\perp}$. The associated natural transformations extend to indexed natural transformations, defined at $R$: $A_0 \leftrightarrow A_1$ by $\eta_R = (\eta_{A_0}, \eta_{A_1})$ and $\mu_R = (\mu_{A_0}, \mu_{A_1})$. They are squares since $\eta_{A_0} [R \Rightarrow R_{\perp}] \eta_{A_1}$ and $\mu_{A_0} [(R_{\perp})_{\perp} \Rightarrow R_{\perp}] \mu_{A_1}$.

The edge category of $\text{Pfin}$ as given by the above recipe has relations as edges and pairs $(f_0, f_1)$ such that $f_0 [R \Rightarrow S_{\perp}] f_1$ as squares. Expanding the notation, we note that $(f_0, f_1):R \to S$ is a square if and only if the following holds.

\[\forall x,y. \ x \land y \Rightarrow (f_0(x) = \perp \land f_1(y) = \perp) \lor (f_0(x) \lceil S \rceil f_1(y))\]

Note that this is quite different from the reflexive graph structure of algebras discussed in the previous example. The algebras for the lift monad are pointed sets $(A, \perp_A)$ and the morphisms are strict

32
functions (functions preserving the chosen point). A T-simulation \( R: (A_0, \bot_{A_0}) \leftrightarrow (A_1, \bot_{A_1}) \) is a relation \( R: A_0 \leftrightarrow A_1 \) such that the bottom elements are related, \( \bot_{A_0}[R] \bot_{A_1} \). Even though every pointed set can be expressed as a free algebra \( A = X_{\bot} \), this is not so for T-simulations. A T-simulation \( R: X_{\bot} \leftrightarrow Y_{\bot} \) is not necessarily of the form \( P_{\bot} \) for a relation \( P: X \leftrightarrow Y \).
Chapter 3

Reflexive Graph Models of Polymorphism

As discussed in Chapter 2, reflexive graph categories form an abstract description of the structure of relational correspondences involved in Reynolds’ theory of parametricity. In this chapter, we explore the structure of reflexive graph categories so as to be able to define categorical models for polymorphic lambda calculus.

The vertex category of the exponent $H^G$ in the 2-category $RG$ consists of the indexed functors (1-cells) $G \to H$. This is similar to the situation in $CAT$, but is not true of all 2-categories. Much like what happens in $CAT$, many of the 2-categorical constructions which can be defined abstractly enjoy direct intuitive descriptions as well. One example is the inclusion of $G$ into $G^H$ (often referred to as the diagonal arrow) which is given in any Cartesian closed category by currying the projection, $\Delta = (\Pi_1)^H \circ \eta_H$. It is more intuitive to describe this as producing constant indexed functors.

In this chapter, similar intuitive descriptions of the limit and colimit as universal indexed natural transformations are shown. These two constructions correspond exactly to the set of polymorphic functions defined by Reynolds [Rey83, see also section 2.2] and the set of equivalence classes defined by Reddy [Red98, see also section 2.2], respectively. These two definitions give us the tools to define a categorical model of the predicative polymorphic lambda calculus. The definitions can be adapted to internal categories as well, leading to a categorical model of System F.

3.1 Parametric Limits and Colimits

Recall from section 2.2 that Reynolds interpreted the type $\forall X.\tau$ as the set of parametric families of elements of $\tau$, not arbitrary families. This resulted in
a highly trimmed down model of the predicative calculus. In this section, we show that Reynolds’ idea corresponds to limits in reflexive graph categories, characterized by a universal property. Likewise, Reddy’s interpretation of ∃X.τ corresponds to colimits.

The inclusion $\Delta: G \to G^H$ (or diagonal) indexed functor gives a way to treat a vertex as an indexed functor that intuitively ‘contains the same information’. The right adjoint to $\Delta$, when it exists, provides a method to do the reverse, that is, produce a vertex that intuitively ‘has the same information’ as a given indexed functor. In standard categories, the right adjoint to the diagonal functor is the Limit functor. The point-wise description of limits (that is, the limit of a particular functor) can be interesting even when the whole of the functor $\text{Lim}: C^D \to C$ does not exist. A similar consideration applies to reflexive graph categories as well.

**Definition 3.1 (Parametric limit)**

Given an indexed functor $F: H \to G$, a parametric limit of $F$ consists of a vertex $\forall Y F(Y)$ of $G$ together with an indexed natural transformation $\omega: \Delta(\forall Y F(Y)) \to F$ which is universal in the following sense:

For any indexed natural transformation $\beta: \Delta(A) \to F$ (for any vertex $A$ of $G$), there is a unique morphism $\Lambda(\beta): A \to \forall Y F(Y)$ such that the following diagram commutes.

\[
\begin{array}{ccc}
\Delta(A) & \xrightarrow{\Delta(\lambda(\beta))} & \Delta(\forall Y F(Y)) \\
\downarrow{\beta} & & \downarrow{\omega} \\
F & & F
\end{array}
\]

As one expects of something defined via such a universal property, parametric limits are unique up to isomorphism. Therefore one typically refers to the parametric limit of $F$.

We calculate parametric limits in some of our example reflexive graph categories.

- For any indexed functor $F: H \to \text{REL}$, where $H$ is small, the parametric limit of $F$ is as follows.

  \[
  \forall Y F(Y) = \left\{ (t_Y) Y \in \Pi Y \in H F(Y) \mid \forall R: A \leftrightarrow B, t_A [F(R)] t_B \right\}
  \]\n
  and $\forall f: A \to B, t_B = F(f) \circ t_A$

  The indexed natural transformation $\omega: \forall Y F(Y) \to F$ has components
as follows.

$$\omega_X(t_Y) = t_X$$

$$\omega_R = \text{the unique square}$$

$$\forall Y F(Y) \xrightarrow{\omega_X} F(X)$$

$$\forall Y F(Y) \xrightarrow{\omega_Y} F(R)$$

The square $\omega_R$ exists since $\omega_X(t_Y) [F(R)] \omega_Y(t_Y)$ for every family $(t_Y) \in \forall Y F(Y)$, by definition.

Given any indexed natural transformation $\beta: A \rightarrow F$ for any set $A$, the function $\Lambda(\beta): A \rightarrow \forall Y F(Y)$ is given by $\Lambda(\beta)(x) = \langle \beta_Y(x) \rangle_{Y \in G}$. Note that $\Lambda(\beta)(x)$ is an element of $\forall Y F(Y)$, since $\beta_R$ is a square of the following shape for every $R: B_0 \leftrightarrow B_1$ in $H_c$.

$$A \xrightarrow{\beta_{B_0}} F(B_0)$$

$$I_A \downarrow \downarrow$$

$$A \xrightarrow{\beta_{B_1}} F(B_1)$$

Note that $\forall Y F(Y)$ only contains “parametric families” as required by Reynolds.

- **CPO** also has small parametric limits. The underlying set of the parametric limit for an indexed functor $F: H \rightarrow \text{CPO}$ is as above.

$$\forall Y F(Y) = \left\{ \langle t_Y \rangle_Y \in \Pi_{Y \in H_c} F(Y) \mid \forall R: A_0 \leftrightarrow A_1, t_{A_0} [F(R)] t_{A_1} \right\}$$

This is a complete partial order when ordered component-wise, that is, $\langle t_Y \rangle_Y \subseteq \langle u_Y \rangle_Y \iff t_Y \leq u_Y$ for every $Y \in H_c$. Completeness is a consequence of every edge in CPO being a complete relation. The projection $\omega$ and factorization $\Lambda(\beta)$ are as for REL above.

**Theorem 3.2**

If $C$ is a category with small limits, such as SET, then $\text{Span}(C)$ has all small parametric limits. That is, if $G$ is a reflexive graph such that $G_e$ is small (which implies that $G_e$ is small as well), then every indexed functor $F: G \rightarrow \text{Span}(C)$ has a parametric limit.

**Proof.** A first attempt to compute the parametric limit might be to consider the limit of the functor $W \circ F_e: G_e \rightarrow C$ in the category $C$ where the
projection \( \pi_X : \lim_R W(F_c(R)) \to W(F_c(X)) \), is used to give the projection \( \omega : \Delta(\lim_R W(F_c(R))) \to F \) as follows.

\[
\omega_A = \pi_{I_A} \quad \omega_R = (\pi_R, \pi_{I_{A_0}}, \pi_{I_{A_1}})
\]

One might hope that this \( \omega \) might be an indexed natural transformation. However, \( \omega_R \) may not be a square, as there is no reason that the parallelograms below need to commute.

\[
\begin{array}{ccc}
\lim_R W(F_c(R)) & \xrightarrow{\pi_{I_{A_0}}} & W(F(I_{A_0})) \quad F(A_0) \\
\downarrow \text{id} & & \quad \downarrow p_0 \\
\lim_R W(F_c(R)) & \xrightarrow{\pi_R} & W(F(R)) \Rightarrow W(F(R)) \\
\downarrow \text{id} & & \quad \downarrow p_1 \\
\lim_R W(F_c(R)) & \xrightarrow{\pi_{I_{A_1}}} & W(F(I_{A_1})) \Rightarrow F(A_1)
\end{array}
\]

To be able to ensure that parallelograms like those above commute, we consider an auxiliary category \( \mathbf{G}^* \) which has all the vertices and edges of \( \mathbf{G} \) as objects. The arrows of \( \mathbf{G}^* \) include not only the morphisms and squares of \( \mathbf{G} \), but also arrows from \( R \) to \( \partial_0(R) \) for all edges \( R \) (and similarly for each of the structural functors of \( \mathbf{G} \)). This construction of the total category from the indexed category \( \mathbf{G} \) due to Grothendieck, is standard (see [Her93], for instance).

The objects of \( \mathbf{G}^* \) consist of the vertices and edges of \( \mathbf{G} \). The arrows of \( \mathbf{G}^* \) are given by pairs \((m, f): X \to Y\) where \( m: x \to y \) is an arrow of \( \mathbf{B} \) and \( f: (\mathbf{G}(m), X) \to Y \) is an arrow of \( \mathbf{G}_y \). (A typical example of an arrow in \( \mathbf{G}^* \) is \( (\partial_0, f): R \to X \) where \( f: \partial_0(R) \to X \) is a morphism of \( \mathbf{G} \).) Identities are inherited from \( \mathbf{G}_v \) and \( \mathbf{G}_e \) as \((\text{id}, \text{id})\). The composite of \((m, f): X \to Y\) and \((m', f'): Y \to Z\) is given by \((m' \circ m, f' \circ (\mathbf{G}(m)f))\).

An indexed functor \( F : \mathbf{G} \to \text{Span}(\mathbf{C}) \) determines a functor \( F^* : \mathbf{G}^* \to \mathbf{C} \) as indicated below. Note that this is different from the Grothendieck construction, since we are looking for a functor into \( \mathbf{C} \) rather than \((\text{Span}(\mathbf{C}))^*\).

\[
\begin{align*}
F^*(A) &= F(A) & F^*(\text{id}_v, f) &= F(f) \\
F^*(I, \sigma) &= W(F(\sigma)) & F^*(\text{id}_e, \sigma) &= W(F(\sigma)) \\
F^*(\partial_0, f) &= F(f) \circ p_0 & F^*(\partial_1, f) &= F(f) \circ p_1 \\
F^*(I \circ \partial_0, \sigma) &= W(F(\sigma)) \circ p_0 & F^*(I \circ \partial_1, \sigma) &= W(F(\sigma)) \circ p_1
\end{align*}
\]

37
Considering the limit of $F^*$ in $\mathbf{C}$ (with projection natural transformation $\pi \colon \lim_X F^*(X) \to F^*$), it is the case that for every edge $R \colon A_0 \leftrightarrow A_1$, the parallelograms below commute.

\[
\begin{array}{c}
\lim_X F^*(X) \\
\downarrow \pi_R \\
\lim_X F^*(X)
\end{array}
\xrightarrow{\pi} \begin{array}{c}
F^*(A_0) \\
\downarrow F^*(\partial_0, id) \\
F^*(R)
\end{array} \xrightarrow{F^*(\partial_1, id)} \begin{array}{c}
W(F(R)) \\
\downarrow p_1 \\
F^*(A_1)
\end{array} \\
\pi_{A_1} \xrightarrow{\pi} F^*(A_0) = F(A_0)
\]

Thus, there is an indexed natural transformation $\omega \colon \Delta(\lim F^*) \to F$ given as follows.

\[
\omega_A = \pi_A \quad \omega_R = (\pi_R, \pi_{\partial_0(R)}, \pi_{\partial_1(R)})
\]

For any object $K$ of $\mathbf{C}$, an indexed natural transformation $\beta \colon \Delta K \to F$ gives rise to a natural transformation $\beta^* \colon \Delta K \to F^*$. For a vertex $A$, we define $\beta^*_A = \beta_A$, while for an edge $R \colon A_0 \leftrightarrow A_1$, the arrow $\beta^*_R$ is given by the witness component of the square $\beta_R$.

\[
\begin{array}{c}
K \\
\downarrow id \\
K
\end{array}
\xrightarrow{\beta} \begin{array}{c}
F(A_0) \\
\downarrow \beta_{A_0} \\
F(\partial_0(R))
\end{array} \xrightarrow{F^*(\partial_1, id)} \begin{array}{c}
W(F(R)) \\
\downarrow \beta_{A_1} \\
F(A_1)
\end{array} \\
\beta_R \xrightarrow{\beta^*_R} K
\]

Moreover, the unique morphism $\Lambda(\beta^*) \colon K \to \lim F^*$ such that $\pi \circ \Lambda(\beta^*) = \beta^*$ is the unique morphism such that the following commutes.

\[
\begin{array}{c}
K \\
\downarrow \beta \\
\lim_X F^*(X)
\end{array}
\xrightarrow{\Lambda(\beta^*)} \begin{array}{c}
\downarrow \omega \\
F
\end{array}
\]

In other words, $\lim_X F^*(X)$ is a parametric limit of $F \colon \mathbf{G} \to \mathbf{Span}(\mathbf{C})$.

The above construction of parametric limits in $\mathbf{Span}(\mathbf{C})$ also works to produce parametric limits in $\mathbf{JMS}(\mathbf{C})$ and $\mathbf{Sp}(\mathbf{C})$.

Let us consider the particular case of parametric limits in $\mathbf{Span}(\mathbf{SET})$ in more detail. Recalling limits in $\mathbf{SET}$, parametric limits in $\mathbf{Span}(\mathbf{SET})$ can be spelled out more explicitly. An element of $\forall_Y F(Y)$ is a collection $q = \langle q_X \in F^*(X) \rangle$, indexed by objects $X$ of $\mathbf{G}^*$, that is preserved by
\(F^*(m, f)\) for every arrow \((m, f)\) of \(G^*\) in the sense that \(q_Y = F^*(m, f)(q_X)\).

Since all vertices of \(G\) are objects of \(G^*\), \(q\) determines a collection of functions \(\rho_A: 1 \rightarrow F(A)\) indexed by vertices of \(G\) satisfying \(\rho_A(*) = q_A\). Since morphisms \(f: A \rightarrow A'\) of \(G\) are arrows of \(G^*\), they commute with the \(\rho\)'s.

\[
\begin{array}{ccc}
1 & \xrightarrow{\rho_A} & F(A) \\
\downarrow{\text{id}} & & \downarrow{F^*(\text{id}_V, f) = F(f)} \\
1 & \xrightarrow{\rho_{A'}} & F(A')
\end{array}
\]

Hence \(\rho: 1 \rightarrow F_v\) is a natural transformation between functors \(G_v \rightarrow \text{SET}\).

For every edge \(R: A_0 \leftrightarrow A_1\) of \(G\), let \(\overline{\rho}_R: 1 \rightarrow W(F(R))\) denote the function which produces the element \(q_R \in W(F(R))\). Thus, there is the following square.

\[
\begin{array}{ccc}
1 & \xrightarrow{\rho_{A_0}} & F(A_0) \\
\downarrow{\overline{\rho}_R} & & \downarrow{p_0} \\
W(F(R)) & \xrightarrow{p_1} & F(A_1) \\
\downarrow{1} & & \downarrow{\rho_{A_1}} \\
1 & \xrightarrow{1} & 1
\end{array}
\]

This square exists since \(p_0 = F^*(\partial_0, \text{id})\) and \(p_1 = F^*(\partial_1, \text{id})\) and the collection \(q\) is preserved by \(F^*(\partial_0, \text{id})\) and \(F^*(\partial_1, \text{id})\). It is easy to see that this collection of squares commutes with \(F_v(\sigma)\) for all squares \(\sigma\) of \(G\), hence there is a natural transformation \(\rho: 1 \rightarrow F_v\) between functors \(G_v \rightarrow \text{Span}(\text{SET})_v\) where \(\rho_R = (\overline{\rho}_R, \rho_{A_0}, \rho_{A_1})\) for any \(R: A_0 \leftrightarrow A_1\). Since \(\rho\) commutes with the structural functors, the arbitrary element \(q\) of \(\forall_Y F(Y)\) determines an indexed natural transformation \(1 \rightarrow F\).

Conversely, any indexed natural transformation \(\eta: 1 \rightarrow F\) determines an element \(\langle n_X \rangle\) of \(\forall_Y F(Y) = \lim_X F^*(X)\) where

\[
n_A = \eta_A(*) \quad \text{for vertices } A \\
n_R = W(\eta_R)(*) \quad \text{for edges } R.
\]

Naturality of the vertex portion ensures that this collection is preserved by arrows of the form \((\text{id}_e, f)\), naturality of the edge portion ensures the preservation by arrows of the form \((\text{id}_e, \sigma)\), and the indexed requirement ensures that this collection is preserved by arrows of the form \((m, \text{id})\). Since these three basic forms generate all the arrows of \(G^*\) through composition, all arrows preserve the collection.

39
The above constructions give a bijection between the set $\forall Y F(Y)$ and the set of indexed natural transformations $1 \to F$. Such a characterization of parametric limits holds in many other reflexive graph categories. Parametric limits in $\textbf{REL}$, $\textbf{CPO}$, and $\textbf{Span}(\textbf{CPO})$ are isomorphic to the set of indexed natural transformations from $1$ to $F$ (ordered pointwise, where appropriate).

The similarity between parametric limits in reflexive graph categories and limits in categories means many results about limits have analogues pertaining to parametric limits. We mention a few such expected results in the following lemma.

**Lemma 3.3**

Suppose $F : \textbf{H} \to \textbf{G}$ is an indexed functor between reflexive graph categories, $f : A \to B$ is a morphism of $\textbf{G}$ and $\tau : \Delta B \to F$ is an indexed natural transformation.

1. $A \cong \forall Y A$

2. $\Lambda(\tau) \circ f = \Lambda(\tau \circ \Delta f)$

**Proof.**

The first point is immediate, as $A$ together with $\Delta \text{Id}$ is a parametric limit of the constant PG-functor $\Delta A$. The second point is ensured by the uniqueness of the factorization $\Lambda(\tau \circ \Delta f)$. Note the following equalities.

$$
\omega \circ \Delta(\Lambda(\tau) \circ f) = \omega \circ \Delta(\Lambda(\tau)) \circ \Delta f = \tau \circ \Delta f
$$

Therefore $\Lambda(\tau) \circ f$ must be the factorization $\Lambda(\tau \circ \Delta f)$.

When working with indexed functors out of a product reflexive graph category, it often arises that one wishes to take the parametric limit over only one component of the product. The other components of the product should be treated as parameters, resulting in an indexed functor, rather than a vertex. This situation can be handled as a special case of parametric limits.

**Definition 3.4**

Let $F : \textbf{G} \times \textbf{G'} \to \textbf{H}$ be an indexed functor. The parametric limit of $F$ with parameters from $\textbf{G}$, denoted $\forall Y F(-, Y)$, is the parametric limit of $\text{curry}(F) : \textbf{G'} \to \textbf{H}^\text{G}$.

**Examples:**

- For any indexed functor $F : \textbf{G} \times \textbf{G'} \to \textbf{REL}$ where $\textbf{G'}$ is small, we define the parametric limit of $F$ with parameters from $\textbf{G}$ for any vertex
A, morphism \( f: A \to B \), edge \( R: A_0 \leftrightarrow A_1 \) and square \( \sigma \) of the following shape.

\[
\begin{array}{c}
A_0 \\
\begin{array}{c}
\hspace{0.5cm} f_0 \downarrow \\
R \\
\hspace{0.5cm} f_1 \\
A_1 \\
\downarrow S
\end{array}
\end{array}
\]

\[
\forall_Y F(A, Y) = \left\{ (t_y)_{Y} \in \Pi_Y F(A, Y) \left| \forall R: Y_0 \leftrightarrow Y_1, \forall f: Y \to Y', t_{Y_0} \left[ F(I_{A}, R) \right] t_{Y_1} \text{ and } t_{Y'} = F(\text{id}, f)(t_Y) \right. \right\}
\]

\[
\forall_Y F(f, Y) = \langle t_Y \rangle_Y \longrightarrow \langle F(f, \text{id}_Y) t_Y \rangle_Y
\]

\[
\langle t_Y \rangle_Y \left[ \forall_Y F(R, Y) \right] \langle u_Y \rangle_Y \iff \forall R: Y_0 \leftrightarrow Y_1, t_{Y_0} \left[ F(R, R') \right] u_{Y_1}
\]

\[
\forall_Y F(\sigma, Y) = \left( \forall_Y F(f_0, Y), \forall_Y F(f_1, Y) \right)
\]

- For any category \( C \) with small limits, if \( G' \) is a reflexive graph category where \( G'_e \) is small, then any indexed functor \( F: G \times G' \to \text{Span}(C) \) has a parametric limit with parameters from \( G \). This is computed in a manner similar to the parametric limits (without parameters) in \( \text{Span}(C) \) (Theorem 3.2) using the total category \( G'^* \). For the vertex portion, we define the auxiliary functor \( F^*: G_e \times G'^* \to C \) as follows.

\[
F^*(A, B) = F(A, B) = W(F(I_A, I_B))
\]

\[
F^*(A, R) = W(F(I_A, R))
\]

\[
F^*(f, (\text{id}_e, g)) = F(f, g)
\]

\[
F^*(f, (I, \sigma)) = W(F(I_f, \sigma))
\]

\[
F^*(f, (\text{id}_e, \sigma)) = W(F(\text{id}_f, \sigma))
\]

\[
F^*(f, (\theta_j, g)) = p_j \circ W(F(\text{id}_f, g))
\]

\[
F^*(f, (1 \circ \theta_j, \sigma)) = p_j \circ W(F(\text{id}_f, g))
\]

The indexed functor \( \forall_Y F(-, Y): G \to \text{Span}(C) \) is defined at a vertex \( A \) to be the limit (in the category \( C \)) of \( F^*(A, -) \). The morphism \( \forall_Y F(f, Y): \forall_Y F(A, Y) \to \forall_Y F(B, Y) \) comes from factoring the projection out of \( \lim_Y F(A, Y) \) followed by \( F(f, \text{id}) \). That is, \( \forall_Y F(f, Y) \) is the unique morphism \( m: \lim_X F^*(A, X) \to \lim_X F^*(B, X) \) such that \( \pi_X \circ m = F^*(f, \text{id}_X) \circ \pi_X \) for all objects \( X \) of \( G^* \).

The edge portion of \( \forall_Y F(-, Y) \) is defined using the auxiliary functor
\[ F^*: \mathcal{G}_e \times \mathcal{G}^{**} \to \mathcal{C} \] defined as follows.

\[
\begin{align*}
F^*(R, B) &= W(F(R, I_B)) \\
F^*(R, R') &= W(F(R, R')) \\
F^*(\sigma, (\text{id}_e, g)) &= W(F(\sigma, I_g)) \\
F^*(\sigma, (1, \sigma')) &= W(F(\sigma, \sigma')) \\
F^*(\sigma, (\text{id}_e, \sigma')) &= W(F(\sigma, \sigma')) \\
F^*(\sigma, (\partial_0, g)) &= p_j \circ W(F(\sigma, I_g)) \\
F^*(\sigma, (1 \circ \partial_1, \sigma')) &= p_j \circ W(F(\sigma, \sigma'))
\end{align*}
\]

The witness object of the span \( \forall_Y F(R, Y) \) is as follows.

\[ W(\forall_Y F(R, Y)) = \lim_X F^*(R, X) \]

The two projection morphisms \( p_0: W(\forall_Y F(R, Y)) \to \forall_Y F(A_0, Y) \) and \( p_1: W(\forall_Y F(R, Y)) \to \forall_Y F(A_1, Y) \) are as follows.

\[ p_0 \text{ is the unique } m: \lim_X F^*(R, X) \to \lim_X F^*(A_0, X) \text{ such that } \pi_X \circ m = F^*(\partial_0, \text{id}) \circ \pi_X \]

\[ p_1 \text{ is the unique } m: \lim_X F^*(R, X) \to \lim_X F^*(A_1, X) \text{ such that } \pi_X \circ m = F^*(\partial_1, \text{id}) \circ \pi_X \]

To specify the square \( \forall_Y F(\sigma, Y): \forall_Y F(R, Y) \to \forall_Y F(S, Y) \), we define the witness morphism \( W(\forall_Y F(\sigma, Y)) \) to be the unique morphism \( m: \lim_X F^*(R, X) \to \lim_X F^*(S, X) \) such that, for all objects \( X \) in \( \mathcal{G}^* \), \( \pi_X \circ m = F^*(\sigma, \text{id}_X) \circ \pi_X \).

Parametric limits with parameters from \( \mathcal{E} \) are useful in describing the edge portion of the right adjoint to \( \Delta \) whenever it exists. This right adjoint is called the parametric limit functor. Recall there is an isomorphism (of categories) \( K: \mathcal{G}_e \to (\mathcal{G}^F) \) that equates indexed functors \( \mathcal{E} \to \mathcal{G} \) with edges of \( \mathcal{G} \) (section 2.3, page 27).
Theorem 3.5
Suppose $\text{Lim} : \mathbf{G}^\mathbf{H} \to \mathbf{G}$ is an indexed functor which is the right adjoint to $\Delta$. For every indexed functor $F : \mathbf{H} \to \mathbf{G}$, the vertex $\text{Lim}(F)$ is a parametric limit of $F$, and for every $\mathcal{F} : \mathbf{E} \times \mathbf{H} \to \mathbf{G}$, $K_{\text{Lim}(\mathcal{F})} : \mathbf{E} \to \mathbf{G}$ is a parametric limit of $\mathcal{F}$ with parameters from $\mathbf{E}$.

Proof. To show that $K_{\text{Lim}(\mathcal{F})}$ is the parametric limit with parameters from $\mathbf{E}$ of $\mathcal{F}$, we must consider the diagonal functor $\Delta' : \mathbf{G}^{\mathbf{E}} \to \mathbf{G}^{\mathbf{E} \times \mathbf{H}}$ (suppressing the isomorphism $(\mathbf{G}^{\mathbf{E}})^{\mathbf{H}} \to \mathbf{G}^{\mathbf{E} \times \mathbf{H}}$) in addition to the diagonal functor $\Delta : \mathbf{G} \to \mathbf{G}^{\mathbf{H}}$ that is the left adjoint to $\text{Lim}$. Direct computation shows that $\Delta_\mathbf{e}$ and $\Delta'_\mathbf{e} \circ K$ are the same functor $G_\mathbf{e} \to \left(G^{\mathbf{E} \times \mathbf{H}} \right)_\mathbf{e}$, and hence $\Delta(\text{Lim}(\mathcal{F})) = \Delta'(K_{\text{Lim}(\mathcal{F})})$ as indexed functors $\mathbf{E} \times \mathbf{H} \to \mathbf{G}$.

The projection indexed natural transformation $\omega : \Delta'(K_{\text{Lim}(\mathcal{F})}) \to \mathcal{F}$ is given by the $\mathcal{F}$ component of the co-unit $\epsilon_\mathcal{F} : \Delta(\text{Lim}(\mathcal{F})) \to \mathcal{F}$ of the adjunction between $\Delta$ and $\text{Lim}$. For any $R$ and $\beta : \Delta(K_R) \to \mathcal{F}$, there is a square $\sigma : R \to \text{Lim}(\mathcal{F})$ given by the composition $\text{Lim}(\beta) \circ \eta_{R \mathbf{e}}$. The indexed natural transformation $K_\sigma : K_R \to K_{\text{Lim}(\mathcal{F})}$ gives the unique factorization $\Lambda(\beta)$ of $\beta$ through $K_{\text{Lim}(\mathcal{F})}$.

Showing that $\text{Lim}(F)$ is the parametric limit of $F$ is quite similar, with $\omega = \epsilon_\mathcal{F}$ and $\Lambda(\beta) = \text{Lim}(\beta) \circ \eta_{\mathbf{A}}$.

Note that parametric limits in functor reflexive graph categories can be given by the limit indexed functor for the codomain, as stated in the following proposition. A consequence of this proposition is that parametric limits with parameters can be reduced to only considering parameters from $\mathbf{E}$.

Proposition 3.6
If $\text{Lim} : \mathbf{G}^\mathbf{H} \to \mathbf{G}$ is a parametric limit functor, then using the isomorphism $i : (\mathbf{G}^\mathbf{K})^\mathbf{H} \cong (\mathbf{G}^\mathbf{H})^\mathbf{K}$, it is the case that $(\text{Lim})^\mathbf{K} \circ i : (\mathbf{G}^\mathbf{K})^\mathbf{H} \to \mathbf{G}^\mathbf{K}$ is a parametric limit functor.

The parametric limit is a generalization of Reynolds’ interpretation for polymorphic types. The explicit description via preserving all relations given in section 2.2 is the same as defining $[\eta \vdash \forall X.\tau]$ as the parametric limit of $[\eta, X \vdash \tau]$ as indexed functors into $\text{REL}$.

The dual construction to parametric limits is that of parametric colimits.

Definition 3.7
Suppose $F : \mathbf{H} \to \mathbf{G}$ is an indexed functor, a parametric colimit of $F$ is a vertex $\exists X F(X)$ of $\mathbf{G}$ together with an indexed natural transformation $\mu : F \to \Delta(\exists X F(X))$ which is universal in the following sense:
For any indexed natural transformation $\beta: F \to \Delta(B)$, there is a unique morphism $\nabla(\beta): \exists_X F(X) \to B$ such that the following diagram commutes.

\[
\begin{array}{c}
F \\
\mu \\
\Delta(\exists_X F(X)) \\
\beta \\
\Delta(\nabla(\beta)) \\
\end{array}
\]

The parametric colimit of $F: G \times G' \to H$ with parameters from $G$ is the parametric colimit of curry($F$): $G' \to H^G$, and is denoted $\exists_X F(-, X)$.

**Examples:**

- The parametric colimit of an indexed functor $F: H \to \text{REL}$ where $H_v$ is small is given by a quotient.

  \[\exists_Y F(Y) = \Sigma_{A \in H_v} F(A) / \approx\]

  The relation $\approx$ is the transitive closure of the following relation $\sim$.

  \[\sim = \{(\text{in}_A(a), \text{in}_B(b)) \mid a \left[ F(R) \right] b \text{ for some } R: A \leftrightarrow B \text{ or } F(f)(a) = b \text{ for some } f: A \to B \text{ or } F(g)(b) = a \text{ for some } g: B \to A \}\]

  The indexed natural transformation $\mu: F \to \exists_Y F(Y)$ maps an element to its equivalence class.

  \[\mu_A(a) = [\text{in}_A(a)] \]

  \[\mu_R = (\mu_{\theta_1(R)}, \mu_{\theta_2(R)})\]

  An indexed natural transformation $\beta: F \to \Delta(B)$ for some set $B$ gives rise to the function $\nabla(\beta): \exists_Y F(Y) \to B$ defined as follows.

  \[\nabla(\beta)[\text{in}_A(a)] = \beta_A(a)\]

  To show this is well defined, since for any $R: A_0 \leftrightarrow A_1$, the square $\beta_R$ implies that if $a_0 \left[ F(R) \right] a_1$ then $\beta_{A_0}(a_0) = \beta_{A_1}(a_1)$.

  Note that this construction incorporates the quotienting by all relations that Reddy used to interpret abstract types [Red98, see also section 2.2].
• For any indexed functor $F: G \to \text{Span}(C)$ where $G_e$ is small and $C$ is co-complete, the parametric colimit of $F$ can be constructed in a manner analogous to the parametric limit (Theorem 3.2). Using the same auxiliary category $G^*$ (the total category of $G$) and functor $F^*$, the parametric colimit is given by $\exists_Y F(Y) = \text{colim}_X F^*(X)$.

• Any indexed functor $F: G \to \text{CPO}$ where $G$ is small has a parametric colimit. The construction presented below is an adaptation of the construction of colimits in the category $\text{CPO}_e$ due to Jung [Jun].

The objects of the colimit will be families of CPOs that respect the structure of morphisms and edges from $G$. A collection $\langle S_A \rangle$ indexed by the vertices of $G$ is called a compatible family of $F$ if

- for every vertex $A$, $S_A$ is a Scott-closed subset of $F(A)$ (that is, $S_A$ is downward closed and directed complete)
- for every morphism $f: A \to B$, $a \in S_A \iff F(f)a \in S_B$
- for every edge $R: A_0 \leftrightarrow A_1$, and any $a_0, a_1$ such that $a_0 \left[ F(R) \right] a_1$, $a_0 \in S_{A_0} \iff b \in S_{A_1}$

The collection $\text{Fam}$ of all compatible families of $F$ can be ordered using a component-wise subset ordering.

$$\langle S_A \rangle \sqsubseteq \langle T_A \rangle \iff S_A \subseteq T_A \text{ for all } A$$

We note that given any family $\langle N_A \rangle$ of arbitrary subsets of the $F(A)$s, there is a least compatible family $\langle S^N_A \rangle$ that covers $\langle N_A \rangle$ (that is, such that $N_A \subseteq S^N_A$ for all $A$). This is given by the component-wise intersection of all compatible families that cover $\langle N_A \rangle$. The least upper bound of a collection $\{ \langle S^f_A \rangle \}_f$ of compatible families is constructed as the least compatible family that covers the component-wise union $\langle \bigcup_f S^f_A \rangle$. Although we will not need them, greatest lower bounds can be constructed by component-wise intersections. Thus $\text{Fam}$ is more than just a CPO, it is a complete lattice. While the entire collection of all compatible families is, in general, too big to be the colimit of $F$, a sub-CPO will be identified to serve that role.

Given any vertex $A$ of $G$, and any element $a \in F(A)$, one can define the principal family $\langle a \rangle$ to be the least compatible family that covers the family whose $A$ component is the singleton $\{ a \}$ and all other families are empty. This mapping of $a$ to $\langle a \rangle$ defines a continuous function $\mu_A: F(A) \to \text{Fam}$. Since the collection of all principal families need
not be complete, we take it’s completion to give the parametric colimit of $F$. The completion can be described in stages, one for each ordinal $\alpha$.

$$P_0 = \{ \langle a \rangle \mid a \in F(A) \text{ for some } A \}$$

$$P_\alpha = \left\{ \bigcup Q \mid Q \subseteq P_\beta \text{ for some } \beta < \alpha \right\}$$

$$\exists_X F(X) = \bigcup_{\alpha \in \text{ORD}} P_\alpha$$

The continuous functions $\mu_A : F(A) \to \exists_X F(X)$ define a natural transformation since for every $f : A \to B$ and $a \in F(A)$, $\langle a \rangle = (F(f))a$. Similarly, for any $R : A_0 \leftrightarrow A_1$ and any $a_0, a_1$ such that $a_0 \left[ F(R) \right] a_1$, the fact that $\langle a_0 \rangle = \langle a_1 \rangle$ gives a unique square of the following shape.

$$F(A_0) \xrightarrow{\mu_{A_0}} \exists_X F(X)$$

$$F(R)$$

$$F(A_1) \xrightarrow{\mu_{A_1}} \exists_X F(X)$$

Defining $\mu_R$ to be the above square makes $\mu : F \to \exists_X F(X)$ an indexed natural transformation.

Given any indexed natural transformation $\tau : F \to \Delta(D)$, we define the factorization $\nabla(\tau) : \exists_X F(X) \to D$ by first defining the unique map $h : R_0 \to D$ such that $h \circ \mu_A = \tau_A$ as follows. For any $a \in F(A)$, we set $h(a) = \tau_A(a)$. Showing that this is well-defined makes use of an auxiliary construction. Given a CPO $X$ and a subset $T \subseteq X$, the downward closure of $T$ is denoted $\downarrow T$.

$$\downarrow T = \left\{ x \in X \mid \exists t \in T. x \subseteq t \right\}$$

For any $b \in F(B)$, the family $\langle \downarrow \tau_A^{-1}(\tau_B(b)) \rangle_A$ is a compatible family, and it contains $b$. Hence $\langle b \rangle \subseteq \langle \downarrow \tau_A^{-1}(\tau_B(b)) \rangle$. For any $c$ in the $C$-component of $\langle b \rangle$, it is the case that $\tau_C(c) \subseteq \tau_B(b)$. This shows not only that $h$ is well-defined (since $\langle c \rangle = \langle b \rangle$ implies $\tau_C(c) \subseteq \tau_B(b) \subseteq \tau_C(c)$), but also that $h$ is monotone. We define an extension of $h$ whose domain is all of $\exists_X F(X)$ by defining a sequence of extensions $h_\alpha : P_\alpha \to D$, as
indicated below.

\[
\begin{align*}
    h_0(x) &= h(x) \\
    h_\alpha \left( \bigsqcup Q \right) &= \bigsqcup_{q \in Q} h_\beta(q)
\end{align*}
\]

The limit of these approximations \( \nabla(\tau) = \bigsqcup_{q \in \text{ORD}} h_\alpha \) is the unique factorization \( \exists_X F(X) \rightarrow D \) of \( \tau \).

As a concrete example of colimits, consider \( F:\textsf{REL} \rightarrow \textsf{REL} \) given by \( F(X) = X \times (X \rightarrow X) \times (X \rightarrow \text{bool}) \). Then \( \exists_X F(X) \) is the type of a switch ADT having operations \textit{initial}, \textit{flick} and \textit{on}?. An intuitive representation of the switch ADT is given using the type \textsf{bool} with operations \textit{initial} = \text{false}, \textit{flick} = \lambda x: \textsf{Bool}.true and \textit{on}? = \lambda x: \textsf{Bool}.x. An alternative representation is given using the type \textsf{N} and operations \textit{initial}' = 0, \textit{flick}' = \lambda x: \textsf{N}.x + 1 and \textit{on}? = \lambda x: \textsf{N}.(x > 0). Even though the \textsf{N} representation keeps track of more information, namely the number of times the switch is flicked, the operations provide no access to this information. These two representations do actually represent the same element of \( \exists_X F(X) \) since there is a relation \( R: \textsf{Bool} \leftrightarrow \textsf{N} \) which is preserved by the operations.

\[
b \left[ R \right] n \iff (b = \text{false} \land n = 0) \lor (b = \text{true} \land n > 0)
\]

The duality between parametric limits and parametric colimits exhibits itself in that results about parametric limits translating into analogous statements about parametric colimits. For instance, the analogue of Theorem 3.5 is as follows.

**Proposition 3.8**

Suppose \( \text{Colim}: \textsf{G}^\textsf{H} \rightarrow \textsf{G} \) is the right adjoint to the diagonal functor, \( \Delta \). For every \( F:\textsf{H} \rightarrow \textsf{G} \), \( \text{Colim}(F) \) is the parametric colimit of \( F \), and for every \( \mathcal{F}: \textsf{E} \times \textsf{H} \rightarrow \textsf{G} \), \( K_{\text{Colim}(x)}: \textsf{E} \rightarrow \textsf{G} \) is the parametric colimit with parameters from \( \textsf{E} \) of \( \mathcal{F} \).

The proof is very similar to that of of Theorem 3.5.

### 3.2 Non-Variant Functors

Before presenting a general construction of models, there is another issue to be addressed. Not all categorical constructions of interest in programming semantics act on arrows so as to form a functor. The canonical counterexample is the exponent \( X \Rightarrow X \), which corresponds to set of endomorphisms.
This fails to be a functor in many categories, such as $\textbf{SET}$, since there is not a general way of producing a morphism $f \Rightarrow f : A \Rightarrow A \rightarrow B \Rightarrow B$ from a morphism $f : A \rightarrow B$. A well known approach to deal with this issue among category theorists is to consider bifunctors $\mathbf{C}^{\text{op}} \times \mathbf{C} \rightarrow \mathbf{D}$ and dinatural transformations between them (as opposed to functors $\mathbf{C} \rightarrow \mathbf{D}$ and natural transformations).

**Definition 3.9**

Given functors $F, G : \mathbf{C}^{\text{op}} \times \mathbf{C} \rightarrow \mathbf{D}$, a dinatural transformation $\tau : F \rightarrow G$ is a collection of morphisms $(\tau_X : F(X, X) \rightarrow G(X, X))_{X \in \text{obj}(\mathbf{C})}$ such that for every $f : A \rightarrow B$, the following hexagon commutes.

\[
\begin{array}{c}
F(id, f) \\
\downarrow \quad \quad \quad \downarrow \\
F(B, A) \\
\downarrow \quad \quad \quad \downarrow \\
F(f, id) \\
\end{array}
\begin{array}{c}
F(B, B) \\
\tau_B \\
G(B, B) \\
\tau_A \\
G(A, A) \\
\downarrow \quad \quad \quad \downarrow \\
G(f, id) \\
G(A, B) \\
G(id, f) \\
\end{array}
\]

Such a notion can be generalized to reflexive graph categories in the obvious manner - an indexed dinatural transformation, $\tau : F \rightarrow G$, consists of a pair of dinatural transformations, $\tau(v) : F_v \rightarrow G_v$ and $\tau(e) : F_e \rightarrow G_e$, which commute with the structural functors ($\partial_0, \partial_1$, and $1$).

With this point of view one can consider a variation of parametric limit so that rather than having a universal indexed natural transformation, one uses a universal dinatural transformation. This leads to the following definition.

**Definition 3.10**

For an indexed functor $F : \mathbf{G}^{\text{op}} \times \mathbf{G} \rightarrow \mathbf{H}$, a parametric end of $F$ consists of a vertex $\int_X F(X, X)$ of $\mathbf{H}$ and an indexed dinatural transformation $\omega : \Delta(\int_X F(X, X)) \rightarrow F$ universal in that for any indexed dinatural transformation $\beta : \Delta(A) \rightarrow F$, there is a unique $\Lambda(\beta) : A \rightarrow \int_X F(X, X)$ such that $\omega \circ \Delta(\Lambda(\beta)) = \beta$.

However, dinatural transformations or even indexed dinatural transformations do not make for an altogether satisfying notion of uniformity. The simplest reason is the well-known fact that the composite of dinatural transformations need not be dinatural. The intuitive understanding of uniformity, on the other hand, is compositional. If $t_A$ does the same thing for all $A$ and $u_A$ does the same thing for all $A$, then $u_A \circ t_A$ should do the same thing at each $A$ as well.
Another approach which is typical in programming semantics that side-steps the issue of type constructors which don’t form functors is to ignore the morphisms altogether. This exhibits itself as considering functors out of discrete categories, such as the so-called environment model of a typed \( \lambda \)-calculus with type variables using a CCC. A similar approach can be taken using reflexive graph categories instead of categories. For ease of exposition, the following terminology will be used.

Definition 3.11
A reflexive graph category \( \mathbf{G} \) is discrete if both the vertex category \( \mathbf{G_v} \) and the edge category \( \mathbf{G_e} \) are discrete categories. In other words, \( \mathbf{G} \) is discrete if the only morphisms are identity morphisms and the only squares are identity squares.

For any reflexive graph category \( \mathbf{G} \), the discrete reflexive subgraph \( |\mathbf{G}| \) consists of the same vertices and edges as \( \mathbf{G} \), but only the identity morphisms and squares. The three functors \( \partial_0, \partial_1 \) and \( \mathbf{I} \) are the obvious restrictions of the corresponding functors from \( \mathbf{G} \).

Definition 3.12
A non-variant functor from one reflexive graph category \( \mathbf{G} \) to another \( \mathbf{H} \) is an indexed functor \( F: |\mathbf{G}| \to \mathbf{H} \).

It still makes sense to talk about indexed natural transformations between non-variant functors, since they are indexed functors. It should be noted that since the only morphisms and squares of \( |\mathbf{G}| \) are identities, naturality conditions only apply to identities, and hence are vacuous.

Since \( |\mathbf{G}| \) is a sub-reflexive graph category of \( \mathbf{G} \), it is obvious that any indexed functor \( F: \mathbf{G} \to \mathbf{H} \) defines a non-variant functor \( \llbracket F \rrbracket: \mathbf{G} \to \mathbf{H} \) by restriction. Since the only morphisms and squares in the image of a non-variant functor are identities, every non-variant functor \( G: \mathbf{G} \to \mathbf{H} \) gives rise to a non-variant functor into the discrete reflexive graph category, \( \llbracket G \rrbracket: \mathbf{G} \to |\mathbf{H}| \). This makes it notationally easy to define the composite of non-variant functors \( F: \mathbf{G} \to \mathbf{H} \) and \( G: \mathbf{H} \to \mathbf{K} \) by the composite of indexed functors \( G \circ F|: |\mathbf{G}| \to |\mathbf{H}| \to |\mathbf{K}| \).

Non-variant functors can be used to produce models of the predicative calculus as we describe in the next section.

### 3.3 Models of Polymorphism

Parametric limits and colimits are categorical constructions which correspond to the parametric interpretations for polymorphic functions and ab-
strict types in the model of 2.2. In this section, we describe a construction of models of the predicative calculus using reflexive graph categories. The 2-categorical description of this model is not new. In this setting, the explicit description shows how the concept of relational parametricity is included. Parametric limits (the right adjoint to the diagonal in this setting) are used to limit the interpretation of polymorphic types to only elements which preserve all edges. This more explicit description of parametric limits will be handy in the subsequent chapters where the consequences of parametricity are addressed.

**Definition 3.13**

An RG setting for the predicative calculus consists of two reflexive graph categories $\mathbf{G}$ and $\mathbf{H}$ such that $\mathbf{G}$ is a Cartesian closed sub-reflexive graph category of the Cartesian reflexive graph category $\mathbf{H}$ and both the parametric limit and colimit indexed functors $\text{Lim}, \text{Colim}: \mathbf{H}^{G} \rightarrow \mathbf{H}$ exist.

We point out that we use the convention that a Cartesian closed reflexive graph category includes a particular choice of the required structure. That is, there is a designated vertex $1$ (the terminal object) as well as indexed functors $\times: \mathbf{G} \times \mathbf{G} \rightarrow \mathbf{G}$ and $\Rightarrow: \mathbf{G}^{op} \times \mathbf{G} \rightarrow \mathbf{G}$ satisfying the usual properties. We shall frequently use indexed natural transformations from the Cartesian closed structure such as the application ap: $(A \Rightarrow B) \times A \rightarrow B$, right cancelation $\rho: A \times 1 \rightarrow A$, left cancelation $\lambda: 1 \times A \rightarrow A$ and symmetry $\sigma: A \times B \rightarrow B \times A$ to denote particular components of the respective transformations without explicitly mentioning the vertices when this information can be inferred.

An RG setting gives a model of the predicative polymorphic lambda calculus where the syntax is interpreted in the following manner.

<table>
<thead>
<tr>
<th>the interpretation of:</th>
<th>is given by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>type judgments</td>
<td>non-variant functors $[\eta \vdash \alpha]: \mathbf{G}^{n} \rightarrow \mathbf{H}$</td>
</tr>
<tr>
<td>simplicity judgments</td>
<td>non-variant functors $[\eta \vdash \tau \text{ simple}]: \mathbf{G}^{n} \rightarrow \mathbf{G}$</td>
</tr>
<tr>
<td>term judgments</td>
<td>indexed natural transformations $[\eta ; \Gamma \vdash M : \phi] : [\eta \vdash \Gamma] \rightarrow [\eta \vdash \phi]$.</td>
</tr>
</tbody>
</table>

(Here $n$ is the number of type variables in $\eta$.) In situations where the typing context and context is apparent or unimportant, the notation of the interpretation may be abbreviated by using only the expression rather than the whole judgment (for example, using $[\tau]$ rather than $[\eta \vdash \tau]$).

The interpretations of type judgments involving simple types are built up from projection non-variant functors (for type variables) by composing
with non-variant functors arising from the Cartesian closed structure of $\mathbf{G}$.

The interpretation of a type judgment $\eta \vdash \tau$ where $\tau$ is a simple type is given by the inclusion of $\mathbf{G}$ into $\mathbf{H}$.

\[
[\eta \vdash \tau] \bar{A} = [\eta \vdash \text{simple}] \bar{A} \\
[\eta \vdash \tau] \bar{R} = [\eta \vdash \text{simple}] \bar{R}
\]

This interpretation of simple types is standard, as are the interpretations of a context as the product of the types of it’s variables and the interpretation of terms with simple types by composition with the units and co-units of the adjunctions. The formal definition is shown in Table 3.1 (page 52).

We use angled brackets to denote both the pairing of two non-variant functors $\mathcal{F}, G: \mathbf{G}^n \to \mathbf{G}$ into a non-variant functor $\langle \mathcal{F}, G \rangle: \mathbf{G}^n \to \mathbf{G} \times \mathbf{G}$ as well as the pairing of two indexed natural transformations $\eta: F \to G$ and $\mu: F \to H$ into an indexed natural transformation with components $\langle \eta, \mu \rangle_X: F(X) \to G(X) \times H(X)$. Since these are applied to different kinds of inputs, it should be possible to determine which is intended. An upper case $\Pi_j$ is used to denote the $j$th projection indexed functor $\Pi_j: \mathbf{G}^n \to \mathbf{G}$ as opposed to a lower case $\pi_j$ for the $j$th projection indexed natural transformation with components $(\pi_j)_X: F_1(X) \times \cdots \times F_n(X) \to F_j(X)$. The inclusion of $\mathbf{G}$ into $\mathbf{H}$ as well as the isomorphism $|\mathbf{G}| \to |\mathbf{G}^{op}|$ are left implicit.

There is a morphism $i^*_\tau$ of $\mathbf{G}^{\mathbf{G}^n}$ that provides a canonical isomorphism between $[\eta \vdash \Gamma] \times [\eta \vdash \tau_1]$ and $[\eta \vdash \Gamma, x: \tau_1]$ for any $\Gamma$ and $\tau_1$. If $\Gamma$ is non-empty, then $i^*_\tau$ is the identity morphism, but something must be done in the case where $\Gamma = \emptyset$, since $[\eta \vdash \Gamma, x: \tau_1] = [\eta \vdash \tau_1]$. Since $[\eta \vdash \emptyset] = \Delta(1)$, which is the terminal object in $\mathbf{G}^{\mathbf{G}^n}$, we use $\lambda \vdash \emptyset = \Lambda: \Delta(1) \times [\eta \vdash \tau_1] \to [\eta \vdash \tau_1]$.

The interpretation of polymorphic types is slightly different from the compositional approach to interpreting simple types. The parametric limit functor $(\text{Lim})^{\mathbf{G}^n}: \mathbf{H}^{\mathbf{G}^{n+1}} \to \mathbf{H}^{\mathbf{G}^n}$ is used directly on the interpretation of $\eta, X \vdash \phi$ to produce the interpretation of $\eta \vdash \forall X. \phi$. The unique factorization of $\beta: [\Gamma] \to [\eta, X \vdash \phi]$ from the definition of the parametric limit corresponds precisely to the $\beta$- and $\eta$-equalities for the $\forall$ type constructor. Similarly, the parametric colimit functor $(\text{Colim})^{\mathbf{G}^n}: \mathbf{H}^{\mathbf{G}^{n+1}} \to \mathbf{H}^{\mathbf{G}^n}$ is used to interpret abstract types, $\eta \vdash \exists X. \phi$.

There are numerous examples of RG settings. Reynolds’ model (section 2.2) arises from using the reflexive graph category $\mathbf{REL}$ and a small, Cartesian closed, full sub-reflexive graph category $\mathbf{G}$ of small sets and relations between them. (In section 2.2, the category $\mathbf{C}$ was used, which we take to be the vertex category of $\mathbf{G}$). The definition given in section 2.2
\begin{align*}
[X_1, \ldots, X_n \vdash X_j] &= |\Pi_j| \\
[\eta \vdash \tau_1 \Rightarrow \tau_2] &= \{ \Rightarrow \circ ([\tau_1], [\tau_2]) \} \\
[\eta \vdash \forall X. \phi] &= (\text{Lim})^{\mathbb{F}_n}(\eta, X \vdash \phi) \\
[\eta \vdash \exists X. \phi] &= (\text{Colim})^{\mathbb{F}_n}(\eta, X \vdash \phi) \\
[\eta \vdash \emptyset] &= |\Delta(1) \\
[\eta \vdash \Gamma, x: \phi] &= \times \circ ([\Gamma], [\phi]) \\
[\eta; x_1: \phi_1 \ldots x_m: \phi_m \vdash x_i: \phi_i] &= \pi_i \\
[\eta; \Gamma \vdash \lambda x: \tau_1.M: \tau_1 \Rightarrow \tau_2] &= \text{curry}(\eta; \Gamma, x: \tau_1 \vdash M: \tau_2 \circ \tau_1^\Gamma) \\
[\eta; \Gamma \vdash MN: \tau_2] &= \alpha p \circ \langle [M: \tau_1 \Rightarrow \tau_2], [N: \tau_1] \rangle \\
[\eta; \Gamma \vdash \Lambda X.M: \forall X.\phi] &= \Lambda([\eta, X; \Gamma \vdash M: \phi]) \\
[\eta; \Gamma \vdash M[\tau]; \phi[\tau/X]]_{\tilde{\lambda}} &= \omega_{\tilde{\lambda},[\tilde{\lambda}]} \circ [M: \forall X.\phi]_{\tilde{\lambda}} \\
[\eta; \Gamma \vdash \text{pack } \tau \text{ with } M: \exists X.\phi]_{\tilde{\lambda}} &= \mu_{\tilde{\lambda},[\tilde{\lambda}]} \circ [M: \exists X.\phi]_{\tilde{\lambda}} \\
[\eta; \Gamma \vdash \text{open } M \text{ as } (X,\phi \text{ in } N):\phi'] &= \nabla([\eta, X; \Gamma, x: \phi \vdash N: \phi']) \circ \text{id}_{\tilde{\lambda}}, [M: \exists X.\phi] \\
\end{align*}

Table 3.1: Interpretation of Polymorphic Lambda Calculus

only differs from the one presented in this section by using explicit descriptions of the relevant non-variant functors. For instance, the interpretation \([\eta \vdash \forall X.\phi] = \text{Lim}\mathbb{F}_n([\eta, X \vdash \phi])\) was given explicitly for vertices \(\tilde{\lambda}\) and edges \(\tilde{R}\) in section 2.2.

\begin{align*}
[\eta \vdash \forall X.\phi]_{\tilde{\lambda}} &= \left\{ p \in \Pi_{B \in S}[\eta, X \vdash \phi](A_1, \ldots, A_n, B) \mid \forall R: B_0 \leftrightarrow B_1 \\
&\quad \text{p}_{B_0} \text{ and } p_{B_1} \text{ are related by} \\
&\quad [\eta, X \vdash \phi](\Delta_{A_1}, \ldots, \Delta_{A_n}, R) \right\} \\
\text{for every } R': B_0 \leftrightarrow B_1, \\
p_{B_0} \left[ [\eta, X \vdash \phi](R_1, \ldots, R_n, R') \right]_{p_{B_1}'} &\Rightarrow \end{align*}

These definitions agree with parametric limits with parameters in \textbf{REL}
(as given in section 3.1, page 40) since the only morphisms in \(|G|\) are iden-
\[ \forall Y [\eta, X \vdash \phi](A_1, \cdots, A_n, Y) = \begin{cases}
\langle t_Y \rangle_Y \in \Pi_Y [\eta, X \vdash \phi](A_1, \cdots, A_n, Y) & \forall R : Y_0 \leftrightarrow Y_1 \text{ and } \forall f : Y \rightarrow Y',
\end{cases}
\]
\[ t_{Y_0} \left[ [\eta, X \vdash \phi](I_{A_1}, \cdots, I_{A_n}, R) \right] t_{Y_1} \text{ and } t_{Y'} = [\eta, X \vdash \phi](\text{id}_{A_1}, \cdots, \text{id}_{A_n}, f) (t_Y) \]
\[ \langle t_Y \rangle_Y \left[ [\eta, X \vdash \phi](R_1, \cdots, R_n, Y) \right] \langle u_Y \rangle_Y \iff \text{ for all } R : Y_0 \leftrightarrow Y_1, \]
\[ t_{Y_0} \left[ [\eta, X \vdash \phi](R_1, \cdots, R_n, R) \right] u_{Y_1} \]

Similar observations are made for all the constructions. Reynolds’ interpretation (defined in section 2.2) agrees with the interpretation defined in table 3.1 for the special case where \( H = \text{REL} \).

There are many other examples of RG-settings that one could use to model the predicative calculus. One could also use \textbf{CPO} and a small, Cartesian closed sub-reflexive graph category of it to give an RG-setting. If \( C \) is a small, Cartesian closed sub-category of \textbf{SET}, then \textbf{Span}(C) and \textbf{Span}(SET) is also an RG-setting. Similarly \textbf{JMS}(C) & \textbf{JMS}(SET) and \textbf{Sp}(C) & \textbf{Sp}(SET) each determine an RG-setting for the predicative calculus.

Reynolds’ use of relations was to trim down the interpretation of polymorphic types to those collections which preserve relations. The use of indexed natural transformations (intuitively collections of morphisms which preserve edges) and parametric limits (universal indexed natural transformations) mimics Reynolds’ trimming down to collections which preserve relations. This is evident in the description of the parametric limit in reflexive graph categories such as \textbf{CPO}, for instance.

Consider an even more concrete example, the interpretation of \( \forall X.X \rightarrow X \) in the \textbf{CPO} RG setting (using a Cartesian closed sub-reflexive graph category \( C \)).

\[ [\forall X.X \rightarrow X] = \left\{ \langle t_A \rangle : \text{ for all } R \text{ in } C_e, t_{(\partial_0(R))} [R \Rightarrow R] t_{(\partial_1(R))} \right\} \]

Since \( t_{A_0} [R \Rightarrow R] t_{A_1} \) happens if and only if there is a square of shape

![Diagram](image)

53
(and if so, that square is unique), $[\forall X. X \Rightarrow X]$ is isomorphic to the collection of all indexed natural transformations $I n \Rightarrow I n$ (where $I n$ is the inclusion $C \rightarrow C P O$). For any CPO $A$ of $C$ and any $a \in A$, the relation $Only_a : A \leftrightarrow A$ relating $a$ to itself and nothing else is a complete relation. Therefore, for any $\langle t_B \rangle \in [\forall X. X \Rightarrow X]$, there is a square of the following shape.

$$
\begin{array}{c}
A \\
\uparrow \text{Only}_a \\
A
\end{array}
\begin{array}{c}
A \\
\downarrow \text{Only}_a \\
A
\end{array}
\begin{array}{c}
A \\
\rightarrow t_A \\
A
\end{array}
\begin{array}{c}
A \\
\rightarrow t_A \\
A
\end{array}

$$

This implies that $t_A(a) = a$. Since $a$ was arbitrary, $t_A$ must be the identity on $A$, and since $A$ was arbitrary, every $t_B$ must be an identity. Thus there is only one element of $[\forall X. X \Rightarrow X]$.

This example achieves the ideal — cutting down the interpretation to only the intuitively uniform things. The family of identity functions is uniform, since it does the same thing (namely echo the input) at every type. It is the only intuitively uniform family of functions $A \Rightarrow A$, indeed the only closed term of type $\forall X. X \Rightarrow X$ in the predicative calculus (or System F, for that matter). It is not the case that $[\emptyset \vdash \forall X. X \Rightarrow X]$ contains only the family of identity functions in all models of the predicative calculus, even of all models which arise from RG settings. Additional axioms on reflexive graph categories which ensure a stronger notion of uniformity will be the focus of Chapter 4.

### 3.4 Modeling Impredicativity

We now address the matter of producing models of System F, rather than the predicative calculus. We define an RG setting for System F to be a reflexive graph category with sufficient structure for us to model System F. We describe one such RG setting for System F using PER by shifting to reflexive graph categories internal the category of $\omega$-sets.

Recall that the difference between System F and the predicative calculus is that System F does not have a distinction between simple types and types. Therefore the above construction of a model for the predicative calculus can be used to produce a model of System F if the distinction between interpretations of simple types and that of types in general is removed, that is, if $G = H$. Since System F does not include existential types, we do not
require parametric colimits.

**Definition 3.14**

An RG setting for System F consists of a Cartesian closed parametricity graph $G$ such that the parametric limit indexed functor $\text{Lim}: G^{\mathbb{G}} \to G$ exists.

Using such an RG setting, the construction of a model given in the previous section will produce a model of System F. Finding a non-trivial reflexive graph category which is Cartesian closed and possessing parametric limit functor $\text{Lim}: G^{\mathbb{G}} \to G$ is not as easy as for the predicative scenario. This difficulty is essentially the same foundational problem that was mentioned regarding set based models of System F in section 2.1.

Categorical models for System F arise from considering categories internal to categories other than $\text{SET}$. A similar approach can be taken with reflexive graph categories. For instance, a model can be given using $\text{PER}$ enriched to be a reflexive graph category internal to $\omega\text{-SET}$ [LM91]. (One could alternatively used the effective topos [Hyl82] as the ambient setting, but $\omega\text{-SET}$ is considered here for ease of exposition.) The basic idea is that the intersection of PERs $\bigcap_A F(A)$ is universal with respect to uniformly realized natural transformations $\tau: F \to \Delta(B)$ (that is, natural transformations having a single partial recursive function $\phi_k$ realizes $\tau_A$ for all $A$). So one shifts to a 2-category where the 2-cells are uniformly realized.

Observe that the criteria for a reflexive graph category to be an RG-setting are 2-categorical. Thus, there is an immediate analogue of RG-setting using internal reflexive graph categories. A $\text{RG-setting internal to } C$ for System F consists of a Cartesian closed reflexive graph category $G$ internal to $C$ with a parametric limit indexed functor $\text{Lim}: G^G \to G$. We shall be considering an RG-setting internal to the category of $\omega$-sets.

Much of the material in this section comes from Longo and Moggi’s construction of a model for System F using a category of PERs internal to $\omega\text{-SET}$ [LM91]. This construction using saturated relations between PERs has been presented [BAC95], although not constructed in a categorical fashion as functors (of some sort) and transformations. We review this material, with a description in terms of reflexive graph categories internal to $\omega\text{-SET}$.

We presume that an encoding of natural numbers $k$ to partial recursive functions $\phi_k: \mathbb{N} \to \mathbb{N}$ has been fixed. Additionally, a fixed recursive bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is fixed, where we use $\langle k, k' \rangle$ to denote the natural number corresponding to the pair of natural numbers $(k, k')$ via that bijection.
An \( \omega \)-set is a set \( X \), together with an assignment of a non-empty set \( r^X(x) \subseteq \mathbb{N} \) of codes for every \( x \in X \). (The superscript on \( r \) may be suppressed when the \( \omega \)-set in question can be inferred.) We say an \( \omega \)-set \( (X, r^X) \) is a sub-\( \omega \)-set of \( (Y, r^Y) \) if \( X \subseteq Y \) and \( r^X \) is the restriction of \( r^Y \) (that is, \( r^X(x) = r^Y(x) \) for all \( x \in X \)). A map between \( \omega \)-sets \( (X, r^X) \) and \( (Y, r^Y) \), called an \( \omega \)-function, is a function \( f : X \rightarrow Y \) such that there exists a natural number \( k \) such that the \( k \)-th partial recursive function \( \phi_k : \mathbb{N} \rightarrow \mathbb{N} \) maps \( r^X(x) \) into \( r^Y(f(x)) \) for every \( x \in X \). Any such natural number \( k \) is said to track the \( \omega \)-function \( f \), while the partial recursive function \( \phi_k \) is said to realize \( f \).

A reflexive graph \( \omega \)-category has four \( \omega \)-sets — for vertices, morphisms, edges and squares. Additionally, there are 14 \( \omega \)-functions giving the source, target, identity and composition in both the vertex and edge categories as well as the object and arrow portions of the structural functors \( \partial_0, \partial_1 \) and \( I \). These \( \omega \)-functions are required to satisfy the usual axioms of reflexive graph categories. We say a reflexive graph \( \omega \)-category \( G \) is a sub-reflexive graph \( \omega \)-category of \( H \) if each of the 4 \( \omega \)-sets for \( G \) is a sub-\( \omega \)-set of the corresponding \( \omega \)-set of \( H \) and the \( \omega \)-functions of \( G \) are restrictions of the corresponding \( \omega \)-functions of \( H \) in the usual manner.

As mentioned above, \( \text{PER} \) can be enriched into a reflexive graph \( \omega \)-category, which we shall denote \( \text{PER}_\omega \). The codes for the components of \( \text{PER}_\omega \) are as follows:

- for \( \text{PERs}, r(A) = \mathbb{N} \)
- for realizable functions, \( r(f) = \{ k \mid \phi_k \text{ realizes } f \} \)
- for saturated relations, \( r(R) = \mathbb{N} \)
- for squares, \( r(f, f') = \{ \langle k, k' \rangle \mid \phi_k \text{ realizes } f \text{ and } \phi_{k'} \text{ realizes } f' \} \)

It should be apparent that the functions completing the reflexive graph category structure of \( \text{PER} \) are \( \omega \)-functions.

Indexed \( \omega \)-functors consist of four \( \omega \)-functions — mapping vertices to vertices, morphisms to morphism, edges to edges and squares to squares — that commute with the \( \omega \)-functions of the reflexive graph \( \omega \)-categories in the usual manner. Note that the \( \omega \)-function on squares uniquely determines the other three \( \omega \)-functions of the indexed \( \omega \)-functor by composition with \( \omega \)-functions from the reflexive graph \( \omega \)-categories. Hence, we say that \( k \) tracks the indexed \( \omega \)-functor \( F \) if and only if \( k \) tracks the \( \omega \)-function giving the action of \( F \) on squares.

An indexed natural \( \omega \)-transformation \( \tau : F \rightarrow G \) between indexed \( \omega \)-functors \( G \rightarrow H \) consists of two \( \omega \)-functions — one from vertices of \( G \)
to morphisms of $H$ (the vertex portion) and the other from edges of $G$ to squares of $H$ (the edge portion). These $\omega$-functions are required to interact with the $\omega$-functions of the $\omega$-categories in the usual manner for indexed natural transformations. Since the $\omega$-function on edges uniquely determines the $\omega$-function on vertices, we say $k$ tracks $\tau$ if and only if $k$ tracks the $\omega$-function on edges portion of $\tau$.

Consider any indexed natural $\omega$-transformation $\tau$ between indexed $\omega$ functors $G \to \text{PER}_\omega$ where $G$ is any sub-reflexive graph $\omega$-category of $\text{PER}_\omega$. Suppose $k$ tracks the vertex portion of $\tau$. All natural numbers are codes for all vertices, in particular $1$ is a code for all vertices. Hence $\phi_k(1)$ is a code for $\tau_A$ for every vertex $A$. (Note $\phi_k(1)$ need not be a code for $\tau_R$, since squares are pairs of functions as compared to morphisms which are lone functions. But it is possible to recursively construct a single code $k'$ from $\phi_k(1)$ such that $k'$ is a code for $\tau_R$ for every edge $R$ since squares are simply pairs of morphisms.) Since $\phi((\phi_k(1))$ realizes $\tau_A$ for all $A$, we say that $\tau$ is uniformly realized.

A particular sub-reflexive graph $\omega$-category we shall be using is the discrete reflexive graph $\omega$-category $[\text{PER}_\omega]$ that has only identity morphisms and squares (but the same vertices, edges, and codes as $\text{PER}_\omega$). Similar to before, we shall use the term non-variant $\omega$-functor on $\text{PER}_\omega$ to refer to an indexed $\omega$-functor $[\text{PER}_\omega] \to \text{PER}_\omega$.

Reflexive graph $\omega$-categories, indexed $\omega$-functors and indexed natural $\omega$-transformations form a 2-category $\text{RG}[\omega\text{-}\text{SET}]$. Much of the discussion about the two category $\text{RG}$ from earlier carries over to the 2-category $\text{RG}[\omega\text{-}\text{SET}]$ in the apparent manner. For instance, the product of reflexive graph $\omega$-categories is given component-wise. Codes for pairs are given in the apparent manner using the bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, such as :

$$r^{G_v \times H_v}(A, B) = \langle r^{G_v}(A), r^{H_v}(B) \rangle$$

To describe the exponent in $\text{RG}[\omega\text{-}\text{SET}]$ as we did the functor graph in $\text{RG}$, Hom$(x, -)$ should be enriched to a reflexive graph $\omega$-category for each object $x$ of $B$. We assign disjoint sets of codes to each arrow of Hom$(x, w)$. For the reflexive graph category $E$, we enrich it to a reflexive graph $\omega$-category $E_\omega$ by using the following codes.

- $r(s) = \{0\}$ $r(t) = \{1\}$
- $r(id_s) = \{0\}$ $r(id_t) = \{1\}$
- $r(I_s) = \{0\}$ $r(I_t) = \{1\}$ $r(E) = \{2\}$
- $r(id_{I_s}) = \{0\}$ $r(id_{I_t}) = \{1\}$ $r(id_{E}) = \{2\}$

57
The sets of codes for different vertices of $E_\omega$ are disjoint. This fact leads to the following observation concerning an indexed natural transformation $\tau: F \to G$ between indexed $\omega$-functors $E_\omega \times |\text{PER}_\omega| \to \text{PER}_\omega$. (Such a $\tau$ will be a square of the functor $\omega$-graph.) Even though there is a partial recursive function realizing the vertex portion of $\tau$, the $\tau_{[s,A]}$ and $\tau_{[t,A]}$ components (for any $\text{PER} A$) need not have the same realizer. Thus, indexed natural $\omega$-transformations $\partial_0(\tau)$ and $\partial_1(\tau)$ given by fixing the $E_\omega$ component of $\tau$ at $s$ and $t$ respectively do not both have to be realized by the same partial recursive function. (Although both $\partial_0(\tau)$ and $\partial_1(\tau)$ will be uniformly realized.)

The exponent in $\text{RG}[\omega\text{-SET}]$ consists of $\omega$-functors, similar to the exponent in $\text{RG}$. For reflexive graph $\omega$-categories $G$ and $H$, the functor $\omega$-graph $H^G$ is as follows.

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Indexed $\omega$-functors $F: G \to H$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>with $r(F) = {k \mid k \text{ tracks } F}$</td>
</tr>
<tr>
<td>Morphisms</td>
<td>Indexed natural $\omega$-transformations between vertices</td>
</tr>
<tr>
<td></td>
<td>with $r(\tau) = {k \mid k \text{ tracks the vertex portion of } \tau}$</td>
</tr>
<tr>
<td>Edges</td>
<td>Indexed $\omega$-functors $\mathcal{F}: E_\omega \times G \to H$</td>
</tr>
<tr>
<td></td>
<td>with $r(\mathcal{F}) = {k \mid k \text{ tracks } \mathcal{F}}$</td>
</tr>
<tr>
<td>Squares</td>
<td>Indexed natural $\omega$-transformations between edges</td>
</tr>
<tr>
<td></td>
<td>with $r(\tau) = {k \mid k \text{ tracks the vertex portion of } \tau}$</td>
</tr>
</tbody>
</table>

Just as for the exponent in $\text{RG}$, $\partial_0$ and $\partial_1$ are given by fixing the $E_\omega$ component at $s$ and $t$, respectively. The $\omega$-functor $I(F)$ ignores the $E_\omega$ component of its input. The Cartesian closed structure of $\text{RG}[\omega\text{-SET}]$ is completely analogous to that of $\text{RG}$. For instance, the diagonal indexed $\omega$-functor $\Delta: H \to H^G$ produces constant indexed $\omega$-functors and indexed natural $\omega$-transformations.

In order for $\text{PER}_\omega$ to be an $\text{RG}$-setting internal to $\omega\text{-SET}$ for System $F$, we need to produce the parametric limit indexed $\omega$-functor. The index $\omega$-functor $\text{Lim: PER}_\omega^{\text{PER}-\text{PER}} \to \text{PER}_\omega$ is defined on vertices $F$, morphisms $\tau: F \to G$, edges $\mathcal{F}$ and squares $\gamma: \mathcal{F} \to \mathcal{G}$ as follows.

\[
\text{Lim}(F) = \{ (n,m) \mid \forall R. n \left[ F(R) \right] n \land n \left[ F(R) \right] m \land m \left[ F(R) \right] m \} \\
\text{Lim}(\tau) = \text{the realizable function } \text{Lim} F \to \text{Lim} G \text{ realized by a realizer of } \tau_A \text{ for all PERs } A \\
\text{Lim}(\mathcal{F}) = \{ (n,m) \mid \forall R. n \left[ \mathcal{F}(E,R) \right] m \} \\
\text{Lim}(\gamma) = \text{the square } (\text{Lim}(\partial_0(\gamma)), \text{Lim}(\partial_1(\gamma)))
\]
A first attempt at producing the parametric limit of an indexed \( \omega \)-functor \( F: \mathbf{G} \rightarrow \mathbf{H} \) was to use the intersection \( \bigcap_R F(R) \) of saturated relations. There have been some results to suggest that this intersection \( \bigcap_R F(R) \) might be a PER rather than a more general relation [Has91, HRR90]. Lacking a proof that \( \bigcap_R F(R) \) is a PER, the parametric limit of \( F \) is given as the largest PER contained in \( \bigcap_R F(R) \).

\[
\left\{ (n, m) \mid \forall \text{edges } R. \ n \left[ F(R) \right] n \land n \left[ F(R) \right] m \land m \left[ F(R) \right] m \right\}
\]

It is not hard to see that every PER contained in \( \bigcap_R F(R) \) is contained in \( \text{Lim}(F) \). The relation \( \text{Lim}(F) \) on \( \mathbb{N} \) is transitive, since there is the following implication for every edge \( R \).

\[
n \left[ \text{Lim}(F) \right] m \land m \left[ \text{Lim}(F) \right] l \implies n \left[ F(R) \right] m \land m \left[ F(\partial_l(R)) \right] l
\]

Since \( F(R) \) is saturated, we can deduce \( n \left[ F(R) \right] l \). Showing that \( \text{Lim}(F) \) is symmetric is similar, since \( n \left[ \text{Lim}(F) \right] m \) implies the following for all edges \( R \).

\[
m \left[ F(R) \right] m \quad \text{and} \quad n \left[ F(\partial_l(R)) \right] m
\]

Since \( \mathcal{F}(\partial_l(R)) \) is a PER, it is symmetric, hence \( m \left[ F(\partial_l(R)) \right] n \). Therefore the saturation of \( F(R) \) implies \( m \left[ F(R) \right] n \).

The definition of \( \text{Lim}(\tau) \) makes use of the fact that \( \tau \) is uniformly realized. If \( \phi_k \) realizes \( \tau_A \) for all PERs \( A \), then for any \( n \left[ \text{Lim}(F) \right] m \), it is the case that \( \phi_k(n) \left[ \text{Lim}(G) \right] \phi_k(m) \). (Recall that for all \( R \), \( n \left[ F(R) \right] m \) implies \( \phi_k(n) \left[ G(R) \right] \phi_k(m) \).) Hence \( \phi_k \) does realize a function from \( \text{Lim}(F) \) to \( \text{Lim}(G) \).

The dual construction is for parametric colimits and it is essentially given by the union. Since the union of saturated relations need not be a PER, one needs to take the symmetric, transitive closure to get a PER (the least PER containing the union). We use \( \text{hub}(R) \) to denote the symmetric, transitive closure of the relation \( R \).

\[
\begin{align*}
\text{Colim}(F) & = \text{hub}(\bigcup_R F(R)) \\
\text{Colim}(\tau) & = \text{the function Colim}(F) \rightarrow \text{Colim}(F') \text{ realized by a realizer of } \tau_A \text{ for all PERs } A \\
\text{Colim}(\mathcal{F}) & = \text{sat}(\bigcup_R \mathcal{F}(E, R)) \\
\text{Colim}(\tau) & = \text{the square realized by a realizer of } \tau
\end{align*}
\]
In those cases where the symmetric, transitive closure is needed to get a
PER, this may result in $\bigcup_R \mathcal{F}(E, R)$ not being saturated with respect to
Colim$\mathcal{F}(s, -)$ and Colim$\mathcal{F}(t, -)$. For this reason, the colimit of an edge $\mathcal{F}$
must relate additional elements as well.

\[
\begin{align*}
n \left[ \text{sat} \left( \bigcup_R \mathcal{F}(E, R) \right) \right] m & \iff \exists n', m' \text{ such that} \\
(n \left[ \text{Colim} \mathcal{F}(s, -) \right] n') \land (n' \left[ \bigcup_R \mathcal{F}(E, R) \right] m') \land (m' \left[ \text{Colim} \mathcal{F}(t, -) \right] m)
\end{align*}
\]

**Theorem 3.15**

The indexed functor $\text{Lim}$ above is the right adjoint to the diagonal indexed
functor $\Delta : \text{PER} \to \text{PER}^{\text{PER}}$. Dually, Colim is the left adjoint.

**Proof.** The co-unit of the adjunction has components given by the projection
indexed natural $\omega$-transformation $\epsilon_F = \omega : \text{Lim}(F) \to F$. Each vertex
component $\omega_A$ is realized by the identity function. The identity function
does map $\text{Lim}(F)$-related numbers to $F(A)$-related numbers, so there is a
realizable function realized by the identity function. Since $n$ and $m$ are related
by $F(R)$ whenever $n \left[ \text{Lim}(F) \right] m$, setting $\omega_R = (\omega_{\delta_0(R)}, \omega_{\delta_1(R)})$
gives an appropriate square for every saturated relation $R$.

Since each $\epsilon_F$ is uniformly realized by the identity function, the vertex
portion of $\epsilon$ is an $\omega$-function (tracked by the code for a constant function).
For each edge $\mathcal{F}$, the indexed natural $\omega$-transformation $\epsilon_F$ is also has every
vertex component realized by the identity function. Hence, $\epsilon$ is an indexed
natural $\omega$-transformation.

The unit $\eta : \text{ID} \to \text{Lim} \circ \Delta$ is the identity natural $\omega$-transformation,
which is reasonable since direct computation shows that $\text{Lim} \circ \Delta$ is the iden-
tity indexed $\omega$-functor.

Showing that $\eta$ and $\epsilon$ are the unit and co-unit of an adjunction boils
down to showing that $\omega \circ \text{Lim}(\tau) = \tau$ for any $\tau : \Delta(A) \to F$. This equality
holds since each $\omega_A$ is realized by the identity and a realizer of $\text{Lim}(\tau)$ is a
realizer of $\tau_A$ for each $A$.

Showing Colim is the left adjoint to $\Delta$ proceeds similarly, with injections
$\mu : F \to \Delta(\text{Colim}(F))$ uniformly realized by the identity function and unique
factorizations of $\tau : F \to \Delta(A)$ realized by a realizer for each $\tau_A$.

**Corollary 3.16**

$\text{PER}_\omega$ is a parametricity setting for System $F$, hence determines a model
of System $F$.

In addition to parametric limits (Theorem 3.15), this corollary asserts that $\text{PER}_\omega$ is Cartesian closed. The Cartesian closed structure of $\text{PER}_\omega$ is
analogous to that of PER. The exponent maps saturated relations \(R\) and \(S\) to the following relation.

\[
n \underbrace{[R \rightarrow S]}_m \quad \text{if and only if } \phi_n \text{ and } \phi_m \text{ map } R\text{-related inputs to } S\text{-related outputs. That is, the following is a square of REL.}
\]

\[
\begin{array}{ccc}
\IN & \phi_n & \IN \\
R & \downarrow & \downarrow & \downarrow \\
\IN & \phi_m & \IN \\
S & & & \\
\end{array}
\]

Products are straightforward, using the recursive bijection from \(\IN \times \IN\) to \(\IN\). Using the Cartesian closed structure and the parametric limits in \(\text{PER}_\omega\), the method of constructing a model in section 3.3 can be used to create a model of System F.

### 3.5 Previous Work on Categorical Models

When Reynolds introduced the initial formalism of relational parametricity [Rey83], it was in the context of modeling a polymorphic lambda calculus. There have been other works in the same direction, using categorical frameworks to describe parametric models. Some of these approaches are mentioned here.

Mitchell & Scrodov [MS93] consider modeling a lambda calculus with implicit polymorphism, that is, one with type variables but not explicit type quantifiers. They describe some models for this calculus where types are interpreted as indexed over a graph category similar to \(\text{JMS}(\mathcal{C})\). (They do not mention the \(I: \mathcal{G}_e \rightarrow \mathcal{G}_e\) functor.)

One of the points pursued by Mitchell & Scrodov is showing that the forgetful functor from categories of indexed functors on \(\text{JMS}(\mathcal{C})\) to categories of functors on \(\mathcal{C}\) preserves all the structure of the models. Any equalities or inequalities between types or terms (programs) that hold with relations also hold without them. In other words, nothing is gained by considering relations in this setting. The work of Mitchell & Scrodov emphasizes the fact that existing computations in a simply typed lambda calculus are parametric, as they do preserve relations. Looking at what their work implies about a polymorphic lambda calculus, we conclude the following. One would not disallow any elements that were constructed in the simply-typed lambda calculus by adding a restriction that elements of polymorphic types preserve relations. To exhibit any advantage of a parametric model, we should consider quantified types, as in System F.
One description of what makes a model of System F is that of PL-categories [See87]. Briefly, a PL-category to model System F is an indexed category $\mathcal{A}: \mathbf{A}^{\mathbf{op}} \to \mathbf{Cat}$ where the base category $\mathbf{A}$ has typing contexts of System F as objects and ‘type substitutions’ as arrows between them. The target category $\mathbf{Cat}$ is the category of Cartesian closed categories and functors which preserve the Cartesian closed structure. The Cartesian closed category $\mathcal{A}(\eta)$ over a typing context provides the interpretation for types (as objects) and terms (as morphisms) in that typing context. Among the properties this indexed category must satisfy is that the image $\mathcal{A}(\pi): \mathcal{A}(\eta) \to \mathcal{A}(\eta, X)$ of the projection $\pi: \eta, X \to \eta$ should have a right adjoint (for interpreting $\forall$). The model of System F described in 3.A can be seen as a PL-category.

**Proposition 3.17**

Every RG setting for System F determines a PL-category.

Suppose $\mathbf{G}$ is an RG setting for System F. The fibre $\mathcal{A}(\eta)$ over a typing context $\eta$ (where $|\eta| = n$) is given by the vertex category of the functor graph $A(\eta) = (G_{\mathbf{G}^n})_v$. The image of a type substitution $\eta \to \eta'$ produces a functor $\mathcal{A}(\eta') \to \mathcal{A}(\eta)$ by composition (with the interpretation of the types to be substituted). The right adjoint is precisely (the vertex portion of) the parametric limit indexed functor (since the image of a projection $\pi: \eta, X \to \eta$ is the diagonal functor).

When Ma & Reynolds [MR92] considered what a parametric model of System F should be, they used the language of PL-categories. A PL-category $\mathcal{A}$ is parametric in the sense of Ma & Reynolds if there is a reflexive graph of PL-categories $\mathcal{A} \xrightarrow{\sim} \mathcal{B}$ such that the respective categories of closed types $\mathcal{A}(\emptyset) \xrightarrow{\sim} \mathcal{B}(\emptyset)$ give a reflexive graph category of a certain form. The form of reflexive graph category used by Ma & Reynolds is a generalization of $\mathbf{JMS} (\mathcal{A}(\emptyset))$ where the witness object and projections lie in some “future” category, $\mathbf{D}$. Since the only requirement on $\mathbf{D}$ is that there be a functor $\mathcal{A}(\emptyset) \to \mathbf{D}$, the reflexive graph structure of $\mathcal{A}(\emptyset)$ could be trivial. Thus all PL-categories are parametric in the sense of Ma & Reynolds.

The work of Ma & Reynolds is more meaningful if one restricts attention to reflexive graph categories that are in some sense non-trivial. For instance, the case were $\mathcal{A}(\emptyset) \xrightarrow{\sim} \mathcal{B}(\emptyset)$ is the reflexive graph category $\mathbf{JMS}(\mathcal{A}(\emptyset))$ does give a useful notion of parametricity, but it is only applicable to closed types. Not being able to reason about open types is a limitation. For example, while it is possible to argue that $\forall X. X \Rightarrow X$ is terminal as a closed type, it is not possible to argue that it is so in the context of a free type
variable.

Our treatment does not suffer from this problem because we do not work in the generality of PL-categories. We do have reflexive graph structures in all typing contexts (that is, the functor graphs $G^{[G]}$). Thus we can argue that $\forall X.X \Rightarrow X$ is terminal in all typing contexts (see section 4.5). We doubt that anything like this is possible in the general setting of PL-categories.

Robinson & Rosolini [RR94] also use PL-categories built from functors which are, in the end, chosen to be the indexed functors of $\mathbf{RG}$. They consider reflexive graph categories of the form $\mathbf{JMS(C)}$, and produce reflexive graphs of PL-categories. While not using this terminology, the reflexive graph categories $A(\eta) \xrightarrow{=} B(\eta)$ they construct are essentially the functor graph $G^{[G]}$ (for the special case $G = \mathbf{JMS(C)}$ they consider). This approach does produce reasonable parametric models for System F. The Robinson & Rosolini model construction is a special case of the construction presented in this thesis.

Our focus is not solely on describing models of System F, but rather on describing an abstract setting for uniformity. Parametric models of polymorphic lambda calculi do provide an application where we can exhibit some consequences of this formalization of uniformity. None of the above mentioned discussions of parametric models addressed the issue of their effectiveness. Is there any sense in which having the extra structure of a reflexive graph of PL-categories is an improvement on simply having a PL-category?

Reynolds [Rey83] provided examples to support the claim that his model trimmed polymorphic types down to only the intuitively uniform families. There have been extensions [RP93, Wad89] to further illustrate how the preservation of relations captures the notion of uniformity. If one wanted to claim to have generalized the notion of relational parametricity to characterize uniformity in more abstract models, it seems reasonable to expect some similar justification.

The settings (generalizations of $\mathbf{JMS(C)}$) described by Mitchell & Scedrov and by Ma & Reynolds are too general to ensure a reasonable amount of uniformity. While jointly monic spans considered by Robinson & Rosolini are strong enough to ensure a reasonable amount of uniformity, there are notions of “relations” over particular categories that we would like to include in our axiomatization but are not of the form of jointly monic spans. Examples include the reflexive graph categories $\mathbf{CPO}$ and $F-\mathbf{Alg}$ mentioned in section 2.3. (Mitchell & Scedrov [MS93] give more examples of exceptions and modifications to the framework of jointly monic spans that one would like to be able to account for.)
This chapter has examined some of the structure of reflexive graph categories. We have shown that parametrically polymorphic families are given by the parametric limit indexed functor. Dually, abstract data types are given by the parametric colimit indexed functor. These indexed functors were used in producing models of the predicative calculus. The parametric limit also can be used to describe models of System F. The structure of reflexive graph categories immediately internalizes. We have used an internal version of reflexive graph categories in describing a particular setting for a reflexive graph model of System F using PERs. While the description of models given in this chapter does allow one to give a more general construction which would include Robinson & Rosolini's models for System F, the real strength of this presentation is the ease with which it melds with our notion of uniformity. In Chapter 4, we present axioms for reflexive graph categories which ensure that the preservation of edges provides a reasonable formalism of uniformity. We shall define parametricity graphs to be the special case of reflexive graph categories that we propose make good general settings for parametricity (section 4.5). Our notion of parametric models for System F arises from using the construction of this chapter specialized to parametricity graphs.
Chapter 4

Parametricity Graphs

The typical notion of uniform families used in categories, naturality, is that of collections of arrows (in the target category) that ‘respect’ all arrows (of the source category). By considering functors out of a discrete category, as in the environment model construction, naturality becomes trivial.

Part of the motivation for this investigation has been to introduce a notion of edges which are to be respected instead. This is exemplified in the CPO interpretation of $\forall X.X \Rightarrow X$ mentioned in the previous chapter — collections of morphisms which ‘respect’ all edges, where respecting the edge $R:A_0 \leftrightarrow A_1$ is meant as giving a square of the following shape.

\[
\begin{array}{c}
A_0 \xrightarrow{t_{A_0}} A_0 \\
R \downarrow \quad \downarrow R \\
A_1 \xleftarrow{t_{A_1}} A_1
\end{array}
\]

It was shown that $[\forall X.X \Rightarrow X]$ contains only the one intuitively uniform collection of morphisms, $t_A:A \to A$. However, the structure of reflexive graph categories doesn’t ensure that parametric limits are trimmed down to just the intuitively uniform elements.

Consider the trivial reflexive graph $\text{Tr} (\text{SET})$ whose vertex category is $\text{SET}$ and has one edge $R:A_0 \leftrightarrow A_1$ for each $A_0$ and $A_1$ — the everywhere true relation. Squares are as in $\text{REL}$, hence, there is a square of the following shape if and only if $f_0:A_0 \to B_0$ and $f_1:A_1 \to B_1$.

\[
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
R \downarrow \quad \downarrow S \\
A_1 \xrightarrow{f_1} B_1
\end{array}
\]
Along with a small Cartesian closed sub-reflexive graph category $G$, we can produce an RG setting $(\text{Tr}(\text{SET}), G)$, and hence a model of the predicative calculus. The interpretation of $\forall X. X \Rightarrow X$ in this model is the set of all indexed natural transformations $\Delta(1) \rightarrow X \Rightarrow X$, which can equivalently be stated as the following collection.

$$\{ (t_X : X \rightarrow X)_{X \in G} : \forall R : X_0 \leftrightarrow X_1, (t_{X_0}, t_{X_1}) \in R \Rightarrow R \}$$

Since $(f, g) \in R \Rightarrow R$ is always true (so long as it type checks), this interpretation consists of all possible collections of $(t_X : X \rightarrow X)$, which is an uncountable set, rather than the singleton set of uniform ones.

The structure of reflexive graph category does ensure that indexed natural transformations (and subsequently elements of parametric limits) contain collections of morphisms which respect all edges. Rather, it is just that the category of edges in $\text{Tr}(\text{SET})$ is not rich enough to ensure any significant uniformity in such a collection. This chapter focuses on what additional axioms are appropriate so that preservation of edges ensures uniformity.

4.1 Relational Reflexive Graph Categories

In Reynolds’ model of the predicative calculus, relations were introduced as a tool to describe the uniformity properties of collections of functions. The focus of the model was still on collections of functions between sets (as the interpretation of terms, or program fragments). The construction of a model in an RG setting in the previous chapter has shifted a little from that focus. The interpretation of terms is not as collections of morphisms (satisfying some properties), but rather as indexed natural transformations — collections of morphisms and squares (satisfying some properties).

Consider the interpretation of $\forall X. X \Rightarrow X$ in $\text{Span}(\text{CPO})$ (using a small, Cartesian closed sub-reflexive graph category $G$ with inclusion indexed functor $\text{In}' : [G] \rightarrow \text{Span}(\text{CPO})$). The interpretation $[\forall X. X \Rightarrow X]$ is given by the collection of indexed natural transformations $1 \rightarrow \text{In}' \Rightarrow \text{In}'$, or equivalently, the collection of indexed natural transformations $\text{In}' \rightarrow \text{In}'$. Unlike in the CPO model of the previous chapter, there is more than one such indexed natural transformation.

To exhibit an indexed natural transformation, $\tau : \text{In}' \rightarrow \text{In}'$, which is not
constantly identities, consider the following span.

\[
\begin{array}{c}
R: \{a, b\} \\
\{\star\}
\end{array}
\]

(Here \(a \) and \(b \) are incomparable in \(R(W)\).) If \(m: \{a, b\} \to \{a, b\} \) denotes the function which interchanges \(a \) and \(b \), then defining \(\tau: \text{In}' \to \text{In}' \) by

\[
\begin{align*}
\tau_X &= \text{id}_X \quad \text{for all vertices } X \\
\tau_R &= \{m, \text{id}_{\{\star\}}, \text{id}_{\{\star\}}\} \\
\tau_S &= \text{id}_S \quad \text{for all edges } S \neq R
\end{align*}
\]

is one of the countless non-identity indexed natural transformation.

Does this mean that \(\text{Span}(\text{CPO})\) does not have enough edges that preservation of edges ensures uniformity? On the contrary, for any indexed natural transformation \(\eta: \text{In}' \to \text{In}'\), and for any CPO \(A\), it is provable that \(\eta_A = \text{id}_A\). (The proof is similar to that of \(\text{CPO}\), using a span analogue of the relation \(\eta_{a'}\)) All the indexed natural transformations of \(\forall X X \Rightarrow X\) have the same vertex part, they only differ in the edge part, that is, they may have different squares of the following shape for any given \(R: A_0 \leftrightarrow A_1\).

\[
\begin{array}{c}
A_0 \\
\eta_{A_0} \downarrow \\
\eta_{A_1} \\
A_1
\end{array}
\]

\[
\begin{array}{c}
\eta_{A_0} \quad \eta_{A_1} \\
\eta_{A_0} \quad \eta_{A_1} \\
R \quad R
\end{array}
\]

It’s not that the collections of morphisms aren’t uniform, but rather that different indexed natural transformations are at liberty to provide different witnesses (or proofs) that edges are respected.

This wasn’t the case in \(\text{CPO}\) since there aren’t multiple witnesses available, as for every edge \(R: A_0 \leftrightarrow A_1\), there is at most one square of the following shape.

\[
\begin{array}{c}
A_0 \\
\eta_{A_0} \downarrow \\
\eta_{A_1} \\
A_1
\end{array}
\]

\[
\begin{array}{c}
\eta_{A_0} \quad \eta_{A_1} \\
\eta_{A_0} \quad \eta_{A_1} \\
R \quad R
\end{array}
\]

For using edges as a tool to reason about the uniformity of collections of
morphisms, it makes sense to have at most one square of a given shape, since they are used to indicate whether a particular pair of morphism respects a particular pair of edges (a yes or no question).

**Definition 4.1**

A reflexive graph category is called relational if, for all edges $R: A_0 \leftrightarrow A_1$, $S: B_0 \leftrightarrow B_1$, and morphisms, $f_0: A_0 \to B_0$, $f_1: A_1 \to B_1$, there is at most one square of the following shape.

\[
\begin{array}{c}
A_0 \\
\downarrow R \\
A_1
\end{array} 
\begin{array}{c}
B_0 \\
\downarrow S \\
B_1
\end{array}
\]

There is a rather rich collection of relational reflexive graph categories. It is obvious that the product of relational reflexive graph categories is again relational. It is not surprising that the functor reflexive graph $H^G$ is relational whenever $H$ is. The sub-2-category $rRG$ of relational reflexive graph categories is Cartesian closed with the inclusion $rRG \to RG$ preserving the Cartesian closed structure.

Many of the natural examples are explicitly defined as relational reflexive graphs, such as $\text{REL}$, $\text{Poset}$, $\text{CPO}$, $\text{CPO}_\perp$, $\text{PER}$, $\text{F}^{-}\text{Alg}$, and the arrow graph $C^{Arr}$ for any category $C$. Spans are a canonical example where there are multiple squares of the same shape. The common restriction to jointly monic spans does put one back in the realm of relational reflexive graph categories. However, this restriction does not preserve the categorical constructions on these reflexive graph structures, and hence $\text{Span}(C)$ and $JMS(C)$ may be seen to give different notions of edges over $C$. There is a relational reflexive graph category which does reflect the structure of $\text{Span}(C)$. This construction applies to a more general setting than just to spans.

For any reflexive graph category $G$, we define an equivalence relation on the collection of squares of $G$ as follows.

\[
\sigma \approx \sigma' \iff \exists R, S. (\sigma: R \to S \land \sigma': R \to S) \land \\
\partial_0(\sigma) = \partial_0(\sigma') \land \partial_1(\sigma) = \partial_1(\sigma')
\]

In other words, two squares are related if they have the same shape. We define the relational reflexive graph category $\mathcal{R}(G)$ to have the same vertices, morphisms and edges as $G$, while the squares of $\mathcal{R}(G)$ are the equivalence classes of $\approx$. There is a square of a given shape in $\mathcal{R}(G)$ if and only if there
exists some square of that shape in \( G \). If such a square exists, there is only one. Clearly, if \( G \) is relational, then \( \mathcal{R}(G) \cong G \).

For any indexed functor \( F: G \to H \), \( \sigma \approx \sigma' \) implies \( F(\sigma) \approx F(\sigma') \). Thus, an indexed functor \( \mathcal{R}(F): \mathcal{R}(G) \to \mathcal{R}(H) \) given by \( \mathcal{R}(F)[\sigma] = [F(\sigma)] \) (and agreeing with \( F \) on vertices, morphisms, and edges) is well-defined. An indexed natural transformation \( \tau: F \to G \) can be mapped to a corresponding \( \mathcal{R}(\tau): \mathcal{R}(F) \to \mathcal{R}(G) \) as follows.

\[
\mathcal{R}(\tau)_A = \tau_A \\
\mathcal{R}(\tau)_R = [\tau_R]
\]

This describes a 2-functor \( \mathcal{R}: \mathcal{RG} \to \mathcal{rRG} \). All 2-categorical constructions (such as adjunctions or monads) involving a reflexive graph category \( G \) produce corresponding constructions involving \( \mathcal{R}(G) \). For instance, if \( G \) is Cartesian closed, so is \( \mathcal{R}(G) \).

The 2-functor \( \mathcal{R} \) is the left adjoint to the inclusion \( r\mathcal{RG} \to \mathcal{RG} \). In other words, \( r\mathcal{RG} \) is a reflective sub-2-category of \( \mathcal{RG} \). The unit of the adjunction has component indexed functors \( N_G: G \to \mathcal{R}(G) \) which map squares \( \sigma \) to their equivalence classes \( [\sigma] \). This is a 2-natural transformation, meaning it commutes with all indexed natural transformations.

\[
\begin{array}{cc}
G & \xrightarrow{\eta} & H \\
\downarrow & \downarrow & \downarrow \\
\mathcal{R}(H) & = & \mathcal{R}(G) \\
\downarrow_{\mathcal{R}(\eta)} & \downarrow_{\mathcal{R}(N_G)} & \downarrow_{\mathcal{R}(\eta)} \\
\mathcal{R}(G) & \xrightarrow{\mathcal{R}(N_G)} & \mathcal{R}(H)
\end{array}
\]

As a result, if one is eventually going to ignore the difference between squares of the same shape, it does not matter whether any 2-categorical constructions are carried out before or after that change.

When considering relational reflexive graph categories, it makes no difference if one selects a square of a given shape or merely acknowledges the existence of one. It seems natural to base our notion of uniformity on the existence of a square rather than the choice of a particular square, as it leads us back to collections of morphisms satisfying some properties. This is the intuition behind the following definition.

**Definition 4.2**

Let \( G \) and \( H \) be reflexive graph categories, and \( F, G: G \to H \) be indexed functors. A collection \( \{ \tau_A: F(A) \to G(A) \mid A \in G_e \} \) of \( H \)-morphisms is a parametric transformation if, for every edge \( R: A_0 \leftrightarrow A_1 \) of \( G \), there is a
square of the following shape in $\mathbf{H}$.

\[
\begin{array}{ccc}
F(A_0) & \xrightarrow{\tau_{A_0}} & G(A_0) \\
\downarrow F(R) & & \downarrow G(R) \\
F(A_1) & \xrightarrow{\tau_{A_1}} & G(A_1)
\end{array}
\]

Natural transformations which additionally are parametric have been used as an alternative characterization of 2-cells [OT95]. If $\mathbf{H}$ is relational, then there is a bijection between parametric natural transformations between indexed functors $\mathbf{G} \to \mathbf{H}$ and indexed natural transformations between the same indexed functors. Indeed, it is typically the case that relational reflexive graph categories are considered when discussing parametric natural transformations. However, the definition makes sense even for non-relational reflexive graph categories. This provides another setting one could consider, the 2-category $\mathbf{pRG}$ of reflexive graph categories, indexed functors, and parametric natural transformations.

This is not really a significantly different setting from $\mathbf{rRG}$. The obvious analogue $\mathcal{R}^*: \mathbf{pRG} \to \mathbf{rRG}$ of the previously mentioned 2-functor $\mathcal{R}: \mathbf{RG} \to \mathbf{rRG}$ along with the inclusion\(^1\) establishes an equivalence between the 2-categories $\mathbf{pRG}$ and $\mathbf{rRG}$. That is, there is a parametric natural isomorphism $\text{ID} \to \mathcal{R}^*: \mathbf{pRG} \to \mathbf{pRG}$ as well as the indexed natural isomorphism $\mathcal{R}^* \to \text{ID}: \mathbf{rRG} \to \mathbf{rRG}$. Therefore, any development in one of these 2-categories can be carried over directly to the other.

Since spans are a canonical construction, we will keep them, or rather their relational counterpart, in mind as a generally applicable example. To facilitate talking about this, we use the notation $\mathbf{Sp}(\mathbf{C})$ which names the image of $\mathbf{Span}(\mathbf{C})$ under $\mathcal{R}$. This reflexive graph category has spans over $\mathbf{C}$ as edges, and there is a unique square of shape on the left if and only if there exists a morphism $w: W(R) \to W(S)$ such that the diagram on the right commutes.

\[\begin{array}{ccc}
A_0 & \xrightarrow{f_0} & B_0 \\
\downarrow R & & \downarrow S \\
A_1 & \xrightarrow{f_1} & B_1
\end{array}\]

\[\begin{array}{ccc}
W(A_0) & \xrightarrow{W(f_0)} & W(B_0) \\
\downarrow W(R) & & \downarrow W(S) \\
W(A_1) & \xrightarrow{W(f_1)} & W(B_1)
\end{array}\]

\(^1\)modulo the bijection between indexed natural transformations and parametric natural transformations mentioned above
4.2 Identity Condition

One place where the trivial example $\text{Tr}(\mathbf{C})$ runs afoul of intuition is that the identity edges don’t seem much like identities. One would like to say that identity edges only relate elements to themselves, as opposed relating everything to everything else as in the $\text{Tr}(\mathbf{C})$ example. Categorically, one states such things for morphisms rather than elements, as in the following definition from [KOP+97].

**Definition 4.3**

A reflexive graph category, $\mathbf{G}$, is said to satisfy the identity condition if $f = g$ whenever there is a square of the following shape.

$$
\begin{array}{c}
A & \xrightarrow{f} & B \\
\downarrow{I_A} & & \downarrow{I_B} \\
A & \xrightarrow{g} & B \\
\end{array}
$$

This condition is either satisfied by construction, explicitly stated, or implied by other requirements in most treatises on parametricity which acknowledge identity edges. Kinoshita et.al. hypothesize that “all naturally occurring” reflexive graph categories satisfy the identity condition. Most of the reflexive graph categories mentioned, with the exception of the admittedly contrived $\text{Tr}(\mathbf{SET})$ example, satisfy the identity condition.

The identity condition does more than merely support the intuitive feel of identity edges. It also provides a way to use the edge category to reason about morphisms. For example, if $\forall_X F(X)$ exists in a reflexive graph category satisfying the identity condition, then the projection $\omega : \forall_X F(X) \to F$ “commutes” with any $f : F(A) \to F(B)$ such that there exists a relation $R : A \leftrightarrow B$ and a square of the following shape.

$$
\begin{array}{c}
F(A) & \xrightarrow{f} & F(B) \\
\downarrow{F(R)} & & \downarrow{\text{id}_{F(B)}} \\
F(B) & \xrightarrow{\text{id}} & F(B) \\
\end{array}
$$

This holds regardless of whether or not $f$ is in the image of $F$. Observe that $\omega_R$ can be composed with above square to get a square of the following

---

2 They admit this is not true for all scenarios. There are natural notions of relations useful in reasoning about program refinement which can be formulated as reflexive graph categories, such as $\mathbf{Poset}$, that do not satisfy the identity condition.
shape.

\[ \forall X F(X) \xrightarrow{f \circ \omega} F(B) \]

\[ \forall X F(X) \xrightarrow{\omega_B} F(B) \]

The identity condition ensures that \( f \circ \omega_A = \omega_B \).

A typical instance where this sort of result would be interesting is dealing with an non-variant functor, \( F: G \rightarrow \mathbf{H} \), as in the environment model construction of section 3.3. This is exemplified in considering the parametric limit of the inclusion \( J: |\text{REL}| \rightarrow \text{REL} \). For any \( f: A \rightarrow B \), consider the relation which is the graph of \( f \), \( \langle f \rangle \). This is such that there is a square of the following shape.

\[ \begin{array}{ccc}
A & \xrightarrow{f} & B \\
\langle f \rangle & & \downarrow \text{id} \\
B & \xrightarrow{\text{id}} & B \\
\end{array} \]

Therefore \( f \circ \omega_A = \omega_B \) for every \( f: A \rightarrow B \) of \( \text{REL} \). (By considering a function with no fixed point, such as the successor function \( \text{succ}: \mathbb{N} \rightarrow \mathbb{N} \), this result implies that \( \forall X J(X) \) is the empty set.)

Reflexive graph categories which satisfy the identity condition may still be weak enough that the preservation of all edges doesn’t provide any uniformity. For any category \( \mathbf{C} \), the reflexive graph category \( \mathbf{C}^* \) given by the corresponding constant functor \( \mathbf{B} \rightarrow \mathbf{CAT} \) has only identity edges and squares. The preservation of all edges in such a setting is vacuous. In order to get a non-trivial characterization of uniformity, there needs to be some more interesting edges.

### 4.3 Subsumption

Parametricity and naturality are both concepts which are attempting to formalize the same idea of uniformity. It seems reasonable to wonder if having two theories of uniformity is necessary, or if there is a single theory which encompasses them both. In the motivating example \( \text{REL} \), observe that the collection of relations includes all the functions (or the graphs of all the functions, if one prefers). Recall that in the \( \forall X J(X) \) example of the previous section, it was graphs of functions which were the relations we
used. The concept of the graph of a function can be expressed in a more
general reflexive graph category, as in the following definition from [Red97].

Definition 4.4
A subsumptive reflexive graph category is a reflexive graph category, $\mathbf{G}$,
together with a mapping of morphisms $f: A \to B$ to edges $\langle f \rangle: A \leftrightarrow B$ such
that:

- for every vertex $A$, $\langle \text{id}_A \rangle = 1_A$.

- there is an edge morphism of shape on the left, below, if and only if
the diagram of morphisms on the right, below, commutes.

\[
\begin{array}{ccc}
  A & \xrightarrow{g} & A' \\
  B & \xrightarrow{h} & B'
\end{array}
\quad \iff \quad
\begin{array}{ccc}
  A & \xrightarrow{g} & A' \\
  B & \xrightarrow{h} & B'
\end{array}
\]

The function $\langle - \rangle$ from vertices to edges is called the subsumption map.

Carrying nomenclature over from $\mathbf{REL}$, $\langle f \rangle$ is called the graph of the
morphism $f$. In addition to $\mathbf{REL}$, there are many other naturally occurring
subsumptive reflexive graph categories.

Examples:

- $\mathbf{CPO}$ is subsumptive, with $\langle f \rangle$ once again given by the usual graph
of the function, $\langle f \rangle = \{(x, f(x)) \mid x \in A\}$ for any morphism $f: A \to B$.
  This relation is complete since $f$ is continuous and hence, is an edge
of $\mathbf{CPO}$.

- For any category $\mathbf{C}$, the arrow graph $\mathbf{C}^{\text{Arr}}$ is subsumptive with the
  graph of a morphism $f$ given by $f$ itself.

- Any $\mathbf{Span}(\mathbf{C})$ or $\mathbf{JMS}(\mathbf{C})$ is subsumptive with the graph of a mor-
  phism $f: A \to B$ given by the following span.

\[
\begin{array}{c}
\text{id} \quad A \\
\downarrow \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \q
if and only if the diagram on the right commutes is immediate.

\[
\begin{array}{c}
\text{id} & \text{id} \\
A & A' \\
f & f' \\
B & B' \\
\end{array}
\quad
\begin{array}{c}
g & g' \\
A & A' \\
f & f' \\
B & B' \\
\end{array}
\]

The witness information of the span morphism must be \(g\) (so the top rectangle commutes), giving that the bottom rectangle is exactly the commuting diagram.

Subsumption is a generalization of the above identity condition by requiring all morphisms be represented by edges as opposed to just the identity edges, which is essentially what the identity condition imposes.

**Lemma 4.5**

A reflexive graph category satisfies the identity condition precisely when the following condition is satisfied: there is a square of the shape below if and only if \(g = h\).

\[
\begin{array}{c}
A & \rightarrow & B \\
\downarrow & & \downarrow \\
I_A & & I_B \\
A & \rightarrow & B \\
\end{array}
\]

**Proof.** For any reflexive graph category, if \(g = h\), then there is a square of the above shape given by \(I_g\) (which is also \(I_h\)). The only if direction is precisely the identity condition.

Having edges corresponding to morphisms can be useful in proving some representation results, such as the following.

**Theorem 4.6**

If \(G\) is a subsumptive relational reflexive graph category such that the parametric limit \(\forall X X\) exists, then \(\forall X X\) is the initial object of \(G_x\).

**Proof.** The projection out of \(\forall X X\) provides a morphism \(\omega_A : \forall X X \rightarrow A\) for every vertex \(A\).
Since $G$ is subsumptive, there is an edge $\langle \omega_A \rangle : \forall X X \leftrightarrow A$ of $G$ and subsequently a square of the following shape for any $A$.

$$
\begin{array}{ccc}
\forall X X & \xrightarrow{\omega_{\forall X X}} & \forall X X \\
L \downarrow & \downarrow {\omega_A} & \downarrow {\langle \omega_A \rangle} \\
\forall X X & \xrightarrow{\omega_A} & A
\end{array}
$$

Therefore, $\omega_A \circ \omega_{\forall X X} = \omega_A$ follows from $G$ being subsumptive. Since there is a unique morphism $\Lambda(\omega) : \forall X X \to \forall X X$ such that $\omega \circ \Lambda(\omega) = \omega$, it is the case that $\omega_{\forall X X} = \Lambda(\omega) = \text{id}_{\forall X X}$.

The uniqueness of the morphism $\omega_A : \forall X X \to A$ (for each $A$) is shown by taking any morphism $f : \forall X X \to A$ and considering the edge $\langle f \rangle : \forall X X \leftrightarrow A$. Note there is a square of the following shape.

$$
\begin{array}{ccc}
\forall X X & \xrightarrow{\omega_{\forall X X}} & \forall X X \\
L \downarrow & \downarrow {\langle f \rangle} & \downarrow {\omega_A} \\
\forall X X & \xrightarrow{\omega_A} & A
\end{array}
$$

It follows from the definition of subsumptive that $\omega_A = f \circ \omega_{\forall X X} = f$. ◦

Having morphisms give rise to edges as in the definition of subsumptive means that collections of morphisms that “respect all edges” will necessarily “respect all morphisms”. But note that there is a slight difference in this manner of respecting morphisms by parametricity as opposed to respecting morphisms in the sense of naturality. Parametricity yields a square (on the left, below) as opposed to the commuting diagram (on the right, below) ensured by naturality.

$$
\begin{array}{ccc}
F(A) & \xrightarrow{t_A} & G(A) \\
F(f) \downarrow & & \downarrow G(f) \\
F(B) & \xrightarrow{t_B} & G(B)
\end{array}
\quad
\begin{array}{ccc}
F(A) & \xrightarrow{t_A} & G(A) \\
F(f) \downarrow & & \downarrow G(f) \\
F(B) & \xrightarrow{t_B} & G(B)
\end{array}
$$

In considering reflexive graph categories with the additional structure of a subsumption mapping, it makes sense to consider indexed functors which respect that additional structure. Since subsumption is saying that morphisms are edges, this amounts to requiring that there be a single image of a morphism, whether it is treated as a morphism or as an edge.
Definition 4.7

Given subsumptive reflexive graph categories, $G$ and $H$, a subsumptive indexed functor from $G$ to $H$ is an indexed functor $F: G \to H$ such that $F(\langle f \rangle) = \langle F(f) \rangle$ for all morphisms $f$.

If $F$ and $G$ are subsumptive indexed functors, then there is a square of the shape on the left if and only the the diagram on the right commutes, and hence the two notions of preserving morphisms coincide.

\[
\begin{array}{ccc}
F(A) & \xrightarrow{t_A} & G(A) \\
\downarrow F(\langle f \rangle) & & \downarrow G(\langle f \rangle) \\
F(B) & \xrightarrow{t_B} & G(B)
\end{array}
\quad
\begin{array}{ccc}
F(A) & \xrightarrow{t_A} & G(A) \\
\downarrow F(f) & & \downarrow G(f) \\
F(B) & \xrightarrow{t_B} & G(B)
\end{array}
\]

Thus, one only needs to consider collections of morphisms which respect all edges as it all such collections will respect all morphisms as well. The full generality of parametric transformations (definition 4.2) is sufficient to ensure naturality, allowing us to use parametric transformations without additionally requiring a naturality assumption. The fact that the preservation of edges is stronger than the preservation of morphisms can be stated formally as in the following proposition from [Red97].

Proposition 4.8

Suppose $F, G: H \to G$ are subsumptive indexed functors. Any parametric transformation $\tau: F \to G$ is a natural transformation.

A subsumptive reflexive graph category has a lower bound on the amount of edges. There must be enough edges to represent all the morphisms. Therefore, we can say something about how strong parametricity is as a notion of uniformity - it is at least as strong as naturality.

4.4 Fibrations

Hermida [Her93] proposed that logical relations are best modeled in terms of fibrations (or fibred categories). Fibrations have a strong connection with predicate logic, thus they can be viewed as generalization of the setting of sets and subsets (recalling that a predicate on a set can be equated with the subset of the set that satisfies the predicate). Since relations can be viewed as predicates (or subsets) of the product, one can see how a general theory of predicates (that is, fibrations) might relate to a general theory of relations. We say that a reflexive graph category $G$ is fibred when
$(\partial_0, \partial_1) : G_e \rightarrow G_e \times G_e$ is a fibration. We tailor the definition of a fibration to this specialized case as follows.

**Definition 4.9**

Consider a square $\phi$ of the following shape.

\[
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
R \\
A_1 \xrightarrow{f_1} B_1 \\
\end{array}
\]

Such a square is said to be cartesian if for any square $\sigma$ of the shape on the left, below, there is a unique factorization of $\sigma$ through $\phi$, denoted $\sigma = \phi \circ \Phi(\sigma)$.

\[
\begin{array}{c}
C_0 \xrightarrow{f_0 \circ g_0} B_0 \\
T \\
C_1 \xrightarrow{f_1 \circ g_1} B_1 \\
\end{array}
\]

A reflexive graph category is called fibred if, for every edge $S : B_0 \leftrightarrow B_1$ and for any morphisms $f_0 : A_0 \rightarrow B_0$, $f_1 : A_1 \rightarrow B_1$, there exists an edge $[f_0, f_1]S : A_0 \leftrightarrow A_1$ with a cartesian square of the following shape.

\[
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
[f_0, f_1]S \\
A_1 \xrightarrow{f_1} B_1 \\
\end{array}
\]

Such an edge is said to be the weakest pre-edge of $S$ along $(f_0, f_1)$.

Dually, there is the notion of cofibred reflexive graph category, where for each edge $R : A_0 \leftrightarrow A_1$ and morphisms $f_0 : A_0 \rightarrow B_0$, $f_1 : A_1 \rightarrow B_1$ there is a strongest post-edge $R[f_0, f_1]$ and co-cartesian square $\phi ; R \rightarrow R[f_0, f_1]$ such that any square of the shape on the left factors uniquely through $\phi$. 

\[
\begin{array}{c}
A_0 \xrightarrow{g_0 \circ f_0} C_0 \\
R \\
A_1 \xrightarrow{g_1 \circ f_1} C_1 \\
\end{array}
\quad
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \xrightarrow{g_0} C_0 \\
R \\
A_1 \xrightarrow{f_1} B_1 \xrightarrow{g_1} C_1 \\
\end{array}
\]

77
Observe that the weakest pre-edge is unique up to isomorphism. Suppose $R$ and $R'$ both have cartesian squares into $S$.

\[
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
A_1 \xrightarrow{f_1} B_1
\end{array}
\quad \sigma : R \quad \begin{array}{c}
\uparrow S \\
\downarrow \quad \uparrow S
\end{array}
\quad \begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
A_1 \xrightarrow{f_1} B_1
\end{array}
\quad \sigma' : R'
\]

The isomorphism $\rho R \to R'$ comes from factoring $\sigma$ through $\sigma'$ (and similarly, its inverse comes from factoring $\sigma'$ through $\sigma$).

\[
\rho : R \quad \begin{array}{c}
\uparrow \quad \uparrow \quad \uparrow \quad \uparrow
\end{array}
\quad \begin{array}{c}
A_0 \xrightarrow{\text{id}} A_0 \\
A_1 \xrightarrow{\text{id}} A_1
\end{array}
\quad R'
\]

A specific choice of all weakest pre-edges is called a *cleavage*. For any edge $S : B_0 \leftrightarrow B_1$, the identity edge $\text{id}_S : S \to S$ is cartesian. Thus one may always select a cleavage such that $[\text{id}_{B_0}, \text{id}_{B_1}]S = S$. Such a cleavage is called *normalized*.

**Examples:**

- **REL** is both fibred and cofibred. The weakest pre-edge $[f_0, f_1]S$ is given as follows.

\[
x_0 \left[ [f_0, f_1]S \right] x_1 \iff f_0(x_0) \left[ S \right] f_1(x_1)
\]

Dually, the strongest post-edge is given below.

\[
x_0 \left[ R[f_0, f_1] \right] x_1 \iff \exists y_0, y_1. \ f_0(y_0) = x_0 \land y_0 \left[ R \right] y_1 \land f_1(y_1) = x_1
\]

- **CPO** is also both fibred and cofibred. The weakest pre-edge is defined exactly as in **REL**. Note that $[f_0, f_1]S$ is a complete relation whenever $f$ and $f'$ are continuous, and $S$ is complete. For the strongest post-edge, the construction used in **REL** need not define a complete relation. Taking $[f_0, f_1]R$ to be the least complete relation such that $y_0 \left[ R \right] y_1$ implies $f_0(y_0) \left[ [f_0, f_1]R \right] f_1(y_1)$ does give us the strongest post-edge in **CPO**.

- For any Cartesian category $\mathbf{C}$ with equalizers (and hence, all finite limits), $\text{Span}(\mathbf{C})$ is fibred. The weakest pre-edge $[f_0, f_1]S$ has witness
object given by the limit of the diagram below.

\[
\begin{array}{ccc}
A_0 & \xrightarrow{f_0} & B_0 \\
\downarrow & \nearrow & \downarrow \\
W(S) & \longrightarrow & \\
A_1 & \xrightarrow{f_1} & B_1 \\
\end{array}
\]

**Span(C)** is always cofibred, for any category C. The co-cartesian edge morphism \(\phi: R \rightarrow R[f_0, f_1]\) is given below.

\[
\begin{array}{ccc}
A_0 & \xrightarrow{f_0} & B_0 \\
\downarrow & \nearrow & \downarrow \\
W(R) & \longrightarrow & \\
A_1 & \xrightarrow{f_1} & B_1 \\
\end{array}
\]

An a non-example, notice that the arrow graph \(C^{Arr}\) is, in general, not fibred. This is the same as saying that there is no way, in general, to find a morphism from \(A_0\) to \(A_1\) making the diagram below commute.

\[
\begin{array}{ccc}
A_0 & \xrightarrow{f_0} & B_0 \\
\downarrow & \nearrow & \downarrow \\
? & \xrightarrow{h} & \\
A_1 & \xrightarrow{f_1} & B_1 \\
\end{array}
\]

In any fibred reflexive graph category, there is a bijection between squares of the following two shapes.

\[
\begin{array}{ccc}
C_0 & \xrightarrow{g_0} & A_0 \\
\downarrow & \nearrow & \downarrow \\
C_1 & \xrightarrow{g_1} & A_1 \\
R & \xrightarrow{[f, f']S} & [S] \\
\end{array}
\]

Thus, if squares into a certain edge \(S\) can be used to reason about morphisms, it may be possible to extend that reasoning to apply to other edges \([f_0, f_1]S\). An example of this is that the identity condition can be extended to a subsumption map.

**Theorem 4.10**

Any fibred reflexive graph category satisfying the identity condition is subsumptive.
Proof. For a fibred reflexive graph category $\mathbf{G}$, we fix a normalized cleavage and reason as follows. For any morphism $g: A \to A'$, define $\langle g \rangle = [g, id_{A'}] I_{A'}$ with a chosen cartesian square $i_g: \langle g \rangle \to I_{A'}$. Note that there is a square $i^g: I_A \to \langle g \rangle$ arising as the unique factor of $I_g: I_A \to I_{A'}$ through $\langle g \rangle$.

$$
\begin{array}{c c c}
A & \xrightarrow{id_A} & A \\
| & h | & l \\
\langle g \rangle & \xrightarrow{g} & \langle h \rangle \\
| & i_g | & l_g \\
A & \xrightarrow{id_{A'}} & A' \\
\end{array}
$$

With this choice of graphs of morphisms, $\mathbf{G}$ can be shown to be subsumptive. Since re-indexing along $(id_A, id_A)$ is the identity, it follows that

$$\langle id_A \rangle = [id_A, id_A] I_A = I_A.$$

All that remains is to show that there is a square of the shape on the left below if and only if the right diagram commutes.

$$
\begin{array}{c c c c}
A & \xrightarrow{f} & B \\
| & \downarrow{g} | & \downarrow{l} \\
A' & \xrightarrow{f'} & B' \\
\end{array} \iff 
\begin{array}{c c c c}
A & \xrightarrow{f} & B \\
| & \downarrow{g} | & \downarrow{l} \\
A' & \xrightarrow{f'} & B' \\
\end{array}
$$

For the left-to-right direction, we compose squares as follows.

$$
\begin{array}{c c c c}
A & \xrightarrow{id_A} & A & \xrightarrow{B} & B \\
| & i_g | & \downarrow{h} | & \downarrow{l_h} | & \downarrow{1_{B'}} \\
A & \xrightarrow{g} & A' & \xrightarrow{B'} \\
\end{array}
$$

The desired equality follows from the identity condition. For the right-to-left direction, by composition we have a square $\sigma = 1_{f'} \circ i_g$ of the following shape.

$$
\begin{array}{c c c}
A & \xrightarrow{g} & A' & \xrightarrow{f'} & B' \\
| & i_g | & \downarrow{1_{A'}} | & \downarrow{1_{B'}} \\
A' & \xrightarrow{1_{A'}} & A' & \xrightarrow{f'} & B' \\
\end{array}
$$

80
Since $f' \circ g = h \circ f$, it is the case that $\partial_0(\sigma) = h \circ f$, and hence $\sigma$ uniquely factors through the cartesian square out of $\langle h \rangle$.

This gives us the desired square.

Note that a fibred reflexive graph category does not necessarily provide a way to use edges and squares to reason about morphisms. The fact that the trivial reflexive graph category $\text{Tr}(\text{SET})$ is fibred emphasizes this point.

If one wishes to consider fibred reflexive graph categories, it is natural to wonder what the appropriate notion of maps between them is. Much the way that indexed functors between subsumptive reflexive graph categories were asked to preserve subsumption maps, there should be requirements for preserving the additional structure of fibrations.

**Definition 4.11**

If $\mathbf{G}$ and $\mathbf{H}$ are fibred reflexive graph categories, we say that an indexed functor $F: \mathbf{G} \to \mathbf{H}$ is fibred if, for every cartesian square $\sigma$ of $\mathbf{G}$, its image $F(\sigma)$ is cartesian in $\mathbf{H}$. Dually, an indexed functor between cofibred reflexive graph categories is cofibred if it maps co-cartesian squares to co-cartesian squares.

Recall that weakest pre-edges are unique up to isomorphism. The following two squares are both cartesian squares.

$$
\begin{array}{ccc}
F(A_0) & \xrightarrow{F(f_0)} & F(B_0) \\
\downarrow & & \downarrow \\
F([f_0,f_1]S) & \xrightarrow{\mathcal{F}(f_1)} & F(S) \\
\downarrow & & \downarrow \\
F(A_1) & \xrightarrow{\mathcal{F}(f_1)} & F(B_1)
\end{array}
$$

$$
\begin{array}{ccc}
F(A_0) & \xrightarrow{F(f_0)} & F(B_0) \\
\downarrow & & \downarrow \\
[F(f_0), F(f_1)]F(S) & \xrightarrow{\mathcal{F}(f_1)} & F(S) \\
\downarrow & & \downarrow \\
F(A_1) & \xrightarrow{\mathcal{F}(f_1)} & F(B_1)
\end{array}
$$

It follows that $F([f_0,f_1]S) \simeq [F(f_0), F(f_1)]F(S)$ for any fibred indexed functor $F: \mathbf{G} \to \mathbf{H}$ and any cleavages on $\mathbf{G}$ and $\mathbf{H}$. If there are fixed cleav-
ages in mind for $\mathbf{G}$ and $\mathbf{H}$, a \textit{cloven} indexed functor is a fibred functor that preserves the chosen re-indexings exactly (this means not only that $F([f_0, f_1]S) = [F(f_0), F(f_1)]F(S)$, but also the chosen cartesian squares are preserved as well).

Requiring that indexed functors between fibred reflexive graph categories preserve cartesian squares allows one to produce more examples of fibred reflexive graph categories. If $F: \mathbf{REL} \to \mathbf{REL}$ is a fibred indexed functor, then $F - \mathbf{Alg}$ is also fibred. For any two morphisms $f_0: \langle A_0, \alpha_0 \rangle \to \langle B_0, \beta_0 \rangle$, and $f_1: \langle A_1, \alpha_1 \rangle \to \langle B_1, \beta_1 \rangle$ and any edge $S: \langle B_0, \beta_0 \rangle \leftrightarrow \langle B_1, \beta_1 \rangle$, the weakest pre-edge is just as in $\mathbf{REL}$, $[f_0, f_1]S$.

Showing that $[f_0, f_1]S: A_0 \leftrightarrow A_1$ is an $F$-simulation amounts to showing that there is a square of the following shape in $\mathbf{REL}$.

\[
\begin{array}{c}
F(A_0) \xrightarrow{\alpha_0} A_0 \\
F([f_0, f_1]S) \downarrow \\
F(A_1) \xrightarrow{\alpha_1} A_1
\end{array}
\]

Suppressing the isomorphism $\sigma: F([f_0, f_1]S) \cong [F(f_0), F(f_1)]F(S)$ which satisfies $\sigma_0(\sigma) = \text{id}$ and $\sigma_1(\sigma) = \text{id}$, there is a square of the following shape arising from the cartesian square and the fact that $S: B_0 \leftrightarrow B_1$ is an $F$-simulation.

\[
\begin{array}{c}
F(A_0) \xrightarrow{F(f_0)} F(B_0) \xrightarrow{\beta_0} B_0 \\
F([f_0, f_1]S) \downarrow \\
F(A_1) \xrightarrow{F(f_1)} F(B_1) \xrightarrow{\beta_1} B_1
\end{array}
\]

Since $\beta_0 \circ F(f_0) = f_0 \circ \alpha_0$ and $\beta_1 \circ F(f_1) = f_1 \circ \alpha_1$, this factors uniquely through $[f_0, f_1]S$.

\[
\begin{array}{c}
F(A_0) \xrightarrow{\alpha_0} A_0 \xrightarrow{f_0} B_0 \\
F([f_0, f_1]S) \downarrow \\
F(A_1) \xrightarrow{\alpha_1} A_1 \xrightarrow{f_1} B_1
\end{array}
\]

Dually, if $F$ is cofibred, then $F - \mathbf{Alg}$ is cofibred as well. Strongest post-edges are again just as in $\mathbf{REL}$.

Given the relationship between fibred reflexive graph categories and subsumptive reflexive graph categories of Theorem 4.10, it is interesting to consider the differences between the notions of fibred indexed functors and
subsumptive indexed functors.

Since normalized cleavages are used to define subsumption mappings, preservation of those cleavages implies the preservation of the graphs of morphisms.

\[ F(\langle f \rangle) = F([f, \text{id}_B]_B) = [F(f), \text{id}_B]_B F(I_B) = [F(f), \text{id}_{F(B)}]_{F(B)} = \langle F(f) \rangle \]

Thus cloven indexed functors are subsumptive. For general fibred indexed functors, one has that graphs of morphisms are preserved up to isomorphism.

Since that isomorphism lies over identity morphisms, this is good enough to ensure that \( F(\langle f \rangle) \) interacts with the subsumption map of \( \mathbf{H} \) in the same way as if it were the graph of \( \mathcal{F}(f) \). That is, there is a square of shape of the shape on the left below if and only if the diagram on the right below commutes.

\[
\begin{array}{ccc}
F(A) & \xrightarrow{g} & C \\
\downarrow F(\langle f \rangle) & & \downarrow \langle h \rangle \\
\mathcal{F}(B) & \xrightarrow{g'} & D \\
\end{array} \quad \begin{array}{ccc}
F(A) & \xrightarrow{g} & C \\
\downarrow F(f) & & \downarrow h \\
\mathcal{F}(B) & \xrightarrow{g'} & D \\
\end{array}
\]

(A similar assertion holds for squares into \( F(\langle f \rangle) \)).

The definition of fibration presented in definition 4.9 was specifically tailored to the setting at hand, but the basic notion is more general. Recalling the generalization from reflexive graph categories to n-ary reflexive graph categories (in section 2.3, page 28), it is straightforward to talk of a fibred n-ary reflexive graph category. In fact, considering the additional properties proposed in this chapter, the only part which is inherently binary is the subsumption map - graphs of functions are binary relations. (Some may argue that the notion of identity edges as the equality relation is inherently binary, but defining an n-ary equality as every pair being equal is a reasonably natural extension.)

### 4.5 Parametricity Graphs

The additional properties mentioned in the previous sections combine to define a reasonably general setting for describing uniformity.

**Definition 4.12**

A parametricity graph is a fibred, relational reflexive graph category which satisfies the identity condition.
A PG-functor is a fibred indexed functor $F: G \to H$ between parametricity graphs.

As has been mentioned while discussing the properties individually, many of the examples from Chapter 2 are parametricity graphs: \textbf{REL, CPO, PER, $F-\text{Alg}$} (provided $F$ is fibred) as well as \textbf{JMS(C)} and \textbf{Sp(C)} for any category with finite limits. Having parametricity graphs as the intended setting for discussing parametricity, the analysis from Chapter 2 can be carried over to parametricity graphs.

We start by defining the 2-category \textbf{PG} to consist of parametricity graphs, PG-functors and parametric transformations (definition 4.2). In general, 2-categorical definitions will be taken in the 2-category \textbf{PG} using the adjective parametric, for instance parametric adjunctions or parametric monads. Much of the translation from \textbf{RG} to \textbf{PG} is straightforward and warrants little or no comment. For instance, the product of two parametricity graphs is a parametricity graph. However, one cannot appeal to the general construction of exponents in indexed categories to get the exponent of parametricity graphs as \textbf{PG} is not an indexed category.

For parametricity graphs $G$ and $H$, there is a functor graph $G^H$ with PG-functors $F: H \to G$ as vertices, PG-functors $F: E \times H \to G$ as edges, and the apparent parametric transformations as morphisms and squares. It is almost immediate that it is relational and satisfies the identity condition. To show that this functor graph is fibred, consider an edge $G: G_0 \leftrightarrow G_1$ and parametric transformations $\tau_0: F_0 \to G_0$ and $\tau_1: F_1 \to G_1$. The weakest pre-edge is defined in the obvious pointwise manner.

$([\tau_0, \tau_1]_G)(E, R) = [(\tau_0)_{A_0}, (\tau_1)_{A_1}](G(E, R))$

Proving that the functor graph gives the exponent in \textbf{PG} is straightforward. The unit $\eta_{H,G}: G \to (G \times H)^H$ is obtained by “fixing the first argument” of the identity indexed functor. “Fixing the first argument” of an indexed functor $F: G \times H \to K$ at an edge $R: X \leftrightarrow X'$ of $G$ to yield an indexed functor $E \times H \to K$ is achieved by composition with the indexed functor $K_R: E \to G$. The co-unit of the adjunction is given directly by application.

Consider the right adjoint Lim and left adjoint Colim to the diagonal PG-functor $\Delta: G \to G^H$. (We are considering adjunctions in the 2-category \textbf{PG}.) One can show that Lim($F$) and Colim($F$) are the parametric limit and parametric colimit of a PG-functor $F: H \to G$. These claims are analogous to Theorem 3.5 and proposition 3.8 except in the 2-category \textbf{PG} rather than
RG. The proofs can be translated to the new 2-category directly.

The construction of a model for the polymorphic lambda calculi presented in Chapter 3 can be translated to the 2-category PG in a straightforward manner. One simply refers to the Cartesian closed structure, exponents, parametric limit functor and parametric colimit functor in PG rather than RG. We therefore shall refine the definition of RG setting to describe an appropriate setting for giving parametric models of polymorphic calculi.

**Definition 4.13**

A parametricity setting for the predicative calculus consists of a pair of parametricity graphs $G$ and $H$ such that $G$ is a Cartesian closed sub-parametricity graph of $H$ and the parametric limit and colimit PG-functors $\text{Lim}, \text{Colim}: H^G \to H$ exist in PG.

A parametricity setting for System F consists of a single parametricity graph $G$, such that the parametric limit PG-functor $\text{Lim}: G^G \to G$ exists in PG.

Having terms interpreted as collections of morphisms which preserve all edges makes it apparent that there is some uniformity requirement on them. Correspondingly, the interpretation of polymorphic types as parametric limits (intuitively collections of parametric transformations) has uniformity requirements imposed as well. Fibred reflexive graph categories that satisfy the identity condition are subsumptive, hence have a rich collection of edges that can be used to reason about morphisms. This allows us to show that the preservation of edges in parametricity graphs does provide a strong notion of uniformity. The following representation result illustrates the uniformity.

**Theorem 4.14**

Let $G$ be a well-pointed, Cartesian closed parametricity graph. The parametric limit of $X \Rightarrow X: |G| \to G$ exists and is given by the terminal object, $1 = \forall X X \Rightarrow X$.

**Proof.** The projection parametric transformation $\omega: 1 \to X \Rightarrow X$ is given by currying $\lambda_A: 1 \times A \to A$ for each vertex $A$ (from the fact that 1 is terminal).

For any parametric transformation $\beta: P \to X \Rightarrow X$ (where $P$ is an object of $G$), the only choice for the factorization $\Phi(\beta): P \to 1$ is the unique morphism $!_P: P \to 1$. To show that $\omega \circ !_P = \beta$, it suffices to show that they agree on any component, $A$, or even that the uncurried versions, $\text{ap} \circ (\beta_A \times \text{id})$ and $\text{ap} \circ ((\omega_A \circ !_P) \times \text{id})$, agree. By the well-pointedness of $G$, it suffices to consider an arbitrary point $\langle f, g \rangle: 1 \to P \times A$. The subsumption map is used to
produce the edge \((g)\): \(1 \leftrightarrow A\) and a square of the following shape.

\[
\begin{array}{c}
1 \xrightarrow{id} 1 \\
\downarrow (id) = 1 \\
1 \xrightarrow{g} A
\end{array}
\]

Since \(\beta: P \to X \Rightarrow X\) is a parametric transformation, there is a square of the following shape.

\[
\begin{array}{c}
P \xrightarrow{\beta_1} 1 \Rightarrow 1 \\
\downarrow I_P \\
I \xrightarrow{\langle g \rangle \Rightarrow \langle g \rangle} A \Rightarrow A
\end{array}
\]

These squares are combined with \(I(f)\), the identity square on \(\langle g \rangle\), and a square guaranteed to exist by the parametricity of \(ap\) to get a square as follows.

\[
\begin{array}{c}
1 \langle f, id \rangle \xrightarrow{P \times 1} \beta_1 \times id \xrightarrow{1 \Rightarrow 1 \times 1} ap \xrightarrow{1} 1 \\
\downarrow I \\
1 \langle f, g \rangle \xrightarrow{P \times A} \beta_A \times id \xrightarrow{A \Rightarrow A \times A} ap \xrightarrow{A} A
\end{array}
\]

Since there is a unique morphism \(!_1: 1 \to 1\), the morphism along the top of this square is \(id_1\). Translating this square into a commuting diagram (from the fact \(G\) is subsumptive) yields that \(ap \circ (\beta \times id) \circ \langle f, g \rangle = g\). We can also show that \(ap \circ (\omega_A \times id) \circ (l_P \times id) \circ \langle f, g \rangle = g\). This uses the definition of \(\omega_A\) as \(\text{curry}(\lambda)\) and the fact that \(ap \circ (f \times id)\) is \(\text{uncurry}(f)\) for any \(f\).

\[
\begin{align*}
ap \circ (\omega_A \times id) \circ (l_P \times id) \circ \langle f, g \rangle &= ap \circ (\text{curry}(\lambda) \times id) \circ (l_P \times id) \circ \langle f, g \rangle \\
&= \text{uncurry}(\text{curry}(\lambda)) \circ (l_P \times id) \circ \langle f, g \rangle \\
&= \lambda \circ (l_P \times id) \circ \langle f, g \rangle \\
&= \lambda \circ \langle f, g \rangle = g
\end{align*}
\]

So, for every point \(\langle f, g \rangle\), we have the following equality, since both morphisms are equal to \(g\).

\[
ap \circ \beta \times id \circ \langle f, g \rangle = ap \circ (\omega_A \times id) \circ (l_P \times id) \circ \langle f, g \rangle
\]

86
Therefore the well pointedness of $G$ ensures the equality of $ap \circ (\beta_A \times \text{id})$ and $ap \circ (\omega_A \circ !_P \times \text{id})$. Since the uncurried versions are equal, we see that $\beta_A$ and $\omega_A \circ !_P$ are equal. This holds for any $A$.

In this chapter, we have presented the axioms of parametricity graphs. These requirements formalize some intuitive notions about relations and their use. The relational requirement ensures that 2-cells are families of morphisms which preserve all edges. The identity condition ensures that identity edges can be used as the equality relation. Being subsumptive means that all morphism are represented by edges, and fibred implies that the collection of edges is closed under substitution. These features combine to define a strong notion of uniformity. Representation results, such as Theorem 4.14, illustrate the strength of the uniformity requirements. In the next chapter, we show that a wide collection of representation results hold for well-pointed parametricity graphs.
Chapter 5

Representation Results

Reflexive graph categories provide adequate structure with which to formalize the preservation of edges as an abstract notion of uniformity. In the previous chapter, we proposed parametricity graphs as a setting where that uniformity is a strong and meaningful notion. In this chapter we provide evidence to support our claim that the preservation of edges in parametricity graphs is a strong notion of uniformity.

There are many instances where one can show that the collections of morphisms that preserve all edges (that is, parametric transformations) coincide with the intuitively uniform operations. For instance, there is only one parametric transformation $\text{ID}_G \to \text{ID}_G$ for any well-pointed parametricity graph $G$. This is equivalent to saying that the parametric limit of $X \Rightarrow X$ is terminal (Theorem 4.14, section 4.5) and corresponds to the intuition that the polymorphic identity function is the only uniform way to map an object of unknown type to something of the same type.

Phil Wadler gave descriptions of what intuitively uniform operations there are of some particular polymorphic types in his paper “Theorems for Free” [Wad89]. When Reynolds introduced his parametric model [Rey83], he included descriptions of the interpretation of some polymorphic types, later generalized in [RP93]. Such descriptions, later dubbed “representation results” [BFSS90, BAC95, OT95], effectively state that the interpretation of these polymorphic types contain only the intuitively uniform operations. We show that models of polymorphism in well-pointed parametricity settings satisfy similar representation results.

We approach the issue of representation results syntactically, in a way that is similar to the approaches taken in [Mai91, PA93, ACC93]. We introduce a language for reasoning with relations, and call this language System P. System F embeds in System P. The relations allow one to prove some representation results in our system. By constructing models for System
P in *well-pointed* parametricity settings, we infer that these representation results hold in all such settings. In other words, we show that certain parametric limits contain only the intuitively uniform operations. (Using the correspondence between parametric limits and parametric transformations, similar representation results could equivalently be stated and proven for parametric transformations.)

System P is modeled in *well-pointed* parametricity settings, thus the question arises as to what happens if the well-pointedness assumption is dropped. We produce an example of a non-well-pointed parametricity setting where the traditional representation results do *not* hold. However, this does not imply that parametric transformations there represent non-uniform families. As far as we have examined, parametric transformations do seem to represent only intuitively uniform families. Rather, the situation is that we do not know how to state representation results for *arbitrary* parametricity settings. We leave this issue for future work. (However, in the next chapter we study a specialized pre-sheaf-like category and the corresponding parametricity graph to give some idea of what happens in the non-well-pointed cases.)

### 5.1 Formal System of Parametricity

In any well-pointed parametricity setting for System F, the interpretations of polymorphic types are trimmed down to contain only the intuitively uniform families. We show this by way of a formal language for reasoning with relations. The polymorphic lambda calculus we introduce here is called System P. System F embeds into System P, so we will be able to use System P to reason about System F terms. Our discussion of System P begins with an informal discussion of some concepts that will appear in the judgments of System P.

In System P, we work with *relations* instead of types. We think of System P's relations as being relations between System F types. Contexts are made of *relation assumptions*, which are expressions of the following form.

$$\frac{x}{R} \frac{x'}{x}$$

Such a relation assumption indicates that $x$ and $x'$ are variables taking values that are related by $R$. Similarly, the analogue of a term judgment makes a statement of the following form to indicate that terms $M$ and $N$ are related
by $R$.

$$M^R_N$$

We are able to include typing information in this notation because every type has a corresponding relation (the identity relation on that type). So, if $A$ is a “type”, the same information as the type assertion $M:A$ is conveyed by the following statement.

$$M^A_N$$

Therefore, we shall abbreviate the above statement as $M:A$.

**Indeterminates:** Relations will in general be formed using certain term identifiers, which are called *indeterminates*. A context of type assumptions $\Gamma_0 = x_1:A_1, \ldots, x_n:A_n$ (similar to a context of System F) is used to specify indeterminates. Such a *context of types* $\Gamma_0$, along with a list $\eta_0$ of the type variables that may appear in $\Gamma_0$, are used in every judgment.

To describe System P formally, we start with two infinite collections of identifiers. (We call these *identifiers* in System P, rather than *variables* as done in System F, to reserve “variable” and “indeterminate” to be used in distinguishing between different uses of identifiers.) The meta-variables $X$, $Y$ and $Z$ are taken to range over *relation identifiers*. The meta-variables $x$, $y$ and $z$ range over *term identifiers*. The collection of relation expressions $R$ and the collection of term expressions $M$ are defined by the following context-free grammar. The type expressions $A$ of System P are identified as a sub-collection of the relation expressions.

$$A ::= X \ | \ A_1 \Rightarrow A_2 \ | \ \forall X. A$$

$$R ::= X \ | \ R_1 \Rightarrow R_2 \ | \ \forall X. R \ | \ \frac{A_0}{A_1} [M, N] R$$

$$M ::= x \ | \ (\lambda x: A. M) \ | \ M_1 \ | \ M_2 \ | \ (\lambda X. M) \ | \ M[R]$$

The types of System P are given by equivalence classes of type expressions, where type expressions that differ only in the choice of type identifier in a quantified type expression are equated in the usual manner. Similarly pre-relations are equivalence classes of relation expressions equating expressions that differ only in the choice of identifier used in quantified expressions. We maintain the usual convention of using a type or relation to denote its equivalence class.

We define functions $\partial_0$ and $\partial_1$ from relation expressions to type expres-
sions. It is obvious that $\partial_0$ and $\partial_1$ defined below preserve equivalence. Therefore these functions also determine functions from pre-relations to types, which we also denote by $\partial_0$ and $\partial_1$.

$$
\begin{align*}
\partial_0(X) &= X \\
\partial_0(R \Rightarrow S) &= \partial_0(R) \Rightarrow \partial_0(S) \\
\partial_0(\forall X.R) &= \forall X.\partial_0(R) \\
\partial_0(_A^0[M,N]R) &= A_0
\end{align*}
$$

$$
\begin{align*}
\partial_1(X) &= X \\
\partial_1(R \Rightarrow S) &= \partial_1(R) \Rightarrow \partial_1(S) \\
\partial_1(\forall X.R) &= \forall X.\partial_1(R) \\
\partial_1(_A^0[M,N]R) &= A_1
\end{align*}
$$

Observe that for any type expression $A$, $\partial_0(A) = A = \partial_1(A)$.

We define a relational context to be a finite sequence of distinct relation identifiers. We denote relational contexts by the meta-variable $\eta$. A relational assumption is an expression $\frac{x}{x'} R$ where $R$ is a pre-relation while $x$ and $x'$ are term identifiers. Note that there is no requirement that $x$ and $x'$ be different term identifiers. Particularly when $R$ is a type (an identity relation over a System F type), it is suggestive to use the same variable in both positions. A relation of the form $\frac{x}{x} A$ is called a type assumption, and we often use the abbreviation $x:A$ for $\frac{x}{x} A$.

A context in System P is a finite sequence of relation assumptions such that no term identifier appears in two different relation assumptions. A special case of context, which we call a context of types, occurs when all the relation assumptions are type assumptions. The metavariable $\Delta$ ranges over contexts and $\Gamma$ ranges over contexts of types. The functions $\partial_0$ and $\partial_1$ on pre-relations can be extended to map contexts to contexts of types in an apparent manner.

$$
\begin{align*}
\partial_0\left(\frac{x_1}{x'_1}, \ldots, \frac{x_m}{x'_m} R_{m} \right) &= x_1: \partial_0(R), \ldots, x_m: \partial_0(R_m) \\
\partial_1\left(\frac{x_1}{x'_1}, \ldots, \frac{x_m}{x'_m} R_{m} \right) &= x'_1: \partial_1(R), \ldots, x'_m: \partial_1(R_m)
\end{align*}
$$

There are two forms of judgment in System P.

$$
\eta \vdash_{\eta_0[\Gamma_0]} \alpha \quad \text{relation judgment}
$$

$$
\eta ; \Delta \vdash_{\eta_0[\Gamma_0]} \frac{M}{N} R \quad \text{relatedness judgment}
$$

91
The metavariable $\alpha$ above denotes either a pre-relation or a context. In all cases, $\eta$ and $\eta_0$ are disjoint relational contexts. For a relatedness judgment, there is the additional stipulation that no term identifier appears in both $\Delta$ and $\Gamma_0$. The relations of System P are those pre-relations $R$ that appear in a derivable relation judgment $\eta \vdash_{\eta_0, \Gamma_0}^M R$. Terms are those pre-terms $M$ or $N$ which appear in derivable relatedness judgments $\eta ; \Delta \vdash_{\eta_0}^M R$.

A relatedness judgment $\eta ; \Delta \vdash_{\eta_0}^M R$ intuitively asserts that the terms $M$ and $N$ are related by $R$ (in the given contexts). When the relation in question is a type (intuitively an identity relation), can be read as asserting $M$ and $N$ are equal. We are therefore motivated to adopt the following conventions.

**Conventions:**

- If $A$ is a type, $M =_A N$ is used to abbreviate $M \equiv N$. The subscript $A$ will frequently be dropped when it can be inferred.

- For any pre-term $M$, we use the type assertion $M : A$ as an abbreviation for $M : A$.

The identifiers that appear in the subscript of the turnstile ($\vdash$) in a given judgment are called indeterminates of that judgment. Therefore the relational context $\eta_0$ and context $\Gamma_0$ appearing in the subscript of the turnstyle are together referred to as the indeterminate zone. In contrast, the relational context $\eta$ (and context $\Delta$, for the case of relatedness judgments) appearing to the left of the turnstyle are called the variable zone. Any type identifier in $\eta$ is called a type variable, and the term identifiers $x$ and $y$ of a relation assumption $x \in R$ in the variable zone are term variables. Using the same terminology as in System F, an identifier of System P is fresh relative to a collection of judgments and expressions provided that identifier does not appear in any of the judgments nor expressions.

The relation judgments, context judgments and relatedness judgments of System P are defined simultaneously. The judgments of System P are those judgment that are the conclusion of finite derivations using the following rules. The rules for relation judgments are given in table 5.1 (page 93).

The first four rules are similar to corresponding rules for System F, with indeterminates being treated just like variables were in System F. The \{rel\prec\} rule is for “pre-image” relations. Intuitively, the relation $\Delta \vdash_{\eta_0}^A[M, N]R$ relates those elements of $A_0$ and $A_1$ which, upon the respective action of $M$
\[ X_1, \ldots, X_n \vdash_{\eta_0} \emptyset \quad 1 \leq j \leq n \] 
\[ X_1, \ldots, X_n \vdash_{\eta_0} X_j \quad \{\text{rel\_var}\} \]
\[ \eta \vdash_{X_1, \ldots, X_n, \emptyset} \emptyset \quad 1 \leq j \leq n \quad \{\text{rel\_ind}\} \]
\[ \eta \vdash_{\eta_0} R \quad \eta \vdash_{\eta_0} S \quad \{\text{rel\_fun}\} \]
\[ \eta \vdash_{\eta_0} R \quad \eta \vdash_{\eta_0} \lambda_0[M, N] R \quad \{\text{rel\_pre}\} \]
\[ \eta \vdash_{\emptyset} \emptyset \quad \{\text{cont\_emp}\} \]
\[ \eta \vdash_{\eta_0} \emptyset \quad (\eta_0, \Gamma_0) \neq (\emptyset, \emptyset) \quad \{\text{cont\_ind}\} \]
\[ \eta \vdash_{\eta_0} \Delta \quad \eta \vdash_{\eta_0} R \quad x \text{ and } y \text{ are fresh} \quad \{\text{cont\_ext}\} \]

Table 5.1: Rules for Relation Judgments

and \( N \), give \( R \)-related elements of \( \partial_0(R) \) and \( \partial_1(R) \). Note that \( M \) and \( N \) can contain indeterminates, but not ordinary term variables.

The rules \{\text{cont\_emp}\} and \{\text{cont\_ext}\} state the obvious formation rules for contexts. The other rule is for forming contexts with indeterminates.

The rules to form relatedness judgments are given in tables 5.2 (page 94) and 5.3 (page95). The rules to form relatedness judgments can be divided into two kinds. The first are term forming rules, and correspond to term judgments of System F. The rule \{\text{ind}\} is for using indeterminates, which are used just like variables are used. The rules \{\text{preJ}\} and \{\text{preE}\} signify that \( M' \)
\[ \lambda_0[M, N] R \quad \text{holds if and only if} \quad M M' \]
\[ N N' \quad \{\text{polyJ}\} \]

adapted from System F. In \{\text{polyJ}\}, the hypothesis \( \eta \vdash_{\eta_0} \Delta \) ensures that the relation variable \( Y \) does not appear free in any relation of \( \Delta \).

The final rules for System P correspond to equational rules of System F. Note that to assert \( M \) and \( N \) are equal terms, we use types (intuitively, identity relations). \( M \) and \( N \) are equal if they are related by an identity relation
\[
\begin{align*}
\eta \models_{\eta_0, \Gamma_0} x_1^{R_1}, \cdots, x_m^{R_m} & \quad \text{1 \leq j \leq m} \quad \text{\{var\}} \\
\eta ; x_1^{R_1}, \cdots, x_m^{R_m} & \quad \text{1 \leq j \leq m} \quad \text{\{ind\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} x_j : A_j \\
\eta ; \Delta, \frac{x}{y} R \models_{\eta_0, \Gamma_0} M S & \quad \partial_0(R) = A \quad \partial_1(R) = B \quad \text{\{fun\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} R \models S & \quad \text{(\lambda x : A.M)} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} \frac{R \models S}{M \models N} & \quad \text{(\lambda y : B.N)} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} M \\ M' \models N' & \quad \eta \models_{\eta_0, \Gamma_0} S \quad \text{\{fun\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} (\lambda Y.M) \forall Y.R & \quad \text{\{poly\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} (\lambda Y.N) & \quad \text{\{poly\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} M \\ A \models \frac{M[A]}{N[B]} \models \frac{R^S[A]}{\text{\{poly}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} M \\ M', A_0[M, N] \models \frac{R \models S}{\text{\{pre\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} A_0[M, N] R & \quad \text{\{pre\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} M \models M' \\ A_0[M, N] \models \frac{R \models S}{\text{\{pre\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} M \models M' \\ \text{\{pre\}} \\
\eta ; \Delta \models_{\eta_0, \Gamma_0} M \models M' \\ \text{\{pre\}} \\
\end{align*}
\]

Table 5.2: Rules for Relatedness Judgments: Term Forming Rules
\[ \eta; \Gamma, x: A \vdash_{\gamma_0} M : B \quad y \text{ is fresh} \]

\[ \eta; \Gamma \vdash_{\gamma_0} \lambda x: A.M =_{A \Rightarrow B} \lambda y: A.M[y/x] \quad \text{[\texttt{alpha\_fun}]} \]

\[ \eta, X; \Gamma \vdash_{\gamma_0} M : B \quad \eta \vdash_{\gamma_0} \Gamma \quad Y \text{ is fresh} \]

\[ \eta; \Gamma \vdash_{\gamma_0} \Lambda X.M =_{\nu X.B} \Lambda Y.M[y/X] \quad \text{[\texttt{alpha\_poly}]} \]

\[ \eta; \Gamma, x: A \vdash_{\gamma_0} M : B \quad \eta \vdash_{\gamma_0} N : A \]

\[ \eta; \Gamma \vdash_{\gamma_0} (\lambda x: A.M) N =_{B^{[A/B]}} M[y/x] \quad \text{[\texttt{beta\_fun}]} \]

\[ \eta; \Gamma \vdash_{\gamma_0} M : A \Rightarrow B \quad x \notin \Gamma, \Gamma \]

\[ \eta; \Gamma \vdash_{\gamma_0} M =_{A \Rightarrow B} (\lambda x: A.M x) \quad \text{[\texttt{eta\_fun}]} \]

\[ \eta, X; \Gamma \vdash_{\gamma_0} M : B \quad \eta \vdash_{\gamma_0} \Gamma \quad \eta; \Gamma \vdash_{\gamma_0} A \]

\[ \eta; \Gamma \vdash_{\gamma_0} ([A] =_{\nu A.B} [A[y/X]] \quad \text{[\texttt{beta\_poly}]} \]

\[ \eta; \Gamma \vdash_{\gamma_0} \forall X.A \quad \eta; \Gamma \vdash_{\gamma_0} \forall X.A \quad \text{[\texttt{eta\_poly}]} \]

\[ \eta; \Gamma \vdash_{\gamma_0} M =_{A} N \quad \eta; \Gamma \vdash_{\gamma_0} N =_{A} M \quad \text{[\texttt{eq\_symm}]} \]

\[ \eta; \Delta \vdash_{\gamma_0} M \quad \eta; \partial_1(\Delta) \vdash_{\gamma_0} N =_{\partial_1(R)} N' \]

\[ \eta; \Delta \vdash_{\gamma_0} M \quad \eta; \Delta \vdash_{\gamma_0} N \quad \text{[\texttt{trans\_w}]} \]

\[ \eta; \Delta \vdash_{\gamma_0} M \quad \eta; \partial_0(\Delta) \vdash_{\gamma_0} M =_{\partial_0(R)} M' \]

\[ \eta; \Delta \vdash_{\gamma_0} M' \quad \eta; \Delta \vdash_{\gamma_0} N \quad \text{[\texttt{trans\_j}]} \]

\[ \eta; \Gamma \vdash_{\gamma_0} M =_{A} N \quad \eta, X; \Gamma \vdash_{\gamma_0} M =_{A} N \quad \text{[\texttt{eq\_rel}]} \]

\[ \eta; \Gamma \vdash_{\gamma_0} M =_{A} N \quad \eta; \Gamma \vdash_{\gamma_0} M =_{B} N \quad \text{[\texttt{eq\_term}]} \]

Table 5.3: Rules for Relatedness Judgments: Equational Rules

95
provided their free variables are related by identity relations. Therefore the equational rules of System P use contexts of types.

The first eight of these equational rules are adaptations of the equational rules of System F. The last two rules imply that equalities do not depend on the distinction between variables and indeterminates. Any equalities using indeterminates are also equalities using corresponding variables. The reverse implication, equalities using variables are equalities using indeterminates, is an admissible rule. An even more general property holds. However, we first present some expected structural properties of System P. One can always add unused identifiers to judgments and also permute the order of identifiers in any context or relational context.

**Lemma 5.1 (Weakening and Permutation)**

Suppose there are derivations for \( \eta \vdash_{\eta_0} \alpha \), \( \eta_0, \eta_1 \vdash \alpha \) and \( A \) is a type. Further suppose \( \tilde{\eta}, \tilde{\eta}_0, \tilde{\Delta}, \tilde{\Gamma}_0 \) are permutations of \( \eta, \eta_0, \Delta, \Gamma_0 \).

1. If \( \eta \vdash_{\eta_0} \alpha \) is derivable, then so is \( \eta, \eta' \vdash_{\eta_0, \eta_1} \alpha \).

2. If \( \eta, \Delta \vdash_{\eta_0} \alpha \) is derivable, then so is \( \eta, \eta' \vdash_{\eta_0, \eta_1} \alpha \).

3. If \( \eta \vdash_{\eta_0} \alpha \) is derivable, then so is \( \eta \vdash_{\eta_0, \alpha} \alpha \).

4. If \( \eta, \Delta \vdash_{\eta_0} \alpha \) is derivable, then so are \( \eta, \Delta \vdash_{\eta_0, \alpha} \alpha \).

5. If \( \eta \vdash_{\eta_0} \alpha \) is derivable, then so is \( \tilde{\eta} \vdash_{\tilde{\eta}_0} \alpha \).

6. If \( \eta, \Delta \vdash_{\eta_0} \alpha \) is derivable, then so is \( \tilde{\eta}, \tilde{\Delta} \vdash_{\tilde{\eta}_0} \alpha \).

**Proof.** The proof proceeds by induction on derivations in a straightforward manner. There are interdependencies between the various points of the lemma, and hence the induction is performed simultaneously for all points. As an instance of a dependency on a separate point, in the step corresponding to \( \{ \text{fun} \} \) for point 4, we require an application of point 6 to reorder the context in the inductive hypothesis, given below.

\[
\eta, \Delta \vdash_{\eta_0} \alpha \quad \eta' \vdash_{\eta_0} \alpha \quad M \quad N
\]
Lemma 5.2

Suppose that \( A \) is a type such that \( \varphi_0 \vdash A \).

- \( \Gamma, \Delta \vdash u : A \) \( \Gamma, \Delta \vdash v : B \) \( \Gamma, \Delta \vdash A \to B \) is derivable, then so is \( \varphi_0 \vdash x_{i_0} \).

- \( \varphi_0 \vdash M \) \( \varphi_0 \vdash N \) \( \varphi_0 \vdash M \to N \) is derivable, then so is \( \varphi_0 \vdash x_{i_0} \).

- \( \varphi_0 \vdash \alpha \) is derivable, then so is \( \varphi_0 \vdash x_{i_0} \).

Proof. This lemma is shown by induction on derivations. There are only three rules where variables are introduced into relational contexts — \( \text{cond} \), \( \text{var} \), and \( \text{eq} \). We only consider these six interesting cases.

- \( \varphi_0 \vdash \alpha \) is derivable as a result of \( \text{eq} \).
- \( \varphi_0 \vdash \alpha \) is derivable as a result of \( \text{var} \).
- \( \varphi_0 \vdash \alpha \) is derivable as a result of \( \text{cond} \).

Weakening is a very useful property which will reappear frequently. As a first instance, weakening will be helpful in proving that for any derivable judgment using variables, there is a corresponding derivable using indeterminates.

All of the steps in the induction are no way surprising and can be reconstructed directly by the reader familiar with similar inductive proofs.
If $\eta ; \Delta, x : A \vdash x_j^{\eta_0} A_j$ is derivable as a result of \{ind\}, then we can conclude $\eta \vdash^{\Gamma_0} \Delta, x : A$ is derivable. The only rule which produces judgments of this form is \{cont\ext\}, so there must be a derivation of the judgment $\eta \vdash^{\Gamma_0} \Delta$. Weakening (lemma 5.1) is used to deduce the judgment $\eta \vdash^{\Gamma_0} \Delta, x : A \vdash x_j^{\eta_0} A_j$ follows from \{ind\}.

If $\eta ; \Delta, x_m : A_m \vdash^{\Gamma_0} x_j^{\eta_0} A_j$ is derivable as a result of \{var\}, then a similar argument as in the previous case can be used to deduce the judgment $\eta \vdash^{\Gamma_0, x_m : A_m} \Delta$ is derivable. The judgment $\eta ; \Delta \vdash^{\Gamma_0, x_m : A_m} x_j^{\eta_0} A_j$ then follows from \{var\} (if $j \leq m - 1$) or \{ind\} (if $j = m$). (Note that in the case $j = m$, then $x_j' = x_j$ since only one variable was used in the $m$th relational assumption $x_m : A_m$.)

Permutations of contexts and relational contexts (lemma 5.1) can be combined with the above lemma to conclude that the position of a variable in a context or relational context is not relevant to whether that variable can be shifted to an indeterminate. Note that the hypothesis $\eta_0 \vdash^{\Gamma_0} A$ requires that all the free relation identifiers of the type $A$ will necessarily be indeterminates in the resulting judgment $\eta ; \Delta \vdash^{\Gamma_0} x : A \vdash^{\Gamma_0} R^N$.  

There are other expected properties of System P that one can prove. These range from basic structural results, which support the well-formedness of the system, to handy shortcuts useful for reasoning in the system. We shall mention a few which will be relevant to future analysis. The first captures the intuition that variables are place holders which can be replaced by values (of the appropriate kind).

**Lemma 5.3 (Substitutions)**

- If $\eta, X \vdash^{\Gamma_0} R$ and $\eta \vdash^{\Gamma_0} S$ are derivable and no relation identifier of $S$ is bound in $R$, then $\eta \vdash^{\Gamma_0} R^{[S/X]}$ is derivable.

- If $\eta, X ; \Delta \vdash^{\Gamma_0} M^N \vdash^{\Gamma_0} S$ are derivable and no identifier of $S$ is bound in $R$, then $\Delta \vdash^{[S/X]} M^{[\delta_0(S)/X]} \vdash^{[S/X]} N^{[\delta_1(S)/X]}$ is derivable.

- If $\eta ; \Delta, x \vdash^{\Gamma_0} M^R$ and $\eta ; \Delta \vdash^{\Gamma_0} M^{N'}$ are derivable, and no identifier of $M'$, $N'$ or $S$ is bound in $M$, $N$ or $R$, then
\[\eta \vdash \frac{M^{[M'/x]}}{N^{[N'/y]}} R \text{ is derivable.}\]

**Proof.** Any derivations of the hypotheses can be used to produce a derivation of the conclusion. For the first point, the derivation of \(\eta, X \vdash \eta_0 \Gamma_0 R\) is modified by replacing all sub-derivations for a relation judgment of \(X\) by a derivation for a relation judgment of \(S\). This might require weakening (lemma 5.1) be applied to the given derivation of \(\eta \vdash \eta_0 \Gamma_0 S\). For instance, a sub-derivation resulting in \(\eta, X, Y \vdash \eta_0 \Gamma_0 X\) is replaced by a derivation of \(\eta, Y \vdash \eta_0 \Gamma_0 S\).

The other points are proved similarly. \(\diamond\)

Indeterminates are also place holders. They can be replaced, although the relations they can be replaced by is limited to only types.

**Proposition 5.4 (Substitution for Indeterminates)**

Suppose both \(\eta \vdash \eta_0 \Gamma_0 B\) and \(\eta_0 \vdash \eta_0 N : B\) are derivable. Suppose further that \(B\) is a type and that no identifier of \(N\) or \(B\) is bound in \(M\), \(M'\) or \(R\).

- If \(\eta \vdash \eta_0 X[\Gamma_0 R\text{ is derivable, then so is } \eta \vdash \eta_0 \Gamma_0 \text{ with } R[M/B/X] R[M'/B/X].\)

- If \(\eta \vdash \eta_0 X[\Gamma_0 R\text{ is derivable, then so is } \eta \vdash \eta_0 \Gamma_0 \text{ with } R[M/B/X] R[M'/B/X].\)

- If \(\eta \vdash \eta_0 \Gamma_0 x:B R\text{ is derivable, then so is } \eta \vdash \eta_0 \Gamma_0 \text{ with } R[N/x].\)

- If \(\eta \vdash \eta_0 \Gamma_0 x:B R\text{ is derivable, then so is } \eta \vdash \eta_0 \Gamma_0 \text{ with } R[N/x].\)

This is proved in a manner analogous to the previous lemma on substitution for variables.

Terms of type \(A \Rightarrow B\) are general thought of, and referred to, as functions from the type \(A\) to the type \(B\). As one expects, there is a canonical identity function for every type \(A\) and functions can be composed.

**Conventions:**

- For any type \(A\), we use \(id_A\) to denote \(\lambda x:A.x\) where \(x\) is fresh. The subscript \(A\) may be suppressed when it can be inferred.

- Suppose \(\eta : \Gamma \vdash \eta_0 \Gamma_0 M : A \Rightarrow B\) and \(\eta : \Gamma \vdash \eta_0 \Gamma_0 N : B \Rightarrow C\). We use \(N \circ M\) to denote \(\lambda x:A.N(M x)\) where \(x\) is fresh.
It is immediate to see that these terms are well-defined. That is, the following derivations are derivable rules of System P.

\[
\frac{\eta \vdash \eta_0 \Gamma_0 A}{\eta ; \emptyset \vdash \eta_0 \Gamma_0 \text{id}_A : A \Rightarrow A}
\]

\[
\frac{\eta ; \Gamma \vdash \eta_0 \Gamma_0 M : A \Rightarrow B \quad \eta ; \Gamma \vdash \eta_0 \Gamma_0 N : B \Rightarrow C}{\eta ; \Gamma \vdash \eta_0 \Gamma_0 \text{id} \circ M}
\]

Just as in System F, identities and composites can be used to identify isomorphic types in the usual categorical fashion.

**Definition 5.5**

For any types A and B such that \( \eta \vdash \eta_0 \Gamma_0 A \) and \( \eta \vdash \eta_0 \Gamma_0 B \) and any context of types \( \Gamma \), a pair of functions \((M, M^{-1})\) is called an isomorphism from A to B if the following judgments are derivable.

\[
\eta ; \Gamma \vdash \eta_0 \Gamma_0 M^{-1} \circ M = \text{id}_A
\]

\[
\eta ; \Gamma \vdash \eta_0 \Gamma_0 M \circ M^{-1} = \text{id}_B
\]

For any relations R and S such that \( \eta \vdash \eta_0 \Gamma_0 R \) and \( \eta \vdash \eta_0 \Gamma_0 S \) and any relational context \( \Delta \), we say R and S are isomorphic relations if there exist isomorphisms \((M, M^{-1})\) from \( \partial_0(R) \) to \( \partial_0(S) \) and \((N, N^{-1})\) from \( \partial_1(R) \) to \( \partial_1(S) \) such that the following judgments hold.

\[
\eta ; \Delta \vdash \eta_0 \Gamma_0 M \Rightarrow S
\]

\[
\eta ; \Delta \vdash \eta_0 \Gamma_0 N^{-1} \Rightarrow R
\]

We use the notation \( \eta \vdash \eta_0 \Gamma_0 R \cong S \) to indicate that R and S are isomorphic.

A special case of isomorphic relations can occur when two relations R and S are both relations between the same two types, say A and B. We can describe the situation when R and S relate the same pairs of elements \((a, b)\) as R and S being isomorphic as witnessed by the isomorphisms \((\text{id}_A, \text{id}_A)\) and \((\text{id}_B, \text{id}_B)\). This can be stated more directly as follows.

**Definition 5.6**

Suppose \( \eta \vdash \eta_0 \Gamma_0 R \) and \( \eta \vdash \eta_0 \Gamma_0 S \) are derivable. We say that R and S are equivalent provided there are derivable judgments \( \eta \vdash \eta_0 \Gamma_0 x R \Rightarrow S \) and \( \eta \vdash \eta_0 \Gamma_0 y S \Rightarrow R \). We use the notation \( \eta \vdash \eta_0 \Gamma_0 R \equiv S \) to denote that R and S are equivalent.
Since equivalent relations contain the same elements, one would expect that they are interchangeable. It is the case that they are, in the following senses.

**Lemma 5.7**

If $\eta \vdash_{\Gamma_0} R \equiv R'$, then:

- $\eta ; \Delta \vdash_{\Gamma_0} M \quad \frac{R}{N} \quad \text{if and only if} \quad \eta ; \Delta \vdash_{\Gamma_0} M \quad \frac{S}{N}$.

- $\eta \vdash_{\Gamma_0} \frac{R}{N} \quad \text{if and only if} \quad \eta \vdash_{\Gamma_0} \frac{S}{N}$.

- For any $\eta, X \vdash_{\Gamma_0} T$, one has $\eta \vdash_{\Gamma_0} T[R/X] \equiv T[S/X]$.

**Proof.** These are direct results of substitution (lemma 5.3). For instance, one direction of the first point uses applies substitution using the following two judgments.

\[
\eta ; \Delta \vdash_{\Gamma_0} M \quad \frac{R}{N} \quad \eta ; \Delta \vdash_{\Gamma_0} M \quad \frac{S}{N}
\]

The judgment on the right comes from weakening (lemma 5.1) one of the judgments defining the equivalence. The substitution gives the following judgment, as desired.

\[
\eta ; \Delta \vdash_{\Gamma_0} M \quad \frac{S}{N}
\]

Recall that re-indexing in a parametricity graph can be used to define a subsumption map (Theorem 4.10). An analogous construction can be used in System P. We therefore introduce the following notation.

**Convention:** For any term $M$ such that $\eta ; \emptyset \vdash_{\Gamma_0} M : A \Rightarrow B$, we use $\langle M \rangle$ to denote the relation $\frac{A}{B}[M, \text{id}_B]B$.

It is immediate that $\langle M \rangle$ gives a valid relation and that it captures the intuitive concept of the graph of $M$ in the following sense.

**Lemma 5.8**

Suppose $\eta ; \emptyset \vdash_{\Gamma_0} M : A \Rightarrow B$.

- $\eta \vdash_{\Gamma_0} \langle M \rangle$ is derivable.

- $\eta ; \Delta \vdash_{\Gamma_0} \langle M \rangle \quad \text{is derivable if and only if} \quad \eta ; \Delta \vdash_{\Gamma_0} M \quad N \overset{N'}{=} N'$ is derivable.
Proof. The relation judgment follows from \{rel_{\text{pre}}\}. The left to right implication is a consequence of \{\text{pre}_E\} while the reverse implication is an instance of \{\text{pre}_J\}.

\[ \]

5.2 Relation to Other Systems

In this section, we compare System P to some other polymorphic lambda calculi. Since our purpose for introducing System P is to reason about models of System F, it is not surprising that we show System F can be embedded into System P. In the context of comparing System F and System P, we produce some expected structural results about System P that will prove useful in later sections. We also contrast System P to System R, a calculus proposed by Abadi, Cardelli and Curien [ACC93] which also extends System F to include relations.

The relationship between System P and System F arises because many of the rules for System P are analogous to rules for System F. In this analogy, indeterminates in System P are treated as variables in System F. It will be handy to identify the rules which correspond to forming term judgments and type judgments in System F.

Definition 5.9

The following rules are called pure rules of System P.

\[
\{
\text{rel}_{\text{var}}, \{	ext{rel}_{\text{ind}}\}, \{	ext{rel}_{\text{fun}}\}, \{	ext{rel}_{\text{poly}}\}, \\
\text{cont}_{\text{gmp}}, \{	ext{cont}_{\text{ind}}\}, \{	ext{cont}_{\text{ext}_{\text{pure}}}\} \\
\{	ext{var}\}, \{	ext{ind}\}, \{	ext{fun}_{\text{f}}\}, \{	ext{fun}_{\text{E}}\}, \{	ext{poly}_{\text{f}}\}, \{	ext{poly}_{\text{E}}\}
\]

The rule \{\text{cont}_{\text{ext}_{\text{pure}}}\} is a restriction of the rule \{\text{cont}_{\text{ext}}\} with the additional stipulation that the same variable is used for both positions.

\[
\eta \vdash_\eta \alpha_0 \Delta \quad \eta \vdash_\eta \alpha_0 R \quad x \text{ is fresh} \\
\eta \vdash_\eta \alpha_0 \Delta, \frac{x}{x} R \\
\]

A derivation of System P is pure if it uses only pure rules.

One handy property of pure derivations is that they are unique. One can say even more about the uniqueness of derivations of relation judgments.

Lemma 5.10 (Unique Derivations)

For any relation judgment \eta \vdash_\eta \alpha_0 \alpha, there is a unique rule having having it as the conclusion.
Any pure derivation of a relation judgment $\eta \vdash_{\Gamma_0} R \alpha$ is the only derivation of that judgment in System P.

For any relatedness judgment $\eta \vdash_{\Gamma_0} M \frac{R}{N}$, there is at most one pure rule having it as the conclusion.

For any relatedness judgment $\eta \vdash_{\Gamma_0} M \frac{R}{N}$, there is at most one pure derivation of it.

**Proof.** If $\alpha$ is a relation $R$, the structure of $R$ uniquely determines the last rule used in any derivation of $\eta \vdash_{\Gamma_0} R$ unless $R$ is a relation identifier. Relation identifiers appear as the relation in the conclusion for both \{rel\} and \{ind\}. These can be distinguished based on whether the identifier is a variable (that is, in $\eta$) or an indeterminate (that is, in $\eta_0$) in the desired judgment $\eta \vdash_{\Gamma_0} X$. If $\alpha$ is a non-empty context $\Delta$, $\frac{R}{R}$, it must have been derived using \{context\}. When $\alpha$ is the empty context, examining the indeterminate zone will determine if this judgment is a result of \{cont\} (if the indeterminate zone is empty) or \{cont\}.

The second claim is shown by induction on derivations. For each pure rule that has a relation judgment for a conclusion, all the hypotheses are themselves relation judgments or meta-level observations (such as $x$ and $y$ being fresh). Since the last rule used in deriving $\eta \vdash_{\Gamma_0} \alpha$ is uniquely determined, and the hypotheses to that rule have unique derivations, the judgment in question has a unique derivation.

An approach similar to the one used for relation judgments is used in proving the claims about relatedness judgments. The structure of the top term $M$ in the relatedness assertion $\frac{M}{N} \frac{R}{R}$ will determine which pure rule (if any) leads to the conclusion $\eta \vdash_{\Gamma_0} \alpha$ unless $M$ is a term identifier. One can determine if \{var\} or \{ind\} leads to the desired judgment based on whether the identifier is a variable or an indeterminate. (Similarly, the bottom term $N$ would determine which pure rule, if any, would lead to the desired judgment.)

Induction on derivations are used to show that pure derivations of relatedness judgments are unique. The uniqueness of pure derivations of relation judgments $\eta \vdash_{\Gamma_0} \alpha$ are used in showing that pure derivations ending in \{var\} and \{ind\} are unique.

The pure rules of System P correspond to rules of System F and can be used to embed the types and terms of System F into System P. This
embedding can be stated formally in the following lemma. We presume that the type variables and term variables of System F are the same as the relation identifiers and term identifiers of System P. To help avoid confusion, judgments of System F will have their turnstyles superscripted with an F.

**Lemma 5.11**

- If $\eta \vdash^F \alpha$ holds in System F, then there is a pure derivation of $\eta \vdash^P \alpha$ in System P.
- If $\eta ; \Gamma \vdash^F M : A$ holds in System F, then there is a pure derivation of $\eta ; \Gamma \vdash^P M : A$ in System P.

**Proof.** The first point is first shown for types $\alpha = A$ by induction on the structure of the type $A$. If $A$ is a type variable, then $\eta \vdash^F A$ arises as the result of an instance of \{rel_var\} (where \{cont_emp\} provides the hypothesis). Function types and polymorphic types follow from applications of \{rel_fun\} and \{rel_poly\} to relation judgments assured by the inductive hypothesis.

Having shown the result for any type $A$, the case where $\alpha$ is a context $\Gamma$ can be shown by induction on the length of $\Gamma$. The hypothesis $\eta \vdash^F A$ needed for \{cont_ext Pure\} is assured since $\eta \vdash^P A$ for any $A$ in the context $\Gamma$.

The other point is shown by transforming a derivation of a term judgment in System F into a derivation in System P. The previous point provides P-derivations necessary to replace the leaves of an F-derivation, while each term forming rule of System F has a corresponding rule in System P which can be used to replace each internal node in the F-derivation. 

The above embedding provides a manner in which to view System P as an extension of System F. We will show that those judgments of System P that have a pure derivation are the judgments in the image of this embedding. As a step in that direction, we note that the judgments that have pure derivations in System P only involve types, contexts of types and terms.

**Lemma 5.12**

- If there is a pure derivation of $\eta \vdash^F_0 R$, then $R$ is a type.
- If there is a pure derivation of $\eta \vdash^F_0 \Delta$, then $\Delta$ is a context of types.
- If there is a pure derivation of $\eta ; \Delta \vdash^F_0 M \frac{N}{R}$, then $M$ and $N$ are syntactically equal and $R$ is a type.

**Proof.** This is shown by induction on derivations. For each of the pure rules with $\eta \vdash^F_0 R$ as the conclusion, the induction hypothesis will ensure
all of the relations appearing in hypotheses are in fact types. Hence \( R \) will remain in the collection of types.

The base cases involving contexts, namely \{cont.smp\} and \{cont.ind\}, only produce the empty context, which is trivially a context of types. The pure rule \{cont.ext.pure\} will produce a context of types as the induction hypothesis ensures \( \Delta \) is a context of types and the previous point ensures the new relation is a type.

The pure rules resulting in relatedness judgments necessarily introduce the same syntax in the top and the bottom term. The induction hypothesis will ensure that the only relations appearing in the hypothesis of the final rule in a pure derivation are types. Therefore the relation \( R \) in a relational assertion \( \frac{M}{N} R \) in the conclusion of a pure derivation must be a type. \( \diamond \)

The relationship between types and relation judgments that have pure derivations is even closer than claimed in the previous lemma. For any type \( A \), all derivations of a \( \eta \vdash_{\eta_0} \alpha \vdash_{\eta_0} A \) are pure.

**Lemma 5.13**

- If there is a derivation of \( \eta \vdash_{\eta_0} \alpha \vdash_{\eta_0} R \) that is not pure, then \( R \) is not a type.

- If there is a derivation of \( \eta \vdash_{\eta_0} \Delta \vdash_{\eta_0} \) that is not pure, then \( \Delta \) is not a context of types.

**Proof.** These are shown by simultaneous induction. The only relation forming rule that is not pure is \{rel.pure\} which produces a relation that is not a type. All the pure relation forming rules will preserve the property of not being a type. The only non-pure context forming rule is the general case of \{cont.ext\}. Thus different identifiers are used, making the resulting context \( \Delta, \frac{x}{y} R \) not a typing context. \( \diamond \)

Lemmas 5.12 and 5.13 imply that the types of System P (intuitively, identity relations over types of System F) are precisely those relations that have a relation judgment with a pure derivation. The correspondence between System P and System F can be made more precise with the following converse of lemma 5.11. Judgments with pure derivations in System P correspond to judgments derivable in System F.

**Lemma 5.14**

- If there is a pure derivation of \( \eta \vdash_{\eta_0} \alpha \vdash_{\eta_0} \), then \( \eta, \eta_0 \vdash_{\alpha} \) holds in System F.
• If there is a pure derivation of \( \eta ; \Gamma \vdash \eta_0 \ M : A \), then one can derive 
\( \eta, \eta_0 ; \Gamma, \Gamma_0 \vdash M : A \) in System F.

**Proof.** The syntax introduced in relations by pure rules is the syntax of System F types. Since the relational context keeps track of all the free relation identifiers (type variables), and contexts are formed by adding type assumptions (a special case of relation assumption) to empty context, relation judgments with pure derivations give rise to type judgments of System F.

A pure derivation of \( \eta ; \Gamma \vdash \eta_0 \ M : A \) can be modified to produce an F-derivation of \( \eta, \eta_0 ; \Gamma, \Gamma_0 \vdash M : A \) by induction on derivations. The hypothesis of each pure term forming rule is transformed into a judgment of System F (either by the induction hypothesis or one of the previous points of this lemma). These judgments of System F provide the hypothesis to a corresponding term forming rule of System F whose conclusion is the desired judgment.

We can think of pure derivations as being System F derivations. Therefore judgments which have pure derivations correspond to type judgments and term judgments of System F. Our intuitive reading of a general relatedness judgment in System P is as an assertion that two terms are related by the stated relation. This relationship between a relatedness judgment and other judgments is but one of many expected results about how judgments relate to one another.

**Lemma 5.15 (Implied Judgments)**

1. If \( \eta \vdash \eta_0 \Delta \), then \( \eta \vdash \eta_0 \ R \) for each \( x \) \( R \) in \( \Delta \).
2. If \( \eta \vdash \eta_0 \Delta \), then \( \eta \vdash \eta_0 \ A \) for each \( x : A \) in \( \Gamma_0 \).
3. If \( \eta \vdash \eta_0 \Delta \), then \( \eta_0 \vdash \Gamma_0 \).
4. If \( \eta ; \Delta \vdash \eta_0 \ R \), then \( \eta \vdash \eta_0 \ R \) and \( \eta \vdash \eta_0 \Delta \).
5. If \( \eta \vdash \eta_0 \alpha \) is derivable in System P, then there are pure derivations for \( \eta \vdash \eta_0 \partial_0 (\alpha) \) and \( \eta \vdash \eta_0 \partial_1 (\alpha) \).
6. If \( \eta \vdash \eta_0 \alpha \) is derivable in System P, then there are pure derivations of \( \eta \ ; \partial_0 (\Delta) \vdash \eta_0 \ M : \partial_0 (R) \) and \( \eta \ ; \partial_1 (\Delta) \vdash \eta_0 \ N : \partial_1 (R) \).

**Proof.** Each of part is shown by induction on derivations, with some of the later points relying on earlier points. Some of the steps in the inductions
will make use of earlier lemmas as well. We mention a few examples of these dependencies.

The Unique Derivations lemma (lemma 5.10) is used frequently, such as in the induction step corresponding to the rule \{funJ\} to prove the fourth assertion of this lemma. For this step, we are required to show \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} \Delta \)
from the hypothesis \( \eta ; \Delta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} R \Rightarrow S \). The induction hypothesis will yield \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} \Delta, \quad x, \ y, \ z \). The Unique Derivations lemma ensures that this judgment had to have come from an application of \{cont\}. Thus, there must be a derivation of \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} \Delta \), as desired.

Similarly, in the induction step corresponding to the rule \{funE\} for the fourth point, one is required to ensure \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} S \) is derivable whenever there is a derivation of \( \eta ; \Delta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} R \Rightarrow S \). The induction hypothesis yields a derivation of \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} R \Rightarrow S \), which had to have arisen from an instance of \{rel\} using \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} S \) as a hypothesis.

The step corresponding to \{polyE\} for the fourth point, one needs to show \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} S^{[S/X]} \) is derivable. The induction hypothesis implies there is a derivation of \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} \forall X : R \). This judgment had to have come from a derivation of \( \eta, X \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} R \) via \{rel\}. Using the hypothesis \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} S \) and substitution (lemma 5.3), there must be a derivation of \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} S^{[S/X]} \) as desired.

For proving the sixth point, an interesting step is the induction step corresponding to the rule \{pre\}. From the hypothesis \( \eta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} A^\lambda[M, N] R \) (which had to have been deduced from an application of \{rel\}), we deduce \( \eta ; \Delta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} M : A \Rightarrow A_0(R) \) (which uses weakening, lemma 5.1). The inductive hypothesis yields a pure derivation of \( \eta ; \Delta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} M^{\prime} : M^{\prime} : A_0(R) \).

Since this arises as an instance of \{funE\}, there is a pure derivation of \( \eta ; \Delta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} N^{\prime} : B \) since \( A_0^\lambda[M, N] R = A \), we have the desired judgment. The pure derivation of \( \eta ; \Delta \vdash_{\eta_0}^{\Gamma \vdash_{\eta_0}} N^{\prime} : B \) is shown to exist similarly.

In proving the sixth point, the induction step corresponding to most rules use almost identical arguments to prove the two claims (one using the function \( \Delta_0 \) and the other using \( \Delta_1 \)). The similarity between \( \Delta_0 \) and \( \Delta_1 \) does not apply to steps corresponding to \( \beta \)- and \( \eta \)-equalities. However, the structure of the respective terms guides one to the pure derivation. For instance, the \( \Delta_0 \) case for \{beta\} uses \{funJ\} followed by \{funE\} as indicated by the term \( (\lambda x : A. M) N \). The \( \Delta_1 \) case uses substitution (lemma 5.3) to produce the derivation of \( \eta ; \Gamma \vdash_{\eta_0}^{\Delta \vdash_{\eta_0}} M^{[N/X]} : B \).
By combining lemmas 5.14 and 5.15, it is immediate that every System P judgment determines a pair of System F judgments.

**Corollary 5.16**

- If $\eta \vdash_{\eta_0} \alpha$ is derivable in System P, then the judgments $\eta, \eta_0 \vdash_0 \partial_0(\alpha)$ and $\eta, \eta_0 \vdash_1 \partial_1(\alpha)$ are derivable in System F.

- If $\eta; \Delta \vdash_{\eta_0} M \rightarrow N$ is derivable in System P, then the judgments $\eta, \eta_0; \partial_0(\Delta), \Gamma_0 \vdash_0 M; \partial_0(R)$ and $\eta, \eta_0; \partial_1(\Delta), \Gamma_0 \vdash_1 N; \partial_1(R)$ are derivable in System F.

By using types of System P (intuitively identity relations over System F types), we can assert the equality of System F terms. (This was the motivation for using the notation $M =_A N$ in System P.) System P includes all the equalities of System F.

**Lemma 5.17**

If $\eta; \Gamma \vdash M =_A N$ can be derived in System F, then $\eta; \Gamma \vdash \emptyset \emptyset M =_A N$ can be derived in System P as well.

**Proof.** Each equational rule of System F has a corresponding rule in System P. The F-derivation of $\eta; \Gamma \vdash M =_A N$ gives the blueprint for a corresponding P-derivation of $\eta; \Gamma \vdash \emptyset \emptyset M =_A N$.

However the converse of lemma 5.17 does not hold — equations in System P need not be equations in System F. Indeed, the point of introducing the additional structure of relations is to be able to prove more equalities. Thus models of System P, beyond merely providing models of System F, can have more said about them. We can syntactically prove representation results in System P which will lead to showing that corresponding representation results hold in parametric settings for System F.

The idea behind System P is to require more uniformity constraints in polymorphic types. The bound variable ranges over all relations, as indicated in the polymorphism elimination rule $\{\text{polyE}\}$. This matches with Reynolds' parametric polymorphic type construction. A major point of Reynolds' idea was that type variables can be instantiated at relations (resulting in relations) and that terms of polymorphic type do respect these relations. Therefore the polymorphic type construction ranges over all relations (and not just over types) in our system.

The fact that quantification ranges over relations as well as types is a major difference between our calculus and the calculus introduced by Abadi, Cardelli, and Curien \[\text{ACC93}\] to reason about uniformity syntactically. In
that calculus, called System R, type quantification and relation quantification are kept separate. A preliminary version of System R did equate the two, but Hasegawa pointed out that system was inconsistent. That proof uses the fact that $\text{Bot} = \forall X.X$ is initial, and under the assumption that there is an $z: \text{Bot}$, an inconsistency ($true = false$) can be proven. This argument is summarized below. (The proof, along with the definition of System R, can be found in [ACC93].)

The fact that $\text{Bot}$ is initial can be used to produce the judgment

$$\begin{align*}
\emptyset; z: \text{Bot} \vdash & true \\
\emptyset; z: \text{Bot} \vdash & false
\end{align*}$$

Consider a typing context with one type variable $Y$ and the context $\Delta = \frac{\emptyset'(f) \Rightarrow Y}{y} \quad \frac{x \Rightarrow (f)}{x'}$

(where $f = \lambda z: \text{Bot}. z[\text{Bool}]$ is the unique term $f: \text{Bot} \Rightarrow \text{Bool}$).

In System R, one can compose $y': \text{Bool} \Rightarrow Y$ and $x: Y \Rightarrow \text{Bot}$ to get $Y; \Delta \vdash x(y'(true)): \text{Bot}$. The judgment $Y; \Delta \vdash x(y'(true)): \text{Bot}$ follows from substitution. Thus by abstracting, and using $T$ to denote the relation $\forall Y.((f) \Rightarrow Y) \Rightarrow (Y \Rightarrow (f)) \Rightarrow \text{Bool}$, the following judgment is derivable.

$$\begin{align*}
\emptyset \vdash & true \\
\emptyset \vdash & false
\end{align*}$$

By equating quantification over type variables with quantification over relation variables, one can instantiate the variable at the $(f): \text{Bot} \leftrightarrow \text{Bool}$, to get the terms related by the relation $T[\text{Bool}] = ((f) \Rightarrow (f)) \Rightarrow ((f) \Rightarrow (f)) \Rightarrow \text{Bool}$.

$$\begin{align*}
\emptyset \vdash & true \\
\emptyset \vdash & false
\end{align*}$$

Note that the identity functions $\text{id}_{\text{Bot}}$ and $\text{id}_{\text{Bool}}$ are related by $(f) \Rightarrow (f)$, so by function application (and beta reductions) one can derive the judgment $\emptyset \vdash true \text{ Bool}$.

To analyze what ‘went wrong’ to allow such a proof, one sees that the functions $y': \text{Bool} \Rightarrow Y$ and $x: Y \Rightarrow \text{Bot}$ are composed to get a function from an inhabited type (Bool) to an uninhabited type (Bot). But actually, the same value is not used for the type variable $Y$ in these two instances when it comes down to finding “actual values” for $y'$ and $x$. By using the codomain of $(f)$ in $y'$ and the domain in $x$, the “function” of type $\text{Bool} \Rightarrow \text{Bot}$ comes
from composing \( y' \), the opposite of the relation \( (f) \), and \( x \). This composite
is a relation (namely the empty relation), not a function.

Abadi, Cardelli, and Curien blame the inconsistency on equating type quantifiers and relation quantifiers, and therefore, provide separate notions for each. An alternative interpretation would be to blame the prior assumption that all instances of a type variable will be instantiated at the same value, that is, there is no need to distinguish between type variables appearing in the domain and type variable appearing in the codomain of relations.

Since the polymorphic type construction uses quantifiers over relations in Reynolds’ parametric model (which is the motivation for our system), System P is built so that type quantification is over relations as well as types. Hence terms in the domain and terms in the codomain of relations are kept distinct despite possibly having the same types. Thus, the typing discipline mentioned above is enforced. Hasegawa’s paradox does not arise in System P since the domain variable \( x \) and the codomain variable \( y' \) can not be composed.

System F uses an impredicative form of polymorphism, hence System P (and System R) are also impredicative. One could produce a language including relations and a predicative form of polymorphism if one so desires.

A modification of System P for reasoning about the predicative calculus (see section 2.1) can be formed by including an additional kind of judgment.

\[
\eta \vdash_{\eta_0} R \text{ simple}
\]

This judgment asserts \( R \) is a relation between simple types of the predicative calculus. The elimination rule for polymorphic types should be restricted to only allow instantiation at relations between simple types.

\[
\begin{array}{ll}
\eta ; \Delta \vdash_{\eta_0} R_0 \stackrel{M}{\Rightarrow} X.R & \eta \vdash_{\eta_0} S \text{ simple} & \partial_0(S) = A & \partial_1(S) = B \\
\hline
\eta ; \Delta \vdash_{\eta_0} R_0 \stackrel{M[A]}{\Rightarrow} \frac{N[B]}{R[S/X]} \\
\end{array}
\]

The predicative calculus contains a type construction not found in System F — that of existential types (for abstract data types). Therefore, a modified version of System P for the predicative calculus would include another rule for forming relation judgments along with corresponding intro-
duction and elimination rules for relatedness judgments as follows.

\[
\begin{align*}
& \frac{\eta, X \vdash \eta_0 \models_\eta R}{\eta \vdash \eta_0 \models_\eta \exists X.R} \\
& \frac{\eta, X \vdash \eta_0 \models_\eta S \text{ simple}}{\eta \vdash \Delta \vdash \eta_0 \models_\eta M \quad \Delta \vdash \eta_0 \models_\eta N \models^{[S/X]} R} \quad \partial_0(S) = A} \\
& \frac{\eta, X \vdash \eta_0 \models_\eta R}{\eta \vdash \Delta \vdash \eta_0 \models_\eta (\text{pack } A \text{ with } M) \quad \exists X.R} \\
& \frac{\eta \vdash \Delta \vdash \eta_0 \models_\eta (\text{pack } B \text{ with } N) \quad \exists X.R}{\eta \vdash \Delta \vdash \eta_0 \models_\eta M \quad \exists X.R} \\
& \frac{\eta, X \vdash \Delta, \eta_0 \models_\eta x : R \models_\eta N \models^{[S]} \eta_0 \models_\eta S}{\eta \vdash \Delta \vdash \eta_0 \models_\eta (\text{open } M \text{ as } (X, x) \text{ in } N) \quad \eta \vdash \Delta \vdash \eta_0 \models_\eta S} \\
& \frac{\eta \vdash \Delta \vdash \eta_0 \models_\eta (\text{open } M' \text{ as } (X, y) \text{ in } N') \quad \eta \vdash \Delta \vdash \eta_0 \models_\eta S}{\eta \vdash \Delta \vdash \eta_0 \models_\eta S} \\
\end{align*}
\]

The associated \( \beta \)- and \( \eta \)-equalities that should also be added are as follows.

\[
\begin{align*}
& \eta, \Gamma \vdash \eta_0 \models_\eta M : A[B/X] \quad \eta, \Gamma, x : A \vdash \eta_0 \models_\eta N : C \quad \eta \vdash \eta_0 \models_\eta C \\
& \eta, \Gamma \vdash \eta_0 \models_\eta N[B/X][M/x] = (\text{open } (\text{pack } B \text{ with } M) \text{ as } (X, x) \text{ in } N) \\
& \eta, \Gamma \vdash \eta_0 \models_\eta M : \exists X.A \quad \text{\( Y, y \) are fresh} \\
& \eta, \Gamma \vdash \eta_0 \models_\eta M = (\text{open } M \text{ as } (Y, y) \text{ in } \text{(pack } Y \text{ with } y)) \\
\end{align*}
\]

Such a modified version of System \( \Pi \) for the predicative calculus has many of the structural properties one expects. For instance, substitution (an analogue of lemma 5.3) is admissible when the relation to be substituted is a relation between simple types. For instance, if \( \eta \vdash \eta_0 \models_\eta S \text{ simple} \) and \( \eta, X \vdash \eta_0 \models_\eta \), are derivable and no relation identifier of \( S \) is bound in \( R \), then \( \eta \vdash \eta_0 \models_\eta R^{[S/X]} \) is derivable.

### 5.3 Consequences of Parametricity

System \( \Pi \) can be used to provide constructions and associated characterizations that are fairly standard parametricity results [Rey83, Wad89, ACC93, PA93, RP93]. These includes, for example, the following.
\[ \forall X. X \Rightarrow X \text{ is terminal.} \]
\[ \forall X. (A \Rightarrow B \Rightarrow X) \Rightarrow X \text{ is the product } A \times B. \]
\[ \forall X. X \text{ is initial.} \]
\[ \forall X. (A \Rightarrow X) \Rightarrow (B \Rightarrow X) \Rightarrow X \text{ is the coproduct } A + B. \]

Here \( A \) and \( B \) are types which do not include \( X \).

A rather general parametricity result due to Reynolds and Plotkin [RP93] gives an encoding of initial algebras. The parametricity results listed above could all arise as instances of the initial algebra result alluded to above. To illustrate how such arguments can be formalized in System P, a limited form of the initial algebra result is carried out in some detail.

We consider types in which a given variable does not occur to the left of \( \Rightarrow \). We say such a type is plain relative to the given variable. The collection of types that are plain relative to \( X \) can be described by the following grammar.

\[
P = X \mid Q \mid Q \Rightarrow P' \mid \forall Y.P'
\]

Here \( Y \) stands for any relation identifier other than \( X \) and \( Q \) stands for any type in which \( X \) does not occur free. We use the notation \( \eta \vdash X. P \) to assert that \( \eta \vdash X. P \) is derivable and \( P \) is a plain type relative to \( X \).

For any plain type \( P \) relative to \( X \), there is a mapping \( P_X \) of functions \( A \Rightarrow B \) to functions \( P[A/X] \Rightarrow P[B/X] \). We first define a term \( P_X(f) \) with the following type.

\[
\eta, Z, Z' ; f : Z \Rightarrow Z' \vdash \exists \eta \vdash X. P_X(f) : P[Z/X] \Rightarrow P[Z'/X]
\]

The definition is by induction on the structure of \( P \) as follows.

\[
\begin{align*}
P = X & \quad P_X(f) = f \\
P = Q & \quad P_X(f) = \lambda x : Q.x \\
P = Q \Rightarrow P' & \quad P_X(f) = \lambda x : P[Z/X]. \lambda y : Q. P_X'(f)(xy) \\
P = \forall Y. P' & \quad P_X(f) = \lambda x : P[Z/X]. \lambda Y. P_X'(f) x[Y]
\end{align*}
\]

One can use substitutions (lemma 5.3) to define the meta-level function \( P_X \) on the syntax of relations and terms.

**Definition 5.18**

For any type \( P \) plain relative to \( X \), the function \( P_X \) on the set of relations \( R \) and terms \( M \) of System \( P \) is defined as follows.

\[
P_X(R) = P[R/X] \quad P_X(M) = P_X(f)[M/f]
\]
It is an immediate corollary of lemma 5.3 (substitution) that $P_x$ preserves relation judgments.

**Corollary 5.19**

Suppose $\eta, X \vdash_{\emptyset} \text{Plain}_X(P)$. If $\forall \emptyset R$ is derivable in System $P$, then so is $\forall \emptyset P_x(R)$.

The definitions of $P_x$ on terms and on relations are similar enough that relatedness judgments are also preserved.

**Proposition 5.20**

Assume $\eta, X \vdash_{\emptyset} \text{Plain}_X(P)$. If one can derive $\eta ; \Delta \vdash \emptyset_0 \vdash M_N R \Rightarrow S$, then

\begin{align*}
P_x(M) & \Rightarrow P_x(R) \Rightarrow P_x(S) \quad \text{as well.}
\end{align*}

Having defined $P_x(R)$ by substitution, it is immediate that it commutes with functional and polymorphic relation constructions. That is, the following syntactic equalities hold.

\begin{align*}
P_x(R) & \Rightarrow P_x(S) = P_x(R \Rightarrow S) \\
\forall Y P_x(R) & = P_x(\forall Y R)
\end{align*}

The same cannot be said about preimage relations. The mapping $P_x$ does preserve preimage relations up to equivalence (definition 5.6).

**Lemma 5.21**

If $\eta, X \vdash_{\emptyset} \text{Plain}_X(P)$ and $\emptyset \vdash \emptyset_0 \vdash A_M[N]R$ is derivable, then the following holds.

$$
\eta \vdash_{\emptyset_0} P_x(A_M[N]R) \equiv P_x(A) P_x(M), P_x(N) P_x(R)
$$

**Proof.** (by induction on the structure of $P$) The base cases for $X$ and $Q$ are immediate.

Considering the case $P = Q \Rightarrow P'$, we note that function application $\{\text{fun_E}\}$ and the inductive hypothesis can be used to derive the following.

\begin{align*}
\eta ; f \quad g x P_x(A_M[N]R), x: Q & \vdash_{\emptyset_0} f x \\
\eta ; g x P_x(A_M[N]R) & \vdash_{\emptyset_0} P_x(A), P_x(M), P_x(N), P_x(R)
\end{align*}

The elimination of preimage relations $\{\text{pre_E}\}$ and function abstraction $\{\text{fun_J}\}$
extend the derivation as follows.

\[
\eta; \frac{f}{g} \frac{P'_x(M) (f x)}{P'_x(N) (g x)} \\
\eta; \frac{\lambda x: Q. P'_x(M) (f x)}{\lambda x: Q. P'_x(N) (g x)} Q \Rightarrow P'_x(R)
\]

Recalling how \( P'_x(f) \) was defined, we see that preimage introduction \{pre.J\} allows one to derive the desired judgment.

\[
\eta; \frac{f}{g} \frac{P'_x(M) (f x)}{P'_x(N) (g x)} P'_x(R)
\]

The other direction is proved similarly, only in the reverse order. As a convenience, we use \( \Gamma \) to denote the lone relational assumption given here.

\[
\Gamma = \frac{f}{g} \frac{P'_x(A)}{P'_x(B)} [P'_x(M), P'_x(N)] P'_x(R)
\]

First, one eliminates the preimage relation using \{pre.E\} and expands out the definition of \( P'_x \).

\[
\eta; \Gamma, x: Q \frac{P'_x(M) (f x)}{P'_x(N) (g x)} P'_x(R)
\]

One can then use function application \{fun.E\} and the \( \beta \)-equality for functions \{beta.fun\} prior to introducing a preimage relation \{pre.J\}.

\[
\eta; \Gamma, x: Q \frac{\lambda y: Q. P'_x(M) (f y)}{\lambda y: Q. P'_x(N) (g y)} (f x)
\]

\[
\eta; \Gamma, x: Q \frac{P'_x(M) (f x)}{P'_x(N) (g x)} P'_x(R)
\]

\[
\eta; \Gamma, x: Q \frac{P'_x(A)}{P'_x(B)} [P'_x(M), P'_x(N)] P'_x(R)
\]

114
One can apply the induction hypothesis and abstract over the variable $x$ using \{$\text{fun}, \text{J}$\} to continue the derivation as follows.

\[
\frac{\eta ; \Gamma, x : Q \vdash \eta_0 \Gamma_0 \quad f \ x \ P_x' (\overline{A}[M, N] R)}{\lambda x : Q. (f \ x)}
\]

\[
\frac{\eta ; \Gamma \vdash \eta_0 \Gamma_0 \quad Q \Rightarrow P_x' (\overline{A}[M, N] R)}{\lambda x : Q. (g \ x)}
\]

Simplifying using \(\eta\)-equality \{$\text{eta} \text{.fun}$\}, one need only recall the definitions of \(P_x (f)\) and of \(\Gamma\) to get the desired judgment.

\[
\frac{\eta ; \Gamma \vdash \eta_0 \Gamma_0 \quad f \ P_x' (\overline{A}[M, N] R)}{g P_x' (\overline{A}[M, N] R)}
\]

\[
\frac{f P_x'(A)[P_x(M), P_x(N)] P_x(R) \vdash \eta_0 \Gamma_0 \quad f \ P_x' (\overline{A}[M, N] R)}{P_x (\overline{A}[M, N] R)}
\]

We also mention the step corresponding to \(P = \forall Y . P'\). We present one direction of the equivalence, and note that each step is invertible, so the other direction would reverse the steps of the derivation. By expanding out the definition of \(P_x (f)\) and eliminating the quantification \{$\text{poly}.E$\}, we can derive the following.

\[
\frac{\eta, Y ; x P_x (\overline{A}[M, N] R) \vdash \eta_0 \Gamma_0 \quad x \forall Y . P_x'(\overline{A}[M, N] R)}{y Y} \]

\[
\frac{\eta, Y ; P_x (\overline{A}[M, N] R) \vdash \eta_0 \Gamma_0 \quad x [Y] \forall Y . P_x'(\overline{A}[M, N] R)}{y [Y]} \]

The induction hypothesis and elimination of preimage relations \{$\text{pre}.E$\} allows one to quantify over the relation variable \(Y\) using \{$\text{poly}.J$\} in the following derivation.

\[
\frac{\eta, Y ; P_x (\overline{A}[M, N] R) \vdash \eta_0 \Gamma_0 \quad x [Y] P_x'(A)[P_x'(M), P_x'(N)] P_x'(R)}{y [Y]} \]

\[
\frac{\eta, Y ; P_x (\overline{A}[M, N] R) \vdash \eta_0 \Gamma_0 \quad P_x'(M) x [Y] P_x'(R)}{y [Y]} \]

\[
\frac{\eta, Y ; P_x (\overline{A}[M, N] R) \vdash \eta_0 \Gamma_0 \quad PM. x [Y] \forall Y . P_x'(R)}{y [Y]} \]

Recalling the definition of \(P_x (f)\) when \(P = \forall Y . P'\), introducing the preimage
relation via \{pre\} allows one to conclude the desired judgement.

\[
\begin{align*}
\eta ; & \quad x \overset{x}{y} P_{\eta}(A)[M,N]R \quad \vdash_{\eta_0} P_{\eta}(M) & \quad \overset{x}{y} P_{\eta}(R) \\
\eta ; & \quad x \overset{x}{y} P_{\eta}(A)[M,N]R \quad \vdash_{\eta_0} P_{\eta}(A)[P_{\eta}(M), P_{\eta}(N)]P_{\eta}(R)
\end{align*}
\]

Recall that graphs of functions are a special case of weakest preimages. Recalling the notation \( \langle M \rangle = A_B[M, \lambda X : B, x][1_B] \) introduced in section 5.1 (page 101), we state that \( P_{\eta} \) preserves the graphs of functions (up to equivalence) as well.

**Corollary 5.22**

If \( \eta, X \not\vdash_{\emptyset} \text{Plain}_X(P) \) and \( \eta ; \emptyset \vdash_{\eta_0} M : A \Rightarrow B \) is derivable, then the following holds.

\[ \eta \vdash_{\eta_0} P_{\eta}(\langle M \rangle) \equiv \langle P_{\eta}(M) \rangle \]

The mapping \( P_{\eta} \) not only preserves the graphs of functions, but it can also be shown to preserve the composition of functions.

**Lemma 5.23**

Suppose \( \eta, X \not\vdash_{\emptyset} \text{Plain}_X(P) \). If \( \eta ; \emptyset \vdash_{\emptyset} f : A \Rightarrow B \) and \( \eta ; \emptyset \vdash_{\emptyset} g : B \Rightarrow C \) are derivable, then the following hold.

\[ \eta ; \emptyset \vdash_{\emptyset} P_{\eta}(g \circ f) = P_{\eta}(g) \circ P_{\eta}(f) \]

**Proof.** These are shown by induction on the structure of \( P \). The base cases of \( P = X \) and \( P = Q \) are immediate.

For the case \( P = Q \Rightarrow Q' \), we begin by recalling the definition to note \( P_{\eta}(g \circ f) = \lambda x : P_{\eta}(A) \lambda y : Q. P_{\eta}(g \circ f)(x,y) \). Therefore, by using the induction hypothesis, we can build up the following judgment.

\[
\eta ; \emptyset \vdash_{\emptyset} P_{\eta}(g \circ f) = \lambda x : P_{\eta}(A) \lambda y : Q. (P_{\eta}(g) \circ P_{\eta}(f))(x,y)
\]

(It should be clear how this follows from the induction hypothesis using weakening (lemma 5.1), function application (fun,E) and then function abstraction (fun,J) twice. In the future, we do not mention such steps, nor the applications of transitivity (trans,r).) By using the definition of composition \( \beta \)-reduction (\{beta\}_\text{fun} ) and \( \eta \)-expansion (\{eta\}_\text{fun} ), we get the following
judgments.

\[ \eta : \Gamma \vdash P_X(g \circ f) = \lambda x : P_X(A).\lambda y : Q.\left( \lambda z : P'_X(A).P'_X(f)(P'_X(g)z) \right)(x y) \]

Recalling the definitions of \( P_X \) and of composition, the above judgment is the same as each of the following judgments.

\[ \eta ; \Gamma \vdash P_X(g \circ f) = \lambda x : P_X(A).\lambda y : Q.\lambda z : P'_X(A).P'_X(f)(P'_X(g)z)(x y) \]

For the case when \( P = \forall Y.P' \), we recall the definition of \( P_X \) to see that \( P_X(g \circ f) = \forall Y.\forall Y.P'(g \circ f) \). Using the induction hypothesis, we get the following judgment.

\[ \eta ; \Gamma \vdash P_X(g \circ f) = \lambda x : P_X(A).\lambda Y.\left( \lambda z : P'_X(A).P'_X(f)(P'_X(g)z) \right)(x [Y]) \]

This can be rewritten using the definition of composition. Using \( \beta \)-reduction \( \{ \text{beta} \text{fun} \} \) and \( \eta \)-expansion \( \{ \text{eta} \text{poly} \} \) will give rise to the following judgments.

\[ \eta ; \Gamma \vdash P_X(g \circ f) = \lambda x : P_X(A).\lambda Y.\left( \lambda z : P'_X(A).P'_X(f)(P'_X(g)z) \right)(x [Y]) \]

Recalling the definitions of \( P_X \) and of composition, this is the desired judgment.

\[ \eta ; \Gamma \vdash P_X(g \circ f) = \lambda x : P_X(A).\lambda Y.\lambda z : P'_X(A).P'_X(f)(P'_X(g)z)(x [Y]) \]

We are now able to present a fairly general type construction in System P — initial algebras.

**Definition 5.24**

Suppose \( \eta, X \vdash P \text{Plain}_X(P) \). For any \( \eta' \vdash \Gamma' \) where \( \eta \subseteq \eta' \), a \( P \)-algebra relative to \( (\eta' ; \Gamma') \) is a pair \((A, h)\) where \( A \) is a type and the judgments \( \eta' \vdash A \) and \( \eta' ; \Gamma' \vdash h : P_X(A) \Rightarrow A \) are derivable.

An initial \( P \)-algebra is a \( P \)-algebra \((A, h)\) relative to \((\eta ; \emptyset)\) such that
for any \( \eta' \), \( \Gamma' \) and any \( P \)-algebra \((A',h')\) relative to \((\eta';\Gamma')\), there exists a unique (up to provable equality) term \( M \) satisfying \( \eta' ; \Gamma' \vdash_{\emptyset} M : A \Rightarrow A' \) that commutes with \( h \) and \( h' \) in the following sense.

\[
\eta' ; \Gamma' \vdash_{\emptyset} h' \circ P_X(M) = P_X(A) \Rightarrow A' \ M \circ h
\]

Explicit mention of the context a given \( P \)-algebra is relative to shall frequently be suppressed whenever it can be inferred. Using standard terminology for algebras, the type \( A \) is called the carrier of the \( P \)-algebra \((A,h)\) and \( h \) is called the structure map.

In standard categories, one often refers to the initial algebra of a functor, since initial algebras are unique up to isomorphism. A similar observation can be made in System \( P \).

**Lemma 5.25**

Suppose \( \eta \vdash_{\emptyset} \text{Plain}_X(P) \). If \((A,h)\) and \((A',h')\) are both initial \( P \)-algebras, then \( \eta \vdash_{\emptyset} A \cong A' \).

**Proof.** Using the initiality of \((A,h)\), we get the term \( M \) of type \( A \Rightarrow A' \) that commutes in the sense of \( \eta ; \emptyset \vdash_{\emptyset} h' \circ P_X(M) = M \circ h \). Similarly there is a term \( M' \) of type \( A' \Rightarrow A \) such that \( \emptyset ; \emptyset \vdash_{\emptyset} h \circ P_X(M') = M' \circ h' \).

To show that the composition \( M' \circ M \) is the identity \( \text{id}_A \), we show that it is the unique morphisms \( A \Rightarrow A \) that comes from factoring the algebra \((A,h)\) through the initial algebra \((A,h)\). We begin by recalling that \( P_X \) preserves compositions (lemma 5.23).

\[
\eta ; \emptyset \vdash_{\emptyset} h \circ P_X(M' \circ M) = h \circ P_X(M') \circ P_X(M)
\]

Using the equalities characterizing \( M' \) and \( M \), we can use transitivity to get the following judgments.

\[
\eta ; \emptyset \vdash_{\emptyset} h \circ P_X(M' \circ M) = M' \circ h' \circ P_X(M)
\]
\[
\eta ; \emptyset \vdash_{\emptyset} h \circ P_X(M' \circ M) = M' \circ M \circ h
\]

Therefore \( M' \circ M \) is the unique morphism \( A \Rightarrow A \) that commutes with structure map \( h \). Since \( \text{id}_A \) also commutes with \( h \), \( M' \circ M \) is the identity. The other composition \( M \circ M' \) is shown to be the identity \( \text{id}_A \) in a similar manner (using \((A', h')\) instead of \((A, h)\)).

Every plain type \( P \) has an initial \( P \)-algebra. If \( \eta,X \vdash_{\emptyset} \text{Plain}_X(P) \), the \( P \)-algebra \((\mu P, \text{in}^P)\) relative to \( \eta ; \emptyset \) is defined as follows.
\[ \mu P = \forall X. (P \Rightarrow X) \Rightarrow X \]

\[ \text{fold}^P = \Lambda X. \lambda f : P \Rightarrow X. \lambda x : \mu P. x[X]f \]

\[ \text{in}^P = \lambda y : P^X. \Lambda X. \lambda f : P \Rightarrow X. f (\text{fold}^P[X]f) y \]

We shall frequently suppress explicit mention of the type \( P \) in the notation for \( \text{in} \) and \( \text{fold} \) when it can be inferred.

The term \( \text{fold} \) given above satisfies the following judgment.

\[ \eta : \emptyset \vdash \emptyset \text{fold} : \forall X. (P \Rightarrow X) \Rightarrow \mu P \Rightarrow X \]

The term \( \text{fold} \) is used to produce the function \( M = \text{fold}[A'][h'] \) from \( \mu P \) to the carrier of any \( P \)-algebra \((A', h')\). Using variables \( Y \) and \( h \) to hold the place of an arbitrary \( P \)-algebra, we get that \( \text{fold}[Y]h \) commutes with \( \text{in} \) and \( h \) as follows.

**Lemma 5.26**

The following judgment is derivable in System \( P \).

\[ \eta, Y ; h : P^X (Y) \Rightarrow Y \vdash_{\emptyset \emptyset} h \circ P^X (\text{fold}[Y]h) = (\text{fold}[Y]h) \circ \text{in} \]

**Proof.** Using the definition of \( \text{fold} \) and beta reductions, we get that the following is derivable.

\[ \eta, Y ; h : P^X (Y) \Rightarrow Y, x : P^X (\mu P) \vdash_{\emptyset \emptyset} (\text{fold}[Y]h)(\text{in} \ x) = (\text{in} \ x)[Y]h \]

Expanding out the definition of \( \text{in} \) and using more beta-reductions, we get the following.

\[ \eta, Y ; h : P^X (Y) \Rightarrow Y, x : P^X (\mu P) \vdash_{\emptyset \emptyset} (\text{fold}[Y]h)(\text{in} \ x) = h \left( P^X (\text{fold}[Y]h)x \right) \]

Recall the definition of composition in System \( P \).

\[ (\text{fold}[Y]h) \circ \text{in} = \lambda x : P^X (\mu P). (\text{fold}[Y]h)(\text{in} \ x) \]

\[ h \circ P^X (\text{fold}[Y]h) = \lambda x : P^X (\mu P). h \left( P^X (\text{fold}[Y]h)x \right) \]

Therefore, we get the following judgment as a consequence of \{fun,\}.

\[ \eta, Y ; h : P^X (Y) \Rightarrow Y \vdash_{\emptyset \emptyset} (\text{fold}[Y]h) \circ \text{in} = h \circ P^X (\text{fold}[Y]h) \]

The desired judgment follows by \{eq_symm\}. 

119
For any \( P \)-algebra \( (A', h') \), one can use substitution (lemma 5.3) to get that \( \text{fold}[A'] h' \) commutes with \( \hat{m} \) and \( h' \) in the desired manner.

\[
\eta'; \Gamma \vdash h' \circ P_X(\text{fold}[A] h') = (\text{fold}[A'] h') \circ \hat{m}
\]

Hence all that remains in order to show that \((\mu P, \hat{m})\) is an initial \( P \)-algebra is to show the uniqueness of \( \text{fold}[A'] h' \).

**Theorem 5.27**

Suppose \( \eta; X \vdash \text{Plain}_X(P) \) and \( \eta'; \Gamma' \vdash h: P_X(A) \Rightarrow A \) where \( \eta \subseteq \eta' \) and \( X \) is not in \( \eta' \). Additionally suppose \( \eta'; \Gamma' \vdash \mu P \Rightarrow A \) is derivable in System \( P \).

If \( \eta'; \Gamma' \vdash \hat{m} \circ P_X(M) = M \circ \hat{m} \) is derivable in System \( P \), then so is \( \eta'; \Gamma' \vdash M = \text{fold}[A] h \).

Consequently, \((\mu P, \hat{m})\) is the initial \( P \)-algebra.

**Proof.** We begin by noting that there is a relation judgment \( \emptyset \vdash_{\eta' \mid \Gamma'} \langle M \rangle \) for the graph of \( M \) as well as \( \emptyset \vdash_{\eta' \mid \Gamma'} \langle P_X(M) \rangle \) for the graph of \( P_X(M) \). The graph of \( P_X(M) \) is such that the judgment on the left below is derivable.

\[
\emptyset ; \ x ; \ P_X(\langle M \rangle) \vdash_{\eta' \mid \Gamma'} P_X(M) \ x = x' \quad \emptyset ; \ x' ; \ P_X(\langle M \rangle) \vdash_{\eta' \mid \Gamma'} h = h
\]

The judgment on the right comes from shifting variables to indeterminates (lemma 5.2) and weakening (lemma 5.1). Function application \{\text{fun}E\} allows one to conclude the following judgment is derivable.

\[
\emptyset ; \ x ; \ P_X(\langle M \rangle) \vdash_{\eta' \mid \Gamma'} h(P_X(M) \ x) = h x'
\]  \hspace{1cm} (5.1)

By shifting variables to indeterminates (via lemma 5.2) and weakening (lemma 5.1), the hypothesis yields that the following judgment is derivable.

\[
\emptyset ; \ x ; P_X(\mu P) \vdash_{\eta' \mid \Gamma'} M \circ \hat{m} = h \circ P_X(M)
\]

Using \{\text{fun}E\} and eta-reduction, we conclude the following.

\[
\emptyset ; \ x ; P_X(\mu P) \vdash_{\eta' \mid \Gamma'} M(\hat{m} \ x) = h(P_X(M) x)
\]

Then \{\text{trans}\} can be applied using judgment 5.1 to deduce the following.

\[
\emptyset ; \ x ; P_X(\langle M \rangle) \vdash_{\eta' \mid \Gamma'} M(\hat{m} \ x) = h x'
\]
This judgment can be rewritten using the graph of $M$ (lemma 5.8) as follows.

$$\emptyset ; \frac{x}{x'} P_x(\langle M \rangle) \vdash_{\eta \Gamma'} \frac{\text{in } x}{h x'}$$

By \{fun\}, we conclude the following.

$$\emptyset ; \emptyset \vdash_{\eta \Gamma' \Gamma'} \frac{\lambda x : P_x(\mu P).\text{in } x}{P_y((\langle M \rangle) \Rightarrow (M))} \frac{\lambda x' : P_y(\langle M \rangle).h x'}{h}$$

This becomes the following by $\eta$-equalities (formally, by using \{trans_r\} and \{trans_l\} with instances of \{eta\_fun\} for hypotheses).

$$\emptyset ; \emptyset \vdash_{\eta \Gamma' \Gamma'} \frac{\text{in } x}{h x'}$$

As an instance of \{poly\}, we get the following.

$$\emptyset ; z : \mu P \vdash_{\eta \Gamma' \Gamma'} \frac{z[\mu P]}{z[\langle M \rangle \Rightarrow (M)] \Rightarrow (M)} \frac{\text{in } x}{z[y]h}$$

The previous two judgments can be used as the hypotheses for \{fun\} to deduce the following judgment.

$$\emptyset ; z : \mu P \vdash_{\eta \Gamma' \Gamma'} \frac{z[\mu P]}{z[\langle M \rangle \Rightarrow (M)] \Rightarrow (M)} \frac{\text{in } x}{z[y]h}$$

Using $\beta$-equalities and the definition of $fold$, we deduce the following.

$$\emptyset ; z : \mu P \vdash_{\eta \Gamma' \Gamma'} \frac{z[\mu P]}{z[\langle M \rangle \Rightarrow (M)] \Rightarrow (M)} \frac{\text{in } x}{z[y]h}$$

We can use \{trans_l\} and the following lemma (5.28) to conclude the following.

$$\emptyset ; z : \mu P \vdash_{\eta \Gamma' \Gamma'} \frac{z[\mu P]}{z[\langle M \rangle \Rightarrow (M)] \Rightarrow (M)} \frac{\text{in } x}{z[y]h}$$

By eliminating the graph of $M$, we get the following.

$$\emptyset ; z : \mu P \vdash_{\eta \Gamma' \Gamma'} \frac{z[M]}{z[\langle M \rangle \Rightarrow (M)] \Rightarrow (M)} \frac{\text{in } x}{z[y]h}$$

Applying \{fun\} and $\beta$-equalities, we get the following.

$$\emptyset ; \vdash_{\eta \Gamma' \Gamma'} M = fold[\langle Y \rangle h]$$
The desired judgment follows from using \{eq,\equiv\} and \{eq,rel\} to clear the indeterminate zone.

\(\textbf{Lemma 5.28}\)

If \(\eta, X \vdash \emptyset\ \text{Plain}_X(P)\), the following is derivable.

\[\eta \cdot z \cdot \mu P \vdash \emptyset\ z = z[\mu P][\text{id}]\]

**Proof.** By symmetry \{eq,\equiv\} and expanding out the identity relation of \(\mu P = \forall X.(P \Rightarrow X) \Rightarrow X\), it suffices to show the following judgment is derivable.

\[\eta, X \cdot z \cdot \mu P, f : P \Rightarrow X \vdash \emptyset\ (z[\mu P][\text{id}])[X] f = z[X] f\]

Since graphs of functions are preserved by \(P_X\) (corollary 5.22), the following is derivable.

\[\emptyset ; \ x' \ P_X((\text{fold}[X] f)) \vdash \eta, X[f : P \Rightarrow X] \ P_X((\text{fold}[X] f) \ x = x'\]

Applying \(f\) to both sides and applying \(\beta\)-equalities to introduce the terms \(\text{id}\) and \(\text{fold}\) yields the following steps.

\[\emptyset ; \ x' \ P_X((\text{fold}[X] f)) \vdash \eta, X[f : P \Rightarrow X] f(\ P_X((\text{fold}[X] f) \ x) = f x'\]

\[\emptyset ; \ x' \ P_X((\text{fold}[X] f)) \vdash \eta, X[f : P \Rightarrow X] (\text{fold}[X] f)(\text{id} \ x) = f x'\]

Rewriting the above using the graph of \(\text{fold}[X] f\) and using \{fun,\} gives the following steps.

\[\emptyset ; \ x' \ P_X((\text{fold}[X] f)) \vdash \eta, X[f : P \Rightarrow X] \ x \ (fold[Z] f)\]

\[\emptyset ; \ x' \ P_X((\text{fold}[X] f)) \vdash \eta, X[f : P \Rightarrow X] \ x \ (fold[Z] f)\]

Using the \(\eta\)-equality \{eta,fun\} and weakening (lemma 5.1), the derivation can
be continued as follows.

\[
\begin{align*}
\emptyset ; \emptyset &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{P_X((\text{fold}[X]f))}{\text{fold}[Z]f}} \\
\emptyset ; z: \mu P &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{P_X((\text{fold}[X]f))}{\text{fold}[Z]f}} \\
\end{align*}
\]

The following judgment follows from \{var\} and \{poly.E\}.

\[
\begin{align*}
\emptyset ; z: \mu P &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{z[\mu P]}{\text{fold}[Z]f}} \\
\emptyset &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{(P_X((\text{fold}[Z]f))=z[\mu P])}{\text{fold}[Z]f}} \\
\emptyset &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{(z[\mu P])}{\text{fold}[Z]f}} \\
\end{align*}
\]

We can apply \{fun.E\} using the previous two judgments as hypotheses to conclude the following relatedness judgement.

\[
\begin{align*}
\emptyset ; z: \mu P &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{z[\mu P]}{\text{fold}[X]f}} \\
\emptyset &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{(z[\mu P]z)}{\text{fold}[X]f}} \\
\end{align*}
\]

This can be simplified to the desire result by eliminating the graph of \text{fold}[X]f (lemma 5.8), using the definition of \text{fold}, and clearing the indeterminate zone (via \{eq.term\} and \{eq.rel\}).

\[
\begin{align*}
\emptyset ; z: \mu P &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{(\text{fold}[X]f)(z[\mu P]z)}{\text{fold}[X]f}} \\
\emptyset &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{(z[\mu P]z)}{\text{fold}[X]f}} \\
\emptyset &\vdash_{\eta,X;f:P\Rightarrow X} \overset{\text{in}}{\genfrac{}{}{0pt}{}{z[\mu P]z)}{\text{fold}[X]f}} \\
\end{align*}
\]

For any type \( Q \) that does not have \( X \) as a free type identifier, \( Q \) is a plain type relative to \( X \). It is easy to see that the type \( Q \) is itself the carrier of the initial \( Q \)-algebra, \((Q, \text{id}_Q)\). For any \( Q \)-algebra \((A,h)\), the unique term \( M \) of type \( Q \Rightarrow A \) from the definition is \( h \) itself. Since initial algebras are unique up to isomorphism (lemma 5.25), the encoding on initial algebras (Theorem 5.27) provides a type isomorphic to \( Q \).

**Corollary 5.29**

For any type \( Q \) such that \( \eta \vdash_{\emptyset} Q \) where \( X \) is not in \( \eta \), the following is derivable.

\[
\eta \vdash_{\emptyset} Q \cong \forall X. (Q \Rightarrow X) \Rightarrow X
\]

If System P had types representing the initial type 0, the terminal type 1, the product \( A \times B \) and the coproduct \( A + B \) satisfying the expected properties of those types, then we would expect the following isomorphisms (where \( X \)
is a fresh identifier).

\[
\begin{align*}
1 & \iff \forall X.(1 \Rightarrow X) \Rightarrow X \iff \forall X.X \Rightarrow X \\
A \times B & \iff \forall X.((A \times B) \Rightarrow X) \Rightarrow X \iff \forall X.(A \Rightarrow B \Rightarrow X) \Rightarrow X \\
0 & \iff \forall X.(0 \Rightarrow X) \Rightarrow X \iff \forall X.1 \Rightarrow X \iff \forall X.X \\
A + B & \iff \forall X.((A + B) \Rightarrow X) \Rightarrow X \iff \forall X.((A \Rightarrow X) \times (B \Rightarrow X)) \Rightarrow X \\
& \iff \forall X.(A \Rightarrow X) \Rightarrow (B \Rightarrow X) \Rightarrow X
\end{align*}
\]

Therefore, we are motivated to use the above encodings as polymorphic types to define 0, 1, \(A \times B\) and \(A + B\) in System F and in System P.

\[
\begin{align*}
1 &= \forall X.X \Rightarrow X & A \times B &= \forall X.(A \Rightarrow B \Rightarrow X) \Rightarrow X \\
0 &= \forall X.X & A + B &= \forall X.(A \Rightarrow X) \Rightarrow (B \Rightarrow X) \Rightarrow X
\end{align*}
\]

These encodings can be shown to satisfy the expected properties for the initial type, terminal type, product and coproduct. For example, the expected properties of product types are as follows.

**Proposition 5.30**

Suppose \(A\) and \(B\) are types with free identifiers from \(\eta\) and that \(X\) is fresh. We define the following notational abbreviations.

\[
\begin{align*}
\pi_1 &= \lambda x : A \times B . x[A](\lambda a : A . \lambda b : B . a) \\
\pi_2 &= \lambda x : A \times B . x[B](\lambda a : A . \lambda b : B . b) \\
\langle x, y \rangle &= \Lambda X . \lambda p : A \Rightarrow B \Rightarrow X . p \ x \ y
\end{align*}
\]

These terms satisfy the following (expected) type judgments.

\[
\begin{align*}
\eta \vdash \emptyset \vdash \pi_1 : A \times B \Rightarrow A \\
\eta \vdash \emptyset \vdash \pi_2 : A \times B \Rightarrow B \\
\eta \vdash x : A, y : B \vdash \langle x, y \rangle : A \times B
\end{align*}
\]

For any terms \(M\) and \(N\), we shall use \(\langle M, N \rangle\) to denote the substitution \(\langle x, y \rangle[M/x][N/y]\). One can derive the usual beta- and eta-equalities, as stated below.

\[
\begin{align*}
\eta \vdash x : A, y : B \vdash \pi_1(x, y) = x \\
\eta \vdash x : A, y : B \vdash \pi_2(x, y) = y \\
\eta \vdash z : A \times B \vdash \langle \pi_1(z), \pi_2(z) \rangle = z
\end{align*}
\]

The two beta equalities are straightforward from \{\texttt{beta.fun}\} and can be shown in System F. The eta-equality does require parametricity arguments very similar to Theorem 5.27 and lemma 5.28.
Another example of this initial algebra result is the encoding of the type of booleans.

\[
\text{Bool} \equiv \forall X. X \rightarrow X \rightarrow X
\]

Using the common notation, there are terms with corresponding judgments as follows.

\[
\begin{align*}
\emptyset ; \emptyset & \vdash_{\emptyset} \text{true: Bool} \\
\emptyset ; \emptyset & \vdash_{\emptyset} \text{false: Bool} \\
Y ; x & \vdash \text{Bool}, a ; Y, b ; Y \vdash_{\emptyset} \text{if } x \text{ then } a \text{ else } b ; Y
\end{align*}
\]

These satisfy expected equalities. In particular, the following \(\eta\)-law for booleans (which is not provable in System F) is derivable in System P.

\[
\emptyset ; x ; \vdash \text{Bool} \vdash_{\emptyset} \text{if } x \text{ then } \text{true else false} =_{\text{Bool}} x
\]

Recall that the predicative polymorphic lambda calculus from section 2.1 contained a type construction not present in System F — existential quantification (for abstract data types). Existential types can also be encoded in System P as follows (where \(Y\) is a fresh identitier).

\[
\exists X. A = \forall Y. (\forall X. A \rightarrow Y) \rightarrow Y
\]

The associated term constructions for packing and opening abstract data types are as follows.

\[
\begin{align*}
\text{pack } B \text{ with } M & = \Lambda Y. \lambda p : (\forall X. A \rightarrow Y). p[B] M \\
\text{open } M \text{ as } (X, x) \text{ in } N & = M[C] (\lambda X. \lambda x : A. N)
\end{align*}
\]

These encodings can be shown to given existential quantification in System P as the \(\beta\)- and \(\eta\)-equalities hold.

**Proposition 5.31**

Using the notation introduced above, the following rules are derivable in System P.

\[
\begin{align*}
\eta ; \Gamma & \vdash_{\emptyset} M : A[^B] [^X] & \eta, X ; \Gamma, x : A & \vdash_{\emptyset} N : C & \eta \vdash_{\emptyset} C \\
\eta ; \Gamma & \vdash_{\emptyset} N [^B] [^X] [^M] [^C] =_{C} (\text{open } (\text{pack } B \text{ with } M) \text{ as } (X, x) \text{ in } N)
\end{align*}
\]

\[
\begin{align*}
\eta ; \Gamma & \vdash_{\emptyset} M : \exists X. A & Y, y & \text{are fresh} \\
\eta ; \Gamma & \vdash_{\emptyset} M = (\text{open } M \text{ as } (Y, y) \text{ in } (\text{pack } Y \text{ with } y))
\end{align*}
\]
The $\beta$-equality (beta_abs) follows from the $\beta$-equalities for $\Rightarrow$ and $\forall$, and thus also holds in System F. The $\eta$-equality (eta_abs) requires the use of parametricity, and is proved in a manner similar to the proof of Theorem 5.27.

The notation of existential types and products is handy in presenting the duals of initial algebras — final co-algebras.

**Definition 5.32**
Suppose $\eta, X \vdash_{\emptyset} Plain_X(P)$. For any $\eta' \vdash_{\emptyset} \Gamma'$ where $\eta \subseteq \eta'$, a $P$-co-algebra relative to $(\eta' ; \Gamma')$ is a pair $(A, k)$ where $A$ is a type and $k$ is a term as indicated below.

$$\eta' ; \Gamma' \vdash_{\emptyset} k : A \Rightarrow P_X(A)$$

A final $P$-co-algebra is a $P$-co-algebra $(A, k)$ relative to $(\eta ; \emptyset)$ such that for any $P$-co-algebra $(A', k')$ relative to $(\eta' ; \Gamma')$, there is a unique (up to provable equality) term $M$ such that the following judgments are derivable.

$$\eta' ; \Gamma' \vdash_{\emptyset} M : A' \Rightarrow A$$

$$\eta' ; \Gamma' \vdash_{\emptyset} P_X(M) \circ k' = k \circ M$$

Final $P$-co-algebras exist for all plain types. The encoding of the final $P$-co-algebra is given as follows.

$$\nu P = \exists X. (X \Rightarrow P) \times X$$

$$\text{out} = \lambda z : \nu P. (\text{open } z \text{ as } (X, x) \text{ in } (P_X(\text{unfold}[X]\pi_1 z) ((\pi_1 z) (\pi_2 z))))$$

$$\text{unfold} = \lambda X. \lambda h : X \Rightarrow P. \lambda x : X. (\text{pack } X \text{ with } \langle h, x \rangle)$$

Much as in the case of initial algebras, it is a simple application of $\beta$-equalities to show that for any co-algebra $(A, k)$, $\text{unfold}[A]k$ commutes with the co-algebra structure. That is, the following rule is admissible in System P.

$$\eta' ; \Gamma' \vdash_{\emptyset} k : A \Rightarrow P_X(A)$$

$$\eta' ; \Gamma' \vdash_{\emptyset} P_X(\text{unfold}[A]k) \circ k = \text{out} \circ \text{unfold}[A]k$$

Parametricity is used to show that $\text{unfold}[A]k$ is the unique such function.

**Theorem 5.33**
Suppose $\eta, X \vdash_{\emptyset} Plain_X(P)$, $(A, k)$ is a $P$-co-algebra relative to $(\eta ; \Gamma')$ and $g$ is a term such that $\eta' ; \Gamma' \vdash_{\emptyset} g : A \Rightarrow \nu P$ is derivable. If one can derive
\[ \eta' : \Gamma' \vdash_{\emptyset} P_X(g) \circ k = \text{out} \circ g, \text{then one can derive the following.} \]

\[ \eta' : \Gamma' \vdash_{\emptyset} \text{unfold}[A]k = g \]

**Proof.** Since the proof is very similar to the proof of the initial algebra result (Theorem 5.27), we shall merely outline this one. We work primarily with the relation \( \langle g \rangle \) where \( \emptyset \vdash_{\emptyset} \langle g \rangle \) is derivable. The hypothesis judgment \( \eta' : \Gamma' \vdash_{\emptyset} P_X(g) \circ k = \text{out} \circ g \) can be translated into the following.

\[ \emptyset ; \emptyset \vdash_{\emptyset} \Gamma' \quad k \quad \langle g \rangle \Rightarrow P_X(\langle g \rangle) \]

Since \( \text{unfold}[A] \) and \( \text{unfold}[\nu P] \) are related by \((\langle g \rangle \Rightarrow P_X(\langle g \rangle)) \Rightarrow \langle g \rangle \Rightarrow \nu P\), we get following.

\[ \emptyset ; \emptyset \vdash_{\emptyset} \Gamma' \quad \text{unfold}[A]k \quad \langle g \rangle \Rightarrow \nu P \]

The following lemma is used to conclude the following.

\[ \emptyset ; \emptyset \vdash_{\emptyset} \Gamma' \quad \text{unfold}[A]k = g \]

After shifting from indeterminates to variables using \{eq\_term\} and \{eq\_rel\}, this gives the desired result.

**Lemma 5.34**

If \( \eta, X \vdash_{\emptyset} \text{Plain}_X(P) \), then the following is derivable.

\[ \eta ; \emptyset \vdash_{\emptyset} \text{unfold}[\nu P] \text{out} = \text{id}_{\nu P} \]

**Proof.** We make use of an indeterminate \( x : (Z \Rightarrow P_X(Z)) \times Z \) (where \( Z \) is fresh) in forming the relation \( R = \langle \text{unfold}[Z]((\pi_1, x)) \rangle \) between \( Z \) and \( \nu P \).

Setting \( \eta_0 = \eta, Z \) and \( \Gamma_0 = x : (Z \Rightarrow P_X(Z)) \times Z \), we have \( \emptyset \vdash_{\emptyset} \Gamma_0 \quad R \).

One can unravel the definitions of \( \text{unfold} \) and \( \text{out} \) using \( \beta \)-equalities (such as \{beta\_abs\}) to get \( P_X(\text{unfold}[Z](\pi_1, x))((\pi_1, x), a) = \text{out}(\text{unfold}[Z]((\pi_1, x)a)). \) Therefore, the following judgment holds.

\[ \emptyset ; a \quad \text{out} \quad \text{fold}[\nu P] \]

This implies that \( \pi_1x \) and \( \text{out} \) are related by \( R \Rightarrow P_X(R) \). Since the definitions (and \( \beta \)-equalities) give \( (\text{unfold}[Z]((\pi_1, x))((\pi_2, x)) = (\text{pack} Z \text{ with } x) \), it follows that \( \pi_2x \) and \( \text{pack} Z \text{ with } x \) are related by \( R \). Therefore the
following judgment is derivable.

\[ \emptyset ; \emptyset \vdash_{\eta_{\otimes}} \Gamma \vdash x (R \Rightarrow \mathcal{P}_x (R)) \times R \]

Since any \( y : \forall X.((X \Rightarrow P) \times X) \Rightarrow Y \) respects \( R \), this gives the following.

\[ Y ; y : \forall X.((X \Rightarrow P) \times X) \Rightarrow Y \vdash_{\eta_{\otimes}} \Gamma \vdash y[Z]x = y \ \nu P (\text{out}, (\text{pack } Z \text{ with } x)) \]

Using the definition of pack and \( \beta \)-equalities, we can get a judgment asserting the following equality.

\[ (\text{pack } Z \text{ with } x)[Y]y = (\text{pack } \nu P \text{ with } (\text{out}, (\text{pack } Z \text{ with } x)))[Y]y \]

By abstracting away the applications and the making use of the equality \((\text{pack } \nu P \text{ with } (\text{out}, (\text{pack } Z \text{ with } x))) = (\text{unfold } \nu P \text{ out})(\text{pack } Z \text{ with } x)\), this gives us the following judgment.

\[ \emptyset ; \emptyset \vdash_{\eta_{\otimes}} \Gamma \vdash (\text{pack } Z \text{ with } x) =_{\nu P} (\text{unfold } \nu P \text{ out})(\text{pack } Z \text{ with } x) \]

Indeterminates can be shifted to variables using \{eq,term\} and \{eq,rel\}. Since \{eq,mm\}, shifts this into a form that can be rewritten using the relation \((\text{unfold } \nu P \text{ out})\), we abstract away variables to get the following relation assertion involving the relation \( S = \forall Z.((Z \Rightarrow \mathcal{P}_x (Z)) \times Z) \Rightarrow (\text{unfold } \nu P \text{ out}) \).

\[ \emptyset ; \emptyset \vdash_{\eta_{\otimes}} \Gamma \vdash \Lambda Z.\lambda x: (Z \Rightarrow \mathcal{P}_x (Z)) \Rightarrow Z.(\text{pack } Z \text{ with } x) S \]

Before introducing a variable \( z: \nu P \), we note the following equalities.

\[ \nu P = \exists X.(X \Rightarrow P) \times X = \forall Y.((\forall X.(X \Rightarrow P) \times X) \Rightarrow Y) \Rightarrow Y \]

Thus, one can use \{poly,E\} at the relation \((\text{unfold } \nu P \text{ out})\) and apply the function \( z[\nu P] \) to the above terms to show that the following term is related to itself by \((\text{unfold } \nu P \text{ out})\).

\[ z[\nu P]((\Lambda Z.\lambda x: (Z \Rightarrow P) \times Z.(\text{pack } Z \text{ with } x))) \]

By the definition of open, this gives the following judgment.

\[ \emptyset ; z: \nu P \vdash_{\emptyset \otimes} \Gamma \vdash (\text{open } z \text{ as } (x, Z) \text{ in } (\text{pack } Z \text{ with } x)) (\text{unfold } \nu P \text{ out}) \]

(\text{open } z \text{ as } (x, Z) \text{ in } (\text{pack } Z \text{ with } x)) \]

128
Since \(\eta_{\alpha, \beta}\) equates \((\text{open } z \text{ as } (x, Z) \text{ in } (\text{pack } Z \text{ with } x))\) and \(z\), this gives the following.

\[
\eta ; z : \nu P \Downarrow \eta_{0,0} (\text{unfold } \nu P)_{out} z = z
\]

By \(\{\text{fun}\}\), we get the following judgment, which gives the desired judgment as a result of \(\beta\)-reduction.

\[
\eta ; \emptyset \Downarrow \eta_{0,0} \lambda z : \nu P.(\text{unfold } \nu P)_{out} z = \text{id}_{\nu P}
\]

\[\Box\]

### 5.4 Soundness

Recall that a parametricity setting for System F consists of a Cartesian closed parametricity graph \(G\) along with the parametric limit PG-functor \(\text{Lim} : G^{\text{Gr}} \rightarrow G\). A model of System F can be given in a parametricity setting for System F using the construction described in section 3.3. In this section, we show that any well-pointed parametricity setting for System F additionally models System P. We show that the relationship between this model and the model of System F is such that provable equality in System P between System F terms implies the equality of the (System F) interpretations of the term judgments involved.

For this section, we fix \(G\) to be a well-pointed parametricity setting for System F. The model of System P is produced in a manner similar to that of System F (section 3.3). Recall that System F is built around types and terms, which give rise to vertices and morphisms of the functor graph \(G^{\text{Gr}}\). System P’s relation judgments and relatedness judgments will give rise to edges and squares. System P judgments are parameterized by indeterminates, which did not appear in System F. The interpretation of System P judgments will be parameterized by points corresponding to the indeterminate context.

Since System F embeds into System P, this model will produce a PL-category [See87], that is, a model of System F. Recalling Ma & Reynolds defined a parametric model of System F [MR92] to be a pair of PL-categories discussed in section 3.5, the model produced in the current section will be the “edge PL-category” over the model of System F using \(G\) as described in section 3.3. The relationship between our models of System P and System F will be discussed further at the end of this section. Models of System P are not described at the level of generality of PL-categories. System P
delves more deeply into the actual make up of the edges to allow meaningful parametricity results to be deduced.

Recall the intuition of System P's relations is that they are relations between System F's types. Since System F's type judgments are interpreted as vertices of the functor graph $G^{|\square|}$, it is not surprising that relation judgments of System P will give rise to edges of $G^{|\square|}$, that is, non-variant functors $E \times G^n \to G$. Relatedness judgments will determine squares.

$$
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
R \\
A_1 \xrightarrow{f_1} B_1
\end{array}
$$

Recall that a square of the above shape intuitively says that the morphisms $f_0$ and $f_1$ map $R$-related inputs to $S$-related outputs. The interpretation of relatedness judgments yield squares asserting that the interpretation of System F terms map related inputs (for the context) to related outputs, in an obvious correspondence with the notation of System P.

To avoid confusion with the model of System F described earlier, we shall use angled brackets to denote the interpretation of System P judgments, as in $[\eta \vdash \Gamma_0 R]$. Square brackets will be reserved for the interpretation of System F judgments given in section 3.3, as in $[\eta \vdash^F A]$.

Judgments of System P are all parameterized by indeterminates. Recall that the indeterminate context $\Gamma_0$ is a context of types such that $\eta_0 \vdash_{\emptyset} \Gamma_0$ for any indeterminate zone $(\eta_0|\Gamma_0)$ that appears in a judgment (lemma 5.15). Therefore $\Gamma_0$ is essentially a System F context satisfying $\eta_0 \vdash^F \Gamma_0$ (see lemma 5.14). The System F interpretation provides a concise manner to describe the points used as parameters for the interpretation of System P judgments.

**Definition 5.35**

For any relational context $\eta_0$ and context of types $\Gamma_0$ such that $\eta_0 \vdash_{\emptyset} \Gamma_0$, a pair of appropriate parameters for $(\eta_0|\Gamma_0)$ consists of an $|\eta_0|$-tuple of vertices $\vec{C}$ of $G$ and a morphism $r:1 \to [\eta_0 \vdash^F \Gamma_0]\vec{C}$. The set of all appropriate parameters $(\vec{C}, r)$ for $(\eta_0|\Gamma_0)$ is denoted $\text{Param}(\eta_0|\Gamma_0)$.

Much of the interpretation of System P mirrors the interpretation of System F. However, the construction of pre-image relations is unlike anything in System F. Pre-image relations will be interpreted using re-indexing in the parametricity graph $G^{|\square|}$. Before defining the model, we outline out how
pre-image relations will be interpreted. Recall the rule \( \{ \text{rel} \}_{\preceq} \).

\[
\begin{align*}
\eta \vdash \eta_0[\Gamma]\ R & \quad \eta : \emptyset \vdash \eta_0[\Gamma]\ M : A \Rightarrow \partial_0(R) & \quad \eta : \emptyset \vdash \eta_0[\Gamma]\ N : B \Rightarrow \partial_1(R) \\
\eta \vdash \eta_0[\Gamma]\ A[B, N][R]
\end{align*}
\]

Note that the hypothesis \( \emptyset \vdash \eta_0[\Gamma]\ M : A \Rightarrow \partial_0(R) \) implies that there is a System F term judgment \( \eta, \eta_0 \vdash \Gamma \vdash M : A \Rightarrow \partial_0(R) \) (corollary 5.16). Hence, there is a morphism \([M] : [\eta, \eta_0 : \Gamma] \rightarrow [\eta, \eta_0 : \Gamma] \) \( A \Rightarrow \partial_0(R) \). Note however, \([M]\) is a morphism of \( G^{\mathbb{N}^{n+n_0}} \) whereas we wish to define an edge of \( G^{\mathbb{N}^{n}} \). This disparity is offset by using the vertices \( \tilde{C} \) from the appropriate parameters for \( (\eta_0[\Gamma])_0 \) to fix additional arguments. In general, one can use a non-variant functor of (at most) \( n \) variables to transform a \((n+1)\)-ary non-variant functor into an \( n \)-ary non-variant functor.

**Definition 5.36**

Let \( n \) be greater than 0. For any non-variant functor \( F : G^n \rightarrow G \), we define \( \text{last}_n(F) \) to be the unique PG-functor \( G^{\mathbb{N}^{n+1}} \rightarrow G^{\mathbb{N}^{n}} \) satisfying the following.

\[
\begin{align*}
\text{last}_n(F)(G)\tilde{A} & = G(A_1, \ldots, A_n, F(\tilde{A})) \\
\text{last}_n(F)(G)\tilde{R} & = G(R_1, \ldots, R_n, F(\tilde{R})) \\
\text{last}_n(F)(\tau)\tilde{A} & = \tau(A_1, \ldots, A_n, F(\tilde{A})) \\
\text{last}_n(F)(\varnothing)(E, \tilde{R}) & = \varnothing(E, (R_1, \ldots, R_n, F(\tilde{R})))
\end{align*}
\]

For any edge \( \mathcal{F} \) of \( G^{\mathbb{N}^{n}} \), we define \( \text{last}_n(\mathcal{F}) : E \times G^{\mathbb{N}^{n+1}} \rightarrow G^{\mathbb{N}^{n}} \) to be the edge \( \text{last}_n(\mathcal{F}) : \text{last}_n(\partial_0(\mathcal{F})) \leftrightarrow \text{last}_n(\partial_1(\mathcal{F})) \) satisfying the following.

\[
\begin{align*}
\text{last}_n(\mathcal{F})(E, \varnothing)(E, \tilde{R}) & = \varnothing(E, (R_1, \ldots, R_n, \mathcal{F}(E, \tilde{R})))
\end{align*}
\]

When \( n = 0 \), for non-variant functors \( F : 1 \rightarrow G \) and \( \mathcal{F} : E \times 1 \rightarrow G \), the PG-functors \( \text{last}_0(F) : G^{\mathbb{N}} \rightarrow G^1 \) and \( \text{last}_0(\mathcal{F}) : E \times G^{\mathbb{N}} \rightarrow E \times G^1 \) are the unique PG-functors characterized as follows.

\[
\begin{align*}
\text{last}_0(F)(G) & = G \circ F \\
(\text{last}_0(F)(\tau))_* & = \tau_{F(*)} \\
\text{last}_0(F)(\varnothing) & = \varnothing \circ I_F \\
(\text{last}_0(\mathcal{F})(E, \varnothing))(E, \tilde{R}) & = \varnothing(E, \mathcal{F}(E, I_*))
\end{align*}
\]

While it is expected that the definition of \( \text{last}_n \) is proper in that there exists such a PG-functor, there is a good deal to be checked to verify that
it is. For instance, to show that last\(_n\)(F)(\(\tau\)) is a parametric transformation
\(\text{last}\(_n\)(F)(G) \rightarrow \text{last}\(_n\)(F)(\tilde{G})\) for any \(\tau: G \rightarrow \tilde{G}\), one needs to show that for
any \(\tilde{R}: \tilde{A} \leftrightarrow \tilde{B}\), there is a square of the following shape.

\[
\begin{array}{ccc}
\text{last}\(_n\)(F)(G) & \text{last}\(_n\)(F)(\tilde{G}) & \\
\text{last}\(_n\)(F)(\tilde{G}) & \text{last}\(_n\)(F)(\tilde{G}) & \\
\text{last}\(_n\)(F)(\tilde{G}) & \text{last}\(_n\)(F)(\tilde{G}) & \\
\end{array}
\]

The above square exists since there is a square of the following shape.

\[
\begin{array}{ccc}
G(A_1, \ldots, A_n, F(\tilde{A})) & \tau(A_1, \ldots, A_n, F(\tilde{A})) & G(A_1, \ldots, A_n, F(\tilde{A})) \\
G(R_1, \ldots, R_n, F(\tilde{R})) & \tau(B_1, \ldots, B_n, F(\tilde{B})) & G(R_1, \ldots, R_n, F(\tilde{R})) \\
G(B_1, \ldots, B_n, F(\tilde{B})) & \tau(B_1, \ldots, B_n, F(\tilde{B})) & G(B_1, \ldots, B_n, F(\tilde{B})) \\
\end{array}
\]

All the steps to be checked are similarly just cases of expanding the definitions.

For an \(m\)-tuple \(\tilde{F}\) of non-variant functors \(G^n \rightarrow G\), we use last\(_m\)(\(\tilde{F}\)) to
denote an iterated series of last\(_n\)s, such as the following.

\[
\text{last}\(_m\)(\(\tilde{F}\))(G) = \text{last}\(_n\)(F_1)(\ldots(\text{last}\(_{n+0}\)(F_{m-1})(G))\ldots)
\]

(We suppress explicit mention of the inclusion \(G^{[n]} \rightarrow G^{[n+i]}\) (ignoring the
final \(i\) inputs) used to make \(F_{i+1}\) be of the appropriate type to apply last\(_{n+i}\)
for each \(i \in \{1 \ldots m-1\}\).) To allow for uniformity in future assertions,
we do allow for the empty tuple of non-variant functors to be used with
this notation. Using \(m = 0\), fixing the last 0 arguments does nothing to a
non-variant functor.

\[
\text{last}\(_0\)(\emptyset)(G) = G
\]

It is immediate that changing the order of functions \(\tilde{F}\) in last\(_m\)(\(\tilde{F}\)) does
the same thing as reordering the inputs to the functor argument. That is, if
\(\sigma: G^{m_1} \times G^{m_2} \rightarrow G^{m_2} \times G^{m_1}\) is the apparent permutation, then the following
equality holds.

\[
\text{last}\(_{m_1+m_2}\)(F_1, \ldots, F_{m_1}, H_1, \ldots, H_{m_2})(G)
= \text{last}\(_{m_1+m_2}\)(H_1, \ldots, H_{m_2}, F_1, \ldots, F_{m_1})(G \circ (\text{ID} \times \sigma)).
\]

We shall frequently drop the subscript \(n\) (and superscript \(m\), if present)
when the number of arguments can be inferred.

Arguments of a non-variant functor can be fixed at a particular vertex by using the corresponding constant non-variant functor. For any \( m \)-tuple \( \mathcal{C} \) of vertices of \( G \), we use the following abbreviation.

\[
\text{Kon}_{\mathcal{C}} = \text{last}^m_\mathcal{C}(\Delta(C_1), \ldots, \Delta(C_m)).
\]

A useful fact about fixing arguments of non-variant functors is that it commutes with the Cartesian closed structure of the functor graphs.

**Lemma 5.37**

For any non-variant functors \( G_1 \) and \( G_2 \) from \( G^{n+n_0} \) to \( G \) and any \( n_0 \)-tuple \( \bar{F} \) of non-variant functors from \( G^n \) to \( G \), the following equations hold.

\[
\begin{align*}
\text{last}(\bar{F})(G_1) \times \text{last}(\bar{F})(G_2) &= \text{last}(\bar{F})(G_1 \times G_2) \\
\text{last}(\bar{F})(G_1) \Rightarrow \text{last}(\bar{F})(G_2) &= \text{last}(\bar{F})(G_1 \Rightarrow G_2)
\end{align*}
\]

**Proof.** Products in any functor graph are defined point-wise. In the special case of non-variant functors, the exponent \( G_1 \Rightarrow G_2 \) is also point-wise. Clearly \( \text{last}(\bar{F}) \) commutes with point-wise defined operations. \( \diamond \)

Using the point \( r: 1 \rightarrow [\eta_0, \Gamma_0]C \) from a pair of appropriate parameters for \( (\eta_0, \Gamma_0) \), there is the corresponding constant parametric transformation \( \Delta r: 1 \rightarrow \text{Kon}_{\mathcal{C}}[\eta, \eta_0, \Gamma] \). Therefore, for any \( M \) such that there is a judgment \( \eta, \eta_0 : \Gamma_0 \Rightarrow M : A \Rightarrow \partial_0(R) \), there is the following morphism of \( G^{\mathcal{C}^\Gamma} \).

\[
\text{Kon}_{\mathcal{C}}[M] \circ \Delta r : 1 \rightarrow \text{Kon}_{\mathcal{C}}[\eta, \eta_0, \Gamma] A \Rightarrow \partial_0(R)
\]

In any Cartesian closed parametricity graph, morphisms \( 1 \rightarrow A \Rightarrow B \) are equivalent to morphisms \( A \rightarrow B \). We shall refer to these transitions as **carrying a function of one variable** and **unwinding a point**. The carrying of \( g: A \rightarrow B \) to \( g^* : 1 \rightarrow A \Rightarrow B \) and the unwinding of \( f: 1 \rightarrow A \Rightarrow B \) to \( f^* : A \rightarrow B \) make use of the left cancellation isomorphism \( \lambda_A : 1 \times A \rightarrow A \) from the Cartesian closed structure.

**Definition 5.38**

For any vertices \( A \) and \( B \), and morphisms \( f: 1 \rightarrow A \Rightarrow B \) and \( g: A \rightarrow B \), the morphisms \( f^* \) and \( g^\circ \) are as follows.

\[
\begin{align*}
f^* &= \text{uncarry}(f) \circ \lambda^i_A : A \rightarrow B \\
g^\circ &= \text{carry}(g \circ \lambda_A) : 1 \rightarrow A \Rightarrow B
\end{align*}
\]

133
It is immediate that \((g^*)^* = g\) and \((f^*)^* = f\). Additional properties that support the intuition that \(f\) and \(f^*\) represent essentially the same function are given below.

**Lemma 5.39**

- For any \(f:1 \to A \Rightarrow B\) and \(g:D \to A\),

\[
f^* \circ g = \text{ap} \circ ((f \circ !D), g)
\]

where \(!D:D \to 1\) is the unique arrow of that type.

- There is a square of the shape on the left below if and only if there is a square of the shape on the right below.

\[
\begin{array}{ccc}
1 & \xrightarrow{f_0} & A_0 \Rightarrow B_0 \\
\downarrow & & \downarrow \\
1 & \xrightarrow{f_1} & A_1 \Rightarrow B_1 \\
\end{array}
\quad
\begin{array}{ccc}
A_0 & \xrightarrow{f_0^*} & B_0 \\
\downarrow & & \downarrow \\
A_1 & \xrightarrow{f_1^*} & B_1 \\
\end{array}
\]

**Proof.** The first point is proved by expanding \(\text{uncurry}(f)\) using its definition in terms of the unit of the adjunction, \(\text{ap}\), and appealing to naturality properties.

\[
f^* \circ g = \text{uncurry}(f) \circ \lambda^A \circ g = \text{ap} \circ (f \times \text{id}) \circ \lambda^A \circ g
\]

\[
= \text{ap} \circ (f \times \text{id}) \circ (!A, \text{id}) \circ g
\]

\[
= \text{ap} \circ ((f \circ !A \circ g), g)
\]

\[
= \text{ap} \circ ((f \circ !D), g)
\]

For the second point, one direction of the bi-implication expands the definitions of \(f_0^*\) and \(f_1^*\).

\[
\begin{array}{ccc}
A_0 & \xrightarrow{\lambda^A} & 1 \times A_0 \\
\downarrow & & \downarrow \\
R & \xrightarrow{I \times R} & (R \Rightarrow S) \times R \\
\downarrow & & \downarrow \\
A_1 & \xrightarrow{\lambda^A} & 1 \times A_1 \\
\end{array}
\quad
\begin{array}{ccc}
(A_0 \Rightarrow B_0) \times A_0 & \xrightarrow{\text{ap}} & B_0 \\
\downarrow & & \downarrow \\
(A_1 \Rightarrow B_1) \times A_1 & \xrightarrow{\text{ap}} & B_1 \\
\end{array}
\]

The other direction expands the definitions of \(f_0^{*\circ}\) and \(f_1^{*\circ}\), using the explicit description of \(\text{curry}(g)\) in terms of the co-unit \(\epsilon_{XY}:X \to Y \Rightarrow (X \times Y)\)
from the adjunction giving the Cartesian closed structure.

\[
\begin{array}{c}
1 \xrightarrow{\epsilon} A_0 \Rightarrow (1 \times A_0) & \xrightarrow{id \Rightarrow \lambda} & A_0 \Rightarrow A_0 & \xrightarrow{id \Rightarrow f^*} & A_0 \Rightarrow B_0 \\
1 \xrightarrow{\epsilon} A_1 \Rightarrow (1 \times A_1) & \xrightarrow{id \Rightarrow \lambda} & A_1 \Rightarrow A_1 & \xrightarrow{id \Rightarrow f^*} & A_1 \Rightarrow B_1 \\
1 \xrightarrow{R \Rightarrow (1 \times R)} & \xrightarrow{R \Rightarrow R} & R \Rightarrow R & \xrightarrow{R \Rightarrow S} & R \Rightarrow S
\end{array}
\]

Using the Cartesian closed structure, we have the following string of equalities.

\[
(id \Rightarrow f_0^*) \circ (id \Rightarrow \lambda) \circ \epsilon = (id \Rightarrow (f_0^* \circ \lambda)) \circ \epsilon = \text{curry}(f_0^* \circ \lambda) = (f_0^* \circ \lambda) = f_0
\]

Similarly one can show $(id \Rightarrow f_1^*) \circ (id \Rightarrow \lambda) \circ \epsilon = f_1$. Therefore the above square is the desired square.

\[\text{Corollary 5.40}\]

For any $f_1 \rightarrow C$, $h_1 \rightarrow A \Rightarrow B$ and any point $g_1 \rightarrow A$, the following equality holds.

\[(h \circ f)^* \circ g = \text{uncurry}(h) \circ \langle f, g \rangle\]

**Proof.** This follows from the first point above by noting that $!_1 = id$.

\[
(h \circ f)^* \circ g = \text{ap} \circ \langle ((h \circ f) \circ !_1), g \rangle = \text{ap} \circ \langle (h \circ f), g \rangle = \text{ap} \circ \langle h \times id \rangle \circ \langle f, g \rangle = \text{uncurry}(h) \circ \langle f, g \rangle
\]

The equivalence between morphisms $1 \rightarrow A \Rightarrow B$ and morphisms $A \rightarrow B$ will be used in interpreting pre-image relations. Terms $M$ and $N$ (of types $A \Rightarrow \partial_0 (R)$ and $B \Rightarrow \partial_1 (R)$, respectively) will be used to determine morphisms $m_1 : 1 \rightarrow F_0 \Rightarrow G_0$ and $n_1 : 1 \rightarrow F_1 \Rightarrow G_1$. The corresponding morphisms $m^*$ and $n^*$ will be used to re-index the edge corresponding to $R$ (between $G_0$ and $G_1$) and thus produce an edge for $\partial_1 [M, N] R$ (between $F_0$ and $F_1$). Recall that re-indexing in the functor graph $G^{\mathbb{Q}^n}$ is defined in the apparent pointwise manner. Hence, for any $n$-tuple $\tilde{S}$ of edges of $G$, the following equation holds.

\[
([\tau, \tau']^F(E, \tilde{S})) = [\tau_{(\tau_0(\tilde{S}))}, \tau_{(\tau_1(\tilde{S}))}]^F(E, \tilde{S})
\]

Formally, the interpretation of System P is defined as follows. We define the interpretation of a relation judgment $\eta_1 \vdash_0 \alpha$ as a function from
\[ \langle X_1, \ldots, X_n \vdash_{\eta_0}^\Gamma_0 X_j \rangle_\langle \tilde{C}, r \rangle = \Pi_j \]
\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 X_j \rangle_\langle \tilde{C}, r \rangle = \Delta(\Gamma_j) \]
\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 R \Rightarrow S \rangle_\langle \tilde{C}, r \rangle = \]
\[ \Rightarrow \circ \langle \langle \eta \vdash_{\eta_0}^\Gamma_0 R \rangle_\langle \tilde{C}, r \rangle, \langle \eta \vdash_{\eta_0}^\Gamma_0 S \rangle_\langle \tilde{C}, r \rangle \rangle \]
\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 \forall X.R \rangle_\langle \tilde{C}, r \rangle = \operatorname{Lim}_e \langle \langle \eta, X \vdash_{\eta_0}^\Gamma_0 R \rangle_\langle \tilde{C}, r \rangle \rangle \]
\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 \Delta \rangle_\langle \tilde{C}, r \rangle = [m^*, n^*]\langle \eta \vdash_{\eta_0}^\Gamma_0 R \rangle_\langle \tilde{C}, r \rangle \]
where \( m = \operatorname{Kon}_\tilde{C}[\eta, \eta_0]; \Gamma_0 \models M: A \Rightarrow \partial_0(R) \circ \Delta r \)
and \( n = \operatorname{Kon}_\tilde{C}[\eta, \eta_0]; \Gamma_0 \models N: B \Rightarrow \partial_1(R) \circ \Delta r \)

\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 \emptyset \rangle_\langle \tilde{C}, r \rangle = \Delta(I_1) \]
\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 \emptyset \rangle_\langle \tilde{C}, r \rangle = \Delta(I_1) \]
\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 \Delta \rangle_\langle \tilde{C}, r \rangle = \]
\[ \times \circ \langle \langle \eta \vdash_{\eta_0}^\Gamma_0 \Delta \rangle_\langle \tilde{C}, r \rangle, \langle \eta \vdash_{\eta_0}^\Gamma_0 R \rangle_\langle \tilde{C}, r \rangle \rangle \]

\textbf{Table 5.4: Interpretation of Relation Judgments}

Param(\( \eta_0|\Gamma_0 \)) to the edges of \( G^{G^n} \) (where \( n = |\eta| \)). We shall denote the application of such a function by subscripting. Thus, for any appropriate parameters \( (\tilde{C}, r) \) for \( (\eta_0|\Gamma_0) \), the interpretation of \( \eta \vdash_{\eta_0}^\Gamma_0 \alpha \) gives a non-variant functor as follows.

\[ \langle \eta \vdash_{\eta_0}^\Gamma_0 \alpha \rangle_\langle \tilde{C}, r \rangle : E \times G^n \rightarrow G \]

The interpretation of a relatedness judgment is defined as a function from Param(\( \eta_0|\Gamma_0 \)) to squares of \( G^{G^n} \) such that for any appropriate parameters \( (\tilde{C}, r) \) for \( (\eta_0|\Gamma_0) \) the following typing holds.

\[ \langle \eta; \Delta \vdash_{\eta_0}^\Gamma_0 M \rangle^{\langle C, r \rangle} \rightarrow \langle \eta \vdash_{\eta_0}^\Gamma_0 R \rangle_\langle \tilde{C}, r \rangle \]

The interpretations of relation judgments are defined inductively on the derivation of relation judgments, as indicated in table 5.4. The notation \( \Pi_j \) is used to denote the \( j \)th projection non-variant-functor \( G^n \rightarrow G \). Angled brackets are used to denote the pairing of non-variant functors \( F, G : E \times G^n \rightarrow G \) into a non-variant functor \( \langle F, G \rangle : E \times G^n \rightarrow G \times G \). In the step corresponding to the rule \{rel, poly\} when \( \eta = \emptyset \), we make tacit use of the isomorphism between \( G \) and \( G^1 \) to permit us to treat the edge \( \operatorname{Lim}_e \langle X \vdash_{\eta_0}^\Gamma_0 R \rangle_\langle \tilde{C}, r \rangle \) as a non-variant functor \( E \times 1 \rightarrow G \).
Recall (corollary 5.16) that a relatedness judgment $\eta ; \Delta \vdash_{\eta_0}^{\Gamma_0} M^R_N$ determines two System F term judgments $\eta, \eta_0 \vdash_0 (\Delta), \Gamma_0 \vdash M : \partial_0 (R)$ and $\eta, \eta_0 \vdash_1 (\Delta), \Gamma_0 \vdash N : \partial_1 (R)$ for the terms involved. The interpretations of these two System F judgments will determine the shape of the interpretation of the relatedness judgment. Since parametricity graphs are relational, the shape of a square uniquely determines that square.

Care must be taken to ensure the proper handling of empty relational contexts. Note that $[\eta \vdash \Gamma] \times [\eta \vdash \Gamma_0]$ and $[\eta \vdash \Gamma, \Gamma_0]$ are not necessarily equal, but they are canonically isomorphic. For example, if $\Gamma_0 = \emptyset$, then $[\eta \vdash \Gamma] \times [\eta \vdash \Gamma_0] = [\eta \vdash \Gamma] \times \Delta (1)$ but $[\eta \vdash \Gamma, \Gamma_0] = [\eta \vdash \Gamma]$. We extend the isomorphism $i_\Gamma^\eta$ (from section 3.3, page 51) to allow a context $\Gamma_0$ in the place of the type $\tau$.

$$i_{\Gamma_0}^\eta = \lambda : \Delta (1) \times [\eta \vdash \Gamma_0] \rightarrow [\eta \vdash \Gamma_0]$$

$$i_{\Gamma}^\eta = \rho : [\eta \vdash \Gamma] \times \Delta (1) \rightarrow [\eta \vdash \Gamma]$$

$$i_{\Gamma_0}^\eta = \text{id} : [\eta \vdash \Gamma] \times [\eta \vdash \Gamma_0] \rightarrow [\eta \vdash \Gamma, \Gamma_0] \quad \text{when } \Gamma \neq \emptyset \neq \Gamma_0$$

For any $(\tilde{C}, r) \in \text{Param}(\eta_0 | \Gamma_0)$, we define $\langle \eta ; \Delta \vdash_{\eta_0}^{\Gamma_0} M^R_N \rangle_{(\tilde{C}, r)}$ to be the unique square of the following shape.

$$\begin{array}{c}
\begin{array}{c}
F \\
\text{Kon}_C [\eta, \eta_0 ; \partial_0 (\Delta), \Gamma_0 \vdash M : \partial_0 (R)] \circ i_{\Gamma_0}^{\partial_0 (\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1 \\
F' \\
\text{Kon}_C [\eta, \eta_0 ; \partial_1 (\Delta), \Gamma_0 \vdash N : \partial_1 (R)] \circ i_{\Gamma_0}^{\partial_1 (\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1
\end{array}
\end{array}$$

where

\[
\begin{align*}
F & = \text{Kon}_C [\eta, \eta_0 \vdash_0 (\Delta)] \\
\mathcal{F} & = \langle \eta \vdash_{\eta_0}^{\Gamma_0} \Delta \rangle_{(\tilde{C}, r)} \\
F' & = \text{Kon}_C [\eta, \eta_0 \vdash_1 (\Delta)] \\
\mathcal{F}' & = \langle \eta \vdash_{\eta_0}^{\Gamma_0} R \rangle_{(\tilde{C}, r)}
\end{align*}
\]

(This makes use of $\rho^1$, the inverse of the right cancellation parametric transformation $\rho_F : F \times \Delta (1) \rightarrow F$ from the Cartesian closed structure.)

While the above definitions do completely define a model of System P, it may not be immediately apparent that the definition is actually possible. We must therefore show that there is a square of the desired shape for every relatedness judgment. It also must be shown that the $\{\text{rel,xre}\}$ step in defining
the interpretations of relation judgments makes sense. The interpretation of \( \eta \downarrow_0 \Gamma_0 \xrightarrow{A} B[M,N]R \) was defined at \((\tilde{C},r)\) by re-indexing \( \langle \eta \downarrow_0 \Gamma_0 \rangle_{(\tilde{C},r)} \) along morphisms of the following type.

\[
m^* : \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} A \rightarrow \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} \partial_0(R) \\
n^* : \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} B \rightarrow \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} \partial_1(R)
\]

Therefore, we start by showing that the interpretation of a relation judgment does give edges between the interpretation of corresponding System F types.

**Lemma 5.41**
Suppose \((\tilde{C},r) \in \text{Param}(\eta_0|\Gamma_0)\) and \(S\) is an \(|\eta|\)-tuple of edges \(S_i : D_i \leftrightarrow D'_i\) of \(G\). For any relation judgment \( \eta \downarrow_0 \Gamma_0 \alpha \), the following equations hold.

\[
\partial_0(\langle \eta \downarrow_0 \Gamma_0 \alpha \rangle_{(\tilde{C},r)}) = \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} \partial_0(\alpha) \\
\partial_1(\langle \eta \downarrow_0 \Gamma_0 \alpha \rangle_{(\tilde{C},r)}) = \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} \partial_1(\alpha)
\]

**Proof.** We show this by induction on the derivation of \( \eta \downarrow_0 \Gamma_0 \alpha \). Direct comparison establishes the equality for the steps corresponding to “base cases” \{cont.lamp\}, \{cont.ind\}, \{rel.war\} and \{rel.ind\}. The steps corresponding to \{cont.ext\}, \{rel.fun\} and \{rel.poly\} are consequences of \(\times\), \(\Rightarrow\) and \(\text{Lim}\) being PG-functors. The \{rel.pre\} step for the first equation holds as follows.

\[
\partial_0(\langle \eta \downarrow_0 \Gamma_0 A \rangle_{(\tilde{C},r)}) = \partial_0([m^*, n^*] \langle \eta \downarrow_0 \Gamma_0 R \rangle_{(\tilde{C},r)}) = \text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} A
\]

The last equality holds because the source of \(m^*\) is \(\text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} A\), as indicated above. The \(\partial_1\) equality is shown using \(n^*\) in a similar manner. \(\diamond\)

The intuition for System P types was that they are identity relations over System F types. The intuition is reflected in the model using identity edges.

**Lemma 5.42**
Suppose \((\tilde{C},r) \in \text{Param}(\eta_0|\Gamma_0)\). For any type \(A\) and any context of types \(\Gamma\), the following equations hold.

\[
\langle \eta \downarrow_0 \Gamma_0 A \rangle_{(\tilde{C},r)} = \text{1}(\text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} A) \\
\langle \eta \downarrow_0 \Gamma_0 \Gamma \rangle_{(\tilde{C},r)} = \text{1}(\text{Kon}_{\tilde{C}}[\eta, \eta_0] \xrightarrow{\tilde{F}} \Gamma)
\]

138
**Proof.** Induction on the derivation of \( \eta \vdash_{\eta_0} \Gamma_0 \ A \) or \( \eta \vdash_{\eta_0} \Gamma_0 \ \Gamma \) is used. The steps corresponding to \{rel\_var\}, \{rel\_ind\}, \{cont\_smp\} and \{cont\_ind\} are easily shown directly. The steps corresponding to \{cont\_ext\}, \{rel\_fun\} and \{rel\_poly\} are assured by the PG-functors used to define the interpretations.

It is an immediate corollary of lemma 5.42 that the P-interpretation of a System F type judgment is the identity edge over the P-interpretation of the same judgment. It follows from the relational criteria on a parametricity graph that the P-interpretation of a term judgment is an identity square over the F-interpretation.

**Corollary 5.43**

Suppose \( \eta ; \Gamma \vdash^T M : A \) is derivable in System F.

\[
\begin{align*}
\langle \eta \vdash A \rangle & = I_{[\eta] \not\in A} \\
\langle \eta ; \Gamma \vdash M : A \rangle & = I_{[\eta] ; [\Gamma] \not\in M : A}
\end{align*}
\]

Having types and context of types interpreted as identity edges, it follows from the identity condition that the interpretation of a relatedness judgment \( \eta ; \Gamma \vdash_{\eta_0} \Gamma_0 \ M =_A N \) would be an identity square for every pair of appropriate parameters. This fact will prove useful in the inductive proof that there is a square as desired for interpreting a relatedness judgment.

**Theorem 5.44**

For any \( \eta ; \Delta \vdash_{\eta_0} \Gamma_0 \ M =_A N \) of System \( P \) and any \((\vec{C}, r) \in Param(\eta_0|\Gamma_0)\), there is a square of the following shape.

\[
\begin{array}{c}
\langle \Delta \rangle \ (\vec{C}, r) \\
\downarrow \\
\langle R \rangle \ (\vec{C}, r)
\end{array}
\]

(We have suppressed writing the typing context \( \eta, \eta_0 \) for the term judgments as well as using just the relational context \( \Delta \) and relation \( R \) as abbreviations for the System \( P \) judgments \( \eta \vdash_{\eta_0} \Gamma_0 \ \Delta \) and \( \eta \vdash_{\eta_0} \Gamma_0 \ \Gamma \).)

**Proof.** (by induction on derivations) The inductive steps corresponding to most term forming rules are rather straightforward. For \{pre\_J\}, the inductive hypothesis ensures a square of from \( S = \langle \Delta \rangle \ (\vec{C}, r) \) to \( S' = \langle R \rangle \ (\vec{C}, r) \) of the
following shape.

\[
\begin{array}{c}
F \\ S \\ F' \\
\end{array}
\xrightarrow{	ext{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M M': \partial_0(R)] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1} \quad H
\end{array}
\]

Note that lemma 5.39 and expanding out the definitions give the following factorization.

\[
m^* \circ \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M': A] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1 = \text{ap} \circ \left( (m \circ !) \circ \left( \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M': A] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1 \right) \right)
\]

\[
= \text{ap} \circ \left( \left( \text{Kon}_C[\Gamma_0 \vdash^e M: A \Rightarrow \partial_0(R)] \circ \Delta r \circ !, \right) \circ \left( \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M': A] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1 \right) \right)
\]

\[
= \text{ap} \circ \left( \left( \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M: A \Rightarrow \partial_0(R)] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1 \right), \right) \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M': A] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1
\]

\[
= \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M M': \partial_0(R)] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1
\]

Similarly one can show the corresponding factorization for \( N \).

\[
n^* \circ \text{Kon}_C[\partial_1(\Delta), \Gamma_0 \vdash N': B] \circ i_{\Gamma_0}^{\partial_1(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1 = \text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash N N': \partial_1(R)] \circ i_{\Gamma_0}^{\partial_1(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1
\]

Therefore the square above factors uniquely through the weakest pre-edge \([m^*, n^*] \langle R \rangle \langle \tilde{C}_F \rangle \) giving a square as desired.

\[
\begin{array}{c}
\langle \Delta \rangle \langle \tilde{C}_F \rangle \\
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle \\
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle \\
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle
\end{array}
\xrightarrow{\text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e M M': \partial_0(R)] \circ i_{\Gamma_0}^{\partial_0(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1} \quad G
\]

\[
\begin{array}{c}
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle \\
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle \\
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle \\
\langle \tilde{A} \rangle \langle \tilde{C}_F \rangle
\end{array}
\xrightarrow{\text{Kon}_C[\partial_0(\Delta), \Gamma_0 \vdash^e N N': \partial_1(R)] \circ i_{\Gamma_0}^{\partial_1(\Delta)} \circ (\text{id} \times \Delta r) \circ \rho^1} \quad G'
\]

The step corresponding to \{pre, \_\} is also a direct result of the fibration condition using the factorization mentioned above.

The steps corresponding to the equational rules make use of lemma 5.42 and the identity condition. For example, the inductive hypothesis in the
step corresponding to \{eq,sym\} guarantees a square of the following shape.

\[
\begin{array}{c}
F \\
\downarrow I_{\text{Kon}_C[F]} \\
F' \\
\end{array}
\begin{array}{c}
\text{Kon}_C[M] \circ i_{\Gamma_0}^{\Gamma} \circ (\text{id} \times \Delta r) \circ \rho^1 \\
\downarrow I_{\text{Kon}_C[A]} \\
\text{Kon}_C[N] \circ i_{\Gamma_0}^{\Gamma} \circ (\text{id} \times \Delta r) \circ \rho^1 \\
\end{array}
\begin{array}{c}
G \\
G' \\
\end{array}
\]

By the identity condition, we have the following equality of morphisms.

\[
\text{Kon}_C[M] \circ i_{\Gamma_0}^{\Gamma} \circ (\text{id} \times \Delta r) \circ \rho^1 = \text{Kon}_C[N] \circ i_{\Gamma_0}^{\Gamma} \circ (\text{id} \times \Delta r) \circ \rho^1
\]

Hence, the above square is exactly the square desired.

\[
\begin{array}{c}
F \\
\downarrow I_{\text{Kon}_C[F]} \\
F' \\
\end{array}
\begin{array}{c}
\text{Kon}_C[N] \circ i_{\Gamma_0}^{\Gamma} \circ (\text{id} \times \Delta r) \circ \rho^1 \\
\downarrow I_{\text{Kon}_C[A]} \\
\text{Kon}_C[M] \circ i_{\Gamma_0}^{\Gamma} \circ (\text{id} \times \Delta r) \circ \rho^1 \\
\end{array}
\begin{array}{c}
G \\
G' \\
\end{array}
\]

Many equational rules express the expected properties of function types and polymorphic types. For example, to get the step corresponding to \{eta_fun\}, we note the following equalities arise from expanding out the definitions.

\[
\begin{align*}
\llbracket \eta, \eta_0 ; \Gamma, \Gamma_0 \Vdash \ell \ (\lambda x : A.M \ x) ; A \to B \rrbracket & = \text{curry} (\llbracket \eta, \eta_0 ; \Gamma, \Gamma_0, x : A \Vdash \ell \ M \ x ; B \rrbracket) \\
& = \text{curry} (\text{ap} \circ (\llbracket \eta, \eta_0 ; \Gamma, \Gamma_0, x : A \Vdash \ell \ M ; A \to B \rrbracket, \llbracket \eta, \eta_0 ; \Gamma, \Gamma_0, x : A \Vdash \ell \ x ; A \rrbracket)) \\
& = \text{curry} (\text{ap} \circ (\llbracket \eta, \eta_0 ; \Gamma, \Gamma_0 \Vdash \ell \ M ; A \to B \rrbracket \times \text{ID}))
\end{align*}
\]

Thus, we see that the morphism \(\llbracket \eta, \eta_0 ; \Gamma, \Gamma_0 \Vdash \ell \ (\lambda x : A.M \ x) ; A \to B \rrbracket\) is equal to \([\eta, \eta_0 ; \Gamma, \Gamma_0 \Vdash \ell \ M : A \to B] \) from the adjunction giving the Cartesian closed structure of \(G\).

The step corresponding to \{eq_term\} uses the well-pointedness of \(G\) in a significant way. Suppose \(\eta ; \Gamma, x : A \Vdash \eta_0 \Gamma_0 M =_B N\) is derivable as a result of \{eq_term\} and \((\tilde{C}, r) \in P_{\text{Param}}(\eta_0 \Gamma_0)\). We consider separate cases (each with two sub-cases) to show the following equation.

\[
\text{Kon}_C[\Gamma, x : A, \Gamma_0 \Vdash \ell \ M ; B] \circ i_{\Gamma_0}^{\Gamma} x : A \circ (\text{id} \times \Delta r) = \text{Kon}_C[\Gamma, x : A, \Gamma_0 \Vdash \ell \ N ; B] \circ i_{\Gamma_0}^{\Gamma} x : A \circ (\text{id} \times \Delta r)
\]

141
• Case I: \( \Gamma_0 \) is not empty. For any point \( r_x:1 \rightarrow [\eta_0 \vdash^L A][C] \), we can pair \( r \) and \( r_x \) to get a point \( \langle r, r_x \rangle:1 \rightarrow [\eta_0 \vdash^L \Gamma_0, x:A][C] \) so that \( \langle C, \langle r, r_x \rangle \rangle \) to get a pair of appropriate parameters for \( \langle \eta_0|\Gamma_0, x:A \rangle \). Thus, the inductive hypothesis ensures that there is a square from \( F = I_{Kon_C[\Gamma]} \) to \( G = I_{Kon_C[B]} \) as follows.

\[
\begin{array}{c}
F \\
\downarrow \text{Kon}_C[\Gamma, \Gamma_0, x:A \vdash^L M:B] \circ i^{_0}_{x:A} \circ (id \times \Delta \langle r, r_x \rangle) \circ \rho_1 \\
\downarrow \text{Kon}_C[\Gamma, x:A, \Gamma_0 \vdash^L M:B] \circ i^{_0}_{x:A, \Gamma_0} \circ (id \times \Delta \langle r, r_x \rangle) \circ \rho_1 \\
\downarrow \text{Kon}_C[\Gamma, x:A, \Gamma_0 \vdash^L N:B] \circ i^{_0}_{x:A, \Gamma_0} \circ (id \times \Delta \langle r, r_x \rangle) \circ \rho_1 \\
\downarrow \text{Kon}_C[\Gamma, \Gamma_0, x:A \vdash^L N:B] \\
G \\
\end{array}
\]

The identity condition assures that the two morphisms this square lies over are equal. Reordering the contexts ensures the following equality holds between morphisms as indicated.

\[
\text{Kon}_C[\Gamma, x:A, \Gamma_0 \vdash^L M:B] \circ i^{_0}_{x:A, \Gamma_0} \circ (id \times \Delta \langle r, r_x \rangle) = \\
\text{Kon}_C[\Gamma, x:A, \Gamma_0 \vdash^L N:B] \circ i^{_0}_{x:A, \Gamma_0} \circ (id \times \Delta \langle r, r_x \rangle)
\]

– Case Ia: \( \Gamma \) is not empty. Note that \( i^{\Gamma}_{x:A, \Gamma_0} = id = i^{\Gamma}_{x:A, \Gamma_0} \). Since every point \( 1 \rightarrow \text{Kon}_C[\eta_0, \eta_0 \vdash^L A] \) is of the form \( \Delta r_x \), the well-pointedness of \( G \) ensures that equation * holds.

– Case Ib: \( \Gamma \) is empty. Note that \( i^{_0}_{x:A, \Gamma_0} = \lambda \circ \lambda = i^{\Gamma}_{x:A, \Gamma_0} \circ \lambda. \)

Therefore, we get the following string of equalities.

\[
\begin{align*}
\text{Kon}_C[\emptyset, x:A, \Gamma_0 \vdash^L M:B] \circ i^{_0}_{x:A} & \circ (\Delta r_x \times \Delta r) \\
& = \text{Kon}_C[\emptyset, x:A, \Gamma_0 \vdash^L M:B] \circ i^{_0}_{x:A, \Gamma_0} \circ \lambda \circ (\Delta r_x \times \Delta r) \\
& = \text{Kon}_C[\emptyset, x:A, \Gamma_0 \vdash^L M:B] \circ i^{_0}_{x:A, \Gamma_0} \circ (id \times \Delta r_x \times \Delta r) \circ \lambda \\
& = \text{Kon}_C[\emptyset, x:A, \Gamma_0 \vdash^L N:B] \circ i^{_0}_{x:A, \Gamma_0} \circ (id \times \Delta r_x \times \Delta r) \circ \lambda \\
& = \text{Kon}_C[\emptyset, x:A, \Gamma_0 \vdash^L N:B] \circ i^{_0}_{x:A, \Gamma_0} \circ \lambda \circ (\Delta r_x \times \Delta r) \\
& = \text{Kon}_C[\emptyset, x:A, \Gamma_0 \vdash^L N:B] \circ i^{_0}_{x:A} \circ (\Delta r_x \times \Delta r)
\end{align*}
\]

Since the above equation holds for all points \( \Delta r_x \), equation * is assured to hold by the well-pointedness of \( G \).

• Case II: \( \Gamma_0 \) is empty. It is necessarily the case that \( r \) is id, since it is the only point \( 1 \rightarrow [\eta_0 \vdash^L \emptyset][C]. \) For any point \( r_x:1 \rightarrow [\eta_0 \vdash^L A][C], \) we have that \( \langle C, r_x \rangle \in \text{Param}(\eta_0|\Gamma_0, x:A). \) Therefore the inductive
hypothesis and identity condition ensure the following equality.

\[
\begin{align*}
\text{Kon}_C[\Gamma, x: A, \Gamma_0 \vdash^\ell M: B] & \circ \iota^\Gamma_{x: A} \circ (\text{id} \times \Delta r_x) = \\
\text{Kon}_C[\Gamma, x: A, \Gamma_0 \vdash^\ell N: B] & \circ \iota^\Gamma_{x: A} \circ (\text{id} \times \Delta r_x)
\end{align*}
\]

- Case IIa: \( \Gamma \) is non-empty. Recalling the definition of the intermediate morphisms, we note the following chain of equalities.

\[
i^\Gamma_{x: A} = \rho^1 = \text{id} \circ \rho^1 = \iota^\Gamma_{x: A} \circ \rho^1
\]

Therefore, the following chain of equalities is valid.

\[
\begin{align*}
\text{Kon}_C[\Gamma, x: A, \emptyset \vdash^\ell M: B] & \circ \iota^\Gamma_{x: A} \circ (\text{id} \times \Delta r_x \times \text{id}) \\
= & \text{Kon}_C[\Gamma, x: A, \emptyset \vdash^\ell M: B] \circ \iota^\Gamma_{x: A} \circ \rho^1 \circ (\text{id} \times \Delta r_x \times \text{id}) \\
= & \text{Kon}_C[\Gamma, x: A, \emptyset \vdash^\ell M: B] \circ \iota^\Gamma_{x: A} \circ (\text{id} \times \Delta r_x) \circ \rho^1 \\
= & \text{Kon}_C[\Gamma, x: A, \emptyset \vdash^\ell N: B] \circ \iota^\Gamma_{x: A} \circ (\text{id} \times \Delta r_x \times \text{id}) \\
= & \text{Kon}_C[\Gamma, x: A, \emptyset \vdash^\ell N: B] \circ \iota^\Gamma_{x: A} \circ (\text{id} \times \Delta r_x \times \text{id})
\end{align*}
\]

Since this holds for all points \( \Delta r_x : 1 \rightarrow \text{Kon}_C[\eta, \eta_0 \vdash^\ell A] \), the well-pointedness of \( \text{G} \) ensures equation \(*\).

- Case IIb: \( \Gamma \) is empty. We make use of the following chain of equalities (where \( \sigma \) gives the commutativity of the product).

\[
\begin{align*}
i^\emptyset_{x: A} \circ (\Delta r_x \times \text{id}) & = \rho^1 \circ (\Delta r_x \times \text{id}) \\
= & \lambda^1 \circ \sigma \circ (\Delta r_x \times \text{id}) \\
= & \lambda^1 \circ (\text{id} \times \Delta r_x) \circ \sigma
\end{align*}
\]

This is used in the following chain of equalities.

\[
\begin{align*}
\text{Kon}_C[\emptyset, x: A, \emptyset \vdash^\ell M: B] & \circ \iota^\emptyset_{x: A} \circ (\Delta r_x \times \Delta r) \\
= & \text{Kon}_C[\emptyset, x: A, \emptyset \vdash^\ell M: B] \circ \iota^\emptyset_{x: A} \circ (\text{id} \times \Delta r_x) \\
= & \text{Kon}_C[\emptyset, x: A, \emptyset \vdash^\ell M: B] \circ \iota^\emptyset_{x: A} \circ (\text{id} \times \Delta r_x) \circ \sigma \\
= & \text{Kon}_C[\emptyset, x: A, \emptyset \vdash^\ell N: B] \circ \iota^\emptyset_{x: A} \circ (\text{id} \times \Delta r_x) \circ \sigma \\
= & \text{Kon}_C[\emptyset, x: A, \emptyset \vdash^\ell N: B] \circ \iota^\emptyset_{x: A} \circ (\Delta r_x \times \text{id}) \\
= & \text{Kon}_C[\emptyset, x: A, \emptyset \vdash^\ell N: B] \circ \iota^\emptyset_{x: A} \circ (\Delta r_x \times \Delta r)
\end{align*}
\]

The well-pointedness of \( \text{G} \) establishes equation \(*\).

Having established equation \(*\) in all cases, we can see that the square
\[ I_{\text{Kon}_C [\Gamma, x: A, \Gamma_0 \vdash^I M: B] \circ i_{\Gamma_0}^\Gamma \circ (\text{id} \times \Delta r) \circ \rho^I } \] is a square of the desired shape.

\[
\begin{array}{c}
F \quad \text{Kon}_C [\Gamma, x: A, \Gamma_0 \vdash^I M: B] \circ i_{\Gamma_0}^\Gamma \circ (\text{id} \times \Delta r) \circ \rho^I \quad G \\
F' \quad \text{Kon}_C [\Gamma, x: A, \Gamma_0 \vdash^I N: B] \circ i_{\Gamma_0}^\Gamma \circ (\text{id} \times \Delta r) \circ \rho^I \quad G'
\end{array}
\]

As mentioned in the above proof, the identity condition allows one to conclude the equality of morphisms from equality judgments in System P.

**Corollary 5.45**

If \( \eta : \Gamma \vdash_{\eta_0|\Gamma_0}^I M =_A N \) is derivable in System P, the following equality of morphisms holds in the model.

\[
[\eta, \eta_0 ; \Gamma, \Gamma_0 \vdash^I M : A] = [\eta, \eta_0 ; \Gamma, \Gamma_0 \vdash^I N : A]
\]

**Proof.** Since \( \eta \vdash_{\eta_0|\Gamma_0}^I \) and \( \eta \vdash_{\eta_0|\Gamma_0}^I \) are both interpreted by identity edges, the theorem gives the following equality for every \((\bar{C}, r) \in \text{Param}(\eta_0|\Gamma_0)\).

\[
\text{Kon}_C [\eta, \eta_0 ; \Gamma, \Gamma_0 \vdash^I M : B] \circ i_{\Gamma_0}^\Gamma \circ (\text{id} \times \Delta r) \circ \rho^I = \text{Kon}_C [\eta, \eta_0 ; \Gamma, \Gamma_0 \vdash^I N : B] \circ i_{\Gamma_0}^\Gamma \circ (\text{id} \times \Delta r) \circ \rho^I
\]

Since \( \rho^I \) is an isomorphism, it can be removed from both sides of the equality. For every \(|\eta_0|\)-tuple of vertices \( \bar{C} \), the quantification over \( \text{Param}(\eta_0|\Gamma_0) \) covers all points \( \Delta r : 1 \to \text{Kon}_C [\eta, \eta_0 | \Gamma_0] \), so the well-pointedness of \( G \) ensures the following equality.

\[
\text{Kon}_C [\eta, \eta_0 ; \Gamma, \Gamma_0 \vdash^I M : B] \circ i_{\Gamma_0}^\Gamma = \text{Kon}_C [\eta, \eta_0 ; \Gamma, \Gamma_0 \vdash^I N : B] \circ i_{\Gamma_0}^\Gamma
\]

Since \( i_{\Gamma_0}^\Gamma \) is an isomorphism, and the above equality holds for all \(|\eta_0|\)-tuples \( \bar{C} \), the desired equality holds.  

The primary consequence of Theorem 5.44 is that the definition of the model of System P is valid in the following sense. All the squares necessary for the interpretation do exist in any well-pointed parametricity setting for System F. We now turn our attention to some consequences of this fact. Recall that \( \text{PER}_\omega \) is a parametricity setting for System F (corollary 3.16). Therefore System P can be modeled in \( \text{PER}_\omega \). Let us consider a little more detail about the interpretation of System P and System F in \( \text{PER}_\omega \).
Recall the type $\textbf{Bool}$ was defined to be $\forall X. X \Rightarrow (X \Rightarrow X)$. Therefore, the interpretation of the type judgment $\emptyset \vdash \textbf{Bool}$ (evaluated at the unique vertex $* \in (\text{PER}_\omega)^0$) is as follows.

$$[[\emptyset \vdash \textbf{Bool}]_\star] = \lim (\{X \vdash X \Rightarrow (X \Rightarrow X)\})$$

$$= \lim (\{X \vdash X \Rightarrow ([X \vdash X \Rightarrow [X \vdash X]])\})$$

$$= \forall Y. Y \Rightarrow (Y \Rightarrow Y)$$

Recalling the definition of parametric limits in $\text{PER}_\omega$ (page 59), we have the following characterization of the PER $[[\emptyset \vdash \textbf{Bool}]_\star]$.

$$n [[\emptyset \vdash \textbf{Bool}]_\star] m \iff \text{ for every edge } R \text{ of } \text{PER}_\omega,$$

$$n [R \Rightarrow (R \Rightarrow R)] n \land n [R \Rightarrow (R \Rightarrow R)] m \land m [R \Rightarrow (R \Rightarrow R)] m$$

There are two canonical terms of type $\textbf{Bool}$, as follows.

$$\text{true} = \Lambda X. \lambda x: X. \lambda y: X. x \quad \text{false} = \Lambda X. \lambda x: X. \lambda y: X. y$$

To discuss the interpretation of the term judgments for $\text{true}$ and $\text{false}$, we begin with the following observations.

$$[X ; x: X \vdash \lambda y: X. x: X \Rightarrow X]$$

is tracked by $k$ such that for all $n$,\n
$$\phi_k(n)$$

is a code for the constantly $n$ function.

$$[X ; x: X \vdash \lambda y: X. y: X \Rightarrow X]$$

is tracked by $h$ such that for all $n$,\n
$$\phi_h(n)$$

is a code for the identity function.

Fix natural numbers $k$ and $h$ as indicated above. Therefore, the following term judgments are interpreted by constant functions, as indicated.

$$[X ; \emptyset \vdash \lambda x: X. \lambda y: X. x: X \Rightarrow X \Rightarrow X]$$

is realized by $\Delta(k)$

$$[X ; \emptyset \vdash \lambda x: X. \lambda y: X. y: X \Rightarrow X \Rightarrow X]$$

is realized by $\Delta(h)$

Recall that for any parametric $\omega$-transformation $\tau: \Delta A \rightarrow F$, the morphism $\Lambda(\tau): A \rightarrow \forall X F(X)$ is realized by a uniform realizer of $\tau$. Therefore, we can characterize the interpretations of $\text{true}$ and $\text{false}$ as realized by constant functions.

$$[[\emptyset ; \emptyset \vdash \text{true}: \textbf{Bool}]$$

is realized by the constantly $k$ function, and

$$[[\emptyset ; \emptyset \vdash \text{false}: \textbf{Bool}]$$

is realized by the constantly $h$ function.

**Lemma 5.46**

*The interpretation in $\text{PER}_\omega$ of the term judgments $\emptyset ; \emptyset \vdash \text{true}: \textbf{Bool}$ and*
\( \emptyset ; \emptyset \vdash false : \text{Bool} \) are distinct.

Proof. To show that \( \emptyset ; \emptyset \vdash true : \text{Bool} \) and \( \emptyset ; \emptyset \vdash false : \text{Bool} \) are different realizable functions \( 1 \to \llbracket \eta \vdash \text{Bool} \rrbracket \), it suffices to show that \( k \) and \( h \) are not related by \( \emptyset \vdash \text{Bool} \). Consider the equivalence relation \( R : \mathbb{N} \leftrightarrow \mathbb{N} \) defined by \( n R m \iff n \mod 2 = m \mod 2 \).

Since \( \phi_{\eta}(0) = 1 \) and \( \phi_{\eta}(0) = 1 \), it is the case that \( \phi_k(0) \) and \( \phi_h(0) \) are not related by \( R \Rightarrow R \). Therefore \( k \) and \( h \) are not related by \( R \Rightarrow (R \Rightarrow R) \) for this particular \( R \). Hence \( k \) and \( h \) are not related by \( \emptyset \vdash \text{Bool} \).

Since \( \emptyset ; \emptyset \vdash \text{false} \neq \emptyset ; \emptyset \vdash \text{true} \), corollary 5.45 implies that the judgment \( \emptyset ; \emptyset \vdash_0 \text{true} = \text{false} \) cannot be derived in System P. Thus System P does not suffer the same weakness as the preliminary version of System R did (see section 5.2).

### 5.5 Semantic Representation Results

We now turn our attention to how System P can be used to reason about parametricity settings for System F. We begin by observing how structural properties of System P are reflected by the interpretation in an intuitive manner. For instance, weakening (lemma 5.1) corresponds to ignoring inputs.

**Proposition 5.47**

Suppose \( \eta \vdash_0 \alpha \) and \( \eta ; \Delta \vdash_0 \alpha \).

\[
\begin{align*}
\langle \eta, \eta' \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C_1, \ldots, C_m, D_1, \ldots, D_m)}, \rho \rangle & = \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', r)} \circ \Pi \\
\langle \eta, \eta' \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C_1, \ldots, C_m, D_1, \ldots, D_m), \rho} & = \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', r)} \circ \Pi \\
\langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', (r, x, y))} & = \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', r)} \\
\langle \eta, \Delta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', \Delta), \rho} & = \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C, r)} \circ \pi \\
\langle \eta ; \Delta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', \Delta), \rho} & = \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', r)} \circ \pi
\end{align*}
\]

where \( \pi : \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', \Delta), \rho} \times \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', (r, x, y))} \to \langle \eta \mid \eta_0 \mid \Gamma_0 \alpha \rangle_{(C', r)} \) is the obvious projection.
\[ \langle \eta \vdash \alpha \rangle_{(C,r)} = \langle \eta \vdash \alpha \rangle_{P_0(\widehat{C},d_0,\sigma_r)} \circ P \]

\[
\left( \langle \eta \vdash \Delta \rangle \left( T \vdash N \right) \right)_{(\tilde{C},\tilde{r})}
\]

where \( P : G^n \to G^n \) and \( P_0 : G^m \to G^m \) are non-variant functors corresponding to the permutations from \( \tilde{\eta} \to \eta \) and \( \tilde{\eta}_0 \to \eta_0 \) in the apparent manner, and \( \Delta: [\tilde{\eta}_0 \vdash \Gamma_0] \to [\eta_0 \vdash \Gamma_0] \) \((P_0(\widehat{C}))\) and \( d: [\eta \vdash \alpha] \to [\eta \vdash \alpha] \) \((P_0(\widehat{C}))\) are the apparent parameteric transformations.

This is shown by induction on derivations. Given the close relationship between the model of System P and that of System F, the above proposition implies analogous statements about the interpretation of System F judgments, such as the following.

**Corollary 5.48**

If \( \eta \vdash \alpha \), then \( [\eta, X \vdash \alpha] = [\eta \vdash \alpha] \circ \Pi \).

Another expected structural property of System P is that it supports substitutions (lemma 5.3). This is reflected in the model by computing the argument corresponding to the variable that was replaced, as indicated in the following lemma.

**Lemma 5.49**

Suppose \( \eta \vdash \alpha \vdash S \) and \( \eta, X \vdash \alpha \vdash M \) \(R \) is a derivable relatedness judgment. Let \( (\tilde{C},\tilde{r}) \) be appropriate parameters for \( (\eta_0,\Gamma_0) \). Using \( \langle S \rangle \) to denote \( \langle \eta \vdash \alpha \vdash S \rangle \) \((\tilde{C},\tilde{r})\), the following equalities hold.

\[
\langle \eta \vdash \alpha[S/X] \rangle_{(\tilde{C},\tilde{r})} = \text{last}(\langle S \rangle) \langle \eta, X \vdash \alpha[S/X] \rangle_{(\tilde{C},\tilde{r})}
\]

\[
\left( \eta \vdash \Delta[S/X] \right) \left( \eta \vdash \alpha \vdash \Delta[S/X] \right)_{(\tilde{C},\tilde{r})}
\]

The corresponding assertions for System F also hold. Consider any type judgment \( \eta, \eta_0 \vdash B \). Suppose \( \eta, \eta_0 \vdash \alpha \) and \( \eta, \eta_0 \vdash \Gamma : A \) are derivable in System F. Then for any \( |\eta| \)-tuple \( \tilde{A} \) and \( |\eta_0| \)-tuple \( \tilde{C} \) of vertices, the
following equalities hold where \( D = [\eta, \eta_0 \vdash B_1, \cdots B_n, C_1, \cdots C_m] \).

\[
[\eta, \eta_0 \vdash \alpha[B/X]](A_1, \cdots A_n, C_1, \cdots C_m)
= [\eta, X, \eta_0 \vdash \alpha](A_1, \cdots A_n, D, C_1 \cdots C_m)
\]

\[
[\eta, \eta_0 ; \Gamma[B/X] \vdash M[B/X]: A[B/X][A_1, \cdots A_n, C_1, \cdots C_m)]
= [\eta, X, \eta_0 ; \Gamma \vdash M: A](A_1, \cdots A_n, D, C_1, \cdots C_m)
\]

**Proof.** The proof proceeds by induction on derivations. The assertion about type judgments in System F follows from the assertion about relation judgments in System P since \( I(Kon_B[\eta, \eta_0 \vdash \alpha]) = \langle \eta \vdash \eta_0 \Gamma \alpha \rangle (\beta, \rho) \) (lemma 5.42). Since the interpretations of relatedness judgments in System P were given using the interpretation of term judgments of System F, we use induction to show the assertion about term judgments and get the assertion about relatedness judgments as a consequence. Since relation judgments can contain terms (in pre-image relations), the proofs for relation judgments and for term judgments proceed by simultaneous induction.

Since \( \{\text{rel}\} \) leads to \( \langle \eta, X \vdash \eta_0 \Gamma_0 X \rangle (\mathcal{C}, \rho) \) being interpreted as the projection, \( \Pi_{n+1} \), we get the base case of the induction from when \( \alpha \) is a relation variable. For an \( n \)-tuple of vertices \( \vec{A} \), this is as follows.

\[
\text{last}(\langle S \rangle)(\langle \eta, X \vdash \eta_0 \Gamma_0 X \rangle (\mathcal{C}, \rho)) \vec{A} = \Pi_{n+1}(A_1, \cdots A_n, \langle S \rangle \vec{A}) = \langle S \rangle \vec{A}
\]

The inductive steps simply propagate the equality along as expected.

For the assertion about term judgments, we begin by noting that the interpretation of \( \{\text{variable}\} \) was given by the projection parametric transformation. Note the assertion about type judgments ensures that the source and target of the following two morphisms are the same.

\[
[\eta, \eta_0 (x_1 : B_1, \cdots x_n : B_n)[B/X] \vdash x_j : B_j[B/X][A_1, \cdots A_n, C_1, \cdots C_m])
\]

and \( [\eta, X, \eta_0 (x_1 : B_1, \cdots x_n : B_n) \vdash x_j : B_j][A_1, \cdots A_n, D, C_1, \cdots C_m] \)

Therefore, these two morphisms are the same projection morphism. The inductive steps propagate the equality as one might expect.

The previous lemma shows that substitution of relations or types is reflected in the model by computing arguments in an expected manner. Similarly, one can show that substitution of terms corresponds to composition of the interpretations as stated in the following lemma.
Lemma 5.50

For any \( \eta ; \Gamma \vdash N : B \) and \( \eta ; \Gamma, x : B \vdash M : A \), the following equality holds.

\[
[\eta; \Gamma \vdash M^{[N/x]}; A] = [\eta; \Gamma, x : B \vdash M; A] \circ (\text{id}, [\eta; \Gamma \vdash N; B])
\]

Consequently, if \( \eta ; \Delta \vdash_{\eta_0\Gamma_0}^{N R} M \) and \( \eta ; \Delta, x \vdash_{\eta_0\Gamma_0}^{N' R} M' \), then for any appropriate parameters \((\tilde{C}, r) \in \text{Param}(\eta_0\Gamma_0)\), the following equality holds.

\[
\left( \left[ \eta; \Delta \vdash_{\eta_0\Gamma_0}^{N R} M^{[N/x]} \right]_{\tilde{C}, r} \right) = \left( \left[ \eta; \Delta, x \vdash_{\eta_0\Gamma_0}^{N' R} M' \right]_{\tilde{C}, r} \right) \circ \left( \text{id}, \left[ \eta; \Delta \vdash_{\eta_0\Gamma_0}^{N R} M^{[N/x]} \right]_{\tilde{C}, r} \right)
\]

Proof. Proceed by induction on the derivation of \( \eta ; \Gamma, x : B \vdash M : A \). If \( M \) is a variable, then one of the following two cases applies.

1. \([\eta; \Gamma, x : B \vdash x : B] \circ (\text{id}, [\eta; \Gamma \vdash N; B])
   \begin{align*}
   &= \pi_{n+1} \circ (\text{id}, [\eta; \Gamma \vdash N; B]) \\
   &= [\eta; \Gamma \vdash N; B] \\
   &= [\eta; \Gamma \vdash x^{[N/x]}; B]
   \end{align*}

2. \([\eta; x_1 : A_1, \ldots, x_n : A_n, x : B \vdash x_j : A_j] \circ (\text{id}, [x_1 : A_1, \ldots, x_n : A_n \vdash N; B])
   \begin{align*}
   &= \pi_j \circ (\text{id}, [x_1 : A_1, \ldots, x_n : A_n \vdash N; B]) \\
   &= \pi_j \circ \text{id} = \pi_j \\
   &= [\eta; x_1 : A_1, \ldots, x_n : A_n \vdash x_n : A_n] \\
   &= [\eta; x_1 : A_1, \ldots, x_n : A_n \vdash x_n^{[N/x]}; A_n]
   \end{align*}

Interpretations corresponding to the other term forming rules use the same construction on both sides of the equation, hence propagate the equality as expected.

Since fixing arguments of a parametric transformation "commutes" with pairing and composition, applications of \( \text{last}(\tilde{F}) \) commute in the expected manner. Specifically, if \( n + n' = |\eta| \) and \( \tilde{F} \) is an \( n' \)-tuple of \( n \)-ary non-variant functors, then we have the following equality.

\[
\begin{align*}
\text{last}(\tilde{F})[\eta; \Gamma \vdash M^{[N/x]}] \\
&= \text{last}(\tilde{F}) \left( [\eta; \Gamma, x : B \vdash M; A] \circ (\text{id}, [\eta; \Gamma \vdash N; B]) \right) \\
&= \left( \text{last}(\tilde{F})[\eta; \Gamma, x : B \vdash M; A] \right) \circ (\text{id}, \left( \text{last}(\tilde{F})[\eta; \Gamma \vdash N; B] \right))
\end{align*}
\]

149
Using the special case $\text{Kon}_C$ of non-variant functors $\text{last}(\vec{F})$, it follows that squares below have the same shape.

$$
\left\langle \eta ; \Delta \vdash \nu_0 \Gamma_0 \quad M[N/x] \quad S \middle| M'[N'/x] \right\rangle_{(\vec{c}, \bar{r})}
$$

$$
\left\langle \eta ; \Delta, \quad x R \vdash \nu_0 \Gamma_0 \quad M \quad S \middle| M' \right\rangle_{(\vec{c}, \bar{r})} \circ \text{id}, \left\langle \eta ; \Delta \vdash \nu_0 \Gamma_0 \quad N \quad R \right\rangle_{(\vec{c}, \bar{r})}
$$

Since $\mathbf{G}$ is relational, there is only one square of that shape.

The correspondence between substitution in System P and composition in the model indicated in the above lemma can be used to show how composition in System P corresponds to composition in the model. This correspondence makes use of the following observation about function application.

**Lemma 5.51**

Suppose $\eta ; \Gamma \vdash M : A \Rightarrow B$ is derivable and $x$ is fresh. If $|\eta| = n + n'$ and $\vec{F}$ is an $n'$-tuple of $n$-ary non-variant functors, then the following equality holds.

$$
\text{last}(\vec{F})[\eta ; \Gamma, x : A \vdash M \ x : B] = \text{uncurry}(\text{last}(\vec{F})[\eta ; \Gamma \vdash M : A \Rightarrow B])
$$

**Proof.** Using the inductive definition of the interpretation of System F terms (see section 3.3) and the way weakening is reflected in the model (proposition 5.47), we get the following string of equalities.

$$
\text{last}(\vec{F})[\eta ; \Gamma, x : A \vdash M \ x : B] = \text{last}(\vec{F}) \left( \text{ap} \circ \left( [\eta ; \Gamma, x : A \vdash M : A \Rightarrow B] \circ [\eta ; \Gamma, x : x : A] \right) \right)
$$

$$
= \text{last}(\vec{F}) \left( \text{ap} \circ \left( \left( [\eta ; \Gamma \vdash M : A \Rightarrow B] \circ \Pi \right), \left( [\eta ; x : A \vdash x : A] \circ \Pi' \right) \right) \right)
$$

$$
= \text{last}(\vec{F}) \left( \text{ap} \circ \left( [\eta ; \Gamma \vdash M : A \Rightarrow B] \times [\eta ; x : A \Rightarrow x : A] \right) \circ \left( \Pi, \Pi' \right) \right)
$$

$$
= \text{last}(\vec{F}) \left( \text{ap} \circ \left( \left( [\eta ; \Gamma \vdash M : A \Rightarrow B] \times [\eta ; x : A \Rightarrow x : A] \right) \right) \right)
$$

$$
= \text{ap} \circ \left( \text{last}(\vec{F})[\eta ; \Gamma \vdash M : A \Rightarrow B] \times \text{id} \right)
$$

Since uncurry($f$) is defined as $\text{ap} \circ (f \times \text{id})$ for any morphism $f$, the above gives the desired equality.

$\diamond$

**Lemma 5.52**

Suppose $\vec{F}$ is an $n'$-tuple of $n$-ary non-variant functors where $n + n' = |\eta|$. For any parametric transformation $\tau : 1 \rightarrow \text{last}(\vec{F})[\eta \vdash \Gamma]$, the following
equality holds,

\[
\left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash M \circ N : A \Rightarrow C \circ \tau \right)^* = \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash M : B \Rightarrow C \circ \tau \right)^* \circ \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B \circ \tau \right)^* 
\]

**Proof.** The above morphisms \( \text{last}(\bar{F})[\eta] \vdash A \Rightarrow \text{last}(\bar{F})[\eta] \vdash C \) are shown to be equal by considering an arbitrary point \( \sigma : 1 \vdash \text{last}(\bar{F})[\eta] \vdash A \).

We first apply corollary 5.40 to rewrite the currying (to a morphism of one variable) of a composition. Then the definition of \( M \circ N \) and the inductive definition of the interpretation (see section 3.3) can be used to simplify.

\[
\left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash M \circ N : A \Rightarrow C \circ \tau \right)^* \circ \sigma = \text{uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma \vdash M \circ N : A \Rightarrow C) \circ \langle \tau, \sigma \rangle \\
= \text{uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma \vdash \lambda x : A. M(N x) : A \Rightarrow C) \circ \langle \tau, \sigma \rangle \\
= \text{uncurry}(\text{curry}(\text{last}(\bar{F})[\eta] ; \Gamma, x : A \vdash M(N x) : C)) \circ \langle \tau, \sigma \rangle \\
= \text{last}(\bar{F})[\eta] ; \Gamma, x : A \vdash M(N x) : C \circ \langle \tau, \sigma \rangle
\]

Using properties of substitution (lemma 5.49) and observation of funtion application given in lemma 5.51, the above string of equalities can be extended.

\[
\begin{align*}
= & \text{last}(\bar{F})[\eta] ; \Gamma, x : A, y : B \vdash M \ y : C) \circ \\
& \langle \text{id, last}(\bar{F})[\eta] ; \Gamma, x : A \vdash N \ x : B) \rangle \circ \langle \tau, \sigma \rangle \\
= & \text{uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma, x : A \vdash M : B \Rightarrow C)) \circ \\
& \langle \text{id, uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B)) \rangle \circ \langle \tau, \sigma \rangle \\
= & \text{uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma, x : A \vdash M : B \Rightarrow C)) \circ \\
& \langle \langle \tau, \sigma \rangle, \ (\text{uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B)) \rangle \circ \langle \tau, \sigma \rangle) \rangle
\end{align*}
\]

Corollary 5.40 about uncurrying a composition to a morphism of one variable can be used twice so that the way weakening is reflected in the model (proposition 5.47) will give the desired composition.

\[
\begin{align*}
= & \left( \text{last}(\bar{F})[\eta] ; \Gamma, x : A \vdash M : B \Rightarrow C) \circ \langle \tau, \sigma \rangle \right)^* \circ \\
& \text{uncurry}(\text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B)) \circ \langle \tau, \sigma \rangle \\
= & \left( \text{last}(\bar{F})[\eta] ; \Gamma, x : A \vdash M : B \Rightarrow C) \circ \langle \tau, \sigma \rangle \right)^* \circ \\
& \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B)) \circ \tau \right)^* \circ \sigma \\
= & \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash M : B \Rightarrow C) \circ \Pi \circ \langle \tau, \sigma \rangle \right)^* \circ \\
& \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B)) \circ \tau \right)^* \circ \sigma \\
= & \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash M : B \Rightarrow C) \circ \tau \right)^* \circ \left( \text{last}(\bar{F})[\eta] ; \Gamma \vdash N : A \Rightarrow B)) \circ \tau \right)^* \circ \sigma
\end{align*}
\]

151
As this holds for all points \(\sigma\), well-pointedness assures the desired equality.

Recall that corollary 5.45 asserted that any equality \(\eta ; \Gamma \vdash_0 M =_A N\) that is derivable in System P implies that the interpretations \([\eta ; \Gamma \vdash^E M ; A]\) and \([\eta ; \Gamma \vdash N ; A]\) are equal. This leads to being able to show more complicated results that hold in System P, such as the initial algebra result of Theorem 5.27, also hold for well-pointed parametricity settings for System F.

For any type \(B\), the F-interpretation of a judgment \(\eta, X \vdash B\) is a PG-functor \(|G|^{n+1} \rightarrow G\) where \(n = |\eta|\). This PG-functor can be used along with last to define a canonical non-variant functor \(\langle B \rangle; G^{[n]} \rightarrow G^{[n]}\).

\[
\langle B \rangle F = \text{last}(F)([\eta, X \vdash B])
\]

When considering a plain type \(P\), one can additionally define a functor \(\langle P \rangle; (G^{[n]})^v \rightarrow (G^{[n]})^v\) that agrees with the above definition on vertices. The mapping of a morphism \(\tau; F \rightarrow G\) of \(G^{[n]}\) (that is, a parametric transformation) to \(\langle P \rangle\tau\) will be defined using the term \(P\tau(f)\).

We first show that for any \(n\)-tuple \(\vec{A}\) of vertices of \(G\), there is a morphism \(\alpha; F(\vec{A}) \rightarrow (\langle P \rangle F)(\vec{A})\) as follows.

\[
\alpha = [\eta, X, Y; f; X \Rightarrow Y \vdash P\tau(f); P\tau(X) \Rightarrow P\tau(Y)]_{\langle A_1, \ldots, A_n, F(\vec{A}), G(\vec{A})\rangle}
\]

This can be broken down into even smaller steps. It is immediate that the first two equalities (below) hold, hence the third equality holds.

\[
\begin{align*}
[\eta, X, Y \vdash X](A_1, \ldots, A_n, F(\vec{A}), G(\vec{A})) &= F(\vec{A}) \\
[\eta, X, Y \vdash Y](A_1, \ldots, A_n, F(\vec{A}), G(\vec{A})) &= G(\vec{A}) \\
[\eta, X, Y \vdash X \Rightarrow Y](A_1, \ldots, A_n, F(\vec{A}), G(\vec{A})) &= F(\vec{A}) \Rightarrow G(\vec{A})
\end{align*}
\]

By substitution (lemma 5.49), we deduce the following equality.

\[
\begin{align*}
[\eta, X, Y \vdash P\tau(X)](A_1, \ldots, A_n, F(\vec{A}), G(\vec{A})) &= [\eta, X \vdash P\tau](A_1, \ldots, A_n, F(\vec{A})) \\
&= (\langle P \rangle F)(\vec{A})
\end{align*}
\]

Similarly, one can show that the corresponding result for \(P\tau(Y)\).

\[
[\eta, X, Y \vdash P\tau(Y)](A_1, \ldots, A_n, F(\vec{A}), G(\vec{A})) = (\langle P \rangle G)(\vec{A})
\]

152
Therefore the following equation holds.

\[
[\eta, X, Y \vdash P(X) \Rightarrow P(Y)](A_1, \ldots, A_n, F(\bar{A}), G(\bar{A})) = (\langle P \rangle F)\bar{A} \Rightarrow (\langle P \rangle G)\bar{A}
\]

Consequently, the morphism \( \alpha \) defined above is of the desired type, namely

\( F(\bar{A}) \Rightarrow G(\bar{A}) \rightarrow P(X) \bar{A} \Rightarrow P(Y) \bar{A} \).

The morphism \( \tau_{\bar{A}} : F(\bar{A}) \rightarrow G(\bar{A}) \) can be used with the \( \alpha \) to define

\( (\langle P \rangle \tau)_{\bar{A}} : (\langle P \rangle F)\bar{A} \rightarrow (\langle P \rangle G)\bar{A} \) in a canonical manner. Formally, the definition is as follows.

\[
(\langle P \rangle \tau)_{\bar{A}} = (\alpha \circ \tau_{\bar{A}}^5)^{r}
\]

We note that, since currying are defined pointwise in the functor graph \( G^{F[n]} \), it is necessarily the case that \( (\tau^o)_{\bar{A}} = (\tau_{\bar{A}})^5 \). Therefore, we can be imprecise about whether the currying is in \( G \) or in \( G^{F[n]} \). A similarly observation applies when uncurrying a point as well.

The above definition gives the \( \bar{A} \)-component of the parametric transformation \( \langle P \rangle \tau : \langle P \rangle F \rightarrow \langle P \rangle G \) that completes the definition of the functor \( \langle P \rangle : (G^{F[n]})_v \rightarrow (G^{F[n]})_v \). Once we have established that the vertex and morphism part of \( \langle P \rangle \) determines a functor, it is not hard to see that \( \langle P \rangle \) is actually an indexed functor. Since \( G \) is relational (hence \( G^{F[n]} \) is relational), the shape on the right below uniquely determines the action of \( \langle P \rangle \) on a square of the shape on the left.

\[
\begin{array}{ccc}
F & \xrightarrow{\tau} & G \\
\downarrow P \quad & & \downarrow \quad P \quad \tau \\
F' & \xrightarrow{\tau'} & G'
\end{array}
\]

\[
\begin{array}{ccc}
\langle P \rangle F & \xrightarrow{\langle P \rangle \tau} & \langle P \rangle G \\
\downarrow P \quad & & \downarrow \quad P \quad \tau \\
\langle P \rangle F' & \xrightarrow{\langle P \rangle \tau'} & \langle P \rangle G'
\end{array}
\]

More than just an indexed functor, it is in fact fibred. In order to show that \( \langle P \rangle \) is a fibred functor (hence a PG-functor), we use induction on the structure of \( P \) to establish some intermediate results.

**Lemma 5.53**

Suppose \( \eta, X \vdash \text{Plain}_X(P) \) and \( n = |\eta| \). Further suppose that \( \bar{R} : \bar{A} \leftrightarrow \bar{A}' \) is an \( n \)-tuple of edges and \( S : C \leftrightarrow C' \) is a single edge of \( G \). Suppose \( g : B \rightarrow C \), \( g' : B' \rightarrow C' \) and \( h : C \rightarrow D \) are morphisms of \( G \). Let \( \eta' = \eta, Y, Z \). The following equalities hold. (Type information for terms and term variables
are suppressed for brevity.)

1. \((\eta'; f \vdash P_X(f))_{(A_1, \ldots, A_n, C, C)} \circ \text{id}_{\eta'}\)∗
   
   \(= \text{id}([\eta; X \vdash P(A_1, \ldots, A_n, C)])\)

2. \((\eta'; f \vdash P_X(f))_{(A_1, \ldots, A_n, B, D)} \circ (h \circ g)\)∗
   
   \(= \left(\left([\eta'; f \vdash P_X(f)]_{\Theta} \circ h\right)^{\circ}\left([\eta'; f \vdash P_X(f)]_{\Theta} \circ g^{\circ}\right)^{\circ}\right)^{\circ}\)
   
   where \(\Theta = (A_1, \ldots, A_n, B, C)\) and \(\Theta' = (A_1, \ldots, A_n, C, D)\)

3. \([([\eta'; f \vdash P_X(f)]_{\Theta} \circ g^{\circ})^{\circ}, ([\eta'; f \vdash P_X(f)]_{\Theta} \circ g^{\circ})^{\circ}]_{\eta}, X \vdash P]_{\mathcal{T}}\)
   
   \(= [\eta, X \vdash P](R_1, \ldots, R_n, [g, g']S)\)
   
   where \(\Theta = (A_1, \ldots, A_n, B, C)\), \(\Theta' = (A'_1, \ldots, A'_n, B', C')\)

   and \(\mathcal{T} = (R_1, \ldots, R_n, S)\)

Proof. These claims are shown by induction on the structure of the type expression \(P\). The induction proceeds simultaneously for the three steps and for all \(\eta\) such that \(\eta, X \vdash P\) is derivable.

As the variable cases are immediate, we focus on the two inductive cases.

When \(P = \forall \vec{Y}. \vec{P} :\) For any morphism \(k: D \to D'\) of \(\mathcal{G}\), we consider the morphism \((\eta'; f \vdash P_X(f))_{\hat{D}} \circ k^{\circ}\)∗ where \(\hat{D} = (A_1, \ldots, A_n, D, D')\). We shall also use the \(n+3\)-tuple \(\vec{E} = (A_1, \ldots, A_n, D, D', V)\) where \(V\) denotes a variable argument. We use \(\bar{\eta}\) to denote \(\eta', \vec{Y}\) and \(\Gamma\) to denote \(f: Y \Rightarrow z, \bar{x}: P_X(Y)\). By using the definition of \(P_X(f)\) and the details of the interpretation, we get the following string of equalities.

\[
([\eta'; f \vdash P_X(f)]_{\hat{D}} \circ k^{\circ})^{\circ}
= ([\eta'; f \vdash \lambda_x. \lambda Y. P_X(f)(x[Y])]_{\hat{D}} \circ k^{\circ})^{\circ}
= (\text{curry}([\eta'; f \vdash \lambda Y. P_X(f)(x[Y])]_{\hat{D}} \circ k^{\circ})^{\circ}
= (\text{curry}(\Lambda[\bar{\eta}; \Gamma \vdash \bar{P}_X(f)(x[\vec{Y}])]_{\vec{E}} \circ k^{\circ})^{\circ}

The expression \([\bar{\eta}; \Gamma \vdash \bar{P}_X(f)(x[\vec{Y}])]_{\vec{E}}\) is treated as a parametric transformation in the variable \(V\) in the apparent manner. After using the naturality of curry, the definition of uncurrying a point (definition 5.38) allows one to make use of the fact that uncurl and curry are inverses.

\[
= \left(\text{uncurry}\left(\text{curry}(\Lambda[\bar{\eta}; \Gamma \vdash \bar{P}_X(f)(x[\vec{Y}])]_{\vec{E}} \circ (k^{\circ} \times \text{id}))\right)^{\circ}\right)^{\circ}
= \text{uncurry}\left(\text{curry}(\Lambda[\bar{\eta}; \Gamma \vdash \bar{P}_X(f)(x[\vec{Y}])]_{\vec{E}} \circ (k^{\circ} \times \text{id}))\right)^{\circ} \circ \lambda^{-1}
= \Lambda[\bar{\eta}; \Gamma \vdash \bar{P}_X(f)(x[\vec{Y}])]_{\vec{E}} \circ (k^{\circ} \times \text{id}) \circ \lambda^{-1}

154
Morphisms can be shifted inside of $\Lambda(-)$ becoming a constant parametric transformation (lemma 3.3). Since currying a morphism of one variable is pointwise in $\mathbf{G}^{\mathbf{G}^n}$, it is immediate that it commutes with $\Delta$, that is, $\Delta(k^n) = (\Delta k)^n$. Thus, one can use the fact that curry and uncurry are inverses and the naturality of uncurry to produce a morphism (in the functor graph) of the form $\text{uncurry}(m) \circ \lambda^{-1} = m^*$. 

\begin{align*}
&= \Lambda(\lbrack \eta \rbrack ; \Gamma \vdash \bar{P}_\chi(f)(x[\bar{Y}])[\bar{E}] \circ \Delta(k^n \times \text{id}) \circ \Delta(\lambda^{-1})) \\
&= \Lambda(\lbrack \eta \rbrack ; \Gamma \vdash \bar{P}_\chi(f)(x[\bar{Y}])[\bar{E}] \circ ((\Delta k)^n \times \text{id}) \circ \lambda^{-1}) \\
&= \Lambda(\text{uncurry}(\text{curry}[\eta] ; \Gamma \vdash \bar{P}_\chi(f)(x[\bar{Y}])[\bar{E}]) \circ ((\Delta k)^n \times \text{id}) \circ \lambda^{-1}) \\
&= \Lambda(\text{uncurry}(\text{curry}[\eta] ; \Gamma \vdash \bar{P}_\chi(f)(x[\bar{Y}])[\bar{E}] \circ (\Delta k)^n \circ \lambda^{-1}) \\
&= \Lambda(\text{curry}[\eta] ; \Gamma \vdash \bar{P}_\chi(f)(x[\bar{Y}])[\bar{E}] \circ (\Delta k)^n)^* \\
\end{align*}

This can be rewritten using the inductive definition of the interpretation (corresponding to $\{\text{fun}_E\}$ and $\{\text{poly}_E\}$) and the way weakening is reflected in the interpretation (proposition 5.47).

\begin{align*}
&= \Lambda(\text{curry}(\lbrack \eta \rbrack ; \Gamma, y \vdash \bar{P}_\chi(f) y)[\bar{E}] \circ (\text{id}, \lbrack \eta \rbrack ; \Gamma \vdash x[\bar{Y}])[\bar{E}] \circ (\Delta k)^n)^* \\
&= \Lambda(\text{curry}(\lbrack \eta \rbrack ; \Gamma, y \vdash \bar{P}_\chi(f) y)[\bar{E}] \circ (\text{id}, (\omega_{\text{poly}} x \times \text{id}) \circ [\eta] ; \Gamma \vdash x)[\bar{E}] \circ (\Delta k)^n)^* \\
&= \Lambda(\text{curry}(\lbrack \eta \rbrack ; \Gamma, y \vdash \bar{P}_\chi(f) y)[\bar{E}] \circ (\text{id}, (\omega_{\text{poly}} \circ \pi')(\text{id}, \omega_{\text{poly}} \circ \pi')) \circ (\Delta k)^n)^* \\
&= \Lambda(\text{curry}(\lbrack \eta \rbrack ; \Gamma, y \vdash \bar{P}_\chi(f) y)[\bar{E}] \circ (\text{id}, (\omega_{\text{poly}} \circ \pi')) \circ (\Delta k)^n)^* \\
\end{align*}

Using an observation about function application (lemma 5.51), weakening (proposition 5.47) and basic properties of Cartesian closed categories, the string of equalities can be extended further.

\begin{align*}
&= \Lambda(\text{uncurry}(\lbrack \eta \rbrack ; \Gamma \vdash \bar{P}_\chi(f)[\bar{E}] \circ (\text{id}, \omega_{\text{poly}} \circ \pi')) \circ (\Delta k)^n)^* \\
&= \Lambda(\text{uncurry}(\lbrack \eta \rbrack ; f \vdash \bar{P}_\chi(f)[\bar{E}] \circ \pi) \circ (\text{id}, \omega_{\text{poly}} \circ \pi')) \circ (\Delta k)^n)^* \\
&= \Lambda(\text{uncurry}(\lbrack \eta \rbrack ; f \vdash \bar{P}_\chi(f)[\bar{E}] \circ (\pi \times \text{id}) \circ (\text{id}, \omega_{\text{poly}} \circ \pi')) \circ (\Delta k)^n)^* \\
&= \Lambda(\text{uncurry}(\lbrack \eta \rbrack ; f \vdash \bar{P}_\chi(f)[\bar{E}] \circ (\pi \times \text{id}) \circ \omega_{\text{poly}} \circ (\pi', \pi')) \circ (\Delta k)^n)^* \\
\end{align*}

Recalling that $\text{uncurry}(g) \circ (\text{id} \times f) = \text{uncurry}((f \Rightarrow \text{id}) \circ g)$ for any $f$ and $g$, we can make use of the fact that curry and uncurry are inverses before expanding out $m^*$ as $\text{uncurry}(m) \circ \lambda^{-1}$.

\begin{align*}
&= \Lambda(\text{curry}(\text{uncurry}(\omega_{\text{poly}} \Rightarrow \text{id}) \circ [\eta] ; f \vdash \bar{P}_\chi(f)[\bar{E}] \circ (\Delta k)^n)^* \\
&= \Lambda((\omega_{\text{poly}} \Rightarrow \text{id}) \circ [\eta] ; f \vdash \bar{P}_\chi(f)[\bar{E}] \circ (\Delta k)^n)^* \\
&= \Lambda(\text{uncurry}(\omega_{\text{poly}} \Rightarrow \text{id}) \circ [\eta] ; f \vdash \bar{P}_\chi(f)[\bar{E}] \circ (\Delta k)^n \circ \lambda^{-1}) \\
\end{align*}

155
We can use naturality properties of uncurry and \( \lambda \) to move \( \omega_V \) outside of \((-)^*\), and rewrite uncurry\((m) \circ \lambda^1\) as the more compact \( m^* \).

\[
\begin{align*}
\Lambda( & \text{uncurry}([\eta]; f \vdash \bar{P}_X(f)]_{\bar{E}} \circ (\Delta k)^\circ \circ (\text{id} \times \omega_V) \circ \lambda^1) \\
& = \Lambda(\text{uncurry}([\eta]; f \vdash \bar{P}_X(f)]_{\bar{E}} \circ (\Delta k)^\circ \circ \lambda^1 \circ \omega_V) \\
& = \Lambda([(\eta]; f \vdash \bar{P}_X(f)]_{\bar{E}} \circ (\Delta k)^\circ)^* \circ \omega_V)
\end{align*}
\]

Recalling how \( \Lambda(-) \) is defined in terms of the PG-functor Lim (in the proof of Theorem 3.5), we use the functorial nature of Lim and the parametric adjunction \( \Delta \vdash \text{Lim} \) to conclude the string of equalities as follows. (Here we use a subscript in \( \text{Lim}_V \) to emphasize that we are discussing parametric transformations and PG-functors in the variable \( V \). To avoid potential confusion between the counit of the adjunction and the typing context \( \eta \), we shall use \( \nu \) to denote the co-unit of the adjunction. Recall that \( \omega \) is given by the unit of the adjunction. We use \( M \) to denote the vertex \( \text{Lim}_V[\eta^{'}, X, \bar{Y} \vdash \bar{P}] (A_1, \cdots A_n, B, C, B, V) \).

\[
\begin{align*}
\text{Lim}_V([(\eta]; f \vdash \bar{P}_X(f)]_{\bar{D}} \circ k^\circ)^* & = \text{Lim}_V([(\eta]; f \vdash \bar{P}_X(f)]_{\bar{E}} \circ (\Delta k)^\circ)^*
\end{align*}
\]

Summing up, when \( P = \forall \bar{Y}.\bar{P} \), the preceding string of equations have established the following equality for any vertices \( D, D' \) and any morphism \( k: D \rightarrow D' \) using the abbreviations \( \bar{D} = (A_1, \cdots A_n, D, D') \) and \( \bar{E} = (A_1, \cdots A_n, D, D', V) \).

The above equality, which we call equation (a), is used to show each of the three points of the lemma. The first two points use the induction hypothesis and the functoriality of \( \text{Lim}_V \). First, we set \( k = \text{id}: B \rightarrow B \).

\[
\begin{align*}
([\eta]; f \vdash \bar{P}_X(f)]_{(A_1, \cdots A_n, B, B)} \circ \text{id}^\circ)^* & = \text{Lim}_V([\eta]; f \vdash \bar{P}_X(f)]_{(A_1, \cdots A_n, B, B, V)} \circ (\Delta \text{id})^\circ)^* \\
& = \text{Lim}_V([\eta]; f \vdash \bar{P}_X(f)]_{(A_1, \cdots A_n, B, B, V)} \circ \text{id}^\circ)^* \\
& = \text{Lim}_V(\text{id}) = \text{id}
\end{align*}
\]

For the second point, the above equation (a) is used for the morphism \( k = (h \circ g) \). Notationally, we shall use \( \bar{E}_1 = (A_1, \cdots A_n, B, D, V), \) \( \bar{E}_2 = (A_1, \cdots A_n, B, C, V) \) and \( \bar{E}_3 = (A_1, \cdots A_n, C, D, V) \). The induc-
tion hypothesis and the fact that $\text{Lim}_V$ is a functor are used before appealing to equation (a) twice more, using $k = g$ and $k = h$.

$$(\eta' ; f \vdash P_y \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ (h \circ g)^o)^*$$

$$= \text{Lim}_V((\eta' ; f \vdash P_y \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ \Delta(h)^o)^* \circ$$

$$(\eta' ; f \vdash P_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ \Delta(g)^o)^*)$$

$$= \text{Lim}_V((\eta' ; f \vdash P_y \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ \Delta(h)^o)^* \circ$$

$$\text{Lim}_V((\eta' ; f \vdash P_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ \Delta(g)^o)^*)$$

$$= (\eta' ; f \vdash P_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ \Delta(g)^o)^*$$

Proving the third equality makes use of the definition of the interpretation and the fact that $\text{Lim}_V$ is fibred before appealing to the induction hypothesis. Notationally, the $n+2$-tuples $\tilde{D} = (A_1,\cdots,A_n,B,C)$ and $\tilde{D}' = (A_1',\cdots,A_n',B',C')$ are used. The subscripts that appear after appealing to equation (a) are the $n+3$-tuples $\tilde{E} = (A_1,\cdots,A_n,B,C,V)$ and $\tilde{E}' = (A_1',\cdots,A_n',B',C',V)$. In addition to using the $n+1$-tuple $\tilde{T} = (R_1,\cdots,R_n,S)$ from the statement of the lemma, we also use the $n+2$-tuple $\tilde{U} = (R_1,\cdots,R_n,S,V)$. We use $\tau$ to denote the parametric transformation $[[\eta ; f \vdash \tilde{P}_x \,(f)\,]]$.

$$([[\eta ; f \vdash P_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ g'^o)^*],[[\eta ; f \vdash P_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ (g'^o)^*)[[\eta,X \vdash P]\tilde{T}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g')^o)^*)[[\eta,X \vdash P]\tilde{T}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g)^o)^*)[[\eta,X,Y \vdash P]\tilde{U}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g')^o)^*)[[\eta,X,Y \vdash P]\tilde{U}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g')^o)^*)[[\eta,X,Y \vdash P]\tilde{U}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g')^o)^*)[[\eta,X,Y \vdash P]\tilde{U}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g')^o)^*)[[\eta,X,Y \vdash P]\tilde{U}$$

$$= \text{Lim}_V(\tau_{\tilde{E}} \circ \Delta(g)^o)^*,\text{Lim}_V(\tau_{\tilde{E}'} \circ \Delta(g')^o)^*)[[\eta,X,Y \vdash P]\tilde{U}$$

When $P = Q \Rightarrow \tilde{P}$: As in the previous case, we begin by considering the morphism $[[\eta ; f \vdash P_y \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ k^o)^* where k is an arbitrary morphism $D \rightarrow D'$ and $\tilde{D} = (A_1,\cdots,A_n,B,C')$. We reason using an arbitrary point $r:1 \rightarrow [\eta' \vdash P_x \,(Y)\,][\tilde{D}$. Expanding the definition of $P_x \,(f)$ and unravelling the definition of the interpretation we get the following equalities.

$$(\eta' ; f \vdash \tilde{P}_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}} \circ k^o)^* \circ r$$

$$= ([\eta' ; f \vdash \lambda x.X. \tilde{P}_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}}(x,y)\,_{\{A_1,\cdots,A_n,B,C\}} \circ k^o)^* \circ r$$

$$= (\text{curry}[[\eta' ; f,x \vdash \lambda y.\tilde{P}_x \,(f)\,_{\{A_1,\cdots,A_n,B,C\}}(x,y)\,_{\{A_1,\cdots,A_n,B,C\}} \circ k^o)^* \circ r$$

The definitions of $M \circ N$ and $m^*$ along with the naturality of curry
are used to extend the string of equalities.

\[
\begin{align*}
&= (\text{curry}[\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ k^o) \circ r \\
&= (\text{curry}((\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id})) \circ r \\
&= \text{uncurry}(\text{curry}((\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id})) \circ \lambda^1 \circ r)
\end{align*}
\]

We make use of the fact that currying and uncurrying are inverses, both for morphisms of several arguments and for morphisms of one argument. The naturality of \(\lambda^1\) is also needed.

\[
\begin{align*}
&= \eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id}) \circ \lambda^1 \circ r \\
&= \left(\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id}) \circ \lambda^1 \circ r\right)^* \\
&= \left(\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id}) \circ \lambda^1 \circ r\right)^*
\end{align*}
\]

The way composition (lemma 5.52) and weakening (proposition 5.47) are reflected by the interpretation can be used along with usual properties of Cartesian categories to extend the string of equalities.

\[
\begin{align*}
&= \left(\left(\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id}) \circ \lambda^1 \circ r \right)^* \circ \\
&= \left(\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id}) \circ \lambda^1 \circ r \right)^* \\
&= \left(\eta'; f, x \vdash \hat{P}_x (f) \circ x]_\mathcal{D} \circ (k^o \times \text{id}) \circ \lambda^1 \circ r \right)^*
\end{align*}
\]

Note that the interpretation \([\eta'; x \vdash x]_\mathcal{D}\) is an identity, as is the discard morphism \(!: 1 \to 1\). The definitions of \(n^o\) and \(m^*\) are used to extend the string of equalities further.

\[
\begin{align*}
&= \left(\left(\eta'; f \vdash \hat{P}_x (f) \circ k^o \circ \pi^* \circ \lambda \right)^* \circ \\
&= \text{curry}((\eta'; f \vdash \hat{P}_x (f) \circ k^o \circ \pi^* \circ \lambda) \\
&= \text{curry}((\eta'; f \vdash \hat{P}_x (f) \circ k^o \circ \text{uncurry}(r) \circ \lambda^1 \circ \lambda)
\end{align*}
\]

Using the naturality of uncurry and the fact that composition of inverses is the identity, the string of equalities concludes as follows.

\[
\begin{align*}
&= \text{curry}(\text{uncurry}(id \Rightarrow (\eta'; f \vdash \hat{P}_x (f) \circ k^o \circ r) \circ \lambda^1 \circ \lambda) \\
&= \text{curry}(\text{uncurry}(id \Rightarrow (\eta'; f \vdash \hat{P}_x (f) \circ k^o \circ r)) \\
&= (id \Rightarrow (\eta'; f \vdash \hat{P}_x (f) \circ k^o \circ r) \circ r
\end{align*}
\]

Since the above equalities hold for all points \(r\), the well-pointedness of \(\mathcal{G}\) ensures that the following equality of morphisms, which we call
equation (b).

\[
([\eta' \ ; \ f \vdash P_X(f)]_B \circ k^\circ)^* = \text{id} \Rightarrow ([\eta' \ ; \ f \vdash P_X(f)]_B \circ k^\circ)^*
\]

This equality is used to prove the three points of the lemma. The first point uses equation (b) and the induction hypothesis.

\[
([\eta' \ ; \ f \vdash P_X(f)]_{(A_1, \ldots, A_n, B, C)} \circ \text{id}^\circ)^* = \text{id} \Rightarrow ([\eta' \ ; \ f \vdash P_X(f)]_{(A_1, \ldots, A_n, B, C)} \circ \text{id}^\circ)^*
\]

The second point additionally uses the fact that \( \Rightarrow \) is functorial in it’s second argument. Here we shall use \( \hat{E}_1 = (A_1, \ldots, A_n, B, C) \) and \( \hat{E}_2 = (A_1, \ldots, A_n, C, C') \).

\[
([\eta' \ ; \ f \vdash P_X(f)]_{(A_1, \ldots, A_n, B, C)} \circ (h \circ g)^\circ)^* = \text{id} \Rightarrow ([\eta' \ ; \ f \vdash P_X(f)]_{(A_1, \ldots, A_n, B, C')} \circ (h \circ g)^\circ)^*
\]

Proving the third point makes use of the equation (b) and the fact that \( F(V) = ([\eta, X \vdash Q][T] \Rightarrow V \) is a fibred functor (defined in the apparent manner). We shall use the \( n+2 \)-tuples \( \tilde{D} = (A_1, \ldots, A_n, B, C) \) and \( \tilde{D}' = (A'_1, \ldots, A'_n, B', C') \). For brevity, we use \( \tau \) to denote the transformation \( [[\eta' \ ; \ f \vdash P_X(f)]]_B \).

\[
[(\tau_{\tilde{D}} \circ g)^\circ, (\tau_{\tilde{D}'} \circ g'^\circ)^\circ][\eta, X \vdash P][\tilde{T}]
\]

The induction hypothesis and the way weakening is reflected in the model (proposition 5.47) can be used to continue the above string of equalities to the desired conclusion.

\[
= [[\eta, X \vdash Q][\tilde{T}] \Rightarrow [\eta, X \vdash \tilde{P}](R_1, \cdots R_n, [g, g'] S)
\]

= [[\eta \vdash Q][\tilde{R}] \Rightarrow [\eta, X \vdash \tilde{P}](R_1, \cdots R_n, [g, g'] S)
\]

= [[\eta, X \vdash Q](R_1, \cdots R_n, [g, g'] S) \Rightarrow [\eta, X \vdash \tilde{P}](R_1, \cdots R_n, [g, g'] S)
\]

= [[\eta, X \vdash P](R_1, \cdots R_n, [g, g'] S)
\]

159
Having shown all three points for both inductive cases (and asserting
them for the variable cases), we conclude that they hold for all plain types

\[ P \].

The above lemma allows one to show that \( \llangle P \rrangle \) is a fibred functor.

**Theorem 5.54**

Suppose \( \eta, X \vdash \text{Plain}_X(P) \) and \( n = |\eta| \).

- For any non-variant functor \( F \colon G^n \to G \), \( \llangle P \rrangle \text{id}_F = \text{id}_{\llangle P \rrangle F} \).

- For any parametric transformations \( \tau \colon F \to G \) and \( \sigma \colon G \to H \),
  \( \llangle P \rrangle (\sigma \circ \tau) = \llangle P \rrangle \sigma \circ \llangle P \rrangle \tau \).

- For any \( \tau \colon F \to G \), \( \tau' \colon F' \to G' \) and \( \mathcal{G} \colon G \leftrightarrow G' \),
  \( \llangle P \rrangle (\llangle \tau, \tau' \rrangle \mathcal{G}) = \llangle \llangle P \rrangle \tau, \llangle P \rrangle \tau' \rrangle \mathcal{G} \rrangle \).

Therefore, \( \llangle P \rrangle \) is a PG-functor \( G^{[n]} \to G^{[n]} \).

**Proof.** As in the previous lemma, we use the abbreviation \( \eta' \) to denote
\( \eta, Y, Z \). For the first two points, we consider any \( n \)-tuple \( \vec{A} \) of vertices of
\( G^n \). In addition to the previous lemma, the first point relies on the definition
of \( \llangle P \rrangle \) on morphisms (see page 153) and the definition of last (definition
5.36). For any \( F \colon G^n \to G \) (that is, any vertex of \( G^{[n]} \)), the following equalities hold.

\[
(\llangle P \rrangle \text{id}_F)_{\vec{A}} = (\llangle \eta, Y, Z \rrangle ; F \vdash P_\lambda(F) \circ \text{id}_\lambda)_{\vec{A}}
\]

\[
= (\llangle \eta, Y, Z \rrangle ; F \vdash P_\lambda(f)_{(A_1, \ldots, A_n, F(A), f(A))} \circ \text{id}_\lambda)_{\vec{A}}
\]

\[
= \text{id}_{\llangle P \rrangle F} \llangle \vec{A} \rrangle
\]

The second point additional relies on the fact that composition in the
functor graph \( G^{[n]} \) is defined component-wise. When applying the previous
lemma, we use the \( n+2 \)-tuples \( \Theta \) and \( \Theta' \) as indicated.

\[
\Theta = (A_1, \ldots, A_n, F(A), G(\vec{A}))
\]

\[
\Theta' = (A_1, \ldots, A_n, G(A), H(\vec{A}))
\]

\[
(\llangle P \rrangle (\sigma \circ \tau))_{\vec{A}}
\]

\[
= (\llangle \eta, Y \rrangle ; f \vdash P_\lambda(f) \circ (\sigma \circ \tau))_{\vec{A}}
\]

\[
= (\llangle \eta, Y \rrangle ; f \vdash P_\lambda(f)_{(A_1, \ldots, A_n, F(A), H(A))} \circ (\sigma_{\vec{A}} \circ \tau_{\vec{A}}))_{\vec{A}}
\]

\[
= (\llangle \eta, Y \rrangle ; f \vdash P_\lambda(f)_{\Theta' \circ \sigma_{\vec{A}} \circ (\eta' ; f \vdash P_\lambda(f)_{\Theta} \circ \tau_{\vec{A}})}_{\vec{A}}
\]

\[
= (\llangle \eta, Y \rrangle ; f \vdash P_\lambda(f) \circ \sigma_{\vec{A}} \circ (\eta' ; f \vdash P_\lambda(f) \circ \tau_{\vec{A}}))_{\vec{A}}
\]

\[
= (\llangle P \rrangle \sigma)_{\vec{A}} \circ (\llangle P \rrangle \tau)_{\vec{A}}
\]

\[
= (\llangle P \rrangle \sigma \circ (\llangle P \rrangle \tau))_{\vec{A}}
\]

160
To show that the edges $\langle P \rangle[\tau, \tau'] G$ and $\langle \langle P \rangle \tau, \langle P \rangle \tau' \rangle G$ are equal, we consider any $n$-tuple $\vec{R} : \vec{A} \leftrightarrow \vec{A}'$ of edges of $G$.

The definition of $\langle P \rangle$ on edges (see page 152) and the definition of last (definition 5.36) give a more explicit description of $\left( \langle P \rangle \tau, \tau' \rangle G \right)(E, \vec{R})$. Corollary 5.43 implies that the edge (of $G^{[n]}$) interpreting the judgment $\eta, X \vdash_0 P$ is an identity edge. We also recall that re-indexing in the functor graph is defined pointwise.

$$\left( \langle P \rangle \tau, \tau' \rangle G \right)(E, \vec{R})$$

$$= \left( \text{last}(\tau, \tau') G^\eta, X \vdash_0 P \right)_{\vec{A}, \vec{A}'}(E, \vec{R})$$

$$= \langle \eta, X \vdash_0 P \rangle_{\vec{A}, \vec{A}'}(E, \langle R_1, \cdots, R_n, ([\tau, \tau'] G)(E, \vec{R}) \rangle)$$

$$= \langle R_1, \cdots, R_n, ([\tau, \tau'] G)(E, \vec{R}) \rangle$$

$$= \langle [\eta, X \vdash P][R_1, \cdots, R_n, ([\tau, \tau'] G)(E, \vec{R})] \rangle$$

The previous lemma is applied, using the specific values as indicated here.

$$S = G(E, \vec{R})$$

$$B = F(\vec{A})$$

$$B' = F'(\vec{A}')$$

$$C = G(\vec{A})$$

$$C' = G'(\vec{A}')$$

For brevity, we also use an $n+1$ tuple $\Phi$ and $n+2$-tuples $\Theta$ and $\Theta'$ as follows.

$$\Phi = (R_1, \cdots, R_n, G(E, \vec{R}))$$

$$\Theta = (A_1, \cdots, A_n, F(\vec{A}), G(\vec{A}))$$

$$\Theta' = (A'_1, \cdots, A'_n, F'(\vec{A}'), G'(\vec{A}'))$$

The desired equality can then be shown using the definition of $\langle P \rangle$ and of re-indexing in the functor graph (see section 4.4).

$$\left( \langle P \rangle \tau, \tau' \rangle G \right)(E, \vec{R})$$

$$= \langle \langle [\eta, X \vdash P][R_1, \cdots, R_n, ([\tau, \tau'] G)(E, \vec{R})] \rangle \tau, \tau' \rangle G(\vec{A}, \vec{A'})$$

$$= \langle \langle [\eta, Y, Z ; f \vdash P_X(f)] \Theta \tau, \tau' \rangle G(\vec{A}, \vec{A'}) \rangle$$

$$= \langle \langle [\langle P \rangle \tau, \langle P \rangle \tau' \rangle G(\vec{A}, \vec{A'}) \rangle, ([\eta, X \vdash P] \Phi) \rangle$$

$$= \langle \langle P \rangle \tau, \langle P \rangle \tau' \rangle G(\vec{A}, \vec{A'}) \rangle$$

The definition of $\langle P \rangle$ on vertices (from page 152) is reminiscent of how substitution is reflected in the model (lemma 5.49). Since $P_X(A)$ was defined by substitution in System P, one can see that it corresponds directly to
functor application.

\[
[\eta \vdash P_x(A)] = \text{last}([\eta \vdash A])[\eta, X \vdash P] = \langle P \rangle[\eta \vdash A]
\]

The correspondence between \( P_x(M) \) and the action of \( \langle P \rangle \) on morphisms is not as close as the above, primarily because the interpretation of a term judgment \( \eta; \Gamma \vdash M: A \Rightarrow B \) is not a morphism from \( [\eta \vdash A] \) to \( [\eta \vdash B] \). However, by considering points into the interpretation of \( \eta \vdash \Gamma \), one can produce a corresponding morphism \( [\eta \vdash A] \to [\eta \vdash B] \). To formally state the relationship, we allow for additional type variables that have been fixed.

**Lemma 5.55**

Suppose that \( \vec{F} \) is an \( |\eta'| \)-tuple of non-variant functors \( G^n \to G \) and that \( m:1 \to \text{last}(\vec{F})[\eta, \eta' \vdash \Gamma] \) is any point. If \( \eta, X, \eta' \vdash \text{Plain}_X(P) \) and the judgment \( \eta, \eta' ; \Gamma \vdash M: A \Rightarrow B \) is derivable, then the following equality between morphisms \( \langle P \rangle/(\text{last}(\vec{F})[\eta, \eta' \vdash A]) \to \langle P \rangle/(\text{last}(\vec{F})[\eta, \eta' \vdash B]) \) holds.

\[
(\text{last}(\vec{F})[\eta, \eta' ; \Gamma \vdash P_x(M); P_x(A) \Rightarrow P_x(B)] \circ m)^* = \langle P \rangle/(\text{last}(\vec{F})[\eta, \eta' ; \Gamma \vdash M: A \Rightarrow B]) \circ m)^*
\]

**Proof.** We shall suppress explicit mention of the type of term variables and terms in the following analysis. Fix the following abbreviations.

\[
\vec{A} = \text{last}(\vec{F})[\eta, \eta' \vdash A] \\
\vec{B} = \text{last}(\vec{F})[\eta, \eta' \vdash B] \\
\vec{M} = \text{last}(\vec{F})[\eta, \eta' ; \Gamma \vdash M]
\]

For any \( |\eta| \)-tuple \( \vec{C} \) of vertices, we use the definitions of \( \langle P \rangle \) (see page 152) and of last (definition 5.36) to get the following string of equalities.

\[
\langle (\vec{P})(\vec{M}) \circ m)^* \rangle_{\vec{C}} = (\vec{M} \circ m)^*_{\vec{C}} = (\text{last}(A, B)[\eta, X, Y; f \vdash P_x(f)])_{\vec{C}} \circ (\vec{M} \circ m)^*_{\vec{C}}
\]

The way weakening and permutations are reflected in the model (proposition 5.47) allows us to reorganize the fixed arguments as follows.

\[
= (\text{last}(A, B)[\eta, X, Y, \eta' ; f \vdash P_x(f)])_{\vec{C}} \circ (\vec{M} \circ m)^*_{\vec{C}}
\]

162
Since composition is pointwise in the functor category $\mathbf{G}^{\mathbf{S}^n}$, we can use standard properties of Cartesian closure and the way composition is reflected in the model to conclude the string of equalities.

\[
\begin{align*}
= (\text{last}(\bar{F})[\eta, \eta'; \Gamma, f \vdash P_X(f)])\bar{\tilde{\gamma}} \circ \bar{\tilde{\gamma}} \circ m\bar{\tilde{\gamma}} \\
= (\text{last}(\bar{F})[\eta, \eta'; \Gamma, f \vdash P_X(f)])\bar{\tilde{\gamma}} \circ \pi_{n+1} \circ (\text{id}, \bar{\tilde{\gamma}}) \circ m\bar{\tilde{\gamma}} \\
= (\text{last}(\bar{F})[\eta, \eta'; \Gamma, f \vdash P_X(M)])\bar{\tilde{\gamma}} \circ (\text{id}, \bar{\tilde{\gamma}}) \circ m\bar{\tilde{\gamma}} \\
= (\text{last}(\bar{F})[\eta, \eta'; \Gamma, f \vdash P_X(M)])\bar{\tilde{\gamma}} \circ m\bar{\tilde{\gamma}} \\
\end{align*}
\]

Since the above equality holds for all $|\eta|$-tuples of vertices, the parametric transformations are equal, as desired.

One can talk of algebras for the functor $\langle P \rangle$ in the usual categorical sense [Mac71] as a vertex $F$ and a morphism $\tau: \langle P \rangle F \to F$. The initial $P$-algebra $\mu P$ in System P can be used to define a $\langle P \rangle$-algebra in $\left(\mathbf{G}^{\mathbf{S}^n}\right)_v$ as follows.

\[
m u P = [\eta \vdash \mu P] = [\eta \vdash \forall X.(P \Rightarrow X) \Rightarrow X] \\
\tilde{\eta} = [\eta; \emptyset \vdash \text{in}: P_X(\mu P) \Rightarrow \mu P]^*
\]

The pair $(muP, \tilde{m})$ is an algebra for $\langle P \rangle$ (that is, $\tilde{m}$ is a morphism from $\langle P \rangle(muP)$ to $muP$) since lemma 5.49 ensures the following equality.

\[
[\eta \vdash P_X(\mu P)] = \text{last}([\eta \vdash \mu P])[\eta, X \vdash P] \\
= \text{last}(muP)[\eta, X \vdash P] \\
= \langle P \rangle muP
\]

Moreover, we will show that $(muP, \tilde{m})$ is initial in the following (usual categorical) sense.

**Definition 5.56**

For any functor $G: \mathbf{C} \to \mathbf{C}$, a weakly initial $G$-algebra is an $G$-algebra $(A, h)$ such that for any $G$-algebra $(B, k)$, there exists a morphism $m^k: A \to B$ such that $k \circ G(m^k) = m^k \circ h$.

An initial $G$-algebra is a weakly initial $G$-algebra such that for every $G$-algebra $(B, k)$, there is a unique morphism $m^k$ as above.

Since similar results have been proven in System P, it is not difficult to show that $(muP, \tilde{m})$ is a weakly-initial $\langle P \rangle$-algebra. For any vertex $F$ and morphism $\tau: \langle P \rangle F \to F$ in $\mathbf{G}^{\mathbf{S}^n}$, there is a morphism $m^\tau: muP \to F$ given.

163
as follows.

\[ m^* = (\text{last}(F)[\eta, Y; P_X(Y) \Rightarrow Y \vdash \text{fold}[Y]h; \mu P \Rightarrow Y] \circ \tau^o)^* \]

**Lemma 5.57**
The \( \langle P \rangle \)-algebra \( (\mu P, \tilde{m}) \) is weakly initial, since for any \( \langle P \rangle \)-algebra \( (F, \tau) \), we have the following equality.

\[ \tau \circ G(m^*) = m^* \circ \tilde{m} \]

**Proof.** Lemma 5.26 states that the following judgment is derivable.

\[ \eta, Y; h; P_X(Y) \Rightarrow Y \vdash h \circ P_X(\text{fold}[Y]h) = (\text{fold}[Y]h) \circ \text{in} \]

Therefore corollary 5.45 implies that the following morphisms of \( G^{[\mathbb{I}^n]} \) are equal.

\[
\begin{align*}
[\eta, Y; h; P_X(Y) \Rightarrow Y \vdash h \circ P_X(\text{fold}[Y]h)]: P_X(\mu P) & \Rightarrow Y \\
[\eta, Y; h; P_X(Y) \Rightarrow Y \vdash (\text{fold}[Y]h) \circ \text{in}]: P_X(\mu P) & \Rightarrow Y
\end{align*}
\]

For brevity, we shall suppress the types of terms and term variables.

We first note that \( \text{last}(F)[\eta, Y \vdash P_X(Y) \Rightarrow Y] = \langle P \rangle F \Rightarrow F \). This can be shown using the definitions of the interpretation and \( \langle P \rangle \).

\[
\begin{align*}
\text{last}(F)[\eta, Y \vdash P_X(Y) \Rightarrow Y] & = \text{last}(F)[(\eta, Y \vdash P_X(Y) \Rightarrow Y, \eta, Y \vdash Y)] \\
& = \langle P \rangle F \Rightarrow \text{last}(F)[\eta, Y \vdash Y] \\
& = \langle P \rangle F \Rightarrow \text{last}(F)\Pi_{n+1} \\
& = \langle P \rangle F \Rightarrow F
\end{align*}
\]

Therefore, by computing the last argument using \( F \) and composing with \( \tau^o : 1 \rightarrow \langle P \rangle F \Rightarrow F \), we have the following equality.

\[
\begin{align*}
\text{last}(F)[\eta, Y; h \vdash h \circ P_X(\text{fold}[Y]h) \circ \tau^o] & \at \equiv \text{last}(F)[\eta, Y; h \vdash (\text{fold}[Y]h) \circ \text{in} \circ \tau^o]
\end{align*}
\]

We shall show that by uncurrying these morphisms \( 1 \rightarrow \langle P \rangle muP \Rightarrow F \) to the corresponding morphisms \( \langle P \rangle muP \rightarrow F \), we get the equation asserted in the lemma.

Let us consider the left hand side of equation \( @ \) first. We use lemma 5.49 to get rid of the composition in System P and lemma 5.55 which associates the meta-function \( P_X \) on the syntax of terms with the PG-functor \( \langle P \rangle \). The definitions of \( m^* \) and the interpretation \( [\eta, Y; h \vdash h] \) complete this string.

164
of equalities.

\[
\begin{align*}
(last(F)[\eta, Y ; h \vdash h \circ P_x (\text{fold}[Y]h)] \circ \tau^o)^* \\
= & \ (last(F)[\eta, Y ; H \vdash h] \circ \tau^o)^* \circ \\
& (last(F)[\eta, Y ; H \vdash h] \circ \tau^o)^* \circ \\
& \langle P \rangle (last(F)[\eta, Y ; h \vdash \text{fold}[Y]h] \circ \tau^o)^* \\
= & \ (last(F)[\eta, Y ; H \vdash h] \circ \tau^o)^* \circ \langle P \rangle (m^r) \\
= & \ (last(F)\text{id} \circ \tau^o)^* \circ \langle P \rangle (m^r) \\
= & \ \tau^o \circ \langle P \rangle (m^r) \\
= & \ \tau \circ \langle P \rangle (m^r)
\end{align*}
\]

For the right hand side, we use properties of composition (lemma 5.52) and weakening (proposition 5.47), as well as the definitions of $m^r$ and $\hat{\text{in}}$.

\[
\begin{align*}
(last(F)[\eta, Y ; h \vdash (\text{fold}[Y]h) \circ \text{in}] \circ \tau^o)^* \\
= & \ (last(F)[\eta, Y ; h \vdash \text{fold}[Y]h] \circ \tau^o)^* \circ (last(F)[\eta, Y ; h \vdash \text{in}] \circ \tau^o)^* \\
= & \ m^r \circ (last(F)[\eta, Y ; h \vdash \text{in}] \circ \tau^o)^* \\
= & \ m^r \circ (\text{id} \circ \tau^o)^* \\
= & \ m^r \circ (\text{id})^* \\
= & \ m^r \circ \hat{\text{in}}
\end{align*}
\]

\[\square\]

In order to show that $\mu \text{P}$ is the initial $\langle P \rangle$-algebra, we first observe that the term $\text{fold}[\mu \text{P}]\text{in}$ gives an encoding of the identity morphism in the following sense.

**Lemma 5.58**

Suppose $\eta, X \vdash \text{Plain}_X (P)$. The interpretation of the term judgment $\eta; \emptyset \vdash \text{fold}[\mu \text{P}]\text{in}; \mu \text{P} \Rightarrow \mu \text{P}$ is equal to $(\text{id}_{\mu \text{P}})^*: 1 \rightarrow \mu \text{P} \Rightarrow \mu \text{P}$.

**Proof.** Lemma 5.28 asserted that the following judgment is derivable.

\[\eta; \emptyset \vdash \lambda z: \mu \text{P} \cdot [\mu \text{P}] \text{in} = z\]

Therefore $\{\text{fun}J\}$ can be used to derive the following.

\[\eta; \emptyset \vdash \lambda z: \mu \text{P} \cdot \lambda z: \mu \text{P}z = \lambda z: \mu \text{P}z\]

165
Recalling that \textit{fold} is used to denote the term \(\Lambda X.\lambda f: P \to X.\lambda x: \mu P. x[X] f\), we note that the beta reductions allow one to derive the following judgment.

\[ \eta; \emptyset \vdash \emptyset \quad \text{fold}[\mu P] \text{in} = \lambda x: \mu P. x[\mu P] \text{in} \]

Since \{alpha fun\} allows one to get around any difference from having chosen different variable names, we can conclude the following judgment is derivable by using \{eq trans\}.

\[ \eta; \emptyset \vdash \emptyset \quad \text{fold}[\mu P] \text{in} = \lambda x: \mu P x \]

Corollary 5.45 allows one to deduce the equality of corresponding term judgments. By using definitions, this gives the desired equality.

\[ \begin{align*}
[\eta; \emptyset \vdash \text{fold}[\mu P] \text{in}: \mu P & \Rightarrow \mu P] \\
= & \quad [\eta; \emptyset \vdash \lambda x: \mu P. x[\mu P] \Rightarrow \mu P] \\
= & \quad \text{curry}([\eta; \emptyset; z: \mu P \vdash z: \mu P] \circ i^0_{\mu P}) \\
= & \quad \text{curry}((id_{\mu P} \circ \lambda) \\
= & \quad (id_{\mu P})^o
\end{align*} \]

\[\blacksquare\]

\textbf{Theorem 5.59}

\textit{Suppose \(\eta, X \vdash \text{Plain}_X(P)\) and that \((F, \tau)\) is a \((P)\)-algebra. For any \(\sigma: \mu P \to F\) such that \(\tau \circ \|P]\sigma = \sigma \circ \|\text{m}\), it must be that \(\sigma = m^\sigma\). Consequently, \((\mu P, \|\text{m})\) in the initial \((P)\)-algebra.}

\textbf{Proof.} Since \(G^{[G]^{n+1}}\) is subsumptive, the hypothesis \(\tau \circ \|P]\sigma = \sigma \circ \|\text{m}\) implies there is a square of the shape on the left below.

\[\begin{array}{c}
\|P]\mu P \quad \overset{\|\text{m}}{\longrightarrow} \\
\|P]\langle \sigma \rangle \\
\|P]\frac{F}{\tau} \quad \overset{\|\sigma}{\longrightarrow} \\
\|P]\langle \sigma \rangle
\end{array}\]

\[\begin{array}{c}
1 \\
\|P]\mu P \Rightarrow \mu P \\
\|P]\langle \sigma \rangle \\
\|P]\frac{F}{\tau} \Rightarrow F
\end{array}\]

Therefore, there is a square of the shape on the right above (lemma 5.39). Note that the interpretation (from section 3.3) gives the following parametric transformation.

\[\begin{align*}
[\eta, Y; h \vdash \text{fold}[Y] h]: [\eta, Y \vdash P^\chi(Y) \Rightarrow Y] & \Rightarrow [\eta, Y \vdash \mu P \Rightarrow Y] \\
\end{align*}\]

By using the PG-functor \(\text{last}(\langle \sigma \rangle): E \times G^{[G]^{n+1}} \to G^{[G]^{n+1}},\) we get that there is a square \(\sigma: \text{last}(\langle \sigma \rangle)(E, I_{\|\text{Y}+P^\chi(Y) \Rightarrow Y}) \to \text{last}(\langle \sigma \rangle)(E, I_{\|\text{Y}+\mu P \Rightarrow Y})\) over
the following morphisms.

\[
\begin{align*}
\partial_0(\sigma) &= \text{last}(\mu_P)[\eta; Y \vdash \text{fold}[Y]h]\n\partial_1(\sigma) &= \text{last}(F)[\eta; Y \vdash \text{fold}[Y]h]
\end{align*}
\]

We examine the edge at the source of this square in more detail. Using the definition of \text{last}(\langle \sigma \rangle) and of identity edges in functor graphs, we can be more explicit of that edge’s value for an \(n\)-tuple \(R\) of edges of \(G\). We use the definition of the interpretation and lemma 5.55 to simplify further.

\[
\begin{align*}
\text{last}(\langle \sigma \rangle)(E, I_{[h: Y \vdash P_X(Y) \Rightarrow Y]}(E, R)) &= I_{[\eta: Y \vdash P_X(Y) \Rightarrow Y]}(E, (R_1, \ldots, R_n, \langle \sigma \rangle(E, \bar{R}))) \\
&= \Pi_{n+1} \Pi_{\eta+1}(R_1, \ldots, R_n, \langle \sigma \rangle(E, \bar{R})) \Rightarrow \langle \sigma \rangle(E, \bar{R}) \\
&= \langle \Pi_{\eta+1}(R_1, \ldots, R_n, \langle \sigma \rangle(E, \bar{R})) \rangle \Rightarrow \langle \sigma \rangle(E, \bar{R})
\end{align*}
\]

Since the source parametricity graph is discrete, exponents in \(G^{	ext{K}^n\pi+1}\) are given pointwise. Using the definition of \(\langle P \rangle\) as well, the string of equalities concludes as follows.

\[
\begin{align*}
\text{last}(\langle \sigma \rangle)(E, I_{[h: Y \vdash P_X(Y) \Rightarrow Y]}(E, R)) &= \text{last}(\langle \sigma \rangle)(E, R) \Rightarrow \langle \sigma \rangle(E, \bar{R}) \\
&= \langle \Pi_{\eta+1}(R_1, \ldots, R_n, \langle \sigma \rangle(E, \bar{R})) \rangle \Rightarrow \langle \sigma \rangle(E, \bar{R})
\end{align*}
\]

Since the PG-functors \text{last}(\langle \sigma \rangle)(E, I_{[h: Y \vdash P_X(Y) \Rightarrow Y]}(E, R)) and \langle P \rangle \langle \sigma \rangle \Rightarrow \langle \sigma \rangle agree for all edges, they are equal. Similarly, one can show the following equality for the target edge.

\[
\text{last}(\langle \sigma \rangle)(E, I_{[h: Y \vdash \mu_P \Rightarrow \eta]}(E)) = I_{\mu_P \Rightarrow \langle \sigma \rangle}
\]

We can compose square (1) with \(\sigma\) to get the following square.

\[
\begin{array}{c}
1 \quad \text{last}(\mu_P)[\eta; Y \vdash \text{fold}[Y]h] \circ \tau^\circ \quad \mu_P \Rightarrow \mu_P \\
\downarrow \quad 1 \\
\text{last}(F)[\eta; Y \vdash \text{fold}[Y]h] \circ \tau^\circ \quad \mu_P \Rightarrow F \\
\end{array}
\]

167
The previous lemma (lemma 5.58) asserted that the morphism along the top of this square is equal to \((\text{id})^\circ\). Recalling the definition of \(m^\tau\) (and that \(n = (n^\tau)^\circ\)), the morphism along the bottom is \((m^\tau)^\circ\). In other words, the above square is of the shape on the left below.

\[
\begin{array}{ccc}
1 & \overset{\text{id}}{\longrightarrow} & muP \Rightarrow muP \\
\downarrow{\mu}P & & \downarrow{\mu}P \\
1 & \overset{(m^\tau)^\circ}{\longrightarrow} & muP \Rightarrow F \\
\end{array}
\]

Therefore there is a square of the shape on the right above (lemma 5.39). By way of subsumption, we deduce \(\sigma \circ \text{id} = m^\tau \circ \text{id}\), which means that \(\sigma = m^\tau\) as claimed.

Every PG-functor arising from a plain type has an initial algebra. While the lemmas and theorems about System P can not be used to reason about functors that can not be encoded in System P, the proofs can be used as an outline for proving similar results semantically. We can show that initial algebras exist for a much larger family of functors than just those arising from plain types.

**Definition 5.60**

A PG-functor \(F: \mathbf{H} \to \mathbf{H}\) on a Cartesian parametricity graph \(\mathbf{H}\) is strong if there exists a parametric transformation \(\text{str}^F\) with components of type \(\text{str}^F_{A,B}: F(A) \times B \to F(A \times B)\) such that the following diagrams commute for all vertices \(A, B\) and \(C\).

\[
\begin{array}{ccc}
F(A) \times 1 & \overset{\text{str}^F_{A,1}}{\longrightarrow} & F(A \times 1) \\
\downarrow{\rho} & & \downarrow{F(\rho)} \\
F(A) & \overset{\text{str}^F_{A,B \times C}}{\longrightarrow} & F(A \times B) \times C \\
\end{array}
\]

\[
\begin{array}{ccc}
F(A \times B) \times C & \overset{\text{str}^F_{(A \times B),C}}{\longrightarrow} & F(A \times B \times C) \\
\end{array}
\]

Such a parametric transformation \(\text{str}^F\) is called a strength for \(F\).

This is an immediate analogue of the notion of a strong functor on a Cartesian category (using a parametric transformation rather than a natural transformation for the strength). The superscript \(F\) will generally be suppressed on the strength when the PG-functor it is a strength for is apparent.

When considering Cartesian closed parametricity graphs, the strength of a strong PG functor \(F\) can be used to define a parametric transformation
with components $\text{es}^F_{A,B} : F(A \Rightarrow B) \rightarrow A \Rightarrow F(B)$ as follows.

$$\text{es}^F_{A,B} = \text{curry}(F(ap) \circ \text{str}^F_{A \Rightarrow B,A})$$

This parametric transformation is called the exponential strength of $F$. The superscript will frequently be suppressed. One property about how the exponential strength interacts with the Cartesian closed structure is the following.

**Lemma 5.61**

*Suppose $F$ is a strong PG functor on a Cartesian closed parametricity graph. For any vertices $A$, $B$ and any point $a : 1 \rightarrow A$, the following diagram commutes.*

$$
\begin{array}{ccc}
F(A \Rightarrow B) & \xrightarrow{F(ap \circ (id \times a) \circ \rho^1)} & F(B) \\
\downarrow \text{es}^F_{A,B} & & \downarrow \text{ap} \\
A \Rightarrow F(B) & \xrightarrow{(id \times a) \circ \rho^1} & A \Rightarrow F(B) \times A
\end{array}
$$

**Proof.** The naturality of $\rho^1$ allows the bottom path of the diagram to be rewritten using $\text{uncurry}(\text{es}^F_{A,B})$.

$$ap \circ (id \times a) \circ \rho^1 \circ \text{es}^F_{A,B} = ap \circ (\text{es}^F_{A,B} \times \text{id}) \circ (id \times a) \circ \rho^1 = \text{uncurry}(\text{es}^F_{A,B}) \circ (id \times a) \circ \rho^1$$

Since the exponential strength is defined using curry, we take advantage of $\text{uncurry}$ and $\text{curry}$ being inverses before appealing to the naturality of the strength. Recall that parametric transformations between PG-functors are natural (proposition 4.8).

$$= \text{uncurry}(\text{curry}(F(ap) \circ \text{str}^F_{A,B})) \circ (id \times a) \circ \rho^1$$

$$= F(ap) \circ \text{str}^F_{A,B} \circ (id \times a) \circ \rho^1$$

$$= F(ap) \circ F(id \times a) \circ \rho^1$$

One of the commuting diagrams characterizing the strength implies that the above is the morphism desired.

$$= F(ap) \circ F(id \times a) \circ F(\rho^1)$$

$$= F(ap) \circ (id \times a) \circ \rho^1$$

$\Diamond$
Any strong PG-functor $F: G^{\mathbb{G}^n} \rightarrow G^{\mathbb{G}^n}$ has an initial algebra. We define that $F$-algebra $(\text{init}(F), \text{struct}^F)$ as follows.

$$
\text{init}(F) = \forall Y (F(Y) \Rightarrow Y) \Rightarrow Y \\
\text{struct}^F = \Lambda \left( \text{curry} \left( \text{ap} \circ \pi_2, \text{uncurry} \left( \text{es}_{F(Y) \Rightarrow Y, Y \circ F(\omega_Y)} \right) \right) \right)
$$

We shall frequently suppress the superscript on struct when the PG-functor in question can be inferred.

For any $F$-algebra $(A, \tau)$, the factorization $m^\tau : \text{init}(F) \rightarrow A$ is given as follows.

$$
m^\tau = \text{ap} \circ (\text{id} \times \tau^\circ) \circ \rho^1 \circ \omega_A
$$

In addition to providing a morphism into $A$ for every morphism from $F(A)$ to $A$, the above construction provides a square into $R$ for every square from $F(R)$ to $R$.

**Lemma 5.62**

*Suppose $F$ is a strong PG-functor. If there is a square of the shape on the left below, then there is a square of the shape on the right.*

\[
\begin{array}{c}
F(A) \xrightarrow{\tau} A \\
F(R) \downarrow \quad \downarrow R \\
F(B) \xrightarrow{\eta} B
\end{array}
\quad \quad \quad
\begin{array}{c}
\text{init}(F) \xrightarrow{m^\tau} A \\
\downarrow I \\
\text{init}(F) \xrightarrow{m^\eta} B
\end{array}
\]

**Proof.** Lemma 5.39 ensures that there is a square of the shape on the left below.

\[
\begin{array}{c}
1 \xrightarrow{\tau^\circ} F(A) \Rightarrow A \\
\downarrow I \\
1 \xrightarrow{\eta^\circ} F(B) \Rightarrow B
\end{array}
\quad \quad \quad
\begin{array}{c}
\text{init}(F) \xrightarrow{\omega_A} (F(A) \Rightarrow A) \Rightarrow A \\
\downarrow I \\
\text{init}(F) \xrightarrow{\omega_B} (F(B) \Rightarrow B) \Rightarrow B
\end{array}
\]

There is a square of the shape on the right since $\omega$ is parametric. The following square comes from the parametricity of $\rho^1$.

\[
\begin{array}{c}
(F(A) \Rightarrow A) \Rightarrow A \xrightarrow{\rho^1} (F(A) \Rightarrow A) \Rightarrow A \times 1 \\
(F(R) \Rightarrow R) \Rightarrow R \\
(F(B) \Rightarrow B) \Rightarrow B
\end{array}
\quad \quad \quad
\begin{array}{c}
(F(R) \Rightarrow R) \Rightarrow R \times I \\
(F(B) \Rightarrow B) \Rightarrow B \times 1
\end{array}
\]

170
Similarly, there is a square of the following shape.

\[( (F(A) \Rightarrow A) \Rightarrow A ) \times (F(A) \Rightarrow A) \xrightarrow{\text{ap}} A \]
\[( (F(R) \Rightarrow R) \Rightarrow R ) \times (F(R) \Rightarrow R) \]
\[( (F(B) \Rightarrow B) \Rightarrow B ) \times (F(B) \Rightarrow B) \xrightarrow{\text{ap}} B \]

Composing these squares, we get a square of the following shape.

\[
\begin{array}{c}
\text{init}(F) \xrightarrow{\text{ap} \circ (\text{id} \times \tau^0) \circ \rho^1 \circ \omega_A} A \\
\text{I} \\
\text{init}(F) \xrightarrow{\text{ap} \circ (\text{id} \times \eta^0) \circ \rho^1 \circ \omega_B} B
\end{array}
\]

Recall that \( m^\tau = \text{ap} \circ (\text{id} \times \tau^0) \circ \rho^1 \circ \omega_A \) by definition and similarly that \( m^\eta = \text{ap} \circ (\text{id} \times \eta^0) \circ \rho^1 \circ \omega_A \). Thus the above square is the one claimed. \( \circ \)

In order to show that \((\text{init}(F), \text{struct})\) is the initial \(F\)-algebra, we begin by showing it is weakly initial.

**Lemma 5.63**

For any strong PG functor \( F \), the \(F\)-algebra \((\text{init}(F), \text{struct})\) is weakly initial.

**Proof.** To show that \( m^\tau \circ \text{struct} = \tau \circ F(m^\tau) \) for an arbitrary \(F\)-algebra \((A, \tau)\), we begin by expanding definitions and using the naturality of \( \rho^1 \). This allows us to take advantage of uncurry and curry being inverses.

\[
m^\tau \circ \text{struct} = (\text{ap} \circ (\text{id} \times \tau^0) \circ \rho^1 \circ \omega_A) \circ \]
\[
A \left( \text{curry} \left( \text{ap} \circ \left( \pi_2, \text{uncurry} \left( \text{es}_{F(Y)} \Rightarrow Y \times Y \circ F(\omega_Y) \right) \right) \right) \right)
\]
\[
= \text{ap} \circ (\text{id} \times \tau^0) \circ \rho^1 \circ \text{curry} \left( \text{ap} \circ \left( \pi_2, \text{uncurry} \left( \text{es}_{F(A) \Rightarrow A} \circ F(\omega_A) \right) \right) \right)
\]
\[
= \text{ap} \circ \left( \text{curry} \left( \text{ap} \circ \left( \pi_2, \text{uncurry} \left( \text{es}_{F(A) \Rightarrow A} \circ F(\omega_A) \right) \right) \times \text{id} \right) \right) \circ (\text{id} \times \tau^0) \circ \rho^1
\]
\[
= \text{uncurry} \left( \text{curry} \left( \text{ap} \circ \left( \pi_2, \text{uncurry} \left( \text{es}_{F(A) \Rightarrow A} \circ F(\omega_A) \right) \right) \right) \right) \circ (\text{id} \times \tau^0) \circ \rho^1
\]
\[
= \text{ap} \circ \left( \pi_2, \text{uncurry} \left( \text{es}_{F(A) \Rightarrow A} \circ F(\omega_A) \right) \right) \circ (\text{id} \times \tau^0) \circ \rho^1
\]

This can be rewritten using naturality properties and the fact that both \( \pi_2 \circ \rho^1 \) and \( ! \circ \text{uncurry} \left( \text{es}_{F(A) \Rightarrow A} \circ F(\omega_A) \right) \circ (\text{id} \times \tau^0) \circ \rho^1 \) are the unique
morphism \(!: F(\text{init}(F)) \to 1\).

\[
\begin{align*}
\lambda & = \text{ap} \circ \left( \pi_2 \circ (\text{id} \times \tau^o) \circ \rho^1 \right), \\
& = \text{ap} \circ \left( \tau^f \circ \pi_2 \circ \rho^1, \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1 \right) \\
& = \text{ap} \circ \left( \tau^f \circ ! \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1, \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1 \right) \\
& = \text{ap} \circ \left( \tau^f \circ !, \text{id} \right) \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1 \\
& = \text{ap} \circ (\tau^f \times \text{id}) \circ (\lambda, \text{id}) \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1
\end{align*}
\]

We note that \(\langle !, \text{id} \rangle = \lambda^1 : F(\text{init}(F)) \to 1 \times F(\text{init}(F))\). Recalling how uncurrying and uncurrying a point are defined, we can make use of the fact that \((\tau^f)^* = \tau\).

\[
\begin{align*}
\lambda & = \text{curry}(\tau^f) \circ \lambda^1 \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1 \\
& = (\tau^f)^* \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1 \\
& = \tau \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A \circ F(\omega_A)) \circ (\text{id} \times \tau^o) \circ \rho^1
\end{align*}
\]

Naturality properties permit one to rewrite the above so that lemma 5.61 gives the desired morphism.

\[
\begin{align*}
\lambda & = \tau \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A) \circ (F(\omega_A) \times \text{id}) \circ (\text{id} \times \tau^o) \circ \rho^1 \\
& = \tau \circ \text{curry}(\text{es}_{F(A)} \Rightarrow A, A) \circ (\text{id} \times \tau^o) \circ \rho^1 \circ F(\omega_A) \\
& = \tau \circ F(\text{ap} \circ (\text{id} \times \tau^o) \circ \rho^1 \circ \omega_A) \\
& = \tau \circ F(m^\tau) \\
& \circ
\end{align*}
\]

Before showing that, for any \(F\)-algebra \((A, \tau)\), the morphism \(m^\tau\) is unique, we observe that the morphism this gives from \((\text{init}(F), \text{struct})\) to itself is the identity.

**Lemma 5.64**

The factorization \(m^\text{struct} : \text{init}(F) \to \text{init}(F)\) is the identity.

**Proof.** We reason using an arbitrary vertex \(Z\). We shall show that the morphisms \(\omega_Z \circ m^\text{struct}\) and \(\omega_Z\) are equal. The argument makes use of an arbitrary point \(p : 1 \to F(Z) \Rightarrow Z\).

Note that \((Z, p^*)\) is an \(F\)-algebra. Since \((\text{init}(F), \text{struct})\) is weakly initial (lemma 5.63), the subsumption criteria implies there is a square of the shape

\[172\]
on the left below.

\[
\begin{array}{ccc}
F(\text{init}(F)) \xrightarrow{\text{struct}} & \text{init}(F) & \text{init}(F) \\
\text{init}(F) \xrightarrow{m^\text{struct}} & \text{init}(F)
\end{array}
\]

Therefore lemma 5.62 ensures that there is a square of the shape on the right. Subsumption ensures that \( m^\tau \circ m^\text{struct} = m^\tau \). Expanding the definition of \( m^\tau \), we can use naturality conditions to rewrite this equation.

\[
\begin{align*}
ap \circ (\text{id} \times p) \circ \rho^1 \circ \omega_Z \circ m^\text{struct} &= ap \circ (\text{id} \times p) \circ \rho^1 \circ \omega_Z \\
ap \circ ((\omega_Z \circ m^\text{struct}) \times \text{id}) \circ (\text{id} \times p) \circ \rho^1 &= ap \circ (\omega_Z \times \text{id}) \circ (\text{id} \times p) \circ \rho^1
\end{align*}
\]

Since \( \rho^1 \) is an isomorphism and uncurry(\( f \)) = \( ap \circ (f \times \text{id}) \) for all \( f \), we note that the following equality holds.

\[
\text{uncurry}(\omega_Z \circ m^\text{struct}) \circ (\text{id} \times p) = \text{uncurry}(\omega_Z) \circ (\text{id} \times p)
\]

Since \( p \) was an arbitrary point, we conclude the equality of the morphisms uncurry(\( \omega_Z \circ m^\text{struct} \)) and uncurry(\( \omega_Z \)). Since uncurry is an isomorphism, we see that \( \omega_Z \circ m^\text{struct} = \omega_Z \). As this equality holds for all vertices \( Z \), we conclude that \( m^\text{struct} = \text{id} \) because the definition of parametric limit ensures there is a unique morphism \( f: \text{init}(F) \to \text{init}(F) \) such that \( \omega \circ \Delta(f) = \omega \circ \).

We can now show that \( (\text{init}(F), \text{struct}) \) is the initial \( F \)-algebra.

Theorem 5.65
For any \( n \) and any strong PG-functor \( F: G^n \to G \), the \( F \)-algebra \( (\text{init}(F), \text{struct}) \) is an initial \( F \)-algebra.

Proof. Having already shown that \( (\text{init}(F), \text{struct}) \) is weakly initial (lemma 5.63), it remains to show that for any \( F \)-algebra \( (A, \tau) \), the morphism \( m^\tau \) of \( G^n \) is the unique morphism \( h \) such that \( h \circ \text{struct} = \tau \circ F(h) \).

Suppose \( h: \text{init}(F) \to A \) is any morphism such that \( h \circ \text{struct} = \tau \circ F(h) \). The subsumption property ensures that there is a square of the shape on the left below.

\[
\begin{array}{ccc}
F(\text{init}(F)) \xrightarrow{\text{struct}} & \text{init}(F) & \text{init}(F) \\
\text{init}(F) \xrightarrow{m^\text{struct}} & \text{init}(F)
\end{array}
\]

173
Lemma 5.62 implies that there is a square of the shape on the right. Therefore the subsumption criteria ensures that $h \circ m^{\text{struct}} = m^\tau$. Since $m^{\text{struct}}$ is the identity (lemma 5.64), we conclude that $h = m^\tau$.

In addition to initial algebras, other representation results of System P hold in $G$. For instance, one can construct parametric colimits of non-variant functors. We shall show that the parametric colimit of $F$ is given by the vertex $\forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y$ and the injection parametric transformation $\mu: F \rightarrow \Delta(\forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y)$ as follows.

$$\mu_Z = \Lambda(\text{curry}(\text{uncurry}(\omega_Z) \circ \sigma))$$

In order to compute the factorization of a parametric transformation $\beta: F \rightarrow \Delta B$ (for some vertex $B$), we use the point $\Lambda(\beta^\circ)!: \rightarrow \forall_X F(X) \Rightarrow B$. The factorization $\nabla(\beta); \forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y \rightarrow B$ is as follows.

$$\nabla(\beta) = \text{ap} \circ (\text{id} \times \Lambda(\beta^\circ)) \circ \rho^1 \circ \omega_B$$

The main step in showing that $\nabla(\beta)$ is the factorization of $\beta$ through $\forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y$ is provided by the following lemma.

**Lemma 5.66**

For any vertex $A$ and any point $F: 1 \rightarrow \forall_X F(X) \Rightarrow A$, the following equality holds for all vertices $Y$.

$$\text{ap} \circ (\text{id} \times f) \circ \rho^1 \circ \omega_A \circ \mu_Y = (\omega_Y \circ f)^*$$

**Proof.** By expanding the definition of $\mu$, we make use of $\text{comp}\Lambda(\tau)\omega_A$ being equal to $\tau_A$ from the definition of parametric limit.

$$\text{ap} \circ (\text{id} \times f) \circ \rho^1 \circ \omega_A \circ \mu_Y = \text{ap} \circ (\text{id} \times f) \circ \rho^1 \circ \omega_A \circ \Lambda(\text{curry}(\text{uncurry}(\omega_Y) \circ \sigma)) = \text{ap} \circ (\text{id} \times f) \circ \rho^1 \circ \text{curry}(\text{uncurry}(\omega_Y) \circ \sigma) = \text{ap} \circ (\text{curry}(\text{uncurry}(\omega_Y) \circ \sigma) \times \text{id}) \circ (\text{id} \times f) \circ \rho^1$$

Recall that $\text{ap} \circ (g \times \text{id}) = \text{uncurry}(g)$ for any $g$ and that curry and uncurry are inverses. The naturality of the symmetry transformation $\sigma$ and the way it interacts with the cancellations can be used as follows.

$$= \text{uncurry}(\omega_Y) \circ \sigma \circ (\text{id} \times f) \circ \rho^1$$

$$= \text{uncurry}(\omega_Y) \circ (f \times \text{id}) \circ \sigma \circ \rho^1$$

$$= \text{uncurry}(\omega_Y) \circ (f \times \text{id}) \circ \lambda^1$$

174
The naturality of uncurry and the definition of uncurrying a point give the desired equality.

\[ \text{uncurry}(\omega_Y \circ f) \circ \lambda^1 = (\omega_Y \circ f)^* \]

\[ \text{Corollary 5.67} \]

For any vertex \( B \) and parametric transformation \( \beta: F \to \Delta(B) \), \( \Delta(\nabla(\beta)) \) is a factorization of \( \beta \) in the following sense.

\[ \Delta(\nabla(\beta)) \circ \mu = \beta \]

\[ \text{Proof.} \] For each vertex \( Y \), we calculate using \( f = \Lambda(\beta^\circ) \) in lemma 5.66.

\[ \nabla(\beta) \circ \mu_Y = \text{ap} \circ (\text{id} \times \Lambda(\beta^\circ)) \circ \rho^1 \circ \omega_B \circ \mu_Y \]

\[ = (\omega_Y \circ \Lambda(\beta^\circ))^* \]

\[ = (\beta_Y^\circ)^* = \beta_Y \]

\[ \text{\circ} \]

The factorization of parametric transformations preserves composition with morphisms in the following sense.

\[ \text{Lemma 5.68} \]

If \( \alpha: F \to \Delta A \) is a parametric transformation and \( h: A \to B \) is a morphism of \( G \), then \( h \circ \nabla(\alpha) = \nabla(\Delta h \circ \alpha) \).

\[ \text{Proof.} \] For an arbitrary edge \( R: X \leftrightarrow Y \) of \( G \), there is an edge of the shape on the left, below.

\[ \begin{array}{c}
F(X) \xrightarrow{\alpha_X} A \\
F(R) \downarrow \quad I_A \\
F(Y) \xrightarrow{h \circ \alpha_Y} B \\
\end{array} \quad \begin{array}{c}
A \xrightarrow{h} B \\
I_A \downarrow \quad I_B \\
A \xrightarrow{h} B \\
\end{array} \]

That square can be composed with the square on the right to get a square from \( F(R) \) to \( I_B \). Recall that the graph of a morphism is defined in terms of re-indexing as \( \langle h \rangle = [h, \text{id}]I \) (Theorem 4.10). Thus the fibration condition

175
ensures that there is a square of the shape on the left, below.

\[
\begin{array}{c}
F(X) \xrightarrow{\alpha_X} A \\
F(R) \xrightarrow{h \circ \alpha_Y} B \\
F(Y) \xrightarrow{1} B
\end{array}
\quad
\begin{array}{c}
1 \xrightarrow{\alpha_X} F(X) \Rightarrow A \\
1 \xrightarrow{1} F(R) \Rightarrow \langle h \rangle \\
1 \xrightarrow{\langle h \circ \alpha_Y \rangle} F(Y) \Rightarrow B
\end{array}
\]

Lemma 5.39 ensures that there is a square of the shape on the right. Since this holds for all edges \( R \) (which includes identity edges), there is a square of the following shape in \( G[G]\).

\[
\begin{array}{c}
\Delta 1 \xrightarrow{\alpha} F \Rightarrow \Delta A \\
1 \xrightarrow{I} F \Rightarrow \Delta \langle h \rangle \\
\Delta 1 \xrightarrow{\langle \Delta h \circ \alpha \rangle} F \Rightarrow \Delta B
\end{array}
\]

We can apply the limit PG-functor \( \text{Lim}: G[G] \) to this square to get a square of \( G \). Composing that square with a square from parametricity of the unit \( \eta: \text{ID} \Rightarrow \text{Lim} \circ \Delta \) of the adjunction between \( \Delta \) and \( \text{Lim} \) gives a square of the following shape.

\[
\begin{array}{c}
1 \xrightarrow{\text{Lim}(\alpha) \circ \eta_1} \text{Lim}(I_F \Rightarrow \Delta A) \\
1 \xrightarrow{I} \text{Lim}(I_F \Rightarrow \Delta \langle h \rangle) \\
1 \xrightarrow{\text{Lim}(\langle \Delta h \circ \alpha \rangle) \circ \eta_1} \text{Lim}(I_F \Rightarrow \Delta B)
\end{array}
\]

Recall that \( \text{Lim}(G) \) is the parametric limit of \( G \), and that the factorization of a parametric transformation \( \tau: \Delta C \Rightarrow G \) is given by \( \text{Lim}(\tau) \circ \eta_C \). Thus the previous square is of the following shape.

\[
\begin{array}{c}
1 \xrightarrow{\Lambda(\alpha)} \forall_X F(X) \Rightarrow A \\
1 \xrightarrow{I} \forall_X F(X) \Rightarrow \langle h \rangle \\
1 \xrightarrow{\Lambda(\langle \Delta h \circ \alpha \rangle)} \forall_X F(X) \Rightarrow B
\end{array}
\]

Composing the above square with the square from \( \forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y \) to \( (\forall_X F(X) \Rightarrow \langle h \rangle) \Rightarrow \langle h \rangle \) among others gives a square of the following shape.

\[
\begin{array}{c}
\forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y \xrightarrow{\text{ap} \circ (\text{id} \times \Lambda(\alpha)) \circ \rho^1 \circ \omega_A} A \\
1 \xrightarrow{I} \forall_X F(X) \Rightarrow \langle h \rangle \\
\forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y \xrightarrow{\text{ap} \circ (\text{id} \times \Lambda(\langle \Delta h \circ \alpha \rangle)) \circ \rho^1 \circ \omega_B} B
\end{array}
\]

176
Note that the morphism along the top of this square is $\nabla(\alpha)$, and the morphism along the bottom of this square is $\nabla(\Delta h \circ \alpha)$. Therefore, we conclude that $h \circ \nabla(\alpha) = \nabla(\Delta h \circ \alpha)$, as claimed.

Before showing that the factorization $\nabla(\beta)$ is unique, we first observe that the factorization of $\mu: F \to \forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y$ is the identity.

**Lemma 5.69**
For any non-variant functor $F$, $\nabla(\mu)$ is the identity on $\forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y$.

**Proof.** For any vertex $Z$, consider an arbitrary point $f:1 \to \forall_X F(X) \Rightarrow Z$. Lemma 5.66 ensures that $ap \circ (id \times f) \circ \rho^1 \circ \omega_Z \circ \mu = (\omega \circ f)^*$. Therefore lemma 5.68 ensures that $ap \circ (id \times f) \circ \rho^1 \circ \omega_Z \circ \nabla(\mu) = \nabla((\omega \circ f)^*)$. Expanding out the definition of $\nabla((\omega \circ f)^*)$ and using naturality properties will transform this equality into one that can be written using uncurry.

\[
\begin{align*}
ap \circ (id \times f) \circ \rho^1 \circ \omega_Z \circ \nabla(\mu) &= ap \circ (id \times \Lambda(\omega \circ f)) \circ \rho^1 \circ \omega_Z \\
ap \circ ((\omega_Z \circ \nabla(\mu)) \times id) \circ (id \times f) \circ \rho^1 &= ap \circ (\omega_Z \times id) \circ (id \times \Lambda(\omega \circ f)) \circ \rho^1 \\
\text{uncurry}(\omega_Z \circ \nabla(\mu)) \circ (id \times f) \circ \rho^1 &= \text{uncurry}(\omega_Z) \circ (id \times \Lambda(\omega \circ f)) \circ \rho^1
\end{align*}
\]

Observe that $\Lambda(\omega \circ f) = \Lambda(\omega \circ f) = f$. Since $\rho^1$ is an isomorphism, we conclude $\text{uncurry}(\omega_Z \circ \nabla(\mu)) \circ (id \times f) = \text{uncurry}(\omega_Z) \circ (id \times f)$. As $f$ is an arbitrary point, the well-pointedness of $G$ allows us to conclude that $\text{uncurry}(\omega_Z \circ \nabla(\mu)) = \text{uncurry}(\omega_Z)$. This suffices to prove the equality of $\omega_Z \circ \nabla(\mu)$ and $\omega_Z$. Since this hold for all $Z$, $\nabla(\mu)$ must be id, the unique factorization of $\omega$ through $\omega$.

**Theorem 5.70**
Suppose $F$ is a non-variant functor on $G$, $B$ is a vertex of $G$, and $\beta$ is a parametric transformation $F \to \Delta B$. Then $\nabla(\beta)$ is the unique morphism $h: \forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y$ such that $\Delta h \circ \mu = \beta$.

Therefore, $\exists_X F(X) = \forall_Y (\forall_X F(X) \Rightarrow Y) \Rightarrow Y$.

**Proof.** Suppose $h$ is any morphism such that $\Delta h \circ \mu = \beta$. Lemma 5.68 ensures that $h \circ \nabla(\mu) = \nabla(\beta)$. Since lemma 5.69 ensures that $\nabla(\mu) = \text{id}$, we conclude that $h = \nabla(\beta)$.

Parametric colimits are used to encode final co-algebras. We begin with the definition of a co-algebra in a parametricity graph.

**Definition 5.71**
For any $\PG$-functor $F: G \to G$, a co-algebra for $F$ consists of a vertex $A$ and a morphism $f: A \to F(A)$. The vertex $A$ is called the carrier of the co-algebra and $f$ is called the structure map.
A final co-algebra for $F$ is a co-algebra $(B, g)$ for $F$ such that for any co-algebra $(A, f)$ for $F$, there is a unique morphism $n^f: A \to B$ such that $g \circ n^f = F(n^f) \circ f$.

For any strong PG-functor $F: G \to G$, we construct a co-algebra having final($F$) = $\exists_X (X \Rightarrow F(X)) \times X$ as its carrier. To produce the structure map, we note that composing the symmetry transformation with the injection gives a parametric transformation with components as follows.

$$\mu_X \circ \sigma: X \times (X \Rightarrow F(X)) \to \text{final}(F)$$

The PG-functor $F$ can be applied to each component of that parametric transformation. Composing with the strength allows one to get a parametric transformation $F(\mu_X \circ \sigma) \circ \text{str} \circ (id \times ap) \circ (\delta \times id)$.

Note that one can produce a parametric transformation that has components $(X \Rightarrow F(X)) \times X \to (X \Rightarrow F(X)) \times F(X)$ that copy the $X \Rightarrow F(X)$ argument and applies it to the $X$ argument. Therefore, there is a transformation with components $(X \Rightarrow F(X)) \times X \to \text{final}(F)$ that we can factor through the parametric colimit to produce the structure map.

$$\text{out}^F = \nabla \left( F(\mu_X \circ \sigma) \circ \text{str} \circ (id \times ap) \circ (\delta \times id) \right)$$

The superscript will be suppressed when the PG-functor can be inferred.

To show that (final($F$), out) is the final co-algebra for $F$, the mapping $n^f: A \to \text{final}(F)$ for any co-algebra $(A, f)$ is given as follows.

$$n^f = \mu_A \circ (f^o \times id) \circ \lambda^1$$

This mapping commutes with the structure maps as desired.

**Lemma 5.72**

For any co-algebra $(A, f)$ of a strong PG functor $F$, out $\circ n^f = F(n^f) \circ f$.

**Proof.** Using the definitions of $n^f$ and out, the left hand side of the equation can be rewritten to make use of the fact that $\nabla(\tau) \circ \mu_A = \tau_A$ for any parametric transformation $\tau$.

$$\text{out} \circ n^f = \nabla \left( F(\mu_X \circ \sigma) \circ \text{str} \circ (id \times ap) \circ (\delta \times id) \right) \circ \mu_A \circ (f^o \times id) \circ \lambda^1$$

$$= F(\mu_A \circ \sigma) \circ \text{str} \circ (id \times ap) \circ (\delta \times id) \circ (f^o \times id) \circ \lambda^1$$

178
Using basic properties of Cartesian closure, we note the following equalities.

\[
(\delta \times \text{id}) \circ (f^0 \times \text{id}) \circ \lambda^1 = (f^0 \times f^0 \times \text{id}) \circ (\delta \times \text{id}) \circ \lambda^1
\]
\[
= (f^0 \times f^0 \times \text{id}) \circ (\text{id} \times \lambda^1) \circ \lambda^1 = (f^0 \times ((f^0 \times \text{id}) \circ \lambda^1)) \circ \lambda^1
\]

Therefore we can rewrite the left hand side of the equation in the lemma to using the notation \(\text{uncurry}(f^0)\).

\[
\begin{align*}
\text{out} \circ n^t & = F(\mu_A \circ \sigma) \circ \text{str} \circ \sigma \circ (\text{id} \times \text{ap}) \circ (f^0 \times ((f^0 \times \text{id}) \circ \lambda^1)) \circ \lambda^1 \\
& = F(\mu_A \circ \sigma) \circ \text{str} \circ \sigma \circ (f^0 \times (\text{ap} \circ (f^0 \times \text{id}) \circ \lambda^1)) \circ \lambda^1 \\
& = F(\mu_A \circ \sigma) \circ \text{str} \circ \sigma \circ (f^0 \times (\text{uncurry}(f^0) \circ \lambda^1)) \circ \lambda^1
\end{align*}
\]

Recall that \(\text{uncurry}(g) \circ \lambda^1 = g^*\). Since \(f^0* = f\), the string of equalities can be extended as follows.

\[
\begin{align*}
& = F(\mu_A \circ \sigma) \circ \text{str} \circ \sigma \circ (f^0 \times f^0^*) \circ \lambda^1 \\
& = F(\mu_A \circ \sigma) \circ \text{str} \circ \sigma \circ (f^0 \times f^0) \circ \lambda^1 \\
& = F(\mu_A \circ \sigma) \circ \text{str} \circ (f \times f^0) \circ \sigma \circ \lambda^1
\end{align*}
\]

Recall that \(\sigma \circ \lambda^1 = \rho^1\). Using the naturality of \(\rho^1\) and of \(\text{str}\), the string of equalities continues as follows.

\[
\begin{align*}
& = F(\mu_A \circ \sigma) \circ \text{str} \circ (f \times f^0) \circ \rho^1 \\
& = F(\mu_A \circ \sigma) \circ \text{str} \circ (\text{id} \times f^0) \circ \rho^1 \circ f \\
& = F(\mu_A \circ \sigma \circ (\text{id} \times f^0)) \circ \text{str} \circ \rho^1 \circ f
\end{align*}
\]

Since \(\text{str} \circ \rho^1 = F(\rho^1)\), we can take advantage of \(\sigma \circ \rho^1 = \lambda^1\) to get the desired equality.

\[
\begin{align*}
& = F(\mu_A \circ \sigma \circ (\text{id} \times f^0) \circ \rho^1) \circ f \\
& = F(\mu_A \circ \sigma \circ (f^0 \times \text{id}) \circ \sigma \circ \rho^1) \circ f \\
& = F(\mu_A \circ (f^0 \times \text{id}) \circ \lambda^1) \circ f \\
& = F(n^t) \circ f
\end{align*}
\]

This is the right hand side of the equality in the statement of the lemma.

The mapping of co-algebras \((A, f)\) to morphisms \(n^t: A \to \text{final}(F)\) preserves relations in the following sense.

**Lemma 5.73**

*Suppose \(F: G \to G\) is a strong PG-functor. If there is a square of the shape*
on the left, then there is a square of the shape on the right.

\[
\begin{array}{ccc}
A_0 & \xrightarrow{f_0} & F(A_0) \\
\downarrow{R} & & \downarrow{F(\cdot)} \\
A_1 & \xrightarrow{f_1} & F(A_1) \\
\end{array}
\quad
\begin{array}{ccc}
A_0 & \xrightarrow{n f_0} & \text{final}(F) \\
\downarrow{R} & & \downarrow{I} \\
A_1 & \xrightarrow{n f_1} & \text{final}(F) \\
\end{array}
\]

**Proof.** By lemma 5.39, there is a square of the following shape.

\[
\begin{array}{ccc}
1 & \xrightarrow{f_0 \circ} & A_0 \Rightarrow F(A_0) \\
\downarrow{I} & & \downarrow{R \Rightarrow F(\cdot)} \\
1 & \xrightarrow{f_1 \circ} & A_1 \Rightarrow F(A_1) \\
\end{array}
\]

This square can be combined with squares from the parametricity of \( \lambda^{-1} \) and \( \mu \) as follows.

\[
\begin{array}{ccc}
A_0 & \xrightarrow{\lambda^{-1}} & 1 \times A_0 \\
\downarrow{R} & & \downarrow{I \times R} \\
A_1 & \xrightarrow{\lambda^{-1}} & 1 \times A_1 \\
\end{array}
\quad
\begin{array}{ccc}
(A_0 \Rightarrow F(A_0)) \times A_0 & \xrightarrow{f_0 \circ \times \text{id}} & (R \Rightarrow F(\cdot)) \times R \\
\downarrow{\mu A_0} & & \downarrow{I} \\
(A_1 \Rightarrow F(A_1)) \times A_1 & \xrightarrow{f_1 \circ \times \text{id}} & \text{final}(F) \\
\end{array}
\]

Since \( n f_0 = \mu A_0 \circ (f_0 \circ \times \text{id}) \circ \lambda^{-1} \) and \( n f_1 = \mu A_1 \circ (f_1 \circ \times \text{id}) \circ \lambda^{-1} \), the above is the desired square.

The morphism \( n^{\text{out}} \) that arises from the co-algebra \((\text{final}(F), \text{out})\) is the identity morphism on \( \text{final}(F) \).

**Lemma 5.74**

*For any strong PG functor \( F : G \to G \), it is the case that \( n^{\text{out}} = \text{id} \).*

**Proof.** For any vertex \( Z \), we’ll show that \( n^{\text{out}} \circ \mu \mu Z = \mu \mu Z \). For an arbitrary point \( f : 1 \to Z \Rightarrow F(Z) \), there is a final co-algebra \((Z, f^*)\). Since lemma 5.72 ensures that \( \text{out} \circ n f^* = F(n f^*) \circ f^* \), there is a square of the shape on the left below.

\[
\begin{array}{ccc}
Z & \xrightarrow{f^*} & F(Z) \\
\downarrow{\langle n f^* \rangle} & & \downarrow{F(\langle n f^* \rangle)} \\
\text{final}(F) & \xrightarrow{\text{out}} & F(\text{final}(F)) \\
\end{array}
\quad
\begin{array}{ccc}
Z & \xrightarrow{n f^*} & \text{final}(F) \\
\downarrow{\langle n f^* \rangle} & & \downarrow{I} \\
\text{final}(F) & \xrightarrow{n^{\text{out}}} & \text{final}(F) \\
\end{array}
\]

180
Therefore, there is a square of the shape on the right (lemma 5.73). The subsumption property implies the following equality of morphisms.

\[ n^{\text{out}} \circ n_f = n_f \]

This equation can be rewritten by expanding out the definition of \( n_f \).

\[ n^{\text{out}} \circ \mu_Z \circ (f \times \text{id}) \circ \lambda^{-1} = \mu_Z \circ (f \times \text{id}) \circ \lambda^{-1} \]

Since \( \lambda^{-1} \) is an isomorphism, \( n^{\text{out}} \circ \mu_Z \circ (f \times \text{id}) = \mu_Z \circ (f \times \text{id}) \). As \( f \) is an arbitrary point, we conclude that \( n^{\text{out}} \circ \mu_Z = \mu_Z \). This holds for all \( Z \), so \( n^{\text{out}} = \text{id} \), the unique factorization of \( \mu \).

**Theorem 5.75**

*Suppose \( F: G \to G \) is a strong PG functor. For any co-algebra \( (A, f) \) for \( F \), \( n_f \) is the unique morphism \( h: A \to \text{final}(F) \) such that \( \text{out} \circ h = F(h) \circ f \). Consequently, \( (\text{final}(F), \text{out}) \) is the final co-algebra for \( F \).*

**Proof.** Suppose \( h \) is any morphism such that \( \text{out} \circ h = F(h) \circ f \). Thus subsumption ensures that there is a square of the shape on the right.

\[
\begin{array}{ccc}
A & \xrightarrow{f} & F(A) \\
\downarrow \langle h \rangle & & \downarrow \langle h \rangle \\
\text{final}(F) & \xrightarrow{\text{out}} & F(\text{final}(F)) \\
\end{array}
\quad
\begin{array}{ccc}
A & \xrightarrow{n_f} & \text{final}(F) \\
\downarrow \langle h \rangle & & \downarrow \langle h \rangle \\
\text{final}(F) & \xrightarrow{n^{\text{out}}} & \text{final}(F) \\
\end{array}
\]

Therefore, lemma 5.73 ensures there is a square of the shape on the right. Thus \( n^{\text{out}} \circ h = n_f \). Since \( n^{\text{out}} = \text{id} \) (lemma 5.74), we conclude that \( h = n_f \) as desired.

In any well-pointed parametricity graph, parametric limits and colimits give encodings of initial algebras and final co-algebras for strong PG-functors. These representation results support our claim that parametricity provides a reasonable formalization of uniformity.

### 5.6 A Negative Result

Being able to model System P in well-pointed parametricity settings exhibits the strong uniformity criteria provided by relational parametricity in the form of representation results. The well-pointed assumption is important for modeling System P. One may wonder how crucial the well-pointed
assumption is to having a sufficiently meaningful notion of uniformity. Simply having a parametricity setting for System F is not enough to ensure the standard representation results. We exhibit a non-well-pointed parametricity setting where the representation results do not hold.

Functor categories provide a canonical example of non-well-pointed categories. We shall consider the functor category $\text{SET}^C$ where $C$ is the category with one object $\star$, and two morphisms $i = \text{id}$ and $j$ such that $j \circ j = j$. A functor $F : C \to \text{SET}$ is equivalently presented as a set $F(\star)$ together with an idempotent $F(j) : F(\star) \to F(\star)$. A natural transformation $\eta : F \to G$ is given by a function $\eta_\star : F(\star) \to G(\star)$ that commutes with the idempotents $\eta_\star \circ F(j) = G(j) \circ \eta_\star$.

The parametricity graph $G$ we consider here has $\text{SET}^C$ as its vertex category. An edge $F : F \to F'$ of $G$ consists of a relation $F(\star) : F(\star) \leftrightarrow F'(\star)$ such that there is a square of the following shape in $\text{REL}$.

$$
\begin{array}{ccc}
F(\star) & \xrightarrow{\mathcal{F}(j)} & F(\star) \\
\mathcal{F}(\star) & & \mathcal{F}_\star \\
\downarrow & & \downarrow \\
F'(\star) & \xleftarrow{\mathcal{F}'(j)} & F'(\star)
\end{array}
$$

Squares $\mathcal{F} \to G$ are given as pairs of morphisms $(\eta, \eta')$ such that there exists a square of the following shape in $\text{REL}$.

$$
\begin{array}{ccc}
F(\star) & \xrightarrow{\eta_\star} & G(\star) \\
\mathcal{F}(\star) & & \mathcal{G}(\star) \\
\downarrow & & \downarrow \\
F'(\star) & \xleftarrow{\eta'_\star} & G'(\star)
\end{array}
$$

This parametricity graph is isomorphic to the functor graph $\text{REL}^{KC}$ where the parametricity graph $KC$ is the graph having $C$ as both the vertex and edge categories. (Describing the parametricity graph as a functor from $B$ to $\text{CAT}$, $KC$ is the constantly $C$ functor.)

The parametricity graph $G$ is Cartesian closed. The terminal object $1$ is given by the single object set $1(\star) = \{\star\}$ (with identity idempotent). The vertex portion of the exponent non-variant functor $\Rightarrow : G \times G \to G$ can be described as pairs of functions:

$$(F \Rightarrow G)^\star = \left\{ (a_i, a_j) \in (F(\star) \Rightarrow G(\star))^2 \mid G(j) \circ a_i = a_j \circ F(j) = G(j) \circ a_j \right\}$$

$$(F \Rightarrow G)j = (a_i, a_j) \mapsto (a_j, a_j)$$

182
The edge portion of the exponent is given in the apparent manner.

\[(a_i, a_j) \left[ (\mathcal{F} \Rightarrow \mathcal{G})^* \right] (a'_i, a'_j) \iff \]

\[
\begin{array}{c}
\xymatrix{
F(*) & G(*) \\
\mathcal{F}(*) \ar[r] & G(*) \\
F'(*) & G'(*) \\
\mathcal{F}'(*) \ar[r] & G'(*)
}
\end{array}
\quad \text{and} \quad
\begin{array}{c}
\xymatrix{
F(*) & G(*) \\
\mathcal{F}(*) \ar[r] & G(*) \\
F'(*) & G'(*) \\
\mathcal{F}'(*) \ar[r] & G'(*)
}
\end{array}
\]

Using \(X \Rightarrow X\) to denote the non-variant functor \(\mathbf{G} \to \mathbf{G}\) given by composing \(\Rightarrow\) with the canonical \(\delta: \mathbf{G} \to \mathbf{G} \times \mathbf{G}\), we show that the parametric limit \(\forall X X \Rightarrow X\) is not terminal. Consider the functor \(T: \mathbf{C} \to \mathbf{SET}\) where \(T(*) = \{i, j\}\) and \(T(j) = \text{id}\). There is a parametric transformation \(\omega: \Delta(T) \to X \Rightarrow X\), where the natural transformation \(\omega_F: T \to F \Rightarrow F\) is given as follows.

\[
i \longmapsto (F(i), F(i)) \quad j \longmapsto (F(j), F(j))
\]

It is straightforward to show that these are elements of \((F \Rightarrow F)^*\) and that \(\omega_F\) is natural. Showing that \(\omega\) is parametric requires showing that there is a square of the following shape for every \(\mathcal{F}: F \to F'\).

\[
\begin{array}{c}
\xymatrix{
T \ar[d]^{\omega_F} & \mathcal{F} \Rightarrow F \\
T \ar[d]_{\omega_{F'}} & \mathcal{F}' \Rightarrow F' \\
& \mathcal{F} \Rightarrow \mathcal{F}'
}
\end{array}
\]

We see that \((\omega_F, \omega_{F'})\) is such a square since for each \(m \in \{i, j\} = T(*)\), it is the case that \((F(m), F(m)) \left[ \mathcal{F} \Rightarrow \mathcal{F} \right] (F'(m), F'(m))\).

In light of the functor (i.e. object of \(\mathbf{SET}^C\)) \(T\), and parametric transformation \(\omega\) presented above, 1 cannot be the parametric limit of \(X \Rightarrow X\). For any parametric transformation \(\eta: 1 \to X \Rightarrow X\), the image of \((\eta_F)_*\) has exactly one element for all \(F\). Since the image of \((\omega_F)_*\) has two distinct elements for some (most) \(F\), there can not be a natural transformation (i.e. arrow of \(\mathbf{SET}^C\)) \(\Lambda(\omega): T \to \top\) such that \(\eta \circ \Lambda(\omega) = \omega\).

So what happened? Why does the intuitive argument not carry over into this setting? Recall the argument:

Given an element \(x\) of an unknown type \(X\), since all we know about \(X\) is that it contains \(x\), the only way to produce an element of type \(X\) is to return \(x\).

Notice there is a problem with the claim “all we know about \(X\) is that it
contains $\varphi$. In this setting, something else is known about $X$, namely that it’s a functor from $C$ to $\text{SET}$. This is enough to produce a (potentially) different return value.

Moving to the more formal (yet still fairly intuitive) argument that Reynolds used in $\text{SET}$ (section 2.2), one may wonder why that wouldn’t carry over. Since $\text{SET}^C$ is a topos (and hence a setting for intuitionistic logic), one would expect that any proof in $\text{SET}$ which does not make use of the Law of the Excluded Middle (that is, is purely intuitionistic) could be translated into any topos. However the use of relations in the parametricity argument goes outside of the logic which gets translated directly. In particular, (the analogue of) the relation $\text{only}_B = \{(b,b)\}$ is not necessarily an edge of $G$. The closest we can come is the edge $\mathcal{F}:B \leftrightarrow B$ where $b\left[F(x)\right]b$ and $B(j)b\left[F(x)\right]B(j)b$. Thus some $\tau_B(b)$ may be $B(j)b$ rather than $b$.

One reason for producing semantics where types are interpreted as something more structured than sets is to take advantage of that structure in modeling phenomenon which can not be captured in $\text{SET}$. One would typically then be interested in “relations” which respect that additional structure to allow for the constructions on types to be mirrored in “relations”. However, by restricting to “relations” which respect the structure, one may get things that are relationally parametric and lie outside of traditional parametric results.

Is this a cause for concern? Does this suggest that relational parametricity does not capture uniformity? No. The “new” parametric elements which appear are uniform. As an intuitive example, when considering pointed sets, the operation of selecting the point is uniform across all pointed sets. From a programming language point of view, in a language with nontermination, diverging at every type is doing the same thing at every type, hence behaving uniformly. The term $\Delta X.\lambda x:X.\text{diverge}$ is a perfectly legitimate term of type $\forall X.X \Rightarrow X$. We have not found examples of parametric elements in parametricity settings which are not apparently uniform. On the other hand, it is not the case that the standard representation results fail in all non-well-pointed settings. Rather, it needs to be considered further which such settings they may hold in. One could approach this issue by considering additional programming language features (that is, extensions to the lambda calculus) to see whether one might expect the representation results to hold. One such feature — local variables — is considered in the following chapter.

In this chapter, we have shown traditional representation results hold for well-pointed parametricity graph models of System F. Our approach was to
define System P, an extension of System F that includes relations to allow formal reasoning using parametricity arguments. The traditional representation results were shown to hold in System P. We showed that System P can be modeled in well-pointed parametricity settings for System F, and there are analogous representation results that hold in the models. The well-pointed assumption on parametricity settings is crucial, as we exhibited a non-well-pointed parametricity graph which does not satisfy one of the most basic representation results — the encoding of the terminal object.
Chapter 6

Imperative Parametricity

The study of imperative programming languages naturally leads one to consider semantics in functor categories [Rey80, Ole82, OT92]. Algol is a programming language which combines imperative programming with the procedure mechanism of a typed lambda calculus [Rey81]. It is therefore a well-suited basic language to use when studying imperative versions of typed lambda calculi.

In the following section we discuss background material on Algol-like languages and define an imperative polymorphic lambda calculus. Discussion of previously existing models of of Idealized Algol and their extension to the polymorphic setting is given in section 6.2. In section 6.3, we present a new parametricity graph model of polymorphic Idealized Algol which gives a better treatment of the uniformity over types.

6.1 Polymorphic Idealized Algol

Algol was introduced in 1960 [NeBB+60] as an algebraic programming language. This was a remarkable development and it greatly influenced the design of later programming languages, such as ISWIM [Lan66], Algol 68 [vMP+75], Gedanken [Rey70], Pascal [Wir71], Euclid [LGH+75], MESA [GMS77], and ADA [IHR79]. In 1981, John Reynolds pointed out some significant features of Algol that he felt make up its essence [Rey81]. Many of the successor languages failed to recognize the importance of these features, and are not, in Reynolds’ usage, Algol-like languages. Here, we introduce a polymorphic Algol-like language.

An important principle of Algol is, according to Reynolds, that the procedure mechanism is based on a typed lambda calculus. In particular, the $\beta$- and $\eta$-laws of function application hold for procedure invocation. The imperative features of Algol do not cause these equational laws to fail. Ide-
alized Algol, the language Reynolds introduced [Rey81] for reasoning about Algol-like languages, combines a simple imperative programming language with a simply typed lambda calculus. We take a similar approach, combining a polymorphic lambda calculus with the imperative language.

Another principle of Algol that Reynolds highlighted is that there are two fundamentally different kinds of types. Data types are the values appropriate for expressions and for storing in memory variables. Phrase types give the values for meanings of program fragments. In particular, entities like procedures and variables will have phrase types, and are not storables values. For simplicity, our language will only have a single data type - the natural numbers. Hence all expressions and variables represent natural numbers. We shall use exp and var to denote the phrase types of expressions and variables, respectively. The other phrase type characteristic of imperative programming languages, which we also include, is the type comm of commands.

Intuitively, imperative features can be described in terms of operations potentially using the state of the computer's memory. Expressions may read locations in memory, ultimately producing a data value (a natural number, in our case). Commands are operations which are executed, potentially modifying the memory. They can read the memory as well as write to the memory, with these operations potentially having significantly different effects depending on the state. In our language, commands are the only things that can write to the memory. Expressions are limited to reading the state — they can not modify it in any way. In other words, we do not allow expressions with side-effects. Variables are the basic building blocks by which the memory is accessed. Variables can be read (or dereferenced) and assigned to (which alters the memory). Term constants and constructs of a simple imperative language are given in table 6.1 (page 188).

The intuitive meanings of these constants and constructs are as follows. The terms 0 and succ are used to construct natural number expressions. The constant skip denotes the ‘do nothing’ command, while ; takes two commands and runs the first, then second. (Application of ; is typically denoted using infix notation, $C_1 ; C_2$.) The construct if $M = 0$ then $C_1$ else $C_2$ runs the command $C_1$ if the expression $M$ evaluates to 0 and runs $C_2$ otherwise.

The two primary operations on variables are dereferencing the value stored in it (deref $v$) and assigning a new value to it ($v := M$). Variables arise from the new $v$ in $C$ construct, where a new variable $v$ is created with the initial value 0, and given to the command $C$ to run.

Considering the allocation of variables (using new $v$ in $C$) brings up an-
\[
\begin{array}{ll}
\eta \vdash \Gamma & \eta \vdash \Gamma \\
\eta ; \Gamma \vdash \text{skip} : \text{comm} & \eta ; \Gamma \vdash 0 : \text{exp} \\
\eta ; \Gamma \vdash \text{comm} \times \text{comm} \Rightarrow \text{comm} & \eta ; \Gamma \vdash \text{succ} : \text{exp} \Rightarrow \text{exp} \\
\eta ; \Gamma \vdash M : \text{exp} & \eta ; \Gamma \vdash C_1 : \text{comm} \\
\eta ; \Gamma \vdash \text{if } M = 0 \text{ then } C_1 \text{ else } C_2 : \text{comm} & \\
\eta ; \Gamma \vdash v : \text{var} & \eta ; \Gamma \vdash M : \text{exp} \\
\eta ; \Gamma \vdash \text{deref } v : \text{exp} & \eta ; \Gamma \vdash v : \text{var} \\
\eta ; \Gamma , v : \text{var} \vdash C : \text{comm} & \eta ; \Gamma \vdash M : \text{exp} \\
\eta ; \Gamma \vdash \text{new } v \text{ in } C : \text{comm} & \\
\end{array}
\]

Table 6.1: Term constants and constructs of the imperative language

other important principle. In Algol-like languages, variable allocations and deallocations obey a stack discipline. Moreover, Reynolds maintains that it should be clear from the definition of the language that this discipline is maintained. This is the case in our setting. Using \texttt{new } v \texttt{ in } C, the variable \( v \) only exists for the duration of the command \( C \). The variable \( v \) is effectively deallocated at the completion of this block. The scope of the variable is fixed upon its allocation and will not escape, all taking place within a single command, \texttt{new } v \texttt{ in } C. Any variable allocated within \( C \) (that is, while \( v \) is allocated) via a command \texttt{new } v' \texttt{ in } C' will be deallocated on completion of the block \( C' \). Thus \( v' \) will be deallocated prior to the deallocation of \( v \). The stack discipline is maintained.

The language we will be using in this chapter is Polymorphic Idealized Algol, or PIA. The type system of PIA expands that of the predicative calculus (section 2.1) by including the imperative types, as indicated in the following grammar.

\[
\tau \ := \ X \mid \exp \mid \text{var} \mid \text{comm} \mid I \mid \tau_1 \times \tau_2 \mid \tau_1 \Rightarrow \tau_2 \\
\phi \ := \ X \mid \exp \mid \text{var} \mid \text{comm} \mid I \mid \phi_1 \times \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \forall X.\phi
\]

For simplicity, the only quantifier in PIA types is the universal quantifier (for polymorphic types). Adding existential quantification (for abstract data types) as in the predicative calculus does not present a problem, but neither does it illuminate any additional issues. Observe that PIA includes products,
which we did not include in predicative calculus. This is not crucial to
the analysis we perform, but is included for similarity with other Algol-like
languages.

There is one principle of Algol-like languages which PIA does not sat-
sify. In Algol, facilities like procedure definition and conditional constructs
are uniformly applicable to all phrase types. While procedures (that is,
\(\Rightarrow \) types) are available at all phrase types in PIA, we limit conditional con-
structs to only produce terms of type \texttt{comm}. The reason for this restriction
is semantic simplicity. Traditional functor category semantics (for instance,
[Ole82]) tend to define the interpretation of universally applicable condi-
tional constructions by induction on the resulting phrase type. It is not
apparent how the interpretation of conditionals could handle type variables
other than by imposing seemingly ad-hoc and awkward restrictions on the
collection of potential interpretations for type judgments like ‘‘... such that
there is a procedure mapping interpretations of \(M, N_1\) and \(N_2\) to an inter-
pretation of if \(M = 0\) then \(N_1\) else \(N_2\)’’. We avoid this distraction by only
having conditional commands rather than having conditionals for arbitrary
types. We shall briefly return to this point, commenting on the effects of
this choice, at the end of this chapter.

PIA is defined in a manner similar to the predicative calculus, starting
from two infinite collections — of type variables \(X\) and term variables \(x\).
There are three forms of judgments in PIA.

\[
\eta \vdash \alpha \quad \text{type judgment} \\
\eta \vdash \tau \quad \text{simple} \quad \text{simplicity judgment} \\
\eta ; \Gamma \vdash M : \phi \quad \text{term judgment}
\]

Just as in the predicative calculus, \(\eta\) represents a typing context, which is a
finite sequence of type variables. The expression \(x : \phi\) is a type assumption,
where \(x\) is a term variable and \(\phi\) is a type. The meta-variable \(\Gamma\) ranges over
contexts, that is, finite sequences of type assumptions where no term variable
is repeated. The meta-variable \(\alpha\) represents either a type or a context, while
\(\tau\) and \(\phi\) range over simple types and types, respectively.

A type judgment \(\eta \vdash \alpha\) holds in PIA if all type variables that appear
free in \(\alpha\) are contained in \(\eta\). A simplicity judgment \(\eta \vdash \tau \quad \text{simple}\) holds in
PIA provided \(\eta \vdash \tau\) holds. (Recall that \(\tau\) must be a simple type for the
judgment \(\eta \vdash \tau \quad \text{simple}\) to be well-formed.) Term judgments of PIA are
formed using term forming rules like those of the predicative calculus (table
2.3, section 2.1) and the simple imperative language (table 6.1). A term
judgment \( \eta ; \Gamma \vdash M : \phi \) is derivable in PIA if it follows from the empty set of hypotheses using a finite number of applications of rules in table 6.2 (page 191).

Terms of polymorphic types in PIA are not that different from terms of polymorphic types in the polymorphic lambda calculus. (Obviously, this does not apply to polymorphic types involving imperative types, such as \( \forall X.X \to \text{comm.} \).) State changes are only possible in terms of type \text{comm.} The state can only have an effect in terms of type \text{exp} or \text{comm}. State plays no role in an arbitrary type \( X \). The traditional representation results are still reasonable to expect. For instance, there is still only one term of type \( \forall X.X \to X \) no matter how many locations for variables are allocated in memory. Hence, it is reasonable to expect \( \forall X.X \to X \) to be interpreted as the terminal object in whatever category one uses to interpret types.

To facilitate our discussion of models for PIA, we shall use a sub-language of PIA corresponding to Idealized Algol with type variables but without polymorphic types. Formally, the Idealized Algol (or IA) types are the simple types of PIA. IA has the same three kinds of judgments as PIA, type judgments, simplicity judgments and term judgments. A type judgment \( \eta \vdash \alpha \) holds in IA if it is a type judgment of PIA and all types appearing in \( \alpha \) are IA types (simple types of PIA). The simplicity judgments of IA are exactly the same as the simplicity judgments of PIA. Simplicity judgments of IA provide no more information that a type judgment of IA about an individual type. (That is, for any IA type \( \phi \), the type judgment \( \eta \vdash \phi \) holds if and only if \( \eta \vdash \phi \) \text{simple} holds.) The term judgments of IA are those that are derivable using the rules in table 6.2 (page 191) except the two rules relating to polymorphic types:

\[
\eta, X ; \Gamma \vdash M : \phi \quad \eta \vdash \Gamma \\
\eta ; \Gamma \vdash (\Lambda X.M) : \forall X.\phi
\]

\[
\eta ; \Gamma \vdash \tau \quad \eta \vdash \tau \text{ simple} \\
\eta ; \Gamma \vdash M[\tau] : \phi[\tau/x]
\]

### 6.2 Models of Idealized Algol

A fairly intuitive categorical model of the imperative language can be given using the functor category \( S^L \) [Rey80, Ole82, OT92]. Here \( L \) denotes the category of finite sets and injections and \( S \) is a Cartesian closed category of sets that contains \( \mathbb{N} \). The intuition behind this is that a finite set \( W \) is the set of labels for memory variables that are currently allocated, called a \text{world}. A \text{state} of the computer’s memory for the world \( W \in L \) gives a natural number for each of the labels \( l \in W \) (intuitively, the value of the variable). We define the set of possible states for the world \( W \) to be the
\[\eta \vdash x_1: \phi_1, \cdots, x_m: \phi_m \quad 1 \leq j \leq m\]
\[\eta; x_1: \phi_1, \cdots, x_m: \phi_m \vdash x_j: \phi_j\]
\[\eta; \Gamma, x: \tau_1 \vdash M: \tau_2\]
\[\eta; \Gamma \vdash (\lambda x: \tau_1. M): \tau_1 \Rightarrow \tau_2\]
\[\eta; \Gamma \vdash M: \tau_1 \Rightarrow \tau_2 \quad \eta; \Gamma \vdash N: \tau_1\]
\[\eta; \Gamma \vdash M \cdot N: \tau_2\]
\[\eta; X; \Gamma \vdash M: \phi \quad \eta \vdash \Gamma\]
\[\eta; \Gamma \vdash (\Lambda X. M) : \forall X. \phi\]
\[\eta; \Gamma \vdash M: \forall X. \phi \quad \eta \vdash \tau \text{ simple}\]
\[\eta; \Gamma \vdash M[\tau]: \phi[\tau/x]\]
\[\eta \vdash \Gamma\]
\[\eta; \Gamma \vdash \text{skip} \text{ comm}\]
\[\eta \vdash \Gamma\]
\[\eta; \Gamma \vdash 0: \text{exp}\]
\[\eta \vdash \Gamma\]
\[\eta; \Gamma \vdash : \text{comm} \times \text{comm} \Rightarrow \text{comm}\]
\[\eta \vdash \Gamma\]
\[\eta; \Gamma \vdash \text{succ}: \text{exp} \Rightarrow \text{exp}\]
\[\eta; \Gamma \vdash M: \text{exp} \quad \eta; \Gamma \vdash C_1: \text{comm} \quad \eta; \Gamma \vdash C_2: \text{comm}\]
\[\eta; \Gamma \vdash \text{if } M = 0 \text{ then } C_1 \text{ else } C_2: \text{comm}\]
\[\eta; \Gamma \vdash v: \text{var} \quad \eta; \Gamma \vdash M: \text{exp}\]
\[\eta; \Gamma \vdash v = M: \text{comm}\]
\[\eta; \Gamma \vdash v: \text{var}\]
\[\eta; \Gamma \vdash \text{deref } v: \text{exp}\]
\[\eta; \Gamma, v: \text{var} \vdash C: \text{comm}\]
\[\eta; \Gamma \vdash \text{new } v \text{ in } C: \text{comm}\]

Table 6.2: Term forming rules of PIA
indexed product \( S(W) = \Pi_{i \in W} \mathbb{N} \).

A categorical model of IA can be built that interprets judgments as follows.

- **type judgment** \( \llbracket \eta \vdash \alpha \rrbracket : [S^L]^n \to S^L \)
- **simplicity judgment** \( \llbracket \eta \vdash \tau \text{ simple} \rrbracket : [S^L]^n \to S^L \)
- **term judgment** \( \llbracket \eta ; \Gamma \vdash M : \phi \rrbracket :: \llbracket \eta \vdash \phi \rrbracket \)

We shall only mention the interpretation of the type judgments, as the type system suffices to illuminate the issues at hand. The interested reader is referred to Oles [Ole82] for details on interpreting term judgments. A complete model (using parametricity graphs) of PIA will be presented in section 6.3. In order to define the interpretation of type judgments, we first define functors \( \mathbb{L} \to \mathbb{S} \) that correspond to intuitive descriptions of \texttt{exp}, \texttt{comm} and \texttt{var}.

Expressions are simply described as functions from states to natural numbers. Commands are functions from states to states. Variables are described in terms of expressions and commands, as they provide a command that changes the stored value and an expression to read the currently stored value. Hence the object portions of the functors corresponding to imperative types are as follows.

\[
[\text{exp}]W = \{ f : S(W) \to \mathbb{N} \}
\]
\[
[\text{comm}]W = \{ f : S(W) \to S(W) \}
\]
\[
[\text{var}]W = (\mathbb{N} \Rightarrow [\text{comm}]W) \times [\text{exp}]W
\]

An injection \( j : W \to W' \) determines a canonical map \( \phi_j : S(W') \to S(W) \) that projects out the components of the smaller world. There is also a map \( \tau_j : (S(W) \Rightarrow S(W)) \to (S(W') \Rightarrow S(W')) \). The function \( \tau_j \) takes a state transformer \( f \) for the smaller world and produces a state transformer for the larger world that does the same thing as \( f \), leaving the new locations untouched.

\[
\phi_j(s_{x'})_{x \in W'} = (s_{j(y)})_{y \in W}
\]
\[
\tau_j(f)(s_{x'})_{x \in W'} = \begin{cases} 
\pi_y(f(\phi_j(s_{x'}))) & \text{if } z = j(y) \text{ for some } y \in W \\
 s_z & \text{otherwise}
\end{cases}_{z \in W'}
\]

These two maps are used to define the arrow portions of the functors corre-
sponding to imperative types.

\[ [\text{exp}] \eta(f) = f \circ \phi_j \]
\[ [\text{comm}] \eta(f) = \tau_j(f) \]
\[ [\text{var}] \eta(f, g) = (\tau_j \circ g, f \circ \phi_j) \]

The Cartesian closed structure of the functor category \( \mathbf{S}^L \) provides a model of a typed lambda calculus (the procedure mechanism of Idealized Algol). We briefly mention the object portions of the Cartesian closed structure.

\[(F \times G)W = FW \times GW\]
\[(F \Rightarrow G)W = \text{Nat}(\text{Hom}(W, -) \times F, G)\]

Examining the exponent, one could curry each component of a natural transformation to get a function from \( \text{Hom}(W, W') \) to \( F(W') \Rightarrow G(W') \). This view of the exponent is easily explained in terms of procedures [Rey80]. A procedure \( F \Rightarrow G \) defined in world \( W \) may be called in a larger world \( W' \) after more variables have been allocated. So for any way to go from world \( W \) to \( W' \) (to allow non-local variables from the world \( W \) to be used in the world \( W' \)), the procedure should be applicable in world \( W' \), mapping \( F(W') \) inputs to \( G(W') \) outputs.

The above constructions provide the tools needed for interpreting the type judgments of IA. We give the inductive definition of the interpretation of type judgments below.

\[ [X_1, \cdots, X_n \vdash X_j] = \Pi_j \]
\[ [\eta \vdash \text{exp}] = \Delta([\text{exp}]) \]
\[ [\eta \vdash \text{comm}] = \Delta([\text{comm}]) \]
\[ [\eta \vdash \text{var}] = \Delta([\text{var}]) \]
\[ [\eta \vdash \phi_1 \times \phi_2] = [\eta \vdash \phi_1] \times [\eta \vdash \phi_2] \]
\[ [\eta \vdash \phi_1 \Rightarrow \phi_2] = [\eta \vdash \phi_1] \Rightarrow [\eta \vdash \phi_2] \]
\[ [\eta \vdash x_1 : \phi_1, \cdots, x_m : \phi_m] = [\eta \vdash \phi_1] \times \cdots \times [\eta \vdash \phi_m] \]

A simplicity judgment is interpreted in the same way as the corresponding type judgment \([\eta \vdash \tau \text{ simple}] = [\eta \vdash \tau] \).

The interpretation of procedure types in the categorical model mentioned above (and analyzed more extensively by Oles [Ole82]) captures the notion that procedures (terms of type \( \phi_1 \Rightarrow \phi_2 \)) should be applicable in the presence of local variables that were not available at the procedure’s definition.
However, it does not capture the intuitive understanding that a procedure should not affect subsequently declared local variables. Intuitively, a procedure should have no effect on variables which were not allocated at the time of the procedure definition, although the arguments to the procedure might be able to. For instance, suppose P is a procedure of type $\text{comm} \Rightarrow \text{comm}$ that does not use z. “Running” $P(z := \text{succ}(z))$ may change the value of the variable z, but P itself would have no other access to z. The value of z would not be reset to 0 by the “running” of $P(z := \text{succ}(z))$ in the command $\eta: \Gamma \vdash \text{new z in } (z := \text{succ}(0); P(z := \text{succ}(z)) \ldots): \text{comm}$ for any P such that $\eta: \Gamma \vdash P: \text{comm} \Rightarrow \text{comm}$ (where z is not in $\Gamma$).

The description of procedure types using the exponent of $\mathcal{S}^L$ does not ensure that elements of type $[\text{comm}] \Rightarrow [\text{comm}]$ at a world W do not themselves have an effect on locations not in W. O’Hearn and Tennent proposed that better independence of procedures from subsequently declared local variables can be assured using parametricity constraints [OT95]. By defining a reflexive graph category over $\mathcal{L}$ and using indexed functors from it into $\text{REL}$, they were able to enforce stronger uniformity criteria on the interpretation of procedure types. The exponent in the category of indexed functors they used (the vertex category of the functor graph) contains indexed natural transformations, rather than merely natural transformations. This allowed them to prove some program equivalences that were expected in the language but had not been satisfied in earlier models.

The reflexive graph category used by O’Hearn and Tennent is not a parametricity graph. It is not fibred, nor even subsumptive, and hence does not yield a setting where parametricity implies naturality. For this reason, an alternative reflexive graph category was presented by Reddy [Red97]. The parametricity graph $W$ (of worlds) is defined as follows. The vertex category of $W$ is the category $\mathcal{L}$ of finite sets and injections.

An edge $R: W_0 \leftrightarrow W_1$ of $W$ consists of a pair of relations, one on the state sets themselves, $R_\mathcal{S}: \mathcal{S}(W_0) \leftrightarrow \mathcal{S}(W_1)$, and the other on state transformers, $R_\mathcal{T}: (\mathcal{S}(W_0) \Rightarrow \mathcal{S}(W_0)) \leftrightarrow (\mathcal{S}(W_1) \Rightarrow \mathcal{S}(W_1))$, such that the identity transformations are related, $R_\mathcal{T}$ is closed under composition, and the following are
squares of REL.

\[
\begin{align*}
S(W_0) \Rightarrow S(W_0) & \xrightarrow{id} S(W_0) \Rightarrow S(W_0) \\
R_t \downarrow & \quad R_s \Rightarrow R_s \downarrow \\
S(W_1) \Rightarrow S(W_1) & \xrightarrow{id} S(W_1) \Rightarrow S(W_1)
\end{align*}
\]

\[D \quad S(W_0) \Rightarrow S(W_0) \Rightarrow S(W_0) \]

\[D \quad S(W_1) \Rightarrow S(W_1) \Rightarrow S(W_1) \]

Here \(D\) denotes the obvious diagonalizing function, \(D(f)(a) = f(a,a)\).

The first square is straightforward, while the motivation behind the second square might be mysterious. A primary use of \(D\) in the semantics is for interpreting assignments \(v := M\). A state is used by the expression \(M\) to produce a number, which in turn, is used by the variable to create a state transformer which will store that number. The state used to evaluate the expression is the same state that we wish to modify in storing the number. The second square above ensures that we only consider those relations that admit a uniformly given assignment operation.

The squares of \(\mathcal{W}\) refer to the maps \(\phi_j\) and \(\tau_j\) determined by an arrow of \(\mathbf{L} \ (= \mathcal{W}_v)\). There is a square from \(R\) to \(R'\) (as indicated below) in \(\mathcal{W}\) if and only if there are squares of the following two shapes in REL

\[
\begin{align*}
W_0 & \xrightarrow{j_0} W'_0 \\
R_t \downarrow & \quad R_t' \downarrow \\
W_1 & \xrightarrow{j_1} W'_1
\end{align*}
\]

\[
\begin{align*}
S(W'_0) & \xrightarrow{\phi_{j_0}} S(W_0) \\
R_s \downarrow & \quad R_s \downarrow \\
S(W'_1) & \xrightarrow{\phi_{j_1}} S(W_1)
\end{align*}
\]

\[
\begin{align*}
S(W_0) \Rightarrow S(W_0) & \xrightarrow{\tau_{j_0}} S(W'_0) \Rightarrow S(W'_0) \\
R_t \downarrow & \quad R_t' \downarrow \\
S(W_1) \Rightarrow S(W_1) & \xrightarrow{\tau_{j_1}} S(W'_1) \Rightarrow S(W'_1)
\end{align*}
\]

The identity edge over a world \(W\) consists of the pair of identity relations \((I_W)_s = I_{S(W)}\) and \((I_W)_t = I_{S(W)} \Rightarrow S(W)\).

It is not difficult to show that \(\mathcal{W}\) is a parametricity graph, where the

195
preimage \([j, j']R\) is given by the postimage of the state-set relation and the preimage of the transformer relation.

\[
([j, j']R)_s = R_s[\phi_j, \phi_{j'}] \\
([j, j']R)_t = [\tau_j, \tau_{j'}]R_t
\]

As a particular case, the graph of a morphism \(j: W \rightarrow W'\) was defined as \([j, \text{id}]_{W'}\).

\[
s \left[ (j)_s \right] s' \iff s = \phi_j(s') \\
t \left[ (j)_t \right] t' \iff \tau_j(t) = t'
\]

A state \(s \in \mathcal{S}(W)\) is related to all states \(s' \in W'\) that project on to it. A state transformer \(t: \mathcal{S}(W) \rightarrow \mathcal{S}(W)\) is related only to the unique transformer \(t': \mathcal{S}(W') \rightarrow \mathcal{S}(W')\) which does the same thing as \(t\), leaving the new components unmodified.

The functor interpretations of the imperative types can easily be extended to PG-functors \(\mathcal{W} \rightarrow \mathbf{REL}\). (These make use of the Cartesian closed structure of \(\mathbf{REL}\).)

\[
[\text{exp}] W = \mathcal{S}(W) \Rightarrow \mathbb{N} \\
[\text{exp}] j = f \mapsto f \circ \phi_j \\
[\text{exp}] R = R_s \Rightarrow \mathbb{I}_\mathbb{N}
\]

\[
[\text{comm}] W = \mathcal{S}(W) \Rightarrow \mathcal{S}(W) \\
[\text{comm}] j = f \mapsto \tau_j(f) \\
[\text{comm}] R = R_t
\]

\[
[\text{var}] W = (\mathbb{I}_\mathbb{N} \Rightarrow [\text{comm}] W) \times [\text{exp}] W \\
[\text{var}] j = (f, g) \mapsto (\tau_j \circ f, g \circ \phi_j) \\
[\text{var}] R = (\mathbb{I}_\mathbb{N} \Rightarrow [\text{comm}] R) \times [\text{exp}] R
\]

The Cartesian closed structure of the functor graph \(\mathbf{REL}^\mathcal{W}\) is described as follows. The terminal object and product of PG-functors are given pointwise. The exponent of PG-functors \(F \Rightarrow G\) is isomorphic to the collection of parametric transformations \(\text{Par.\,Trans.}(\mathcal{H}(W, -) \times F, G)\) in analogy with the categorical model. We prefer the more intuitive characterization of \(\mathcal{F} \Rightarrow G\) as a collection of functions indexed by morphisms out of the current
world, as given below.

\[
1(W) = \{\ast\} \quad (F \times G)W = FW \times GW \\
1(j) = \text{id} \quad (F \times G)j = Fj \times Gj \\
1(R) = 1 \quad (F \times G)R = FR \times GR
\]

\[
(F \Rightarrow G)W = \left\{ \langle f_j; F(W') \rightarrow G(W') \rangle; j:W \rightarrow W' \mid \text{for any square in } W \right. \\
of the shape on the left, there is a square of the shape on the right in \textbf{REL}
\]

\[
W \xrightarrow{j} W' \\
I \downarrow \quad R \downarrow \\
W \xleftarrow{k} W''
\]

\[
F(W') \xrightarrow{f_j} G(W') \\
F(R) \downarrow \quad G(R) \downarrow \\
F(W'') \xleftarrow{f_k} G(W'')
\]

\[
(F \Rightarrow G)j = \langle f_k; k:W \rightarrow W' \mid \rightarrow \langle f_{k,j}; k:W' \rightarrow W \rangle
\]

\[
(F \Rightarrow G)R = \left\{ \langle f_j; j:W \rightarrow V \mid \rightarrow \langle g_k; k:W' \rightarrow V \rangle \mid \text{for any square in } W \text{ of the} \right. \\
\text{shape on the left, there is a square of the shape on the right in } \textbf{REL}
\]

\[
W \xrightarrow{j} V \\
R \downarrow \quad S \downarrow \\
W' \xleftarrow{k} V'
\]

\[
F(V) \xrightarrow{f_j} G(V) \\
F(S) \downarrow \quad G(S) \downarrow \\
F(V') \xleftarrow{g_k} G(V')
\]

Not surprisingly, the product and exponent of parametric transformations are given pointwise and by composition, respectively.

\[
(\tau \times \tau')_W = \tau_W \times \tau'_W \\
(\tau \Rightarrow \tau')_W = \eta \leftarrow \tau' \circ \eta \circ (\text{id} \times \tau) \\
\]

\[
\quad = \langle f_j; j:W \rightarrow W' \mid \rightarrow \langle \tau'_W \circ f_j \circ \tau_W \rangle; j:W \rightarrow W' \rangle
\]

While the definition of products and exponents of edges in \textbf{REL}^W is not necessary for interpreting IA (neither O’Hearn & Tennent nor Reddy even acknowledged the functor graph), we include it here for completeness. It will be used when interpreting PIA. Since most of an edge \mathcal{F}: F \leftrightarrow F' can be inferred from the PG-functors \( F, F' : \mathcal{W} \rightarrow \textbf{REL} \) it relates, we only need
to spell out the definition of the edges $F \times G$ and $F \Rightarrow G : E \times W \rightarrow \text{REL}$ on the $E$ component of $E$.

$$(F \times G)(E, R) = F(E, R) \times G(E, R)$$

$$(F \Rightarrow G)(E, R) = \left\{ \langle f_j \rangle_{j : W \rightarrow V}, \langle g_j \rangle_{j : W' \rightarrow V} \right\} \text{ for every } R \quad \begin{array}{ccc} W & \xrightarrow{j} & V \\ \downarrow & & \downarrow \\ W' & \xrightarrow{k} & V' \end{array}$$

there is $F(E, S) \xrightarrow{f_j} G(s, V)$

$G(E, S)$ in $\text{REL}$

$F(t, V') \xrightarrow{g_k} G(t, V')$

The preservation of edges (between worlds) in $F \Rightarrow G$ is explicitly on display in the “indexed by morphisms” characterization presented above. This extra measure of uniformity (going beyond the categorical model mentioned previously) is useful in proving that procedures are independent of subsequently declared local variables [OT95, Red97].

The parametricity graph model of IA mentioned above can be extended to a model of PIA which is analogous to the model of the predicative calculus using a parametricity setting (section 3.3). Suppose $R$ is a small, Cartesian closed sub-parametricity graph of $\text{REL}$ containing $\mathbb{N}$ as a vertex such that $R^W$ is a sub-Cartesian closed parametricity graph of $\text{REL}^W$. One can model PIA using the functor graph $G = R^W$ for simple types and $H = \text{REL}^W$ for types. (More precisely, simplicity judgments in a typing context with $n$ type variables are interpreted as non-variant functors $G^n \rightarrow G$ and type judgments in a typing context with $n$ type variables are interpreted as non-variant functors $G^n \rightarrow H$.)

By currying and uncurrying PG-functors, parametric limits in a functor graph can be described in terms of parametric limits with parameters in the codomain. Recalling parametric limits in $\text{REL}$, the parametric limit $\text{Lim: } H^{\mathbb{C}|} \rightarrow H$ is computed at any vertex $F$ and any edge $F$ of $H^{\mathbb{C}|}$ as

198
follows.

\[ \forall x F(x) W = \{ (p_G)_{G \in G_v} \mid \text{for all } G: G \leftrightarrow G', \ p_G [F(G)(E, I_W)] p_{G'} \} \]

\[ \forall x F(x) j = \{ (p_G)_{G \in G_v} \mid (F(G) j(p_G))_{G \in G_v} \} \]

\[ \forall x F(x) R = \{ ((p_G), (q_G)) \mid \text{for all } G: G \leftrightarrow G', \ p_G [F(G)(E, R)] q_{G'} \} \]

\[ \forall x \mathcal{F}(x) (E, R) = \{ ((p_G), (q_G)) \mid \text{for all } G: G \leftrightarrow G', \ p_G [\mathcal{F}(E, G)(E, R)] q_{G'} \} \]

Parametric limits make use of edges between PG-functors (that is, potential instantiations of type variables) to enforce uniformity. This uniformity is at a distinctly different level than the uniformity criteria in exponents (preserving edges between worlds), although the same technique is used for both levels of uniformity. One can see the preservation of edges at both levels when one considers polymorphic types involving the exponent. For instance, the interpretation of the type $\emptyset \vdash \forall X. X \Rightarrow X$ is given at any world $W$ as follows.

\[ [\forall X. X \Rightarrow X] W = \{ (p_G \in (G \Rightarrow G) W)_{G: W \Rightarrow W} \mid \text{for every } G: G \leftrightarrow G', \ p_G [(G \Rightarrow G)(E, I_W)] p_{G'} \} \]

\[ \cong \{ (f^j_{G}: G(W') \rightarrow G(W'))_{G: W \Rightarrow W, j: W \Rightarrow W'} \mid \text{for any } G: G \leftrightarrow G' \text{ and square of the shape on the left in } W_v \text{ there is a square of the shape on the right in } \text{REL} \}

\[
\begin{align*}
W & \xrightarrow{j} V & G(W') & \xrightarrow{f^j_{G}} G(W') \\
I_W & \quad | \quad | \quad | & \quad | \quad | \quad | & \quad | \quad | \quad | \\
W & \xrightarrow{j'} V' & G'(W'') & \xrightarrow{f^j'_{G'}} G'(W'')
\end{align*}
\]

All edges between worlds are preserved, ensuring that procedures of type $(X \Rightarrow X) W$ only use variables from the world $W$. The preservation of edges between non-variant functors (potential interpretations of types) ensures that polymorphic procedures $(\forall X. X \Rightarrow X) W$ do the same thing for all types $X$. However, these uniformity requirements are not sufficient to ensure that the only polymorphic procedure $(\forall X. X \Rightarrow X) W$ is the family of identity functions.
Theorem 6.1
For any world $W$ with $|W| \geq 2$, $[\forall X.X \Rightarrow X]W$ is not a singleton set.

Proof. Any endomorphism $m: W \rightarrow W$ in $\mathcal{W}$ can be extended along any morphism $j: W \rightarrow V$. (Recall a morphism of $\mathcal{W}$ is an injection.) The resulting endomorphism $j\{m\}: V \rightarrow V$ simply performs $m$ on the image of $W$.

$$j\{m\}x = \begin{cases} j(m(y)) & \text{if } x = j(y) \\ x & \text{otherwise} \end{cases}$$

For any morphism $m: W \rightarrow W$, these $j\{m\}$ can be used to describe an element $\tilde{m} \in [\forall X.X \Rightarrow X]W$ that corresponds to $m: W \rightarrow W$. The component of $\tilde{m}$ at an arbitrary PG-functor $G: W \rightarrow \text{REL}$ is given as follows.

$$\tilde{m}_G = \langle G(j\{m\}) \rangle_{j: W \rightarrow V} \in (G \Rightarrow G)W$$

Verifying that $\tilde{m}_G$ preserves relations uses the fact that for any square of shape (a) below, there is a square of shape (b) to which we can apply $G$.

\begin{figure}[h]
\centering
\begin{tikzpicture}[node distance=2cm,auto]
  \node (W) {$W$};
  \node (V) [right of=W, xshift=2cm] {$V$};
  \node (U) [below of=W, yshift=-2cm] {$U$};
  \node (R) [above of=V, yshift=2cm] {$R$};

  \draw[->] (W) -- node {$j$} (V);
  \draw[->] (W) -- node {$k$} (U);
  \draw[->] (V) -- node {$R$} (R);
  \draw[->] (U) -- node {$R$} (R);

  \node (W') [below of=V, yshift=-2cm] {$W'$};
  \node (V') [right of=W', xshift=2cm] {$V'$};
  \node (U') [below of=W', yshift=-2cm] {$U'$};
  \node (R') [above of=V', yshift=2cm] {$R'$};

  \draw[->] (W') -- node {$j\{m\}$} (V');
  \draw[->] (W') -- node {$k\{m\}$} (U');
  \draw[->] (V') -- node {$R'$} (R');
  \draw[->] (U') -- node {$R'$} (R');

\end{tikzpicture}
\end{figure}

Recalling what it means for there to be a square in $\mathcal{W}$, we need to show that there are squares (c) and (d) in $\text{REL}$.

\begin{figure}[h]
\centering
\begin{tikzpicture}[node distance=2cm,auto]
  \node (V) {$S(V)$};
  \node (U) [below of=V, yshift=-2cm] {$S(U)$};
  \node (S) [above of=U, yshift=2cm] {$S(S(V))$};
  \node (T) [right of=S, xshift=2cm] {$S(S(U))$};

  \draw[->] (V) -- node {$\phi_{j\{m\}}$} (S);
  \draw[->] (V) -- node {$\phi_{k\{m\}}$} (U);
  \draw[->] (S) -- node {$\tau_{j\{m\}}$} (T);
  \draw[->] (U) -- node {$\tau_{k\{m\}}$} (T);

\end{tikzpicture}
\end{figure}

An endomorphism $m: W \rightarrow W$ defines a canonical state transformer $\overline{m}: S(W) \rightarrow S(W)$ which reorders the components of the state, as indicated by $m$.

$$\overline{m}(s_x)_{x \in W} = \langle s_{m(x)} \rangle_{x \in W}$$

Recall that morphisms of $\mathcal{W}$ are injections on finite sets, hence an endomorphism $m: W \rightarrow W$ is an isomorphism. It is clear that the state transformer corresponding to $m^{-1}$ is the inverse of the state transformer $\overline{m}$ corresponding to $m$. These state transformers can be used to give an alternate characteri-
zation of the functions $\phi_j(m)$ and $\tau_j(m)$ as follows.

**Proposition 6.2**

For any injections (morphisms of $W$) $m: W \rightarrow W$, $j: W \rightarrow W'$ and any function $f: S(W') \rightarrow S(W')$, 

$$\phi_j(m) = \tau_j(m)$$

$$\tau_j(m)f = \tau_j(m'f) \circ f \circ \tau_j(m)$$

With these characterizations, we can easily show that for any square of shape (a), we have that $\tau_j(R_i) R_k(R_i)$ and $\tau_j(R_i) R_k(R_i)$. Now square (c) follows since $R_i \subseteq R_s \Rightarrow R_s$ and square (d) follows since $R_i$ is closed under composition.

Note that the terminal object in the functor graph $\text{REL}^W$ is a constant PG-functor which produces the terminal object of $\text{REL}$ (a one element set). Therefore, the interpretation of $\emptyset \models \forall X.X \Rightarrow X$ is not the terminal object.

### 6.3 Representation Results in PIA

The interpretation of $\forall X.X \Rightarrow X$ is not terminal in $\text{REL}^W$. In this sense, the model is not an accurate depiction of PIA. The elements $\tilde{m}$ of $[\forall X.X \Rightarrow X]W$ constructed in the previous section are not elements that will arise from the interpretation of any term of PIA. In the current section we examine this aspect of the model, and describe a parametricity graph model which more closely relates to PIA.

We begin by considering the non-identity elements $\tilde{m}$ constructed in the previous section. Are they an indication that the preservation of edges does not ensure (intuitive) uniformity? We think not, as each $\tilde{m}$ is intuitively uniform. The family $\tilde{m}$ does the same thing at every type — namely performs the identity on its input and swaps memory locations around in the store. An element $\tilde{m}$ is an identity function with side-effects. The same side-effect happens at all types. These elements $\tilde{m}$ are elements of polymorphic type in $\text{REL}^W$ that are uniform operations, just not operations of PIA.

We can produce a slightly different setting that avoids these side-effecting polymorphic functions.

The counterexamples of the previous section arose from non-identity endomorphisms in $W$. With the intended meaning of the vertices as sets of location names, a permutation $m: W \rightarrow W$ represents reassigning names to locations. This is not an operation that is used in managing the store.
\[(F \times G)W = F(W) \times G(W)\]
\[(F \Rightarrow G) W = \left\{ (f_j: F(W') \to G(W')) \mid j: W \to W' \right\} \text{ for any square} \]
\[\text{in } L \text{ of the shape on the left, there is a square in REL of the shape on the right}\]
\[
\begin{array}{c|c|c}
W & f_j & W' \\
\hline
L & F(W') & \text{for squares} \\
\hline
W & f_k & W''
\end{array}
\]
\[(\forall Y F(Y)) W = \left\{ (p_G)_{G: L \to R} \mid \text{for every } G: G \leftrightarrow G', \right. \\
\left. p_G \left[ F(G)(E, I) \right], p_G' \right\}
\]

Table 6.3: Cartesian closed structure & parametric limits of REL\(^L\)

shapes in PIA. Since the only way to alter the shape of the store in PIA is to allocate new variables, we only need to be able to increase the set of locations.

Our model will use a parametricity graph of worlds which has only inclusions as morphisms. Let \(L\) denote the sub-parametricity graph of \(W\) with the same vertices and edges, but with the morphisms restricted to only the inclusions. Correspondingly, the squares of \(L\) are those squares \(\sigma\) of \(W\) such that \(\partial_0(\sigma)\) and \(\partial_1(\sigma)\) are inclusions.

The Cartesian closed structure and parametric limits in REL\(^L\) are defined in a manner completely analogous to those of \(REL^W\). The vertex portions are given in table 6.3. One can restrict the PG-functors \([\texttt{exp}], \quad \text{[comm], and [var]}\]: W \to REL to provide suitable interpretations of the imperative types as PG-functors \(L \to REL\). A model of PIA is produced using the parametricity graphs \(G = R^L\) and \(H = REL^L\) for simple types and types, respectively, where \(R\) denotes a small, sub-Cartesian closed parametricity graph of \(REL\) that contains the vertex \(\text{N}\).

Terms are interpreted as parametric transformations. Constructs of the polymorphic lambda calculus are interpreted by the units and co-units of the Cartesian closed structure and parametric limits as usual. The interpretations of most of the the imperative constants and constructs are given in table 6.4. Note that the \(\tilde{A}\) component of a parametric transformation \(\eta : \Gamma \vdash M : \tau\) is itself a parametric transformation (between PG-functors \(L \to REL\)). We use double subscripts \(\eta : \Gamma \vdash M : \tau\),\(_{W}\) to denote the \(W\)
component of the $\bar{A}$ component.

The use of variables in \texttt{deref} $v$ and $M:=v$ is rather straightforward, selecting the appropriate component from the interpretation of the variable to use. The interesting content of variables occurs in \texttt{new} $v$ \texttt{in} $C$ where a new variable is created. Intuitively, given a collection of labels $W$, the interpretation of \texttt{new} $v$ \texttt{in} $C$ selects a new label $r$ and runs $C$ in the larger world $W \cup \{r\}$, giving it the ability to update and read $r$. (An arbitrary function \texttt{next} such that $\texttt{next}(W) \notin W$ is used to select the new label, $r = \texttt{next}(W)$, and $i$ is used to denote the inclusion $i:W \to W \cup \{r\}$.)

To fix some notation, for any state $s \in S(W')$, we use $s[^n/r]$ to denote the state of $W' \cup \{r\}$ whose $l$-component is as follows.

$$\left( s[^n/r] \right)_l = \begin{cases} n & \text{if } l = r \\ s_l & \text{if } l \neq r \end{cases}$$

This changes the value of the $r$-component to $n$ if $r \in W'$ and adds an $r$ component with value $n$ otherwise.

The ability to update and read the new location is given by an argument $\langle \texttt{update}^W_r, \texttt{read}^W_r \rangle \in \llbracket \texttt{var} \rrbracket W \cup \{r\}$ as follows.

$$\texttt{update}^W_r \ n = s \mapsto s[^n/r]$$

$$\texttt{read}^W_r = s \mapsto s_r$$

The additional parameters $x \in \llbracket \eta \vdash \Gamma \rrbracket \bar{A}$ have to be mapped to the larger world, $W \cup \{r\}$. This is done using the image of the inclusion $i:W \to W \cup \{r\}$.

\begin{table}[h]
\centering
\begin{tabular}{|c|}
\hline
Table 6.4: Interpretation of Imperative Terms \\
\hline
\end{tabular}
\end{table}
under the PG-functor $[\eta \vdash \Gamma] \bar{A}$.

$$y = [\eta \vdash \Gamma] \bar{A} \ x \in [\eta \vdash \Gamma] \bar{A} \ W \ u \{r\}$$

$$[\eta ; \Gamma \vdash \text{new } v \ \text{in} \ C : \text{comm}] \bar{A} \ W \ x =
\begin{array}{c}
s \mapsto \phi_i([\eta ; \Gamma, v : \text{var} \vdash \text{C : comm}] \bar{A} \ W \ (y, \langle \text{update}_r^W, \text{read}_r^W \rangle \ s_i \ s_r)
\end{array}$$

In showing that $[\text{new } v \ \text{in} \ C] = [\eta ; \Gamma \vdash \text{new } v \ \text{in} \ C : \text{comm}] \bar{A}$ is a parametric transformation, one needs to show an arbitrary edge $R : W \leftrightarrow W'$ is respected. We make use of an auxiliary edge $\bar{R} : U \leftrightarrow U'$ (where $U = W \ u \{r\}$ and $U' = W' \ u \{r'\}$). The edge $\bar{R}$ is essentially $R$ extended to relate the new locations $r$ and $r'$ to each other.

$$s \left[ R_s \right] s' \iff \phi_i \left[ \bar{R}_s \right] \phi_\bar{R} s' \land s_r = s_{r'}$$

$$f \left[ \bar{R} \right] f' \iff \text{for all } s \left[ R_s \right] s', \ (f \ s)_r = (f' \ s')_{r'} \text{ and } \exists g \left[ \bar{R}_g \right] g' \text{ such that the following diagrams commute}$$

$$\begin{array}{ccc}
S(U) & \xrightarrow{f} & S(U') \\
\phi_i \downarrow & & \phi_i' \\
S(W) & \xrightarrow{g} & S(W')
\end{array} \quad \begin{array}{ccc}
S(U) & \xrightarrow{f'} & S(U') \\
\phi_\bar{R} \downarrow & & \phi_\bar{R}' \\
S(W) & \xrightarrow{g'} & S(W')
\end{array}$$

Observe that if any transformers $g$ and $g'$ exist such that $\phi_i \circ f = g \circ \phi_i$ and $\phi_\bar{R} \circ f' = g' \circ \phi_\bar{R}$, they are necessarily unique. The definition of $\bar{R}_g$ is more general than having $f = \tau_i(g)$ and $f' = \tau_{r'}(g')$. Recall that $\tau_i(g)$ necessarily leaves the new $r$ component unmodified, while $f$ is permitted to modify it, so long as $f'$ makes the same modification on $r'$.

It is straightforward to check that both $\text{read}_r^W \left[ [\text{exp}] \bar{R} \right] \text{read}_{r'}^W$ and $\text{update}_r^W n \left[ [\text{comm}] \bar{R} \right] \text{update}_{r'}^W n$ for all $n \in \mathbb{N}$. The parametricity of $[C] = [\eta ; \Gamma, v : \text{var} \vdash \text{C : comm}] \bar{A}$ implies that the following relationship holds for any $y \left[ [\eta \vdash \Gamma] \bar{A} \bar{R} \right] y'$.

$$[C] (y, \langle \text{update}_r^W, \text{read}_r^W \rangle \left[ \bar{R}_g \right] [C] (y', \langle \text{update}_{r'}^W, \text{read}_{r'}^W \rangle))$$

Since $[\text{new } v \ \text{in} \ C]_W x$ and $[\text{new } v \ \text{in} \ C]_W x'$ are the unique transformers lying under them when $y = [\eta \vdash \Gamma] \bar{A} \ x$ and $y' = [\eta \vdash \Gamma] \bar{A} \ x'$, this shows the following relationship holds, as desired.

$$[\text{new } v \ \text{in} \ C]_W x \left[ [\text{comm}] \bar{R} \right] [\text{new } v \ \text{in} \ C]_W x'$$

204
Much can be said about this model of the PIA. Not only do the parameterity criteria in $\textbf{REL}^\mathcal{L}$ ensure uniformity at polymorphic types, but there are few enough uniform operations in the model that they all arise in the interpretation of terms from the language. This is shown by way of traditional representation results. A simple example corresponds to the fact that there is a unique term (up to provable equality) of type $\forall X . X \rightarrow X$.

**Theorem 6.3**

The interpretation of $\forall X . X \rightarrow X$ in $\textbf{REL}^\mathcal{L}$ is terminal.

**Proof.** In order to show that $\llbracket \forall X . X \rightarrow X \rrbracket$ is terminal, we show that $\llbracket \forall X . X \rightarrow X \rrbracket W$ is a singleton set for every world $W$.

$$\llbracket \forall X . X \rightarrow X \rrbracket W = \left\{ \left( f_G^j . G(W') \rightarrow G(W') \right)_{G : \mathcal{L} \rightarrow \mathcal{R}, j : W \rightarrow W'} \mid \text{for any edge } G : G \leftrightarrow G' \text{ and square in } \mathcal{L} \text{ of the shape on the left there is a square in } \textbf{REL} \text{ of the shape on the right} \right\}$$

$$\begin{array}{c}
W \xrightarrow{j} W' \\
I_W \quad R \quad G(R) \quad G(R) \quad G'(W'') \\
W \xrightarrow{j'} W'' \\
\end{array}$$

We consider an arbitrary tuple $\langle f_G^j \rangle \in \llbracket \forall X . X \rightarrow X \rrbracket W$. For any $G : \mathcal{L} \rightarrow \mathcal{R}$ and $j : W \rightarrow W'$, we show $f_G^j$ is the identity on $G(W')$ as follows. For an arbitrary $a \in G(W')$, the edge $g^a_G : G \leftrightarrow G$ is defined at any $R : U \leftrightarrow U'$ of $\mathcal{L}$ by the following relation.

$$g^a_G(E, R) = \left\{ (G i a, G i' a) \mid i : W' \rightarrow U, i' : W' \rightarrow U' \right\}$$

(Recall that the $(s, -)$ and $(t, -)$ components of an edge of $\textbf{REL}^\mathcal{L}$ agree with the source and target vertices, respectively.)

Since the only morphisms are inclusions, there is at most one pair of morphisms $i : W \rightarrow U, i' : W' \rightarrow U'$. (There may not be such a pair, in which case the relation is empty.) In particular, $g^a_G(I_W, I_W) = \{(a, a)\}$. Note that there is a square of the following shape in $\textbf{REL}$.

$$\begin{array}{l}
G(W') \xrightarrow{f_G^j} G(W') \\
G(I_W) \xrightarrow{f_G^j} G(I_W) \\
G(W') \xrightarrow{f_G^j} G(W') \\
\end{array}$$
Thus, it follows that \( f^G_t(a) = a \). 

The other representation results which one might expect in models of predicative polymorphic lambda calculus, such as \( \forall X.X \) being initial and \( \forall X.(A \rightarrow B \rightarrow X) \rightarrow X \) being the product of \( A \) and \( B \), can similarly be shown to hold for this model of the polymorphic imperative lambda calculus.

This model also treats the imperative features of the language in a reasonable manner. The test equivalences of [OT95] can similarly be shown to hold in the \textbf{REL}^C model. Note this is still not an ideal treatment of imperative programming, as it has the same weaknesses as the O’Hearn-Tennent model, such as the presence of temporary state changes (or snapshot). O’Hearn and Reynolds have shown that it is possible to avoid snapshot with a linear treatment of state [RO00]. Linearity will be discussed further in the following chapter.

Since the obvious changes one makes to worlds — allocating new labels — are only inclusions, it is natural to think of inclusions as the natural choice. One may wonder if there was a particular reason for selecting injections as morphisms in the first place rather than inclusions. In the categorical model, there is a need to have injections as morphisms, as pointed out in [OT92]. To show that \([\text{new } v \in C]\) preserves a morphism \( j: W \rightarrow W' \), one relies on \([C]\) preserving \( j: W' \{r\} \rightarrow W' \{r'\} \) given as follows.

\[
\hat{j}(x) = \begin{cases} 
  j(x) & \text{if } x \in W \\
  r' & \text{if } x = r
\end{cases}
\]

Even if \( j \) is an inclusion, \( \hat{j} \) need not be an inclusion since \( r = \text{next}(W) \) may not equal \( r' = \text{next}(W') \), that is, different labels for the new variable may have been selected at the different worlds. Thus, in order for \([\text{new } v \in C]\) to preserve inclusions, \([C]\) must preserve injections. In a categorical model, this means that injections are needed as morphisms so that naturality can ensure this uniformity.

There is no such need for injections as morphisms in parametricity graph models. There is still the need for \([C]\) to preserve injections, but by having edges corresponding to injections, this is ensured. Parametricity graphs have the additional component of edges in addition to morphism. One can view these as “properties to be respected” in contrast to the view of morphisms as “operations that can be performed”. In categories, arrows play both roles. Parametricity graphs (or more generally, reflexive graph categories) allow one to strengthen uniformity constraints by adding edges without having to increase the potential operations. It is not surprising that there are

206
relationships that one desires as correspondences, but not as computations, such as the injections mentioned above.

When introducing PIA, we pointed out that conditionals are only available at \texttt{comm} types. Since it is not applicable for all phrase types, PIA is not truly an Algol-like language in Reynolds’ sense [Rey81]. The reason for our imposing this limitation was to avoid awkward and unenlightening restrictions on the models. One might wonder if this restriction is significant to the rest of the analysis of this chapter.

A polymorphic Algol-like language PAL can be defined just like PIA, except with a conditional construct available at all phrase types.

\[
\eta \triangleright \Gamma \vdash M : \text{exp} \quad \eta \triangleright \Gamma \vdash N_1 : \phi \quad \eta \triangleright \Gamma \vdash N_2 : \phi \\
\eta \triangleright \Gamma \vdash \text{if } M = 0 \text{ then } N_1 \text{ else } N_2 : \phi
\]

Just as in PIA, in PAL there is a unique term (up to provable equality) of type \(\forall X. X \Rightarrow X\). Intuitively, for an unknown type \(X\) and any input \(a\) of that type, if one could produce some output of type \(X\) other than \(a\) by using \(\text{if } M = 0 \text{ then } N_1 \text{ else } N_2\), then one could have used either \(N_1\) or \(N_2\) to produce something other than \(a\). While the terminal object representation result is still expected, some other representation results are no longer reasonable since terms of arbitrary type can depend on the store. For instance, the type \(\forall X.(A \Rightarrow X) \Rightarrow (A \Rightarrow X) \Rightarrow X\) contains more than just two copies of \(A\). Using conditionals, two input functions of type \(A \Rightarrow X\) can be combined in ways depending on the store, thus many additional functions can be produced.

The \texttt{REL}^W and \texttt{REL}^C models of PIA can be modified to produce models of PAL by restricting to sub-parametricity graphs of PG-functors “such that conditionals can be interpreted” as alluded to in section 6.1. This has no significant impact on the interpretation of the type \(\forall X. X \Rightarrow X\) mentioned in sections 6.2 and 6.3. The interpretation in \texttt{REL}^C contains only the polymorphic identity function while in \texttt{REL}^W, \([\forall X. X \Rightarrow X]W\) contains additional elements for all the permutations of \(W\).

This chapter has focused on parametricity and imperative programming. We introduced PIA, a polymorphic imperative programming language, and produced a pre-sheaf-like parametricity graph model for it. Not all parametricity graph models of PIA satisfy the traditional representation results. Previously proposed categorical and parametricity graph models of Idealized Algol (with the apparent extensions to PIA) do not satisfy the traditional representation results since they do contain intuitively uniform families which are not definable in the calculus. The parametricity graph model
we describe does capture categorically the intuitive description of how the language works. This model does satisfy the traditional representation results, showing that imperative programming features can coexist with the traditional representation results. This is not the case for another feature typical of Algol but missing from PIA — recursion. Recursion does require major alterations to the models, which leads one to consider linear logic. This is the focus of the next chapter.
Chapter 7

Linear Parametricity

A significant feature of many programming languages is recursion. A defining characteristic of recursion is the presence of fixed point operators. In an effort to describe parametric settings which support recursion, we are led to consider a polymorphic linear lambda calculus. We describe here the structures appropriate for producing parametric models and provide a few examples. These models are settings where parametricity provides a reasonable notion of uniformity and they support fixed point operators.

The canonical categorical setting for modeling a lambda calculus with recursion is the Cartesian closed category \( \text{pcpo} \). The objects are pointed CPOs, that is, complete partial orders having a least element, \( \bot \). The arrows of \( \text{pcpo} \) are continuous functions. The category \( \text{pcpo} \) has a fixed point operator \( Y_A : A \to A \) for every pointed CPO \( A \) given by computing the least upper bound of repeated applications.

\[
Y_A(f) = \bigsqcup_n f^n(\bot)
\]

If we wished to extend this to a parametric model of a polymorphic lambda calculus with recursion, the apparent approach would be to identify an appropriate notion of edges over \( \text{pcpo} \). The edges should be chosen to make the family \( \{Y_A\}_A \) into a parametric transformation because the family of fixed point operators is intuitively uniform. One should be free to build polymorphic functions using fixed points. Obvious candidates, such as complete relations or spans, must be rejected because the fixed point operator fails to be a parametric transformation. The reflexive graph category \( \text{PCPO} \), in which edges are pointed complete relations (that is, relating \( \bot \) to \( \bot \)) and squares are as in \( \text{REL} \), does not have this problem. For any pointed complete relation \( R : A \to B \), if \( f[R \Rightarrow R] g \), then \( f^n(\bot) \left[ R \right] g^n(\bot) \) for all \( n \). (The pointed assumption on the relation is precisely the base case,
\( \bot \left[ R \right] \bot \). By virtue of \( R \) being complete, \( Y_A(f) = \bigcup_n f^n(\bot) \) is related by \( R \) to \( \bigcup_n g^n(\bot) = Y_B(g) \).

While it is known that the category \text{pcpo} does not have arbitrary small limits, the parametric limit \( \text{PCPO}^G \rightarrow \text{PCPO} \) does exist in the 2-category \text{RG} for any small, discrete reflexive graph category \( G \). This restriction to a discrete source is not a problem, since the interpretation of the polymorphic lambda calculus only uses parametric limits out of discrete reflexive graph categories.

The fact that this is in the 2-category \text{RG} rather than \text{PG} is a more serious concern. The reflexive graph category \text{PCPO} is not a parametricity graph. (It is not subsumptive, because the graph of an arbitrary continuous function need not be a pointed relation.) It is not apparent that the preservation of edges in \text{PCPO} provides a good notion of uniformity. (It is not as strong as naturality.) Most of the traditional representation results fail. For instance, \( \left[ \forall X. X \Rightarrow X \right] \) (a two-element CPO) is not terminal and \( \left[ \forall X.X \right] \) (a one element CPO) is not initial.

The failure in producing an adequate parametric model is not limited to the setting mentioned above, but rather is representative of a tension between recursion and parametric polymorphism. Recall that the parametricity of System P ensures the existence of coproducts (which does rely on impredicativity). Huwig and Poigne [HP90] have shown that any Cartesian closed category with coproducts and recursion is trivial (in that it has a single object and a single arrow). Since it is desirable to keep the special case of impredicative polymorphism available as an instance of our development of parametricity, we look for something else to give, permitting recursion and parametricity to coexist.

Thus, we shall not consider Cartesian closed models. A sum-like construction typically used in \text{pcpo} is that of the coalesced sum \( A \oplus B \). This is the disjoint union of the pointed CPOs \( A \) and \( B \) where the two bottom elements are identified. The coalesced sum is the coproduct in the linear category \( \text{cpo}_\perp \) of pointed CPOs and strict (that is, bottom preserving) continuous functions, a sub-category of \text{pcpo} having the same objects. The fact that the disjoint union is the coproduct in a linear category motivated Plotkin to propose a polymorphic linear lambda calculus as a setting for combining parametricity and recursion [Pl93]. Plotkin’s proposal led to considering parametricity for linear categories of domains.

O’Hearn and Reynolds have also considered parametricity in a linear setting [RO00]. Their motivation for using linearity was to capture the irreversibility of state changes in an Algol-like language. They modeled
their polymorphic linear lambda calculus using complete pointed relations over $\text{cpo}_L$. They remarked that it is not clear how to define the relational action for the smash product. The obvious definition which one expects to use does not yield a complete relation in general. We are able to obtain the right relation by taking the completion of this candidate relation. Taking the completion of a relation leads to additional elements being related, and it is not immediately clear why this should be the correct choice. An alternative approach is to consider a different setting due to O’Hearn [O’H], using spans as edges rather than complete relations.

These two approaches lead to different treatments of the smash product, but we find that similar results are obtained. One would like to know if there are significant differences in the resulting models, that is, do spans and relations produce the same class of polymorphic functions. This remains as an open problem.

## 7.1 A Polymorphic Linear Lambda Calculus

We introduce a polymorphic linear lambda calculus. This will provide a framework for discussing parametric models which support recursion. Therefore, we also define the structures in parametricity graphs appropriate for modeling this linear calculus.

Linear logic was introduced by Girard [Gir87] as a resource sensitive logic. The implication $A \rightarrow B$ of linear logic is intuitively understood as consuming $A$ to produce $B$. The resource $A$ will no longer be available after having been used to produce $B$. This notion of implication is reasonable for dealing with resources rather than stable truths. For instance, after using a dollar ($A$) to purchase a ticket to the fun house ($B$), one does not still have the dollar.

The implication $C \Rightarrow D$ of intuitionistic logic applies to stable truths, asserting that if $C$ is true, then $D$ is true as well. The fact that $C$ is true remains valid after being used to deduce $D$. Even in a resource sensitive logic, there is still a role for stable truths. Purchasing a ticket for the roller coaster requires not only a dollar, but also that the purchaser is at least 48 inches tall. While the dollar is no longer available to purchase other tickets, the height requirement does remain satisfied.

The linear lambda calculus we consider here is based on the formalism of Dual Intuitionistic Linear Logic (DILL) from Barber and Plotkin [BP97]. Contexts $\Gamma \mid \Delta$ are split into two zones, the linear zone $\Delta$ and the intuitionistic zone $\Gamma$. Variables declared in the linear zone $\Delta$ occur precisely once in
a term, whereas variables declared in the intuitionistic zone \( \Gamma \) may be used arbitrarily many times. The type system of the polymorphic linear lambda calculus (PLLC) we consider is given as follows.

\[
\begin{align*}
\tau & := X \mid I \mid \tau_1 \otimes \tau_2 \mid \tau_1 \to \tau_2 \mid !\tau \\
\phi & := X \mid I \mid \phi_1 \otimes \phi_2 \mid \phi_1 \to \phi_2 \mid !\phi \mid \forall X.\phi \mid \exists X.\phi
\end{align*}
\]

The modality ! is used to indicate a type whose elements may be copied arbitrarily often. In effect, terms of type !\( \phi \) are those terms which can be substituted for an intuitionistic assumption \( x: \phi \).

As was the case for the predicative calculus (section 2.1), there are three kinds of judgments in PLLC.

\[
\begin{align*}
\eta \vdash \alpha & \quad \text{type judgment} \\
\eta \vdash \tau \text{ simple} & \quad \text{simplicity judgment} \\
\eta ; \Gamma \mid \Delta \vdash M : \phi & \quad \text{term judgment}
\end{align*}
\]

In all cases, \( \eta \) is a typing context (a finite sequence of type variables). A context of PLLC consists of two finite sequences of type assumptions \( x_i: \phi_i \) such that no term variable is repeated. These sequences are separated by a vertical bar, as in \( \Gamma \mid \Delta \). The meta-variable \( \alpha \) for type judgments ranges over types and contexts.

A type judgment \( \eta \vdash \alpha \) holds in PLLC if all the type variables that occur free in \( \alpha \) are contained in \( \eta \). The simplicity judgment \( \eta \vdash \tau \text{ simple} \) holds whenever \( \eta \vdash \tau \) and \( \tau \) is a simple type. The term judgments of PLLC are those that are derivable from the rules listed in Table 7.1 (page 213).

In the rules \{\text{-elim}\}, \{\otimes\text{-intro}\}, \{\otimes\text{-elim}\}, \{\to\text{-elim}\} and \{\exists\text{-elim}\}, the convention that \( \Delta' \) is a merge of \( \Delta_1 \) and \( \Delta_2 \) is presumed. This means that the variables (and associated types) of \( \Delta' \) are precisely those of \( \Delta_1 \) and \( \Delta_2 \), with their orders intermingled.

A distinction between the intuitionistic zone \( \Gamma \) and the linear zone \( \Delta \) of a context can be seen by considering those term forming rules that use multiple sub-terms (such as \{\otimes\text{-intro}\}). While different linear zones are used in the hypotheses for the sub-terms, the same intuitionistic zone is used in both hypotheses. Therefore, variables from the intuitionistic zone may appear repeated in terms. This repetition is crucial for the structural rule of contraction. The structural rule of weakening indicates that variables do not have to appear in the term. PLLC supports both contraction and weakening of intuitionistic assumptions. The following contraction and weakening rules
\[
\begin{align*}
\eta \vdash x_1; \cdots; x_m; \phi_m | \emptyset, & \quad 1 \leq j \leq m \quad \{\text{Int}_\text{var}\} \\
\eta; x_1; \cdots; x_m; \phi_m | \emptyset \vdash x_j; \phi_j & \\
\eta \vdash \Gamma \mid x; \phi \quad \{\text{Lin}_\text{var}\} \\
\eta; \Gamma \mid x; \phi \vdash x; \phi & \\
\eta \vdash \Gamma \mid \emptyset \quad \{\text{I-intro}\} \\
\eta; \Gamma \mid \emptyset \vdash *: I & \\
\eta; \Gamma \mid \Delta_1 \vdash M; I & \quad \eta; \Gamma \mid \Delta_2 \vdash N; \phi \quad \{\text{I-elim}\} \\
\eta; \Gamma \mid \Delta' \vdash \text{let } * \text{ be } M \text{ in } N; \phi & \\
\eta; \Gamma \mid \Delta_1 \vdash M; \phi_1 & \quad \eta; \Gamma \mid \Delta_2 \vdash N; \phi_2 \quad \{\otimes\text{-intro}\} \\
\eta; \Gamma \mid \Delta' \vdash M \otimes N; \phi_1 \otimes \phi_2 & \\
\eta; \Gamma \mid \Delta_1 \vdash M; \phi_1 \otimes \phi_2 & \quad \eta; \Gamma \mid x; \phi_1, y; \phi_2, \Delta_2 \vdash N; \phi_3 \quad \{\otimes\text{-elim}\} \\
\eta; \Gamma \mid \Delta' \vdash \text{let } x \otimes y \text{ be } M \text{ in } N; \phi_3 & \\
\eta; \Gamma \mid \Delta, x; \phi_1 \vdash M; \phi_2 & \\
\eta; \Gamma \mid \Delta \vdash (\lambda x; \phi_1.M); \phi_1 \rightarrow \phi_2 \quad \{\rightarrow\text{-intro}\} \\
\eta; \Gamma \mid \Delta_1 \vdash M; \phi_1 \rightarrow \phi_2 & \quad \eta; \Gamma \mid \Delta_2 \vdash N; \phi_1 \quad \{\rightarrow\text{-elim}\} \\
\eta; \Gamma \mid \emptyset \vdash M; \phi & \\
\eta; \Gamma \mid \emptyset \vdash !M; \phi & \quad \{!\text{-intro}\} \\
\eta; \Gamma \mid \Delta_1 \vdash M; !\phi_1 & \quad \eta; \Gamma, x; \phi_1 \mid \Delta_2 \vdash N; \phi_2 \quad \{!\text{-elim}\} \\
\eta; \Gamma \mid \Delta' \vdash \text{let } x \text{ be } M \text{ in } N; \phi_2 & \\
\eta, X; \Gamma \mid \Delta \vdash M; \phi & \quad \{\forall\text{-intro}\} \\
\eta; \Gamma \mid \Delta \vdash (\Lambda X.M); \forall X.\phi & \\
\eta; \Gamma \mid \Delta \vdash \forall X.\phi & \quad \eta \vdash \tau \text{ simple} \quad \{\forall\text{-elim}\} \\
\eta; \Gamma \mid \Delta \vdash M[\tau]; \phi[\tau/X] & \\
\eta \vdash \tau \text{ simple} & \quad \eta; \Gamma \mid \Delta \vdash M[\tau/X] \quad \{\exists\text{-intro}\} \\
\eta; \Gamma \mid \Delta \vdash (\text{pack } \tau \text{ with } M); \exists X.\phi & \\
\eta; \Gamma \mid \Delta_1 \vdash M; \exists X.\phi_1 & \quad \eta, X; \Gamma \mid x; \phi_1, \Delta_2 \vdash N; \phi_2 & \quad \eta \vdash \phi_2 \quad \{\exists\text{-elim}\} \\
\eta; \Gamma \mid \Delta' \vdash \text{(open } M \text{ as } (X, x) \text{ in } N); \phi_2 & \\
\end{align*}
\]

Table 7.1: Term rules for PLLC

213
are admissible in PLLC.

$$\eta ; \Gamma, x: \phi_1, y: \phi_1 \mid \Delta \vdash M: \phi_2$$

Together, contraction and weakening imply that variables from the intuitionistic zone may be reused arbitrarily many times. A key feature of linear logic is that contraction and weakening are not universally available. In PLLC, the linear zone supports neither contraction nor weakening. This leads to the characterization that variables from the linear zone are used precisely once.

The !-elim rule effectively allows variables of type $\phi$ from the intuitionistic zone to be instantiated by terms of type $!\phi$. This is the only rule where variables from the intuitionistic zone are removed. By passing through a linear variable of type $!\phi$, one can effectively abstract away an intuitionistic variable of type $\phi$, as in the following derivation.

$$\eta ; \Gamma, x: \phi_1 \mid \Delta \vdash M: \phi_2 \quad \eta ; \Gamma \mid y: !\phi_1 \vdash y: !\phi_1$$

$$\eta ; \Gamma \mid \Delta \vdash \lambda y: !\phi_1. \text{let } x \text{ be } y \text{ in } M: !\phi_1 \multimap \phi_2$$

The type $!\phi_1 \multimap \phi_2$ can be thought of as the type of non-linear functions from $\phi_1$ to $\phi_2$. These non-linear functions can use an argument of type $\phi_1$ arbitrarily often. We shall use the notation $\phi_1 \Rightarrow \phi_2$ as an abbreviation for the type $!\phi_1 \multimap \phi_2$ to suggest this analogy with non-linear functions. (For more discussion of the function-like usage of this type, the interested reader is referred to the analogous construction in DILL [BP97].)

Categorical models for linear lambda calculi are given using certain symmetric monoidal closed categories [See87, BBdPH93, Ben94]. Since the relevant properties are described 2-categorically, they can be carried over to parametricity graphs. We briefly mention the necessary structure.

**Definition 7.1**

A symmetric monoidal closed (SMC) parametricity graph consists of

- a parametricity graph $G$

- a designated vertex $I$

- a pair of PG-functors $\otimes: G \times G \rightarrow G$ and $\rightarrow: G^{op} \times G \rightarrow G$ such that for every vertex $B$, $\otimes B$ is the left adjoint to $B \rightarrow -$

- and parametric transformations $\lambda$, $\rho$, $\alpha$, $\sigma$ giving the :
left cancellation \( \lambda_A : I \otimes A \to A \)
right cancellation \( \rho_A : A \otimes I \to A \)
associativity \( \alpha_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C \)
and symmetry \( \sigma_{A,B} : A \otimes B \to B \otimes A \)
of the tensor product \( \otimes \), subject to the standard equational axioms
(see [Mac71] for instance).

A monoidal PG-functor from one SMC parametricity graph \( (G, I, \otimes, \leadsto) \) to
another \( (\bar{G}, \bar{I}, \bar{\otimes}, \bar{\leadsto}) \) is a PG-functor \( G \to \bar{G} \) together with a specified
parametric isomorphism \( m_{A,B} : FA \otimes FB \to F(A \otimes B) \) and an isomorphism
\( m' : \bar{I} \to F(I) \) which commute with \( \lambda, \rho, \alpha \) and \( \sigma \).

A monoidal parametric transformation is a parametric transformation
which commutes with the relevant \( m \)'s and \( m' \)'s.

Multiple tensor products are typically presented without bracketing,
with implicit uses of associativity isomorphisms allowing one to treat a mul-
tiple tensor product \( A \otimes B \otimes C \otimes D \) as bracketed however is convenient at the
time. (Note that every Cartesian closed parametricity graph is symmet-
tric monoidal closed, using the terminal object \( 1 \), Cartesian product \( \times \) and
exponentiation \( \Rightarrow \) for the SMC structure.) “Monoidal” can be applied to
any other 2-categorical construction (such as adjunctions) in the obvious
manner, using the 2-category of SMC parametricity graphs, monoidal PG-
functors and monoidal parametric transformations. We use this language
in defining structures appropriate for producing parametric models of linear
lambda calculi.

**Definition 7.2**

An LNL setting consists of:

- an SMC parametricity graph \( (G, I, \otimes, \leadsto) \)
- a Cartesian closed parametricity graph \( (\bar{G}, 1, \times, \Rightarrow) \)
- a monoidal parametric adjunction \( G \xleftarrow{F} \bar{G} \).

The above definition is an immediate translation of Benton’s concept
of a Linear/Non-Linear (LNL) model [Ben94] from \textbf{CAT} to \textbf{PG}. LNL
models provide sufficient structure to model a linear lambda calculus in
the SMC category, or in other words, the SMC category is a linear cat-
egory\(^1\) [BDPH93, Ben94]. A similar situation occurs with parametricity

\(^1\) Conversely, a category is a linear category if and only if it is the SMC portion of an
LNL model
graphs — the SMC parametricity graph has the analogous structure we’ll call a *linear parametricity graph*. This includes a monodial parametric co-monad \( (L = (F \circ G), \epsilon; L \to \text{ID}, \delta; L \to (L \circ L)) \), which we shall use to model \(!\), and parametric transformations \( e; L \to \Delta_t \) and \( d; L \to L \otimes L \), which allow discarding and copying of \(!\) types, subject to some commuting criteria.

**Definition 7.3**

A linear parametricity graph is a SMC parametricity graph \( G \) together with a monodial parametric co-monad \( (L, \epsilon, \delta) \) on \( G \) such that there exists an LNL setting \( (G, \bar{G}, F, G) \) with \( L = F \circ G \).

The Cartesian closed parametricity graph \( \bar{G} \) of an LNL setting will not be relevant to interpreting linear lambda calculi other than as a convenient manner of asserting desired properties of the co-monad \( L = F \circ G \). We shall primarily talk of LNL settings in the remainder of this chapter, bearing in mind that the particular choice of Cartesian closed parametricity graph is irrelevant.

The ability to readily transform constructions on categories to constructions on parametricity graphs is a strength of the framework developed in this thesis. This ability is made possible by the fact that parametricity graphs form a 2-category.

To produce a parametric model of PLLC, we use a sub-LNL setting to model simple types. There is an immediate analogue of sub-SMC categories (or sub-Cartesian closed categories) to sub-SMC parametricity graphs. An SMC parametricity graph \( G' \) is a sub-SMC parametricity graph of \( G \) if \( G' \) is a sub-parametricity graph of \( G \) and all the PG-functors and parametric transformations giving the SMC structure on \( G' \) are simply restrictions of the corresponding PG-functors and parametric transformations of \( G \). A similar approach can be taken for LNL settings.

**Definition 7.4**

An LNL setting \( (G', G', F', G') \) is a sub-LNL setting of another LNL setting \( (G, \bar{G}, F, G) \) provided

- \( G' \) is a sub-SMC parametricity graph of \( G \)
- \( \bar{G}' \) is a sub-Cartesian closed parametricity graph of \( \bar{G} \)
- and the adjunction \( \begin{array}{c} \text{G} \leftarrow \hspace{1cm} \text{G} \rightarrow \text{G'} \\ \hspace{1cm} \bar{G} \end{array} \begin{array}{c} F \hspace{1cm} \bar{F} \end{array} \) restricts to \( \begin{array}{c} \text{G} \leftarrow \hspace{1cm} \text{G} \rightarrow \text{G}' \\ \hspace{1cm} \bar{G}' \end{array} \begin{array}{c} F \hspace{1cm} \bar{F}' \end{array} \).

Since the adjoint PG-functors of the sub-LNL setting are restrictions of the adjoint PG-functors in the larger LNL setting, we typically use the same names for both parametric adjunctions.
Given a sub-LNL setting \((\mathbf{H}, \mathbf{H}, F, G)\) of \((\mathbf{G}, \mathbf{G}, F, G)\) such that the parametric limit and parametric colimit PG-functors \(\text{Lim}, \text{Colim}: \mathbf{G} \Rightarrow \mathbf{G}\) exist, one can give a model of PLLC in a manner analogous to the model of the predicative calculus described in section 3.3. Judgments of PLLC shall be interpreted as follows.

The co-monad \(L = F \circ G\) is used in the interpretation of the intuitionistic zone of a context. We use \(L\) since the adjunction provides the following parametric transformations.

\[
e: L \rightarrow \Delta_I \quad \text{and} \quad d: L \rightarrow L \otimes L
\]

These parametric transformations are given by \(e_A = (m^I)^{-1} \circ F(\text{discard}_{G(A)})\) and \(d_A = (m_{G(A), G(A)})^{-1} \circ F(\text{copy}_{G(A)})\). These are defined using the canonical \(\text{discard}_B: B \rightarrow 1\) and \(\text{copy}_B: B \rightarrow B \times B\) maps from the Cartesian closed structure of \(\mathbf{G}\). The parametric transformations \(e\) and \(d\) permit the discarding and copying of elements of \(L(A)\) (for any vertex \(A\)).

The interpretation of type judgments are given as follows.

\[
[\eta \vdash X] = \Pi \quad \text{(the appropriate projection)}
\]

\[
[\eta \vdash I] = \Delta_I \quad \text{(the constant functor 1)}
\]

\[
[\eta \vdash \phi_1 \otimes \phi_2] = \otimes \circ \langle[\eta \vdash \phi_1], [\eta \vdash \phi_2]\rangle
\]

\[
[\eta \vdash \phi_1 \rightarrow \phi_2] = \rightarrow \circ \langle[\eta \vdash \phi_1], [\eta \vdash \phi_2]\rangle
\]

\[
[\eta \vdash \forall x. \phi] = \forall_x [\eta, x \vdash \phi]
\]

\[
[\eta \vdash \exists x. \phi] = \exists_x [\eta, x \vdash \phi]
\]

\[
[\eta \vdash \Gamma | \Delta] = L([\Gamma]^{\eta}) \otimes [\Delta]^{\eta}
\]

where \([x_1: \phi_1, \ldots, x_m: \phi_m]^{\eta} = [\eta \vdash \phi_1] \otimes \cdots \otimes [\eta \vdash \phi_m]\)

and \([\emptyset]^{\eta} = \Delta_I\)

Observe that for any simple type \(\tau\), the interpretation of the type judgment
\( \eta \vdash \tau \) will map into \( \mathbf{H} \) (since \( \otimes, \rhd, \) and \( L \) restrict to PG-functors on \( \mathbf{H} \)). The interpretation of simplicity judgments comes from restricting the professed codomain of the interpretation of the corresponding type judgment.

\[
\bar{[\eta \vdash \tau \text{ simple}]} \bar{R} = \bar{[\eta \vdash \tau]} \bar{R}
\]

Since variables from the intuitionistic zone of a context may appear arbitrarily often in a term, the parametric transformations \( e \) and \( d \) are used frequently in interpreting term judgment. The interpretations of many \{elim\} rules (and \{intro\}) make use of a parametric isomorphism

\[
\sigma : \bar{[\Delta']} \longrightarrow \bar{[\Delta_1]} \otimes \bar{[\Delta_2]}
\]

The exact structure of symmetry transformations (and inverses of cancellation transformations, if either \( \Delta_1 \) or \( \Delta_2 \) is empty) that define \( \sigma \) depends on the exact ordering of \( \Delta' \), \( \Delta_1 \) and \( \Delta_2 \). The inductive definition of term judgments is given in Table 7.2 (page 219).

Interpretations corresponding to the variable rules of PLLC are significantly more complex than the interpretation of the variable rule in the predicative calculus. The step corresponding to \{linvar\} uses the monoidal structure of the co-monad \( L \) to isolate the variable of interest before stripping away unwanted variables.

\[
L([\Gamma_1] \otimes [\phi] \otimes [\Gamma_2]) \otimes I \xrightarrow{m^{-1} \otimes \text{id}} L([\Gamma_1] \otimes [\phi]) \otimes L([\Gamma_2]) \otimes I \xrightarrow{m^{-1} \otimes \rho} L([\Gamma_1]) \otimes L([\phi]) \otimes L([\Gamma_2]) \xrightarrow{e \otimes \epsilon \otimes e} [\phi] \xrightarrow{\lambda \otimes I} I \otimes [\phi] \otimes I
\]

The step corresponding to \{linvar\} is simpler as one need only discard the intuitionistic zone of the context.

\[
l([\Gamma]) \otimes [\phi] \xrightarrow{e \otimes \text{id}} I \otimes [\phi] \xrightarrow{\lambda} [\phi]
\]

Similar considerations and complications are made throughout the interpretation.

In looking for LNL settings that one could use to model PLLC, we note
\[
\begin{align*}
\llbracket \eta \mid \Gamma, x : \phi, \Gamma_2 \mid \emptyset \vdash x : \phi \rrbracket &= \rho \circ (\lambda \otimes \text{id}) \circ (e \otimes e) \circ (m^{-1} \otimes \rho) \circ (m^{-1} \otimes \text{id}) \\
\llbracket \eta \mid \Gamma \mid x : \phi \vdash x : \phi \rrbracket &= \lambda \circ (e \otimes \text{id}) \\
\llbracket \eta \mid \Gamma \mid \emptyset \vdash *:1 \rrbracket &= e \circ \rho \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash * \text{ be } M \text{ in } N: \phi \rrbracket &= \llbracket \eta \mid \Gamma \mid \Delta_2 \vdash N: \phi \rrbracket \circ \left( (\rho \circ [\eta \mid \Gamma \mid \Delta_1 \vdash M: I]) \otimes \text{id} \right) \circ (\text{id} \otimes \delta) \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash \lambda x : \phi_1.M : \phi_1 \rightarrow \phi_2 \rrbracket &= \text{ curry}(\llbracket \eta \mid \Gamma \mid \Delta, x : \phi_1 \vdash \phi_2 \rrbracket) \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash M \otimes N : \phi_2 \rrbracket &= \text{ ap } \left( \llbracket \eta \mid \Gamma \mid \Delta_1 \vdash M : \phi_1 \rightarrow \phi_2 \rrbracket \otimes \llbracket \eta \mid \Gamma \mid \Delta_2 \vdash N : \phi_1 \rrbracket \right) \circ (\text{id} \otimes \delta) \\
\llbracket \eta \mid \Gamma \mid \emptyset \vdash !M : \phi_1 \rrbracket &= \text{ L}(\llbracket \eta \mid \Gamma \mid \emptyset \vdash M : \phi_1 \rrbracket) \circ \text{ L}(\rho^{-1}) \circ \delta \circ \rho \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash \lambda x : \phi_1.M \mid \forall X. \phi \rrbracket &= \text{ L}(\llbracket \eta \mid \Gamma \mid \Delta \vdash M : \phi \rrbracket) \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash M \upharpoonright [\tau] : \phi \upharpoonright [\tau/x] \rrbracket &= \omega_{\text{[\tau/x simple]}} \circ \llbracket \eta \mid \Gamma \mid \Delta \vdash M : \forall X. \phi \rrbracket \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash \text{ (pack } \tau, \text{ with } M) : \exists X. \phi \rrbracket &= \mu_{\text{[\tau/x simple]}} \circ \llbracket \eta, X \mid \Gamma \mid \Delta \vdash M : \phi \upharpoonright [\tau/x] \rrbracket \\
\llbracket \eta \mid \Gamma \mid \Delta \vdash \text{ (open } M, \text{ as } (X, x), \text{ in } N) : \phi_2 \rrbracket &= \nabla(\llbracket \eta, X \mid \Gamma \mid \Delta_1 \vdash \exists X. \phi_1 \rrbracket) \circ (\text{id} \otimes \llbracket \eta \mid \Gamma \mid \Delta_2 \vdash N : \phi_2 \rrbracket) \\
\end{align*}
\]

Table 7.2: Interpretation of PLLC terms
that every Cartesian closed parametricity graph gives an LNL setting where
the adjunction is trivial \( \text{G} \xrightarrow{\text{ID}} \text{ID} \xleftarrow{\text{ID}} \text{G} \). Hence, any model of the predicative
calculus gives rise to an analogous model of PLLC. More interesting are
settings which model PLLC where the symmetric monodial structure is not
Cartesian.

7.2 Domain Models

Domains have played an important role in programming semantics for the
past 30 years, dating back to Scott’s modeling of typed and untyped lambda
calculi with domains [Scö69, Scö70]. Ordered sets arise naturally in situations
where non-termination is possible, where a larger element terminates
(is defined) more often than a smaller element. Having such an ordering of
the elements is crucial for modeling recursion. A fixed point operator is typ-
ically handled by making a series of approximations, starting with a totally
undefined element and gradually becoming more and more defined. The
fixed point is given by the limit of those approximations. Various categories
of complete partial orders have been used to model many programming lan-
guages. Some of these provide non-Cartesian SMC categories. Here we
consider one such category and describe two LNL settings using different
notions of edges.

A canonical example of a SMC category is the category \( \mathbf{cpo}_\bot \) of complete
partial orders necessarily having a bottom element, \( \bot \). The arrows of \( \mathbf{cpo}_\bot \)
are strict (that is, bottom-preserving) continuous functions. The collection
of all strict continuous functions \( f : A \to B \) is itself a pointed CPO under
the pointwise ordering. This internal hom will be denoted as \( A \to B \). Its
left adjoint \( A \otimes B \) is given by the smash product — pairs \( (a, b) \in A \times B \)
where all pairs with at least one component \( \bot \) are equated. For any \( a \in A \)
and \( b \in B \), we use \( [a, b] \) to denote the equivalence class of the pair \( (a, b) \).
The unit of \( \otimes \) is the two-point CPO \( I = \{ \bot, \top \} \) with \( \bot \subseteq \top \). This makes
\( \mathbf{cpo}_\bot \) into a symmetric monoidal closed category. The arrow portions of
the functors \( \to : \mathbf{cpo}_\bot \to \mathbf{cpo}_\bot \) and \( \otimes : \mathbf{cpo}_\bot \times \mathbf{cpo}_\bot \to \mathbf{cpo}_\bot \) are
given by composition and component-wise application, respectively.

\[
(f \to f')(g) = f' \circ g \circ f
\]

\[
(f \otimes f')(x, x') = [f(x), f'(x')]
\]

The category \( \mathbf{cpo}_\bot \) is a sub-category of \( \mathbf{Cpo} \), the category of complete
partial orders and continuous functions. This inclusion has a left adjoint \( \text{lift}: \mathbf{Cpo} \to \mathbf{cpo}_\bot \) which adjoins a new element \( \bot \) to a CPO strictly less than all previously existing elements. We typically denote the image of \( A \) under \( \text{lift} \) as \( A_\bot \). In cases were \( A \) was already a pointed CPO, there is the potential for confusion between the ‘new’ bottom element of \( A_\bot \) and the ‘old’ bottom element of \( A \), which is also an element of \( A_\bot \). Therefore \( \bot \) may be subscripted by the CPO it is the bottom element for when necessary to avoid confusion. For instance, \( \bot_{A_\bot} \subseteq \bot_A \subseteq a \) (for any \( a \in A \)).

It is straightforward to show that \( \text{lift} \) is the left adjoint to the inclusion of \( \mathbf{cpo}_\bot \) into \( \mathbf{Cpo} \). Since \( \mathbf{Cpo} \) is Cartesian closed, this gives an LNL model (in the sense of [Ben94]). The category \( \mathbf{cpo}_\bot \) is therefore a linear category with the restriction of \( \text{lift} \) to \( \mathbf{cpo}_\bot \) as the co-monad. The transformations \( \epsilon_A: A_\bot \to A \), \( \delta_A: A_\bot \to (A_\bot)_\bot \), \( \epsilon_A: A_\bot \to I \) and \( \delta_A: A_\bot \to A_\bot \otimes A_\bot \) used in modeling PLLC are given below (where we use \( a \) to denote any non-bottom element of \( A_\bot \)).

\[
\begin{align*}
\epsilon_A(\bot_{A_\bot}) &= \bot_A & \epsilon_A(a) &= a \\
\delta_A(\bot_{A_\bot}) &= \bot_{A_{\bot\bot}} & \delta_A(a) &= a \\
e_A(\bot_{A_\bot}) &= \bot_I & e_A(a) &= \top \\
\delta_A(\bot_{A_\bot}) &= \bot_{A_\bot \otimes A_\bot} & \delta_A(a) &= [a,a]
\end{align*}
\]

The type of non-linear functions \( \phi_1 \Rightarrow \phi_2 \) was defined to be \( !\phi_1 \circ \phi_2 \). We can look at what this might look like in \( \mathbf{cpo}_\bot \), using the co-monad \( \text{lift} \) to interpret !. The adjunction \( \mathbf{cpo}_\bot \xrightarrow{\text{lift}} \mathbf{Cpo} \) ensures an isomorphism between the strict continuous functions \( A_\bot \to B \) and the continuous functions \( A \to B \). With \( \phi_1 \circ \phi_2 \) interpreted as the internal hom in \( \mathbf{cpo}_\bot \) (that is, the set of strict continuous functions), it follows that the non-linear function type \( \phi_1 \Rightarrow \phi_2 \) can be thought of as the collection of continuous functions, that is, the internal hom in \( \mathbf{Cpo} \). Non-linear function terms are essentially morphisms of the Cartesian category \( \mathbf{Cpo} \), as opposed to the linear function terms which are morphisms of the linear category \( \mathbf{cpo}_\bot \).

We look for a suitable notion of edges over \( \mathbf{cpo}_\bot \) to produce a linear parametricity graph which we can use to model PLLC. Not only does this require parametric limits and parametric colimits, but the above transformations \( \epsilon, \delta, e \) and \( d \) need to be parametric. We present two such parametricity graphs — one using relations and the other using spans.

221
7.2.1 Using Relations

In considering what an appropriate notion of relation over \( \text{cpo}_\bot \) is, it seems reasonable to expect that the extra structure (beyond that of sets) should be respected. Fixed point operators (recursion) are defined in terms of least upper bounds (LUBs or limits) of chains starting with \( \bot \). This suggests that relations should respect both least upper bounds as well as the bottom elements of CPOs. The reflexive graph category \( \text{CPO}_\bot \) used by O’Hearn and Reynolds [RO00] uses complete pointed relations as edges. These are relations \( R: A \leftrightarrow B \) with the following properties:

- for directed sets \( \{a_i\}_I \subseteq A \) and \( \{b_i\}_I \subseteq B \) such that \( \forall i \in I. \ a_i [R] b_i \), the least upper bounds are related as well: \( \bigcup_I a_i [R] \bigcup_I b_i \).

\( \bot \) \( [R] \) \( \bot \).

Squares of \( \text{CPO}_\bot \) are as in \( \text{REL} \) — pairs of morphisms that map related elements to related elements. This is a parametricity graph, with weakest pre-edges just as in \( \text{REL} \) and \( \text{CPO} \).

\[
x \ [f, f'] [R] x' \iff f \ [R] f' \ x'
\]

The internal hom functor of \( \text{cpo}_\bot \) (strict, continuous function space) extends to a PG-functor \( \text{CPO}_\bot^{\text{op}} \times \text{CPO}_\bot \to \text{CPO}_\bot \) in a straightforward manner. For edges \( R \) and \( S \), strict continuous functions \( f \) and \( f' \) are related by \( R \circ f = S \) if and only if for every \( x \ [R] x' \), it is the case that \( f \ [S] f' \ x' \). O’Hearn and Reynolds point out that it is not so obvious how the smash product is extended to complete pointed relations. One might expect to use \( R \ast S; (A \circ B) \leftrightarrow (A' \circ B') \) given as follows.

\[
x \ [R \ast S] x' \iff \exists [a, b] = x \land [a', b'] = x' \text{ such that } a \ [R] a' \land b \ [S] b'
\]

However, \( R \ast S \) may not be a complete relation even though \( R \) and \( S \) both are.

As an example of this (from [RO00]), consider the CPOs \( I \), \( \mathbb{N}_\bot \), and \( \text{Vnat} \) where \( \mathbb{N}_\bot \) is the lift of the flat natural numbers \( (n \not\equiv m \text{ for any } n \neq m) \) and \( \text{Vnat} \) is the vertical natural numbers \( (\bot = 0 \leq 1 \leq 2 \leq \cdots \leq \omega) \). Let \( R; \mathbb{N}_\bot \leftrightarrow \text{Vnat} \) relate each \( n \) to itself as well as \( \bot \in \mathbb{N}_\bot \) to \( \bot = 0 \in \text{Vnat} \). Let \( S; I \leftrightarrow I \) relate only \( \bot [S] \bot \) and \( [S] \top \). Both \( R \) and \( S \) are trivially complete, but \( R \ast S \) relates the directed subsets \( \{[n, \bot]\} \subseteq \mathbb{N}_\bot \circ I \) and \( \{[n, \top]\} \subseteq \text{Vnat} \circ I \) in the apparent pointwise manner. (The first is a directed set because \( [n, \bot] = [\bot, \bot] \in \mathbb{N}_\bot \circ I \) for all \( n \), and hence the one element set
$\{[n, \bot]\}$ is trivially directed.) The least upper bounds of these directed sets $[\bot, \bot]$ and $[\omega, \top]$ are not related by $R \ast S$.

One can define the correct relational action $R \otimes S$ as the completion of $R \ast S$, that is, the intersection of all compete pointed relations containing it. This can be shown to be the edge portion of the PG-functor $\otimes$. Given any squares $\sigma$ and $\tau$ of the following shapes, we show that there is a square $\sigma \otimes \tau$ of the following shape.

$$
\begin{array}{ccc}
\sigma : & A_0 \xrightarrow{f_0} A'_0 & \quad \tau : & B_0 \xrightarrow{g_0} B'_0 \\
R & A_1 \xrightarrow{f_1} A'_1 & S & B_1 \xrightarrow{g_1} B'_1
\end{array}
$$

$$
\begin{array}{c}
\sigma \otimes \tau : \quad R \otimes S \\
A_0 \otimes B_0 \xrightarrow{f_0 \otimes g_0} A'_0 \otimes B'_0 \\
A_1 \otimes B_1 \xrightarrow{f_1 \otimes g_1} A'_1 \otimes B'_1
\end{array}
$$

Define the relation $P$: $A_0 \otimes B_0 \leftrightarrow A_1 \otimes B_1$ as $P = [f_0 \otimes g_0, f_1 \otimes g_1] \rightarrow R' \otimes S'$. More explicitly, $P$ is given as follows.

$$
z_0 P z_1 \iff (f_0 \otimes g_0)z_0 \left[ R' \otimes S' \right] (f_1 \otimes g_1)z_1
$$

Note that the following relationship holds for every $x_0 [R] x_1$ and $y_0 [S] y_1$.

$$
[f_0(x_0), g_0(y_0)] \left[ R' \otimes S' \right] [f_1(x_1), g_1(y_1)]
$$

Therefore, it follows that $R \ast S$ is contained in $P$. Since $P$ is a complete pointed relation, $R \otimes S \subseteq P$, which gives the desired square by composition.

$$
\begin{array}{ccc}
A_0 \otimes B_0 & \xrightarrow{id} & A_0 \otimes B_0 & \xrightarrow{f_0 \otimes g_0} & A'_0 \otimes B'_0 \\
R \otimes S & \xrightarrow{P} & R' \otimes S' \\
A_1 \otimes B_1 & \xrightarrow{id} & A_1 \otimes B_1 & \xrightarrow{f_1 \otimes g_1} & A'_1 \otimes B'_1
\end{array}
$$

The above definition of $R \otimes S$ is the correct definition for the PG-functor $\otimes$ in the following sense.

**Theorem 7.5**

$I, \otimes$ and $\rightarrow$ give a symmetric monoidal structure on $\mathbf{CPO}_\bot$.

**Proof.** One need only show that the natural transformations from the SMC structure of $\mathbf{cpo}_\bot$ are actually parametric transformations in $\mathbf{CPO}_\bot$. 223
In order for $\eta$ to be parametric, there must be a square of the following shape for any $R: A_0 \leftrightarrow A_1$ and $S: B_0 \leftrightarrow B_1$.

$$
\begin{array}{c}
A_0 \xrightarrow{\eta_{A_0,B_0}} B_0 \xrightarrow{\epsilon_{A_0,B_0}} (A_0 \otimes B_0) \\
R \downarrow \quad S \downarrow \\
A_1 \xrightarrow{\eta_{A_1,B_1}} B_1 \xrightarrow{\epsilon_{A_1,B_1}} (A_1 \otimes B_1)
\end{array}
$$

This follows from $R \ast S$ being contained in $R \otimes S$. For any $x R x'$ and $y S y'$, we have $[x,y] \left[ R \ast S \right] [x',y']$ (recalling $\eta_{A,B}(x)y = [x,y]$).

Showing that $\epsilon$ is parametric amounts to producing a square of the following shape.

$$
\begin{array}{c}
(A_0 \circ B_0) \otimes A_0 \xrightarrow{\epsilon_{A_0,B_0}} B_0 \\
(R \circ S) \otimes R \downarrow \quad S \downarrow \\
(A_1 \circ B_1) \otimes A_1 \xrightarrow{\epsilon_{A_1,B_1}} B_1
\end{array}
$$

Such a square can be exhibited using the preimage argument. Define the edge $P = [\epsilon_{A_0,B_0}] S$. It is straightforward to see that $(R \circ S) \ast R$ is contained in $P$. Since $(R \circ S) \otimes R$ is the least complete relation containing $(R \circ S) \ast R$, it must be the case that $(R \circ S) \otimes S$ is contained in $P$ as well. The desired square then arises from composition.

These parametricity arguments give insight into why the $\circ$ relation is defined the way it is. The parametricity of $\eta$ forces $R \otimes S$ to contain $R \ast S$. The parametricity of $\epsilon$ does not rule out $R \otimes S$ relating additional elements whose difference is not observable by the action of strict, continuous functions. We shall return to this point in the next section, after completing description of the structure of $\mathbf{CPO}_{\perp}$ necessary for modeling the linear lambda calculus.

The functor $\text{lift}$ extends to a PG-functor $\text{LIFT}: \mathbf{CPO} \rightarrow \mathbf{CPO}_{\perp}$ by relating new bottom elements to each other.

$$x \left[ R_{\perp} \right] x' \iff (x = \perp_{A_{\perp}} \land x' = \perp_{B_{\perp}}) \lor x \left[ R \right] x'$$

Since this gives a parametric adjunction, it is straightforward to show that the natural transformations $\epsilon$, $\delta$, $\epsilon$ and $\delta$ are in fact parametric.

Let $G \xrightarrow{\perp} G$ be a small, sub-ILNL setting of $\mathbf{CPO}_{\perp} \xrightarrow{\perp} \mathbf{CPO}$. Parametric limits of functors $F: G \rightarrow \mathbf{CPO}_{\perp}$ (and corresponding edges $\mathcal{F}$ of
\( \text{CPO}_\perp^G \) are given in a manner analogous to \( \text{CPO} \).

\[
\forall Y F(Y) = \{ p \in \Pi Y \in \text{CPO}_G F(Y) \mid \text{for all } R; X \leftrightarrow Y, \, p_X \left[ F(R) \right] p_Y \}
\]

\[
p \left[ \forall Y F(Y) \right] q \iff \text{for all } R; X \leftrightarrow Y, \, p_X \left[ F(E, R) \right] q_Y
\]

The projection parametric transformation \( \omega : \forall Y F(Y) \to F \) selects the indicated component, \( \omega_Y(p) = p_Y \in F(Y) \).

Parametric colimits are also constructed in a manner similar to those of \( \text{CPO} \) (section 3.1). The main difference is the use of compatible pointed families. These are indexed families \( \langle S_A \rangle \) of non-empty complete downward closed subsets \( S_A \subseteq F(A) \) such that for every \( R; A \leftrightarrow B \), if \( a \left[ F(R) \right] b \), then \( a \in S_A \iff b \in S_B \). (Because \( \text{CPO}_\perp \) is subsumptive, the compatibility with morphisms is a consequence of compatibility with edges.) Parametric colimits are defined in stages, starting from principal families (that is, compatible pointed families that cover singletons), \( P_0^F \). The additive completion of \( P_0^F \) can be defined as the limit of approximations \( P_\alpha^F \).

\[
P_0^F = \{ \langle a \rangle \mid a \in F(A) \text{ for some } A \}
\]

\[
P_\alpha^F = \bigcup \{ \{ q_n \} \mid \text{a bounded subset of } P_\beta^F \text{ for some } \beta < \alpha \}
\]

\[
\exists Y F(Y) = \bigcup_\alpha P_\alpha^F
\]

For edges \( F; F \leftrightarrow G \) of \( \text{CPO}_\perp^G \), the relation \( \exists Y F(Y) ; \exists Y G(Y) \) is also built in stages, \( Q_\alpha : P_\alpha^F \leftrightarrow P_\alpha^G \).

\[
x \left[ Q_0 \right] y \iff \exists R; A \leftrightarrow B, \, a \in F(A), \, b \in G(B).
\]

\[
\langle a \rangle = x \land \langle b \rangle = y \land a \left[ F(E, R) \right] b
\]

\[
x \left[ Q_\alpha \right] y \iff \exists \beta < \alpha, \text{ bounded } \{ a_n \} \subseteq P_\beta^F, \, \{ b_n \} \subseteq P_\beta^G.
\]

\[
\forall n. \, a_n \left[ Q_\beta \right] b_n \land \bigsqcup a_n = x \land \bigsqcup b_n = y
\]

\[
x \left[ \exists Y F(Y) \right] y \iff \exists \alpha \in \text{ORD. } x \left[ Q_\alpha \right] y
\]

Since parametric limits and parametric colimits exist, the LNL settings \( (G, G, \text{LIFT}, \text{INC}) \) and \( (\text{CPO}_\perp, \text{CPO}, \text{LIFT}, \text{INC}) \) can be used to construct a model of PLLC as described in section 7.1.

### 7.2.2 Using Spans

The intuition behind \( A \otimes B \) is that an “item” of \( A \otimes B \) consists of an item of \( A \) and an item of \( B \). For correspondences, the intuition of \( R \otimes S \) is to relate
those pairs of items consisting of an $R$-related pair and an $S$-related pair. When using complete relations for edges, one needs to take the completion in computing $R \otimes S$. This results in relating pairs that do not consist of an $R$-related pair and an $S$-related pair. It is occasionally somewhat inconvenient to deal with the extra elements which appear from taking the completion, such as in the preimage arguments above. It is possible to remain closer to the intuition mentioned above by not taking the extra completion step to define the smash product. This is the situation when one considers spans, rather than relations, over $\text{cpo}_\bot$.

The parametricity graph $\text{Sp}(\text{cpo}_\bot)$ is symmetric monoidal closed. Recall that there is a unique square $\sigma: R \to S$ over $f$ and $g$ in $\text{Sp}(\text{cpo}_\bot)$ if and only if there exists a morphism $W(\sigma): W(R) \to W(S)$ so that the diagrams below commute.

\[
\begin{align*}
A & \xrightarrow{f} B \\
W(R) & \xrightarrow{W(\sigma)} W(S) \\
A' & \xleftarrow{g} B'
\end{align*}
\]

Even though there may be several different witnesses $W(\sigma)$, there is at most one square from $R$ to $S$ over $f$ and $g$.

The smash product of spans $R; A \leftrightarrow A'$ and $S; B \leftrightarrow B'$ is given in a straightforward manner. The witness CPO of the smash product is the smash product of the witness CPOs. $W(R \otimes S) = W(R) \otimes W(S)$.

Every span $R; A \leftrightarrow A'$ determines an underlying relation $U(R); A \leftrightarrow A'$.

\[
x \left[ U(R) \right] x' \iff \exists w \in W(R) \text{ such that } p_0(w) = x \land p_1(w) = x'
\]

For an arbitrary span, the underlying relation need not be a complete relation. For instance, consider an analogue of the O’Hearn-Reynolds smash product example.

\[
\begin{align*}
\begin{array}{ccc}
\mathbb{N} & \xleftarrow{id} & I \\
\bar{R} & \xrightarrow{\text{inc}} & \mathbb{N} \\
\end{array} & \xrightarrow{\bot} & \begin{array}{ccc}
\mathbb{N} & \xrightarrow{id} & I \\
\bar{S} & \xrightarrow{\text{inc}} & \mathbb{N} \\
\end{array}
\end{align*}
\]

The underlying relations for these are exactly the complete relations mentioned earlier $R = U(\bar{R})$ and $S = U(\bar{S})$. Taking the smash product of
these spans and then the underlying relations yields the incomplete relation $R \ast S = \bigcup (\hat{R} \otimes \hat{S})$. There is no reason for the directed sets $\{[n, \bot]\} \subseteq \mathbb{N} \otimes I$ and $\{[n, \top]\} \subseteq \mathbb{V} \otimes I$ to have their least upper bounds $[\bot, \bot] \in \mathbb{N} \otimes I$ and $[\omega, \top]$ related. These directed sets do not arise in from a directed set of witnesses in the CPO $W(\hat{R} \otimes \hat{S})$.

In a sense, the directed set $\{[n, \bot]\} \subseteq \mathbb{N} \otimes I$ is not a significant direct set in the tensor product. It arises as a consequence of equating unordered elements rather than as the product of directed sets. As such, it will not arise as a sequence of increasing defined approximations. Thus not having these least upper bounds preserved by $\hat{R} \otimes \hat{S}$ will not be a detriment to the parametricity of a fixed point operator. In effect, the witness CPO of a span is there to keep a record of which directed sets are significant, and would need to have their least upper bounds respected.

While the smash product of spans is straightforward and component-wise, the construction of $R \rightrightarrows S$ is a little more complicated. One can not simply use $W(R) \rightrightarrows W(S)$ as the witness CPO, since there is not a reasonable candidate for the projections $W(R) \rightrightarrows W(S) \rightarrow A_0 \rightrightarrows B_0$ and $W(R) \rightrightarrows W(S) \rightarrow A_1 \rightrightarrows B_1$. The witness of $R \rightrightarrows S$ needs to contain at least as much information as squares $R$ to $S$ do (that is, pairs of functions $f_0: A_0 \rightarrow B_0$ and $f_1: A_1 \rightarrow B_1$). Rather than just using the collection of squares, we commit to a particular witness. The CPO $W(R \rightrightarrows S)$ is given by triples of strict, continuous functions.

$$W(R \rightrightarrows S) = \{(f, f_0, f_1) \mid \text{commutes}\}$$

The order on $W(R \rightrightarrows S)$ is given component-wise, so that the obvious projections $W(R \rightrightarrows S) \rightarrow A_0 \rightrightarrows B_0$ and $W(R \rightrightarrows S) \rightarrow A_1 \rightrightarrows B_1$ are both strict and continuous.

**Theorem 7.6**

$I$, $\otimes$ and $\rightrightarrows$ given above describe an SMC structure on $\mathbf{Sp}(\mathbf{cpo}_\bot)$.

**Proof.** Showing that the natural transformations from $\mathbf{cpo}_\bot$ are parametric in $\mathbf{Sp}(\mathbf{cpo}_\bot)$ means producing appropriate witness morphisms. For the unit of the adjunction, a morphism $\eta: W(R) \rightarrow W(S \rightrightarrows (R \otimes S))$ maps a witness $x \in W(R)$ to the triple $(\eta x, \eta a_0, \eta a_1)$ where $\partial_0(x) = a_0$ and $\partial_1(x) = a_1$. It is obvious that this $\eta$ is the witness of a span morphism over $\eta_{A_0, B_0}$ and
\( \eta_{A_1, B_1} \):

The parametricity of \( \epsilon \) requires one to produce a witness morphism
\( e: W(S \rightarrow T) \otimes W(S) \rightarrow W(T) \). This is given as
\( e[(f, f_0, f_1), y] = f(y) \). We observe that this is well-defined since
\( f(\bot) = \bot \) for any strict function \( f \) and \( \bot(y) = \bot \) for all \( y \). We must also show that there is a square using \( e \) as
the witness morphism. Thus, we show that the following square commutes.

\[
\begin{array}{c}
\pi \otimes p_0 \\
\downarrow \\
W(S \rightarrow T) \otimes W(S) \\
\downarrow e \\
W(T) \\
\downarrow p_0 \\
\end{array}
\]

\( (B_0 \rightarrow C_0) \otimes B_0 \xrightarrow{\epsilon_{B_0, C_0}} C_0 \)

This is shown by straightforward calculation.

\[
(\epsilon \circ (\pi \otimes p_0)) [(f, f_0, f_1), y] = \epsilon [f_0, p_0(y)] = f_0(p_0(y)) = (p_0 \circ e)[(f, f_0, f_1), y]
\]

The other square is shown to commute analogously.

The parametricity of \( \epsilon \) gives insight into why the definition of \( R \rightarrow S \) is
the way that it is. One needs the explicit choice of a witness morphism
in \( W(R \rightarrow S) \) to be able to give the witness morphism \( e \). Simply using the
collection of squares \( R \rightarrow S \) for the witness of \( R \rightarrow S \) does not suffice.

We also need parametric limits to be able to model the polymorphic
linear lambda calculus. The parametric limit and colimit of a PG-functor
\( F: G \rightarrow \text{Sp eo}_{\bot} \) (for a small parametricity graph \( G \)) are constructed in
a familiar manner.

\[
\forall Y F(Y) = \left\{ f \in \Pi Y_{\text{Sim}(G)} F(Y) \mid f_Y \left[ U(F(R)) \right] f_Y', \text{for all } R: Y \leftrightarrow Y' \right\}
\]

\[
P^F_0 = \left\{ \{a\} \mid a \in F(A) \text{ for some } A \in |G| \right\}
\]

\[
P^F_{\alpha} = \left\{ \bigcup Q \mid Q^\beta \subseteq P^F_{\beta} \text{ for some } \beta < \alpha \right\}
\]

\[
\exists Y F(Y) = \bigcup_{\alpha \in \text{ORD}} P^F_{\alpha}
\]

Note that the compatible pointed families used here to define the parametric
colimit should be compatible with respect to relations underlying the spans.

\[
\text{for all } R: A \leftrightarrow B, \text{ if } a \left[ U(F(R)) \right] b, \text{ then } a \in S_A \iff b \in S_B
\]

228
Parametric limits of edges of $\text{Sp}(\text{cpo}_\bot)^G$ are given by jointly monic spans $\forall_Y \mathcal{F}(Y) : \forall_Y F_0(Y) \leftrightarrow \forall_Y F_1(Y)$ whose witness CPO are given below.

$$W(\forall_Y \mathcal{F}(Y)) = \left\{ (p, q) \mid \text{ for all } R: A \leftrightarrow B, \, \left[ U(\mathcal{F}(E, R)) \right] q_B \right\}$$

For the dual, parametric colimit of edges are spans whose witness CPO is given by the indexed coalesced sum.

$$W(\exists_Y \mathcal{F}(Y)) = \sum_{R \in \text{obj}(\mathcal{C})} W(\mathcal{F}(E, R))$$

Recall that the coalesced sum of pointed CPOs is the disjoint union where all the bottom elements are identified. The order is inherited from the components.

$$\text{in}_R(x) \sqsubseteq \text{in}_{R'}(x') \iff (R = R' \land x \sqsubseteq x') \lor \text{in}_R(x) = \bot$$

As a consequence, for any directed subset \( \{y_n\} \) of the coalesced sum, there exists an \( R \) such that \( y_n = \text{in}_R(x_n) \) for a directed subset \( \{x_n\} \) of \( W(\mathcal{F}(E, R)) \). The projections \( p_i: W(\exists_Y \mathcal{F}(Y)) \to \exists_Y F_i(Y) \) of the span make use of the injections \( \mu^i: F_i \to \exists_Y F_i(Y) \) as well as the projections \( p^R_i \) from the span \( \mathcal{F}(E, R) \) to \( F_i(R) \) (for the appropriate \( R: A_0 \leftrightarrow A_1 \)).

$$p_i(\text{in}_R(x)) = \mu^i_{A_1}(p^R_i(x))$$

To use \( \text{Sp}(\text{cpo}_\bot) \) to model PLLC, there needs to be a parametric adjunction with a Cartesian closed parametricity graph. Observe that the adjunction \( \text{cpo}_\bot \xrightarrow{\epsilon, \delta} \text{Cpo} \) in \( \text{CAT} \) extends to a parametric adjunction \( \text{Sp}(\text{cpo}_\bot) \xrightarrow{\epsilon, \delta} \text{CPO} \) in the obvious manner. The definitions of \( \epsilon, \delta, e \) and \( d \) given at the beginning of this section are parametric transformations. The model construction in section 7.1 gives a \( \text{Sp}(\text{cpo}_\bot) \)-model of PLLC.

### 7.3 Linear Models

In this section, we present parametricity graphs which have bounded complete partial orders as vertices. These produce LNL settings which are truly linear in that they model neither the structural rule of contraction nor weakening.

The smash product in \( \text{cpo}_\bot \) is not a Cartesian product, hence does not model the (full) lambda calculus. But it does model more that just a
linear calculus. It is well known that there is a diagonal $\partial_A : A \to A \otimes A$ in $\mathbf{cpo}_\perp$ given by $\partial_A(x) = [x, x]$. This is a parametric transformation in both parametricity graphs, $\mathbf{CPO}_\perp$ and $\mathbf{Sp}(\mathbf{cpo}_\perp)$. Hence, these $\mathbf{cpo}_\perp$-based models also model the contraction rule. These are more accurately described as models of a polymorphic relevant lambda calculus. O’Hearn and Reynolds [RO00] are satisfied with this as they point out that many of the troublesome examples, such as snapback, rely on both contraction and weakening. As the $\mathbf{cpo}_\perp$-based models do not support weakening, they were able to use this to prove some previously unsupported equivalences.

There are some truly linear models. One category of domains which leads to such models is the category $\mathbf{bcpo}$ of bounded-complete partial orders. Objects are non-empty partial orders such that not only does every directed subset have a least upper bound, but also every subset which is bounded above has a least upper bound. Note that every directed subset is necessarily a bounded subset, so least upper bounds of bounded sets will include the least upper bounds of directed sets. Bottom elements are also subsumed as they are the least upper bounds of empty subsets. The arrows of $\mathbf{bcpo}$ are additive functions — functions which preserve the least upper bounds of all bounded sets.

The symmetric monoidal closed structure of $\mathbf{bcpo}$ starts with the internal hom, $A - \bowtie B = \text{Hom}(A, B)$, ordered pointwise. The associated tensor product has equivalence classes as elements, as was the case in $\mathbf{cpo}_\perp$. Unlike the $\mathbf{cpo}_\perp$ case, these are not equivalence classes of the product partial order $A \times B$, but rather equivalence classes of bounded subsets of $A \times B$. The equivalence relation we are interested in amounts to saying taking least upper bounds in either component is irrelevant. Formally, the equivalence relation $\equiv$ on the bounded power set $\mathcal{P}_B(A \times B)$ is the least equivalence relation such that the following hold.

\[
\{(x_i, y_i)\}_{i \in I} \equiv \left\{ \bigsqcup_{i \in I} x_i, y_i \right\}
\]

\[
\{(x, y_i)\}_{i \in I} \equiv \left\{ (x, \bigsqcup_{i \in I} y_i) \right\}
\]

We denote the equivalence class of $S = \{(x_i, y_i)\}_{i \in I}$ as either $\{\{(x_i, y_i)\}_{i \in I}$ or $\{\{S\}$, perhaps dropping mention of the indexing set where appropriate. The partial order of $A \otimes B = \mathcal{P}_B(A \times B)/\equiv$ is given as follows.

\[
\{S\} \subseteq \{T\} \iff \exists \{\{(x_i, y_i)\}_{i \in I} \equiv S, \{\{(x'_i, y'_i)\}_{i \in I} \equiv T, \forall i \in I, \quad x_i \subseteq x'_i \land y_i \subseteq y'_i
\]

230
This does make $A \otimes B$ into a bounded-complete partial order with least upper bounds given by unions, $\bigcup_{i \in I} \llbracket S_i \rrbracket = \llbracket \bigcup_{i \in I} S_i \rrbracket$.

The arrow portion of the functor $\otimes : \text{bcpo} \times \text{bcpo} \to \text{bcpo}$ is given pointwise. $(f \otimes g) \llbracket (x_i, y_i) \rrbracket_i = \llbracket (f(x_i), g(y_i)) \rrbracket_i$. This is the left adjoint to $- \circ: \text{bcpo}^{op} \times \text{bcpo} \to \text{bcpo}$. The unit $\eta_{A,B}: A \to B \circ (A \otimes B)$ and co-unit $\epsilon_{B,C}: (B \circ C) \otimes B \to C$ are given as follows.

$$\eta_{A,B}(a) = \lambda b: B. \llbracket (a, b) \rrbracket$$

$$\epsilon_{B,C} \llbracket (f_i(x_i)) \rrbracket_{i \in I} = \bigsqcup_{i \in I} f_i(x_i)$$

The category $\text{bcpo}$ is a reflective sub-category of $\text{Cpo}$. The collection of all bounded subsets of an arbitrary CPO $A$ (denoted $\mathcal{P}_b(A)$) is a bounded CPO under the subset order. We can define an equivalence relation $\approx$ on $\mathcal{P}_b(A)$ intuitively by equating any directed subset with the singleton set of its least upper bound. More formally, we take the transitive closure of

$$S \cup \{a_i\}_{i \in I} \approx S \cup \Big\{ \bigsqcup_{i \in I} \{a_i\} \Big\}$$

for any directed set $\{a_i\}_{i \in I}$ and any set $S$ such that $S \cup \{a_i\}$ is bounded. Equivalence classes of $\approx$ inherit the subset order of $\mathcal{P}_b(A)$.

$$x \subseteq y \iff \exists \{a_i\}_{i \cup J} = y \text{ such that } \{a_i\} = x$$

This can be seen to make $T(A) = \mathcal{P}_b(A)/\approx$ into a bounded complete partial order, were least upper bounds are given by union (of representatives of the classes). This defines the object part of a functor. The arrow portion is straightforward, as any continuous function can be applied pointwise to any representative of an equivalence class to yield an additive function.

$$T(f) \{a_i\}_{i \in I} = \llbracket f(a_i) \rrbracket_{i \in I}$$

The restriction of $T$ to $\text{bcpo}$ provides the desired symmetric monoidal co-monad. This yields the following structure transformations of the co-monad $\epsilon, \delta$, as well as the discarding $e$ and duplication $d$ transformations.

$$\epsilon_A \{a_i\}_{i \in I} = \bigsqcup \{a_i\}_{i \in I}$$

$$\delta_A \{a_i\}_{i \in I} = \{ \{a_i\} \}_{i \in I}$$

$$e_A \{a_i\}_{i \in I} = \top \iff \{a_i\}_{i \in I} \neq \bot$$

$$d_a \{a_i\}_{i \in I} = \{ \{a_i\}, \{a_i\} \}_{i \in I}$$

231
Again, the interpretation of non-linear function types consists of continuous functions, being the morphisms of the Cartesian category \textbf{Cpo} from the adjunction.

An appropriate notion of relations between bounded CPOs are additive relations. A relation \( R: A \leftrightarrow B \) is additive if for all bounded sets \( \{ a_i \}_{i \in I} \subseteq A \) and \( \{ b_i \}_{i \in I} \subseteq B \) such that \( a_i [R] b_i \) for all \( i \in I \), then \( \bigsqcup_{i \in I} a_i [R] \bigsqcup_{i \in I} b_i \) as well. These are the edges of a parametricity graph \textbf{BCPO} with vertex category \textbf{bcpo} and squares as in \textbf{REL} (that is, pairs of additive functions which map related arguments to related results).

The SMC structure of \textbf{BCPO} is analogous to that of \textbf{CPO} \(_\perp\). Two additive functions \( f: A \to A' \) and \( g: B \to B' \) are related by \( R \odot R' \) if for every \( a [R] b \), one has \( f(a) [R'] g(b) \). The tensor product proceeds by taking the additive completion of the anticipated relation \( R \star R' \).

\[
\{ [S] \} \left[ R \star R' \right] \{ [T] \} \iff \exists \{(x_i, y_i)\}_{i \in I} = S, \quad \exists \{(x'_i, y'_i)\}_{i \in I} = T, \\
\forall i \in I, \quad x_i [R] x'_i \wedge y_i [R'] y'_i
\]

\[
\{ [S] \} \left[ R \otimes R' \right] \{ [T] \} \iff \forall \text{additive } P: A \otimes A' \leftrightarrow B \otimes B' \\
R \star R' \subseteq P \implies \{ [S] \} [P] \{ [T] \}
\]

Proving that this makes \textbf{BCPO} into a SMC parametricity graph is analogous to the arguments in \textbf{CPO} \(_\perp\). The unit \( \eta \) is parametric because \( R \otimes R' \) contains \( R \star R' \), and \( \epsilon \) is parametric since additive functions into additive relations can not differentiate between \( R \star R' \) and \( R \otimes R' \). The symmetry, associativity and identity for \( \odot \) are straightforward. Parametric limits are also produced in a familiar manner.

\[
\forall Y F(Y) = \{ p \in \Pi_Y F(Y) \mid \text{for all } R: Y \leftrightarrow Y', \quad p_Y \left[ F(R) \right] p_{Y'} \} \\
p \left[ \forall Y F(Y) \right] q \iff \text{for all } R: Y \leftrightarrow Y', \quad p_Y \left[ F(E, R) \right] q_{Y'}
\]

Parametric colimits are built in a manner analogous to those of \textbf{CPO} (section 3.1) and \textbf{CPO} \(_\perp\). The corresponding notion of compatible family used to make the parametric colimits in \textbf{BCPO} are indexed collections \( \langle S_A \rangle \) satisfying the following properties.

- \( S_A \) is a bounded-complete, downward closed subset of \( F(A) \) and
- For all \( R: A \leftrightarrow B \), if \( a [F(R)] b \), then \( a \in S_A \iff b \in S_B \).

The parametric colimit of vertices \( F \) and edges \( \mathcal{F} \) are defined as the bounded

232
completions \( \exists \gamma F(Y) = \bigsqcup \alpha P^F_\alpha \) and \( x \left[ \exists \gamma F(Y) \right] y \iff \exists \alpha. x \left[ Q_\alpha \right] y \) of the initial approximations using principal families below.

\[
P^F_0 = \{ (a) \mid a \in F(A) \text{ for some } A \}\]

\[x \left[ Q_0 \right] y \iff \exists R: A \leftrightarrow B, a \left[ F(R) \right] b \land \langle a \rangle = x \land \langle b \rangle = y\]

The adjunction between \( \text{bepo} \) and \( \text{Cpo} \) extends to a parametric adjunction \( \text{BCPO} \xrightarrow{T} \text{CPO} \). Any complete relation \( R: A \leftrightarrow B \) defines the relation \( T(R): T(A) \leftrightarrow T(B) \) as follows.

\[x \left[ \hat{T}(R) \right] y \iff \exists \left\{ a_i \right\}_I = x \land \exists \left\{ b_i \right\}_I = y \text{ such that } \forall i \in I. a_i \left[ R \right] b_i\]

Since least upper bounds in \( T(A) \) and \( T(B) \) are given by unions, and the equivalence relation defining \( T(A) \) and \( T(B) \) is given by unions, it follows that \( \hat{T}(R) \) is additive. The requirement that relations are additive plays an important role in showing the adjunction is parametric. Recall that the co-unit \( \epsilon \) of the adjunction between \( \text{bepo} \) and \( \text{Cpo} \) is given by least upper bounds \( \epsilon_A \left[ \left\{ a_i \right\}_I \right] = \bigsqcup_i a_i \). From knowing that \( a_i \left[ R \right] b_i \) for bounded subsets \( \{ a_i \}_I \) and \( \{ b_i \}_I \), the additive assumption on \( R \) allows one to conclude that \( \epsilon_A \left[ \left\{ a_i \right\}_I \right] = \bigsqcup_i a_i \) and \( \epsilon_B \left[ \left\{ b_i \right\}_I \right] = \bigsqcup_i b_i \) are related by \( R \). Showing the rest of the structure is parametric proceeds in a straightforward manner.

Concerns about the inconvenience of taking the completion in the tensor product of relations would apply to the \( \text{BCPO} \) model just as it did for \( \text{CPO} \). The complexity of \( \otimes \) can be traded for additional bookkeeping for \(-\alpha\) by using a span model in \( \text{Sp(bepo)} \). As the construction and arguments are not significantly different from those of \( \text{Sp(cpo)} \), we merely reiterated the witness portion of the edge constructions for \(-\alpha\) and \( \otimes \).

\[
\begin{array}{c}
A_0 \xrightarrow{f_0} B_0 \\
W(R \otimes R') = \{ (f, f_0, f_1) \mid W(R) \xrightarrow{f} W(R') \text{ commutes} \}
\end{array}
\]

\[
W(R \otimes R') = W(R) \otimes W(R') = W(R) \times W(R') / \equiv
\]

These \( \text{bepo} \)-based LNL settings (with suitable sub-LNL settings) provide models for the polymorphic linear lambda calculus. These models are truly linear in that neither weakening nor contraction are modeled. Weak-
ening fails since the unit of the tensor product is the two element BCPO, T, which is not terminal. Contraction fails since there is not a diagonal. The apparent candidate function \( \delta_A : A \to A \otimes A \) is not additive. This can be seen by considering three distinct elements such that \( \bigsqcup \{a, b\} = c \).

\[
\bigsqcup \{\delta_A(a), \delta_A(b)\} = \bigsqcup \{\delta_A(a), \delta_A(b)\} = \bigsqcup \{\delta_A(c)\} = \delta_A(c)
\]

We have exhibited several settings for model of PLLC. Relevant models were constructed using two different notions of edges over \( \text{cpo}_\bot \) — relations and spans. Similarly, two linear models arose over \( \text{bcpo} \). We next re-address the issue of recursion and parametricity in these settings.

### 7.4 Recursion and Parametricity

Recursion was used in the motivation for considering linearity. We show that the family of fixed point operators over \( \text{cpo}_\bot \) is parametric in both \( \text{CPO}_\bot \) and \( \text{Sp(cpo}_\bot \) ). These provide settings where recursion does coexist with a reasonable notion of parametricity. We exhibit the strength of parametricity in LNL settings in the form of representation results. We use this to reexamine the parametricity graph \( \text{PCPO} \) over \( \text{pcpo} \) mentioned in the introduction.

Recall that least fixed points in \( \text{pcpo} \) were calculated as the least upper bounds of repeated applications as follows.

\[
Y_A = \lambda f : A \Rightarrow A. \bigcup_{n \in \mathbb{N}} f^n(\bot)
\]

This is a strict continuous function, that is, an arrow \( Y_A : A \Rightarrow A \to A \) in \( \text{cpo}_\bot \). Considering the models of PLLC in the LNL settings from section 7.2, the family of fixed point operators preserve both relations and spans.

**Lemma 7.7**

There is a parametric transformation \( \langle Y_A \rangle_A : [X \Rightarrow X] \to [X] \) in \( \text{CPO}_\bot \) and in \( \text{Sp(cpo}_\bot \) ).

**Proof.** For \( \text{CPO}_\bot \), the argument from the introduction applies. Given any pointed complete relation \( R : A_0 \leftrightarrow A_1 \), any pair of continuous functions such that \( f_0 [ R \Rightarrow R ] f_1 \) will satisfy \( f_0^n(\bot) [ R ] f_1^n(\bot) \) for all \( n \) (with the pointed assumption supplying the \( n = 0 \) case). The completeness of \( R \) ensures \( Y_{A_0}(f_0) [ R ] Y_{A_1}(f_1) \).

---

234
Spans over pointed CPOs do not necessarily represent complete relations, so a slightly different approach is taken to show that \( \langle Y_A \rangle_A \) is parametric in \( \text{Sp}(\text{cpo}_\perp) \). It is still a straightforward and immediate calculation once the definitions have been unraveled. For any span \( S: B_0 \leftrightarrow B_1 \), the witness CPO of the span \( S \Rightarrow S \) has triples \((f, f_0, f_1)\) of continuous functions such that the following diagram commutes.

\[
\begin{array}{ccc}
B_0 & \xrightarrow{f_0} & B_0 \\
W(S) & \xrightarrow{f} & W(S) \\
B_1 & \xrightarrow{f_1} & B_1 \\
\end{array}
\]

(This follows from the definition of \( S \Rightarrow S \) as \( !S \circ S \).) We desire to show that there is a square of the following shape.

\[
\begin{array}{ccc}
B_0 & \xrightarrow{Y_{B_0}} & B_0 \\
S \Rightarrow S & \downarrow{S} & \downarrow{S} \\
B_1 & \xrightarrow{Y_{B_1}} & B_1 \\
\end{array}
\]

For there to exist such a square, there must be a strict continuous function \( h; W(S \Rightarrow S) \rightarrow W(S) \). Taking \( h(f, f_0, f_1) = Y_{W(S)} f \) does yield the commuting diagram below.

\[
\begin{array}{ccc}
A & \xrightarrow{Y_A} & A \\
W(S \Rightarrow S) & \xrightarrow{h} & W(S) \\
B & \xrightarrow{Y_B} & B \\
\end{array}
\]

As there is such an \( h \) for every span \( S \), the family of least fixed point operations is parametric in \( \text{Sp}(\text{cpo}_\perp) \). \( \diamond \)

Our characterization of uniformity as parametricity does include the family of least fixed point operators of \( \text{cpo}_\perp \) as uniform (using both relations and spans). This is not the first attempt to characterize the uniformity of \( \langle Y_A \rangle_A \). Plotkin described the uniformity of least fixed points for \( \text{cpo}_\perp \) in what has come to be known as Plotkin’s Axiom.

**Lemma 7.8 (Plotkin’s Axiom)**

*For any strict continuous function \( h; A \rightarrow B \) and continuous functions*
\( f \colon A \to A \) and \( g \colon B \to B \) such that \( h \circ f = g \circ h \), it is the case that \( h(Y_A(f)) = Y_B(g) \).

**Proof.** This is an immediate consequence of parametricity (for any parametricity graph over \( \text{cpo}_\perp \)) by using the graph of \( h \). The assumption \( h \circ f = g \circ h \) implies the existence of a square with the following shape.

\[
\begin{array}{c}
A \\
\downarrow h \\
B
\end{array} \xrightarrow{f} \begin{array}{c}
A \\
\downarrow h \\
B
\end{array} \xrightarrow{g} \begin{array}{c}
A \\
\downarrow h \\
B
\end{array}
\]

Thus we know that the corresponding constant functions map into \( \langle h \rangle \Rightarrow \langle h \rangle \).

\[
\begin{array}{c}
I \\
\downarrow \text{curry}(f)
\end{array} \xrightarrow{A \Rightarrow A} \begin{array}{c}
I \\
\downarrow \langle h \rangle \Rightarrow \langle h \rangle
\end{array} \xrightarrow{Y_A} \begin{array}{c}
I \\
\downarrow \langle h \rangle \Rightarrow \langle h \rangle
\end{array} \xrightarrow{Y_B} \begin{array}{c}
I \\
B \Rightarrow B
\end{array}
\]

Composing with the square from the parametricity of \( (Y_X)_X \), we get a square of the following shape.

\[
\begin{array}{c}
I \\
\downarrow \text{curry}(f)
\end{array} \xrightarrow{A \Rightarrow A} \begin{array}{c}
I \\
\downarrow \langle h \rangle \Rightarrow \langle h \rangle
\end{array} \xrightarrow{Y_A} \begin{array}{c}
I \\
\downarrow \langle h \rangle \Rightarrow \langle h \rangle
\end{array} \xrightarrow{Y_B} \begin{array}{c}
I \\
B \Rightarrow B
\end{array}
\]

The subsumption criteria ensures that the following is a commuting square.

\[
\begin{array}{c}
I \\
\downarrow \text{curry}(f)
\end{array} \xrightarrow{id} \begin{array}{c}
I \\
\downarrow \langle h \rangle \Rightarrow \langle h \rangle
\end{array} \xrightarrow{Y_A} \begin{array}{c}
I \\
\downarrow \langle h \rangle \Rightarrow \langle h \rangle
\end{array} \xrightarrow{Y_B} \begin{array}{c}
I \\
B \Rightarrow B
\end{array}
\]

This gives the desired equality.  

\[ \diamond \]

LNL settings provide models of linear lambda calculi using parametricity graphs. Parametricity graphs provide a reasonable notion of uniformity by having a rich collection of edges to be preserved. Even in linear settings, we can show that parametric limits are trimmed down to include only the intuitive uniform families.

The representation results of section 5.3 provide precise characterizations of some parametric limits. The result mentioned there were for a non-linear

236
setting, and cannot be carried over directly to models of PLLC which we are currently considering. However, there are representation results for parametric models of linear polymorphic calculi which bear a resemblance to the non-linear ones (as pointed out by Plotkin [Plot93]).

\[
\begin{align*}
\forall X.X & \quad \text{is initial} \\
\forall X.X \to X & \cong I \\
\forall X.(\tau_1 \to \tau_2 \to \circ X) & \cong [\tau_1] \otimes [\tau_2] \\
\forall X.(\tau_1 \to X) \Rightarrow (\tau_2 \to \circ X) & \cong [\tau_1] + [\tau_2] \\
\forall X((\tau_1 \circ X) + (\tau_2 \circ X)) & \cong [\tau_1] \times [\tau_2] \\
\forall Y.\forall X.(\tau \to \circ Y) \Rightarrow Y & \cong \exists X.\tau' \\
[\eta \vdash \forall X.(\tau \to \circ X) \Rightarrow X] & \text{is the initial } [\eta, X \vdash \tau'] \text{ algebra}
\end{align*}
\]

Here \(\tau_1\) and \(\tau_2\) denote types that do not include \(X\). In the encoding of products, \(\tau_1 + \tau_2\) is used as an abbreviation for \(\forall X.(\tau_1 \circ X) \Rightarrow (\tau_2 \circ X) \Rightarrow X\), in light of the previous encoding of coproducts.

To illustrate how the above representation results do capture the intuitively uniform families characterization, we read what a few of these say in \texttt{cpo}. There is only one uniform way to pick an element from each pointed CPO — pick the bottom element. This is in agreement with the initial object, since the initial object of \texttt{cpo} is a one element set \(\{\bot\}\). There are two uniform families of morphisms \(A \to A\), the family of identity morphisms and the family of constant bottom morphisms. Since the unit of the tensor product is a two element set \(I = \{\top, \bot\}\), this also agrees.

Given an LNL setting which has parametric limits and colimits over itself, PLLC can be modeled using the same LNL setting for simple types as polymorphic types. In any such LNL setting where the SMC parametricity graph is well-pointed, the above stated representation results can be proven. Many of these results do rely on impredicativity — the lack of a distinction between simple types and polymorphic types.

One can prove the above representation results in a predicative setting if one has additional information so as to infer that the interpretation of the polymorphic type in question is (at least isomorphic to) the interpretation of a simple type. Since \(I\) and \(\tau_1 \otimes \tau_2\) are simple types, the corresponding two representation results hold in all well-pointed LNL settings for modeling PLLC. (We say that an LNL setting is well-pointed whenever the SMC parametricity graph portion of it is well-pointed.) As an example of how we can prove further representation results under additional assumptions on the
settings for simple types, we prove the coproducts representation result in $\mathbf{CPO}_\bot$. The coproduct in $\mathbf{cpo}_\bot$ is given by the coalesced sum $A \oplus B$ which, if you recall, is the disjoint union with the bottom elements equated.

**Theorem 7.9**

Suppose that $\mathbf{G} \xrightarrow{r} \mathbf{\bar{G}}$ is a small sub-LNL setting of $\mathbf{CPO}_\bot \xrightarrow{\bot} \mathbf{CPO}$ which is closed under coalesced sums. For any vertices $A$ and $B$ of $\mathbf{G}$, the parametric limit $\forall_X (A \rightarrow X) \Rightarrow (B \rightarrow X) \Rightarrow X$ in $\mathbf{CPO}_\bot$ is isomorphic to $A \oplus B$.

**Proof.** Mapping $a \in A$ to the tuple $\left< \lambda f:A \rightarrow C. \lambda g:B \rightarrow C. f(a) \right>_C$ is a strict continuous function $\gamma_A: A \rightarrow \forall_X (A \rightarrow X) \Rightarrow (B \rightarrow X) \Rightarrow X$. For each $C$, it is clear that $(\gamma_A)_C f$ is a constant function $\lambda g:B \rightarrow C. f(a): (B \rightarrow C) \rightarrow C$. This is obviously continuous, but not strict (unless $f(a)$ happens to be $\bot$). The family $\left< \lambda f. \lambda g. f(a) \right>_C$ is parametric since for any $R:C \leftrightarrow C'$, the following implication holds.

$$f \left[ I_A \rightarrow R \right] f' \text{ implies } f(a) \left[ R \right] f'(a).$$

The analogous strict function $\gamma_B: B \rightarrow \forall_X (A \rightarrow X) \Rightarrow (B \rightarrow X) \Rightarrow X$ is given by $\gamma_B b = \left< \lambda f:A \rightarrow C. \lambda g:B \rightarrow C. g(b) \right>_C$. In this case, it is the function $(\gamma_B)_C: \left< (A \rightarrow C) \rightarrow (B \rightarrow C) \Rightarrow C \right>$ which is continuous but not strict. The morphism $\gamma: A \oplus B \rightarrow \forall_X (A \rightarrow X) \Rightarrow (B \rightarrow X) \Rightarrow X$ is given by the choice of these two.

$$\gamma(x) = \text{ case } x \text{ of } \text{in}(a) \longmapsto \gamma_A a ; \text{inr}(b) \longmapsto \gamma_B b$$

The inverse mapping $\gamma^{-1}: \forall_X (A \rightarrow X) \Rightarrow (B \rightarrow X) \Rightarrow X \rightarrow A \oplus B$ comes from application to the injections $\text{inl}: A \rightarrow A \oplus B$ and $\text{inr}: B \rightarrow A \oplus B$.

$$\gamma^{-1}(p_C) = (p_{A \oplus B} \text{inl}) \text{inr}$$

The composite $\gamma^{-1} \circ \gamma$ is easily seen to be the identity directly, while $\gamma \circ \gamma^{-1} = \text{id}$ makes use of parametricity arguments. For any $(p_C)_C$, there is the following equality.

$$(\gamma \circ \gamma^{-1})(p_C) = \text{ case } (p_{A \oplus B} \text{inl}) \text{inr} \text{ of }$$

$$\text{inl}(a) \longmapsto \left< \lambda f. \lambda g. f(a) \right>_C$$

$$\text{inr}(b) \longmapsto \left< \lambda f. \lambda g. g(b) \right>_C$$

For any pointed CPO $D$ and strict functions $f:A \rightarrow D$ and $g:B \rightarrow D$, we
compute the result of the application to be the following.

\[
((\gamma \circ \gamma^{-1})(p_C)_C) D f g = \text{case } (p_{A \oplus B} \text{inl}) \text{inr of}
\]

\[
\begin{align*}
\text{inl}(a) \mapsto f(a) \\
\text{inr}(b) \mapsto g(b)
\end{align*}
\]

We define a strict relation \( R_{f,g} : A \oplus B \leftrightarrow D \) as follows.

\[
x [R_{f,g}] y \iff \text{case } x \text{ of } \text{inl}(a) \mapsto f(a) ; \text{inr}(b) \mapsto g(b) = y
\]

Showing that \( R_{f,g} \) is a complete relation makes use of the fact that directed subsets of \( A \oplus B \) must be completely within one of the two components. (This holds since non-bottom elements of \( A \) are incomparable to non-bottom elements of \( B \) in \( A \oplus B \).) It is immediate that both \( \text{inl } [I_{A \rightarrow \circ} R_{f,g}] f \) and \( \text{inr } [I_{B \rightarrow \circ} R_{f,g}] g \). Note that the parametricity of \( \langle p_C \rangle_C \) ensures the following relationship.

\[
p_{A \oplus B} \left[(I_{A \rightarrow \circ} R_{f,g}) \Rightarrow (I_{B \rightarrow \circ} R_{f,g}) \Rightarrow R_{f,g}\right] p_D
\]

Therefore, we get that \( (p_{A \oplus B} \text{inl}) \text{inr } [R_{f,g}] (p_D f \text{g}) \). Expanding the definition of \( R_{f,g} \) and abstracting away applications and instantiations gives the desired equality \( (\gamma \circ \gamma^{-1})(p_C)_C = (p_C)_C \).

Analogous statements can be made and proved for all well-pointed LNL settings for PLLC. Proving such results in the general setting would require arguments parameterized by points, as in System P (see Chapter 5). We have stated the above result for the more familiar and intuitive setting of \( \mathbf{CPO} \) merely to avoid the distraction of abstract generalities.

We were motivated to find a setting for a parametric model which supports recursion. We have exhibited some LNL settings with achieve this goal, such as one built around \( \mathbf{CPO} \). Since there is a rich structure of edges in parametricity graphs, the preservation of edges in these models provide good notions of uniformity.

LNL settings provide models of a linear lambda calculus rather than the (intuitionistic) setting originally discussed. We can return to an intuitionistic setting by merely restricting attention to the intuitionistic subset of PLLC (that is, where the linear zone of contexts are necessarily empty). Hence, the predicative polymorphic lambda calculus together with recursion can be modeled in \( \mathbf{CPO} \) — with terms interpreted as morphisms \( L[\Gamma] \rightarrow [A] \). By way of the parametric adjunction, these can be viewed
as morphisms $[\Gamma] \to [A]$ in $\text{CPO}$ (where $[\Gamma]$ and $[A]$ map into the image of $\text{CPO}_\perp$). One could consider the full (on morphisms and squares) sub-reflexive graph category of $\text{CPO}$ on (the vertices and edges) of $\text{CPO}_\perp$. This is precisely the reflexive graph category $\text{PCPO}$ over $\text{pcpo}$ mentioned previously. The preservation of edges does provide a reasonable notion of uniformity — the uniformity of $\text{CPO}_\perp$.

The reflexive graph category $\text{PCPO}$ is not a parametricity graph. It does not fit into our categorical axiomatization of parametricity, but we are able to show that preservation of edges in $\text{PCPO}$ is a good notion of uniformity. This situation is not surprising. When considering the category $\text{pcpo}$, Abramsky and Jung [AJ94] point out that one “quite often must resort to detailed proofs at the element level and cannot simply apply general category theoretic principles”. Some presentations completely ignore the category $\text{pcpo}$, preferring to work with the categories $\text{Cpo}$ and $\text{cpo}_\perp$, such as in [Plo83]. We do not see it as a weakness that $\text{PCPO}$ is not included in our categorical formulation of parametricity graphs.

### 7.5 Comparisons

The setting of pointed CPOs is a fairly familiar and canonical setting for programming semantics. However, having two fairly canonical notions of edges which treat the basic constructs $\otimes$ and $\multimap$ of linear logic differently, it is only natural to ask if these provide different notions of uniformity. While the full answer is not known, we show that there are many examples where the consequences of parametricity using relations coincide with the consequences of parametricity using spans. We show that the notions of parametricity using relations and spans agree for some second order types. This includes parametric transformations from an LHS type to an RHS type, where LHS and RHS are given by the following context-free grammar.

\[
\begin{align*}
\text{LHS} & := \text{order } 0 \mid \text{JA} \\
\text{JA} & := X \mid 1 \mid \text{!JA} \mid \text{order } 0 \multimap \text{JA} \\
\text{RHS} & := \text{order } 0 \mid \text{JAd} \mid \text{order } 0 \multimap \text{JAd} \mid \forall X. \text{RHS} \\
\text{JAd} & := X \mid 1 \mid \text{!X} \mid \text{JA} \multimap \text{JAd}
\end{align*}
\]

We are not satisfied with this limited state of understanding, and consider this an area for future research.

While the parametricity graph using pointed complete relations over $\text{cpo}_\perp$ seems like a natural adaptation of relations over $\text{SET}$, the more cate-
gorical formulation of spans is also attractive. Before entertaining thoughts of which is the more appropriate setting, it seems reasonable to ask if they are in effect providing different notions of uniformity. We approach this issue by first noting that every complete pointed relation determines a span. For any relation $R: A \leftrightarrow B$ of $\mathbf{CPO}_\bot$, we define the span $\{R\}: A \leftrightarrow B$ with witness set as follows.

$$W(\{R\}) = \{(a, b) \mid a \ [R] b\}$$

The above set is ordered pointwise and we use the obvious projections. It is easy to see this is a span of $\mathbf{Sp(cpo}_\bot)$. This defines the edge portion of a PG-functor $\{-\} : \mathbf{CPO}_\bot \rightarrow \mathbf{Sp(cpo}_\bot)$ that is the identity on vertices and morphisms.

Every span over pointed complete relations determines a pointed complete relation. Recall the underlying relation of a span.

$$a \ [U(S)] b \iff \exists c \in W(S). p_0(c) = a \land p_1(c) = b$$

This is not necessarily complete (although it is strict). We define $\text{comp} S$ to be the least complete relation containing $U(S)$. Obviously, for any span $\{R\}$ arising from a pointed complete relation, the underlying relation, namely $R$ itself, is already complete. The above construction $\text{comp}$ gives the edge portion of a PG-functor whose vertex and morphism portions are the identity. Since there is a span morphism from $S$ to $\{\text{comp} S\}$ over the identities, these PG functors form an adjunction $\mathbf{Sp(cpo}_\bot) \rightleftharpoons \mathbf{CPO}_\bot$ which is the identity on the vertex categories.

To compare the effects of parametricity constraints in these two parametricity graphs, we shall use models of PLLC as a common ground for talking about PG functors. We fix $G \rightarrowtail \hat{G}$ to be a small sub-LNL setting of $\mathbf{CPO}_\bot \rightarrowtail \mathbf{CPO}$. This defines not only a model of PLLC in $\mathbf{CPO}_\bot$, but also a corresponding span model in $\mathbf{Sp(cpo}_\bot)$ using $\mathbf{Sp(G}_\bot)$ for simple types. Superscripts will be used to distinguish the interpretation using relations $[-]^R$ from the interpretation using spans $[-]^S$.

By virtue of the adjunction between $\mathbf{Sp(cpo}_\bot)$ and $\mathbf{CPO}_\bot$, every parametric transformation $p: F \rightarrow G$ in $\mathbf{Sp(cpo}_\bot)$ is also a parametric transformation $p: (\text{comp} \circ F \circ \{\}) \rightarrow (\text{comp} \circ G \circ \{\})$ in $\mathbf{CPO}_\bot$. (Conversely, every parametric transformation in $\mathbf{CPO}_\bot$ is also a parametric transformation in $\mathbf{Sp(cpo}_\bot)$.) However, this is not what we want. Note that some constructions like ones involving the tensor product $\otimes$ and additive function space

241
\( \neg \circ \) are not preserved by the adjunction.

\[
[X \vdash X \xrightarrow{\circ} (X \otimes X)]^R \neq \text{comp} \circ [X \vdash X \xrightarrow{\circ} (X \otimes X)]^S \circ \{\}
\]

To demonstrate this, consider the pointed complete relation \( R: \mathbb{N} \rightarrow \circ \mathbb{N} \) given as follows.

\[
R = \{ (n, n) \mid n \in \mathbb{N} \} \cup \{ (\bot, \bot), (\bot, 1) \}
\]

Consider the constant function \( f \) in \( \mathbb{N} \rightarrow \circ (\mathbb{N} \otimes \mathbb{N}) \) that sends each \( n \) to \( \bot \). There is also a nearly constant function \( f' \) in \( \mathbb{N} \rightarrow \circ (\mathbb{N} \otimes \mathbb{N}) \) defined as follows.

\[
f'(\bot) = \bot \quad f'(n) = [1, \omega] \quad \text{for } n \neq \bot
\]

It is the case that \( f \left[ [X \vdash X \xrightarrow{\circ} (X \otimes X)]^R \right] f' \). However \( f \) and \( f' \) are not related by \( \text{comp}( [X \vdash X \xrightarrow{\circ} (X \otimes X)]^S \{R\}) \).

Simply appealing to the adjunction to say every parametric transformation in one system is a parametric transformation in the other does not necessarily preserve the type of the parametric transformation. We are more interested in results like “all parametric transformations \( p:T(X) \rightarrow X \otimes X \) in \( \text{CPO}_\bot \) are parametric transformations \( p:T(X) \rightarrow X \otimes X \) in \( \text{Sp}(\text{cpo}_\bot) \)” using the corresponding parametric co-monad \( T \) and tensor product for the respective parametricity graph. The previously mentioned representation results translate into results of this kind using the correspondence between parametric transformations and parametric limits.

**Lemma 7.10**

For any non-variant functor \( F: G \rightarrow \text{CPO}_\bot \), the parametric limit \( \forall Y F(Y) \) in \( \text{CPO}_\bot \) is isomorphic to the pointed CPO of parametric transformations \( \tau : I \rightarrow F \), ordered pointwise.

For any non-variant functor \( \overline{F}: G \rightarrow \text{Sp}(\text{cpo}_\bot) \), the parametric limit \( \forall Y \overline{F}(Y) \) in \( \text{Sp}(\text{cpo}_\bot) \) is isomorphic to the pointed CPO of parametric transformations \( \tau : I \rightarrow \overline{F} \), ordered pointwise.

For example, this shows that the collection of parametric transformations \( q: I \rightarrow [X \vdash X \xrightarrow{\circ} X]^R \) in \( \text{CPO}_\bot \) is the same as the collection of parametric transformations \( q: I \rightarrow [X \vdash X \xrightarrow{\circ} X]^S \) in \( \text{Sp}(\text{cpo}_\bot) \) since both are the same as the CPO \( I \) by the representation result \( [\forall X, X \xrightarrow{\circ} X] \cong I \). (This argument actually shows that they are isomorphic, due in part to the “unique up to isomorphism” characterization of parametric limits. Using the actual choice

242
of parametric limits as parametric families in both parametricity graphs does allow one to exhibit this as an actual equality.

Such an exact correspondence does rely on the fact that $[X \vdash X \rightarrow X]^R_A$ and $[X \vdash X \rightarrow X]^S_A$ are the same (for any pointed CPO $A$). This ensures that the components of a parametric transformation in one setting are at least of the right type to form a parametric transformation in the other setting. For any simple types, it is immediate that the interpretations in $\text{CPO}_\bot$ and $\text{Sp}(\text{cpo}_\bot)$ agree on CPOs, since all the simple type constructors are defined as extensions to the functors on $\text{cpo}_\bot$.

In some cases one can show that the interpretations of polymorphic types are the same on pointed CPOs in both settings. The above lemma lets us show this by way of the equality of parametric transformations into the unquantified type. In presenting the following results that lead to the equality of parametric transformations, we do include polymorphic types in the statements of the lemmas and theorems. In many cases, the stated lemma or theorem does rely on the fact that the interpretation of type judgments for these polymorphic types are the same on CPOs in both systems. Therefore, the induction proofs necessary for most of the lemmas and theorems in the rest of the section need to be proved by simultaneous induction with each other.

Existential types are built from the collection of compatible families both in $\text{CPO}_\bot$ and $\text{Sp}(\text{cpo}_\bot)$. If one could show that the collections of compatible families was the same in the two settings, then one could conclude that the interpretations of existential types are the same. While the equality of parametric transformations into $[\eta, X \vdash \phi]$ does not itself imply that the interpretation of $\eta \vdash \exists X.\phi$ is the same for all pointed CPOs, some of the intermediate lemmas we use to establish that the parametric transformations are the same do suffice to show the compatible families are the same. Therefore we shall address the equality of existential types after addressing the equality of parametric transformations, knowing it too will be part of the simultaneous induction.

As a result of having $\llbracket \phi \rrbracket^R_{\mathcal{T}} = \llbracket \phi \rrbracket^S_{\mathcal{T}}$ for all pointed CPOs $\mathcal{T}$ for the types $\eta \vdash \phi$ that we consider, we shall not bother with the distinguishing superscript when applying these PG-functors to vertices.

In considering whether parametricity in $\text{CPO}_\bot$ differs from parametricity in $\text{Sp}(\text{cpo}_\bot)$ at a given type, it will be relevant to consider how deeply $\rightarrow$’s are nested in that type. It is customary to classify types into orders, counting the depth of $\rightarrow$’s. The types of a given order are defined inductively, according to the structure of the types.

243
Definition 7.11

- $X$ is order 0 for any type variable $X$.
- If $\phi$ is order $n$, then $!\phi$ is order $n$.
- If $\phi_1$ and $\phi_2$ are both order $n$, then $\phi_1 \otimes \phi_2$ is order $n$.
- If $\phi$ is order $n$, then $\forall X. \phi$ and $\exists X. \phi$ are both order $n$.
- If $\phi_1$ is order $n$ and $\phi_2$ is order $n+1$, then $\phi_1 \rightarrow \phi_2$ is order $n+1$.
- If $\phi$ is order $n$, then $\phi$ is order $n+1$ as well.

Observe that the order of a function type is not necessarily greater than the order of its codomain. A consequence of this is that the isomorphic types $\phi_1 \otimes \phi_2 \Rightarrow \phi_3$ and $\phi_1 \Rightarrow (\phi_2 \Rightarrow \phi_3)$ are of the same order. Since we will mainly be interested in those types that are interpreted the same on pointed CPOs (which may or may not include all types), we will simplify the discussion by including this criteria in our classification scheme.

Definition 7.12

A type $\phi$ is said to be level $n^*$ if and only if it is order $n$ and the equality $[\phi]^R \tilde{A} = [\phi]^S \tilde{A}$ holds for all $|\eta|$-tuples of pointed CPOs $\tilde{A}$.

Appealing to the adjunction to translate parametric transformations from one setting to the other is useful when the adjunction does commute with the interpretation of types. In some simple instances, one can show that the adjunction does preserve constructions. This is the case when there are no function types involved.

Lemma 7.13

Suppose that $\phi$ is an order 0 type. For any spans $\overline{S} \in \text{Sp}(\text{cpo}_\perp)^n$, it is the case that $\text{comp}([\phi]^S \overline{S}) = [\phi]^R \text{comp}(\overline{S})$.

A consequence of this lemma is that $\text{comp} \circ [\phi]^S \circ \{ \}^R = [\phi]^R$ (since $\overline{\text{comp}(R)} = \overline{R}$). An intuitive description of the above lemma is that it does not matter if you take the completion early or late in computing those types. This does rely on the absence of function types, as functions could map into the completion, and it is not obvious that continuous functions could be defined to approximate it. This problem does not arise when the witness set $W([\phi]^S \overline{R})$ contains all the data (including order) of $[\phi]^R \overline{R}$. We therefore also identify those types that commute with $\{ - \}$, that is, those types such that there is an isomorphism $[\phi]^S \overline{R} \cong [\phi]^R \overline{R}$ over the identities for every $|\eta|$-tuple of additive relations $\overline{R}$. For notational brevity,

244
we presume that assertions of edges $S$ and $S'$ being isomorphic includes the stipulation that the isomorphism $\sigma: S \rightarrow S'$ and in $G_e$ lie over identities, that is, $\partial_0(\sigma) = \text{id}_{\partial_0(S)}$ and $\partial_1(\sigma) = \text{id}_{\partial_1(S)}$. By asserting the commutativity of a diagram of PGfunctors, we intend that the vertex portion commutes exactly and the edge portion commutes up to isomorphism.

**Lemma 7.14**

- The interpretations of $\eta \vdash I$ commute with $\{\}$.  
- The interpretations of $\eta \vdash X$ commute with $\{\}$ for every type variable $X$.  
- The interpretations of $\eta \vdash !\phi$ commute with $\{\}$ if the interpretations of $\eta \vdash \phi$ do.  
- The interpretations of $\eta \vdash \phi_1 \circ \phi_2$ commute with $\{\}$ if there is the equality $\text{comp} \circ [\eta \vdash \phi_1]^S \circ \{\}^n = [\eta \vdash \phi_1]^R$ and the interpretations of $\eta \vdash \phi_2$ commute with $\{\}$.

For brevity, we shall suppress mention of the contexts when stating interpretations of PLLC judgments.

**Proof.** For each of the assertions about a type judgment $\eta \vdash \phi$, it suffices to show $W([\phi]^S \overline{R}) \cong \{x, y \mid x [\phi]^R y\}$, ordered pointwise, and this isomorphism commutes with the projections. We show this for this interesting case of $\phi_1 \circ \phi_2$. Recall the $\circ$ PG-functor on $\textbf{Sp(cpo}_+\text{)}$.

\[
W([\phi_1 \circ \phi_2]^S \overline{R}) = \left\{ (f, f_0, f_1) \mid \text{the following commutes} \right\}
\]

\[
\begin{array}{c}
\phi_1 \hspace{1cm} \phi_2 \\
\downarrow{\phi_1} \hspace{1cm} \downarrow{\phi_2} \\
\phi_1 \circ \phi_2 \hspace{1cm} \phi_1 \circ \phi_2 \\
\end{array}
\]

(1)

\[
W([\phi_1]^S \overline{R}) \xrightarrow{f} W([\phi_2]^S \overline{R})
\]

The assumption placed on $\phi_2$ ensures the existence of an isomorphism $\text{iso}: W([\phi_2]^R \overline{R}) \rightarrow W([\phi_2]^S \overline{R})$ that is the witness of a square over identity morphisms. Thus, for any $(f, f_0, f_1)$ in $W([\phi_1 \circ \phi_2]^S \overline{R})$, it follows that $f(c) = \text{iso} \left( f_0(p_0(c)), f_1(p_1(c)) \right)$ for all $c \in W([\phi_1]^S \overline{R})$. Thus, for any $x [\phi_1]^S \overline{R} y$, we get $f_0(x) [\phi_2]^R \overline{R} f_1(x)$. Since $f_0$ and $f_1$ are continuous, and $[\phi_2]^R \overline{R}$ is continuous, this extends to a square of the following
shape.

\[
\begin{aligned}
  \phi_1 \xrightarrow{\phi_2} \phi_3 \xrightarrow{f_0} \phi_4 \xrightarrow{f_1} \phi_5
\end{aligned}
\]

Using the hypothesis \(\text{comp} \circ [\phi_1]_S \circ \{-\}^n = [\phi_1]_R\), the above square of \(\text{CPO}_\bot\) gives us that \(\phi_0 \circ [\phi_1 \circ \phi_2]_R \circ [\phi_2]_R \circ f_1\).

Conversely, for any \(\phi_0 \circ [\phi_1 \circ \phi_2]_R \circ f_1\), we have the above parametricity square, (2). We define \(f(c) = \text{iso}(f_0(p_0(c)), f_1(p_1(c)))\). It is apparent this is a strict continuous function \(f: W([\phi_1]_S \circ [\phi_2]_R) \rightarrow W([\phi_2]_S \circ [\phi_2]_R)\) and that diagram (1) commutes. Therefore, \((f, f_0, f_1) \in W([\phi_1 \circ \phi_2]_S \circ [\phi_2]_R)\). Since the order of \(W([\phi_1 \circ \phi_2]_S \circ [\phi_2]_R)\) is defined to be component-wise, this does define a strict continuous function.

It is not difficult to show that these mappings are inverses, and commute with the projections. Therefore the interpretations of \(\eta \vdash \phi_1 \circ \phi_2\) commute with \(\{-\}\).

This provides another collection of types \(\phi\) such that we can prove \(\text{comp} \circ [\phi_1]_S \circ \{-\}^n \equiv [\phi]_R\) (since \(\text{comp} \circ \{-\} = \text{ID}\)). For these types, we have that parametricity in \(\text{Sp(cpo}_\bot\) implies parametricity in \(\text{CPO}_\bot\).

**Theorem 7.15**

Suppose that \(\text{comp} \circ [\phi_i]_S \circ \{-\}^n\) is isomorphic to \([\phi_i]_R\) for \(i \in \{1, 2\}\). Any parametric transformation \(p: [\phi_1]_S \rightarrow [\phi_2]_S\) is also a parametric transformation \(p: [\phi_1]_R \rightarrow [\phi_2]_R\).

**Proof.** By composition, we get that the following is a parametric transformation.

\[
\text{comp} \circ p \circ \{\}^n: (\text{comp} \circ [\phi_1]_S \circ \{\}^n) \rightarrow (\text{comp} \circ [\phi_2]_S \circ \{\}^n)
\]

Since \(\{\}\) and \(\text{comp}\) are both the identity on the vertex category, we have that \(\text{comp} \circ p \circ \{\}^n = p\). By pre-/post-composition with the isomorphism/inverse of the isomorphism from the hypotheses, we conclude that \(p\) is a parametric transformation \(p: [\phi_1]_R \rightarrow [\phi_2]_R\), as desired.

A slightly different approach is taken when proving that parametricity in \(\text{CPO}_\bot\) implies parametricity in \(\text{Sp(cpo}_\bot\). The basic idea is based on translating span morphisms \((h, f_0, f_1)\) into pairs of parametricity squares in \(\text{CPO}_\bot\). This is accomplished via treating the witness set of a span as a
pointed CPO. We use the fact that for any order 0 type $\phi$, the interpretation $[\phi]^R$ can be extended to a PG-functor $[[\phi]]^*$. This is immediate since each of the type constructors for level 0 types was interpreted by restricting a PG-functor to a non-variant functor.

**Proposition 7.16**
Suppose $\phi$ is an order 0 type. For any spans $\overline{S}:\overline{A} \leftrightarrow \overline{B} \in \text{Sp}(\text{cpo}_\perp)^n$, there is an isomorphism (in $\text{cpo}_\perp$) $m: W([[\phi]^S\overline{S}]) \cong [\phi] W(\overline{S})$ that commutes with projections as follows.

$$\begin{align*}
W([[\phi]^S\overline{S}]) & \xrightarrow{m} [\phi] W(\overline{S}) \\
p_0 & \xrightarrow{p_0} [\phi] \overline{A} \\
p_1 & \xrightarrow{p_1} [\phi] \overline{B}
\end{align*}$$

For the case of variables, the mediating morphism is the identity. Since the PG-functors $\otimes$ and $T$ each commute with the witness mapping from edges to vertices of $\text{Sp}(\text{cpo}_\perp)$, the $\phi_1 \otimes \phi_2$ and $! \phi$ cases are immediate as well. The universal properties of parametric limits and parametric colimits provide the mediating morphism for the case of polymorphic types and existential types, respectively.

Once one starts considering function spaces, one can no longer hope for an isomorphism. The function space $W(S) \to W(T)$ contains all strict continuous functions from $W(S)$ to $W(T)$, even the ones that can not possibly commute with any strict continuous functions $\partial_0(S) \to \partial_0(T)$ and $\partial_1(S) \to \partial_1(T)$. But a witness in $W(S \to T)$ does contain a strict continuous function $W(S) \to W(T)$. Thus there is still a strict continuous function $W([[\phi]^S\overline{S}]) \to [\phi] W(\overline{S})$ for all level 1* types.

**Lemma 7.17**
For any level $1^*$ type $\phi$ and any spans $\overline{S}:\overline{A} \leftrightarrow \overline{B} \in \text{Sp}(\text{cpo}_\perp)^n$, there is a strict continuous function $m: W([[\phi]^S\overline{S}]) \to [\phi] W(\overline{S})$ such that there are parametricity squares in $\text{CPO}_\perp$ as follows.

$$\begin{align*}
W([[\phi]^S\overline{S}]) & \xrightarrow{m} [\phi] W(\overline{S}) \\
p_0 & \xrightarrow{p_0} [\phi] \overline{A} \\
p_1 & \xrightarrow{p_1} [\phi] \overline{B}
\end{align*}$$

**Proof.** This is shown by induction on the structure of the type expression. We only discuss the case of function types $\lambda_1 \to \phi_2$, as the other cases are
similar to the isomorphism in the previous proposition.

For \( \phi_1 \to \phi_2 \) types, the mediating morphism is constructed as follows. A witness \((f, f_0, f_1) \in W([\phi_1 \to \phi_2]^{S_\mathcal{S}})\) provides a strict continuous function \(f \in W([\phi_1]^{S_\mathcal{S}}) \to W([\phi_2]^{S_\mathcal{S}})\). The induction hypothesis provides a strict continuous function \(m_2: W([\phi_2]^{S_\mathcal{S}}) \to [\phi_2]^{W(S)}\). Since \(\phi_1\) is necessarily a order 0 type, proposition 7.16 ensures there is an isomorphism \(i_1: W([\phi_1]^{S_\mathcal{S}}) \to [\phi_1]^{W(S)}\). The mediating morphism \(m\) maps the witness \((f, f_0, f_1)\) to the composite \(m_2 \circ f \circ i_1^{-1}: [\phi_1]^{W(S)} \to [\phi_2]^{W(S)}\). Since each \(i_1^{-1}\) and \(m_2\) commutes with the projections, the mapping \(m\) commutes with projections as well. Since orders are given pointwise, the mapping \(m\) is a strict continuous function.

Having the witness mapping commute with the type constructors allows us to show that parametric transformations in \(\mathbf{CPO}_\bot\) are indeed parametric transformations in \(\mathbf{Sp} (\mathbf{cpo}_\bot)\).

**Theorem 7.18**

Suppose \(\phi_1\) is an level 1* type and \(\phi_2\) is an level 0* type. Any parametric transformation \(q: [\phi_1]^R \to [\phi_2]^R\) is also a parametric transformation \(q: [\phi_1]^S \to [\phi_2]^S\).

**Proof.** For any spans \(S: \overline{A} \leftrightarrow \overline{B}\), the parametricity in \(\mathbf{CPO}_\bot\) of \(q\) ensures that there are parametricity squares as indicated below.

\[
\begin{array}{ccc}
\|\phi_1\|^{W(S)} & \overset{q_{W(S)}}{\longrightarrow} & \|\phi_2\|^{W(S)} \\
\downarrow [\phi_1]^R(p_0) & & \downarrow [\phi_2]^R(p_0) \\
[\phi_1]A & \overset{q_A}{\longrightarrow} & [\phi_2]A \\
\end{array}
\]

\[
\begin{array}{ccc}
\|\phi_1\|^{W(S)} & \overset{q_{W(S)}}{\longrightarrow} & \|\phi_2\|^{W(S)} \\
\downarrow [\phi_1]^R(p_1) & & \downarrow [\phi_2]^R(p_1) \\
[\phi_1]B & \overset{q_B}{\longrightarrow} & [\phi_2]B \\
\end{array}
\]

Let \(m_1: W([\phi_1]^{S_\mathcal{S}}) \to [\phi_1]^{W(S)}\) be the mediating morphism of lemma 7.17 and \(m_2: W([\phi_2]^{S_\mathcal{S}}) \to [\phi_2]^{W(S)}\) be the isomorphism from proposition 7.16. The commuting diagrams in lemma 7.17 and proposition 7.16 can give rise to parametricity squares via the subsumption property. One such square can be composed with a parametricity square above to get the following
square.

\[
\begin{array}{ccc}
W([\phi_1]S) & \xrightarrow{m_1} & [\phi_1]W(S) & \xrightarrow{q_{W(s)}} & [\phi_2]W(S) \\
 I & \Rightarrow & [\phi_1]^R(p_0) & \Downarrow & [\phi_2]^R(p_0) \\
W([\phi_1]S) & \xrightarrow{p_0} & [\phi_1]A & \xrightarrow{q_A} & [\phi_2]A
\end{array}
\]

Similarly, there is the corresponding square involving \(p_1\) and \(B\). Since the parametricity graph \(\text{CPO}_\bot\) is subsumptive, we can conclude the following.

\[
[\phi_2]^R p_0 \circ q_{W(S)} \circ m_1 = q_A \circ p_0 \\
[\phi_2]^R p_1 \circ q_{W(S)} \circ m_1 = q_B \circ p_1
\]

Since \([\phi_2]^R p_0 = p_0 \circ m_2^{-1}\) and \([\phi_2]^R p_1 = p_1 \circ m_2^{-1}\) from proposition 7.16, we get a pair of commuting diagrams, below.

\[
\begin{array}{ccc}
[\phi_1]A & \xrightarrow{q_A} & [\phi_2]A \\
W([\phi_1]S) & \xrightarrow{m_2^{-1} \circ q_{W(S)} \circ m_1} & W([\phi_2]S) \\
[\phi_1]B & \xrightarrow{q_B} & [\phi_2]B
\end{array}
\]

Since there is such a witness morphism for all spans, \(S\), \(q\) is parametric.

We now turn our attention to parametric colimits, that is, the interpretation of existential types. If \(\eta, X \vdash \phi\) is a type that we already know is interpreted the same on pointed CPOs in \(\text{CPO}_\bot\) and \(\text{Sp(cpo}_\bot)\), then for all \(|\eta| + 1\)-tuples of pointed CPOs \((A_1, \cdots A_n, B)\), the non-empty complete downward closed subsets of \([\eta, X \vdash \phi](A_1, \cdots A_n, B)\) are the same as well. The next step is to show that those families \(\langle S_B \rangle_B\) that are compatible with \([\eta, X \vdash \phi]^R(A_1, \cdots A_n, Z)\) as a PG-functor on \(\text{CPO}_\bot\) in the variable \(Z\) are the same as those families that are compatible with \([\eta, X \vdash \phi]^S(A_1, \cdots A_n, Z)\) on \(\text{Sp(cpo}_\bot)\).

**Lemma 7.19**

If \(\eta, X \vdash \phi\) is a level 1* type, and \(\overline{A}\) is an \(|\eta|\)-tuple of pointed CPOs, then every family \(\langle S_B \rangle_B\) compatible with \([\eta, X \vdash \phi]^R(A_1, \cdots A_n, Z)\) on \(\text{CPO}_\bot\) is compatible with \([\eta, X \vdash \phi]^S(A_1, \cdots A_n, Z)\) on \(\text{Sp(cpo}_\bot)\).

**Proof.** Suppose that the elements \(x \in [\eta, X \vdash \phi](A_1, \cdots A_n, B_0)\) and \(y \in [\eta, X \vdash \phi](A_1, \cdots A_n, B_1)\) are such that there is a span \(S: B_1 \leftrightarrow B_0\) with \(x \left[ U([\eta, X \vdash \phi]^S(1A_1, \cdots 1A_n, S)] \right] y\). (We wish to show \(x \in S_{B_0}\) holds if and
only if \( y \in S_{B_1} \) holds.) From the definition of the underlying relation of a span, there is some \( c \in W([\eta, X \vdash \phi]^S(I_{A_1}, \cdots I_{A_n}, S)) \) with \( p_0(c) = x \) and \( p_1(c) = y \). Using the \( m \) from \( W([\eta, X \vdash \phi]^S(I_{A_1}, \cdots I_{A_n}, S)) \) to \([\eta, X \vdash \phi]|_{I_{A_1}, \cdots I_{A_n}, W(S)}\) asserted to exist by lemma 7.17, we have the following relationships.

\[
m(c) \left[ [\eta, X \vdash \phi]^R(p_0) \right] x \quad \text{and} \quad m(c) \left[ [\eta, X \vdash \phi]^R(p_1) \right] y.
\]

Since \( p_0 = p_1 = \text{id}_A \) for the identity span on any vertex \( A \), it is the case that \([\eta, X \vdash \phi]^R(p_0) = [\eta, X \vdash \phi]^R(\text{id}_{A_1}, \cdots \text{id}_{A_n}, p_0)\) (and similarly for \( p_1 \)). Since \( \langle S_B \rangle_B \) is compatible with \([\eta, X \vdash \phi]^S(A_1, \cdots A_n, Z)\), we get the following.

\[
x \in S_{B_0} \iff m(c) \in S_{W(S)} \iff y \in S_{B_1}
\]

Since \( x, y \) and \( S \) were arbitrary, the family \( \langle S_B \rangle_B \) is compatible with the PG-functor \([\eta, X \vdash \phi]^S(A_1, \cdots A_n, Z)\) as desired.

\* Lemma 7.20 *

For any \( \eta, X \vdash \phi \) such that \( \text{comp} \circ [\eta, X \vdash \phi]^S \circ [\eta, X \vdash \phi]^R \) for every \( n \)-tuple of pointed CPOs \( \overline{A} \), every family \( \langle S_B \rangle_B \) that is compatible with the PG-functor \([\eta, X \vdash \phi]^S(A_1, \cdots A_n, Z)\) on \( \text{Sp(cpo}_\perp) \) is also compatible with \([\eta, X \vdash \phi]^R(A_1, \cdots A_n, Z)\) on \( \text{CPO}_\perp \).

**Proof.** For any relation \( R: B_0 \leftrightarrow B_1 \), \( x \in [\eta, X \vdash \phi](A_1, \cdots A_n, B_0) \) and \( y \in [\eta, X \vdash \phi|(A_1, \cdots A_n, B_1) \) such that \( x \left[ [\eta, X \vdash \phi]^R(I_{A_1}, \cdots I_{A_n}, R) \right] y \), it is the case that \( x \left[ \text{comp}([\eta, X \vdash \phi]^S(I_{A_1}, \cdots I_{A_n}, \{R\})) \right] y \). Since there is an isomorphism \( (\text{id}_{A_i}, \text{id}_{A_j}):(I_{A_i} \to I_A) \) in \( \text{Sp(cpo}_\perp) \) for any pointed CPO \( A \), we have \( x \left[ \text{comp}([\eta, X \vdash \phi]^S(I_{A_1}, \cdots I_{A_n}, \{R\})) \right] y \).

Therefore, there are directed sets \( \{x_i\}_I \) and \( \{y_i\}_I \) with \( \bigcup_i x_i = x \), \( \bigcup_i y_i = y \) and \( \forall i \in I, x_i \left[ U([\eta, X \vdash \phi]^S(I_{A_1}, \cdots I_{A_n}, \{R\})) \right] y_i \). This gives that \( x \) is in \( S_{B_0} \) if and only if \( y \) is in \( S_{B_1} \).

\[
x \in S_{B_0} \iff \forall i \in I, x_i \in S_{B_0} \iff \forall i \in I, y_i \in S_{B_1} \iff y \in S_{B_1}
\]

Since \( R, x \) and \( y \) were arbitrary, \( \langle S_B \rangle_B \) is compatible with the PG-functor \([\eta, X \vdash \phi]^R(A_1, \cdots A_n, Z)\).

Once it is established that the collections of compatible families are the same, it is not difficult to see that the interpretations of existential types are the same. Since \( \langle b \rangle \) the least compatible family of non-empty complete downward closed subsets containing \( b \in [\eta, X \vdash \phi](A_1, \cdots A_n, B) \) is the same in both settings, the parametric colimits are the same. (In both settings,
the colimits are built as the completion of the same initial approximation \( \{ \{ b \} \mid b \in \eta, X \vdash \phi \{ A_1, \ldots, A_n, B \} \) for some \( B \).) 

The agreement of the interpretations of polymorphic types on pointed CPOs is a direct consequence of the coincidence of parametric transformations. Suppose \( \eta, X \vdash \phi \) is such that \( p: 1 \to [\eta, X \vdash \phi] \) is parametric in \( \text{CPO}_\perp \) if and only if \( p: 1 \to [\eta, X \vdash \phi]_S \) is parametric in \( \text{Sp}(\text{cpo}_\perp) \). It follows from lemma 7.10 that \( [\eta \vdash \forall X. \phi]_S^\text{\text{CPO}_\perp} = [\eta \vdash \forall X. \phi]_S^\text{\text{Sp}(\text{cpo}_\perp)} \) for all \( |\eta| \)-tuples of pointed CPOs \( \overline{A} \).

We have characterized some types for which we know parametricity in \( \text{CPO}_\perp \) implies parametricity in \( \text{Sp}(\text{cpo}_\perp) \). We have also characterized some types for which we know the converse. The intersection of these two collections is at present rather limited. We can see that the parametric transformations between order 0 terms is the same. By currying, this yields the equivalence of the interpretations of \( \forall X. \phi_1 \equiv \phi_2 \) for any order 0 types \( \phi_1 \) and \( \phi_2 \). Repeated currying allows some nesting of \( \equiv \)'s in the codomain, but this is still significantly weaker than having the equivalence of interpretations of \( \forall X. \phi \) for all order 1 types \( \phi \). We can extend to some order 2 types, provided that the domains use little more that just \( \equiv \).

**Definition 7.21**

The collection of just arrow (or JA) types are given by the following context free grammar.

\[
\text{JA} \quad ::= \quad X \mid 1 \mid \text{!JA} \mid \text{order 0} \equiv \text{JA}
\]

The JA types are those order 1 types \( \phi \) for which we have shown \( \phi \) commutes with \( \{ \} \). While this will allow one to show the equality of the interpretations for some order 2 polymorphic types, such as \( \forall (X \Rightarrow X) \equiv X \) (the type of the fixed point operator), it is not powerful enough to include some of the anticipated results (from knowing the representation results), for instance, \( \forall X. (Y \equiv X) \Rightarrow (Z \equiv X) \Rightarrow X \) (which we know is \( Y \equiv Z \) in both parametricity graphs). This can be accounted for by currying JA types into the codomain.

The just arrow domain (or JAd) types consist of types having just arrow types in the domain of function types. They are formally specified by the following context free grammar.

\[
\text{JAd} \quad ::= \quad X \mid 1 \mid \text{!X} \mid \text{JA} \equiv \text{JAd}
\]

The types for which we have shown parametricity in \( \text{CPO}_\perp \) is the same
as parametricity in $\text{Sp}(\text{cpo}_\perp)$ can be summarized by the following context-free grammar.

$$
\begin{align*}
\text{LHS} & := \text{order 0} \mid \text{JA} \\
\text{JA} & := X \mid 1 \mid \text{JA} \mid \text{order 0} \rightarrow \text{JA} \\
\text{RHS} & := \text{order 0} \mid \text{JA} \mid \text{order 0} \rightarrow \text{JA} \mid \forall X. \text{RHS} \\
\text{JA} & := X \mid 1 \mid !X \mid \text{JA} \rightarrow \text{JA} \\
\text{RHS}' & := \text{RHS} \mid \text{order 0} \rightarrow \text{RHS}' \mid \forall X. \text{RHS}'
\end{align*}
$$

**Theorem 7.22**

If $\phi_1$ is a LHS type and $\phi_2$ is a RHS type, then any $p$ is a parametric transformation $p: [\phi_1]^{R} \rightarrow [\phi_2]^{R}$ in $\text{CPO}_\perp$ if and only if it is a parametric transformation $p: [\phi_1]^{S} \rightarrow [\phi_2]^{S}$ in $\text{Sp}(\text{cpo}_\perp)$.

The same equivalence between parametric transformations $p: [\phi_1] \rightarrow [\phi_2]$ holds whenever $\phi_1$ is order 0 and $\phi_2$ is a RHS' type.

**Proof.** It is easy to see that all LHS types $\phi_1$ are both order 1 and satisfy $\text{comp} \circ [\phi_1]^{S} \circ \{\}^{n} = [\phi_1]^{R}$. The proof proceeds by induction on the structure of $\phi_2$.

If $\phi_2$ is order 0, then $\text{comp} \circ [\phi_2]^{S} \circ \{\}^{n} = [\phi_2]^{R}$, and Theorems 7.15 and 7.18 apply to establish the equivalence. This includes the base cases of $\phi_2$ being a JA type.

If $\phi_2$ is either order 0 $\rightarrow$ JA or JA $\rightarrow$ JA, say $\phi_3 \rightarrow \phi_4$, then it commutes with $\{\}$. Therefore parametricity in $\text{Sp}(\text{cpo}_\perp)$ implies parametricity in $\text{CPO}_\perp$. By currying, we can show that the parametricity in $\text{CPO}_\perp$ of $\text{uncurry}(p): [\phi_1 \otimes \phi_3]^{R} \rightarrow [\phi_1]^{R}$ implies the parametricity in $\text{Sp}(\text{cpo}_\perp)$ of $\text{uncurry}(p): [\phi_1 \otimes \phi_3]^{S} \rightarrow [\phi_1]^{S}$. This uses a slightly stronger induction hypothesis.

**Lemma 7.23**

If $\phi_1$ is order 1 and $\phi_4$ is a JA type, then $q: [\phi_1]^{R} \rightarrow [\phi_4]^{R}$ is parametric in $\text{CPO}_\perp$ implies that $q: [\phi_1]^{S} \rightarrow [\phi_4]^{S}$ is parametric in $\text{Sp}(\text{cpo}_\perp)$.

If $\phi_2 = \forall X. \phi_3$, then by induction, we know that parametric transformations $q: [\eta \vdash \phi_1] \rightarrow [\eta, X \vdash \phi_3]$ are the same in both settings. The equivalence with parametric transformations $\Lambda(q): [\eta \vdash \phi_1] \rightarrow [\eta \vdash \forall X. \phi_3]$ from the parametric limit gives the desired correspondence.

If $\phi_2 = \phi_3 \rightarrow \phi_4$ where $\phi_3$ is order 0 and $\phi_4$ is a RHS' type, then we uncurry to uncurry($p$): $[\phi_1 \otimes \phi_3] \rightarrow [\phi_4]$. Since $\phi_1 \otimes \phi_3$ is order 0 and $\phi_4$ is RHS', the induction hypothesis gives the desired correspondence.
The above correspondence between parametric results in $\mathbf{CPO}_\bot$ and $\mathsf{Sp}(\mathbf{cpo}_\bot)$ was stated for certain non-variant functors arising from the types of PLLC. This should not be taken as an indication that one is only interested in comparing parametricity for those type constructors. The collection of non-variant functors arising from type judgments of PLLC is merely convenient starting point for collections non-variant functors in the two systems that correspond to each other. One could also consider additional non-variant functors (or PG-functors) that intuitively capture the same idea. Constructions that are defined 2-categorically, such as products and coproducts, provide additional candidates. Both products and coproducts are definable in terms of $\sim$ and $\forall$, and therefore could arise as consequences of the linear constructions. The above characterization does include coproducts of some functors. However, by treating coproducts as primitives, we can directly show that all the analysis for order 0 types is closed under coproducts. Therefore coproducts can be included as order 0 types (rather than order 2) in the above characterization. Similarly, one can directly show products may be used as order 0 constructions are used in the above characterizations rather than considering them as order 4 that arises from the encodings.

Of course, the limitations of the correspondence between parametric transformations one can show will depend on the particular non-variant functors. Not surprisingly, it is function-like constructions, such as $\sim\sim$ (and would include $\Rightarrow$), that primarily raise difficulties while comparing $\mathbf{CPO}_\bot$ and $\mathsf{Sp}(\mathbf{cpo}_\bot)$. Even limiting just to transformations between order 1 types, it is not clear whether parametricity in $\mathbf{CPO}_\bot$ is the same as parametricity in $\mathsf{Sp}(\mathbf{cpo}_\bot)$. While the proof methods considered so far have not been able to extend consistently to higher types, neither has a counter example been found to exhibit that they are indeed different.

The desire to describe parametric models that support recursion led us to consider parametric models of PLLC, a polymorphic linear lambda calculus. LNL settings were defined, describing parametricity graphs that can be used to model PLLC. A collection of representation results hold in well-pointed LNL settings, exhibiting the strength of parametricity. We defined LNL settings using pointed CPOs and also using bounded-complete partial orders that model PLLC plus recursion. These give parametric models with recursion. Since relations and spans give two canonical parametricity graphs for modeling PLLC plus recursion, we considered whether or not these produced significantly different models or not. No definitive answer has been found, leaving this open as an area for future research.

253
Chapter 8

Conclusions

The aim of this thesis is to give a categorical axiomatization of parametricity. This axiomatization allows one to apply parametricity as a notion of uniformity to a wide range of mathematical structures.

The axioms for parametricity graphs are quite general. There are intuitive parametricity graphs built using many of the structures common to programming semantics. These include sets (REL), domains (CPO and CPO⊥), partial equivalence relations (PER), algebras (F–Alg for any PG-functor F: REL → REL) and free algebras (REL∧ for any parametric monad ⟨T, η, μ⟩ on REL). Spans provide a means to apply parametricity to any category C since Sp(C) and JMS(C) are parametricity graphs.

The 2-category PG of parametricity graphs is an appropriate setting for an abstract description of parametricity. We have shown that the intuitive concept of parametric transformations — families of morphisms which preserve all edges — coincides with the categorical construction of indexed natural transformations between PG-functors. Similar correspondences hold for the intuitive descriptions of parametrically polymorphic families and abstract data types. These are given by the limit and colimit PG-functors in PG.

The strength of the axioms for parametricity graphs can be illustrated by representation results for models of System F. We have shown that traditional representation results, such as the encoding of initial algebras and final co-algebras hold in well-pointed parametricity graph models of System F. These results show that certain parametric limits are trimmed down to include only the intuitively uniform families. While these representation results do not extend to non-well-pointed parametricity graphs in general, we have not found any examples of the parametric limits containing non-uniform families.

We have also investigated a polymorphic Algol-like language where one
does expect the traditional representation results to hold. Not all parametricity graph models of this polymorphic Algol-like language satisfy the traditional representation results. We've shown some parametricity graph models that contain intuitively uniform families which contradict the traditional representation results. We produce a pre-sheaf-like (hence, non-well-pointed) parametricity graph model for this language which agrees with one's intuitions. The traditional representation results hold in this model. This shows that the traditional representation results can coexist with Algol-like imperative features. This is not the case with recursion.

Recursion is a programming language feature which one expects to contradict the traditional representation results. For instance, the constantly divergent family gives a second uniform family of type \(X \Rightarrow X\) (in addition to the family of identity functions), hence \(\forall X. X \Rightarrow X\) is not terminal. By considering recursion in a linear framework, we can use parametricity graphs to produce parametric models for a language with recursion. Recursion can coexist with linear representation results. These linear representation results hold in all well-pointed parametricity graph models of the polymorphic linear lambda calculus. Therefore we are able to describe parametric models of the polymorphic linear lambda calculus plus recursion. This does give models where parametricity and recursion coexist.

Two parametric models of the polymorphic linear lambda calculus plus recursion arise from using relations and from using spans over \(\mathbf{cpo}_\perp\). These two approaches treat the basic connectives \(\otimes\) and \(\to\) of linear logic differently, so it is not clear if the preservation of edges in these two settings give the same collection of parametric families. We presented some limited examples of types where the parametric transformations are the same in both settings. Investigating the potential difference between these two settings is an area for future research.

Another avenue for future research involves investigating representation results for non-well-pointed settings. Recall that the traditional representation results do not hold for non-well-pointed parametricity graphs in general, but that there are certain instances where they do hold, such as in the functor graph \(\mathbf{REL}^C\) of Chapter 6. It may be possible to identify a larger collection of parametricity graphs where the traditional representation results do hold. Perhaps more useful, it may also be possible to discover more general representation results which describe the uniform families in a wider range of settings.
Index

\(\omega\)-set, 56
\(\forall \gamma F(Y)\), 35
\(\exists \gamma F(Y)\), 43
\(F\), 49
\(G\), 49
\(\langle M \rangle\) (in System P), 101
\(\langle f \rangle\) (subsumption), 73
\(\equiv\) (in System P), 100
\(\cong\) (in System P), 100
\(E^{[F/X]}\) (substitution), 12
\([f, fg]R\) (re-indexing), 77
\(\partial_0\), 25
\(\partial_1\), 25
algebra
\(\text{in } G\), 163
\(\text{in System P}, 117\)
Algol, 186
\(\text{Idealized}, 187\)
Polymorphic Idealized, 188
appropriate parameters, 130
arrow graph, 29
assumption
\(\text{relational, 91}\)
\(\text{type, 12, 91}\)
BCPO, 232
bcpo, 230
C\(^{\text{arr}}\), 30, 68
cartesian, 77
Cartesian closed, 27
category
\(\text{edge, 25}\)
\(\text{indexed, 26}\)
\(\text{vertex, 25}\)
category
\(\text{total, 37}\)
cleavage, 78
\(\text{normalized, 78}\)
co-algebra
\(\text{in } G\), 177
\(\text{in System P, 126}\)
cofibred
\(\text{indexed functor, 81}\)
\(\text{reflexive graph category, 77}\)
compatible family
\(\text{in } \text{BCPO}, 232\)
\(\text{in } \text{CPO}, 45\)
\(\text{in } \text{CPO}\_\perp, 225\)
\(\text{in } \text{Sp(cpo}\_\perp), 228\)
complete partial order, 28
\(\text{pointed, 28, 209}\)
context, 12
\(\text{in System P, 91}\)
\(\text{of types, 91}\)
\(\text{relational, 91}\)
\(\text{typing, 12}\)
contraction, 212
CPO, 28
\(\text{pointed, 28, 209}\)
CPO, 28, 68, 78, 84
Cpo, 28, 220
CPO\_\perp, 28, 68, 222
cpo\_\perp, 28, 220
data types, 187
derereference, 187
discrete, 49
Dual Intuitionistic Linear Logic, 211
\(E\), 27
edge, 25
\(\text{identity, 25}\)
edge category, 25
equivalent relations, 100
exponent, 27
exponential strength, 169
F-algebra, 31, 68, 82, 84
F-simulation, 31
fibred
indexed functor, 81, 83
reflexive graph category, 77, 83
final co-algebra
in G, 178
in System P, 126
free algebra, 31
fresh, 12, 92
functor
indexed, 26
cloven, 82
cofibred, 81
fibred, 81, 84
subsumptive, 75, 83
non-variant, 49
PG-functor, 83
functor graph, 27, 68, 84

graph
of a morphism, 73
trivial, 30
$i^r$, 51
$i^r_{in}$, 137
Idealized Algol, 187
identity condition, 71, 74, 83
identity edge, 25
$m$, 163
indeterminates, 90, 92
indexed category, 26
initial $P$-algebra
in System P, 117, 120
initial algebra
in G, 163, 166, 173
initial object, 74
instantiate, 2
instantiation, 13
isomorphic relations, 100

JA, 240, 251
JMS(C), 30, 68, 84
judgment
relatedness, 91
relation, 91
term, 12
type, 12

$K_R$, 27
lambda calculus
polymorphic, 11
linear, 211
predicative, 17
System F, 11
System P, 89
languages
Algol-like, 186
last, 131
lift, 221
Lim, 42
linear category, 215
linear logic, 211
linear parametricity graph, 216, 216
LNL setting, 215

$m^r$, 164

model
in REL, 20–24
in RG setting, 50–51
in Set, 14–16, 18
of Idealized Algol, 190, 194–197
of PIA, 198–199, 202–204
of PLLC, 216–218
monodial
parametric transformation, 215
PG-functor, 215
morphism, 25
muP, 163
non-variant, 49
non-well-pointed categories, 182
$\omega$-set, 56
$P$-algebra, 117
initial, 117, 120
parametric, 69, 84
adjunction, 84
colimit, 43
with parameters, 44, 47
end, 48
family, 22
limit, 35
functor, 42
with parameters, 40
monad, 84
transformation, 69
parametricity
  relational, 5, 20–24
parametricity graph, 83
parametricity setting, 85, 89
partial equivalence relation, 29
PCPO, 209
pcpo, 209
PER, 29, 56, 68, 84
Pfn, 32
PG, 84
PG-functor, 83
  strong, 168
phrase types, 187
PIA, 188
PL-category, 62
plain type, 112
PLLC, 211–212
Poset, 28, 68
predicative, 16
predicative calculus, 17–18
pRG, 70
principal family, 45
pure derivation, 102
$P_x$, 113
realize, 56
reflexive graph category, 5, 25
  discrete, 49
REL, 26, 68, 78, 84
Rel, 24
rel, 32
relational, 68, 83
relations
  isomorphic, 100
RELr, 32
RG, 26
RG setting
  for System F, 55
  for predicative calculus, 50
rRG, 68
Span(C), 30, 68, 78
span, 30
  jointly monic, 30
Sp(C), 31, 70, 84
square, 25
  cartesian, 77
  co-cartesian, 77
state, 190
strength, 168
strongest post-edge, 77
subsumption mapping, 83
subsumptive
  indexed functor, 75, 83
  reflexive graph category, 73, 79
Symmetric monoidal closed, 214
System F, 11–14
System P, 89–96
terminal object, 85
total category, 37
Tr(C), 30
track, 56
transformation
  dinatural, 48
  indexed natural, 26
  parametric, 69, 76, 84
type assertion, 92
vertex, 25
vertex category, 25
weakening, 212
weakest pre-edge, 77
well-pointed, 85, 89, 181
witness (of a span), 30
world, 190
$Y_A$, 234
zone
  indeterminate, 92
  variable, 92
References


260


261


263


[Sco69] D. S. Scott. A type theoretical alternative to CUCH, ISWIM and OWHY. Unpublished manuscript, Oxford University, 1969. Reprinted as [Sco93].


Vita

Brian Patrick Dunphy was born in Ames, Iowa on March 17, 1971. He graduated from North Carolina State University in 1993 with a Bachelor's degree in Mathematics. In 1994, he received a Master's degree in Applied Mathematics with a Computer Science from North Carolina State University. While working towards his Ph.D., Dunphy spent two years visiting the School of Computer Science at the University of Birmingham, England.