

Parametricity and Naturality in the Semantics of Algol-like Languages

Uday S. Reddy

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Voice: +1 217 333 3412 Fax: +1 217 244 6869
reddy@cs.uiuc.edu

December 10, 1998

Abstract

We examine the relationship between two notions of uniformity for polymorphic functions, viz., relational parametricity and naturality. While naturality applies only to first-order function types, parametricity is more general. We axiomatize parametricity at a categorical level so that it subsumes naturality, and provide examples of this situation. Both parametricity and naturality are key components in giving denotational semantics to Algol-like languages. The known models for these languages do not have strong enough parametricity properties to subsume naturality. We give new models where such subsumption takes place. The parametricity properties of the new models automatically incorporate solutions for some of the thorny issues in semantics such as passivity and irreversible state change.

Keywords Type theory, polymorphism, parametricity, categorical models, denotational semantics, Algol-like languages.

1 Introduction

Relational parametricity was proposed by Reynolds [31] as a notion of uniformity appropriate for polymorphic functions, capturing the notion of data abstraction in an elegant way. This idea contrasts with the principle of *naturality* traditionally used in category theory as the notion of uniformity. Intuitively, parametricity is more general than naturality because the latter is essentially limited to first-order types. When higher-order types are involved, type variables occur in both positive and negative positions and the resulting type constructions are not functorial. Thus naturality does not apply. (The generalization to dinatural transformations runs into the problem that dinaturals don't compose in general.) On the other hand, parametricity applies to higher-order types without any problem. Thus, our intuition suggests that parametricity should be a generalization of naturality for higher-order types.

In this paper, we substantiate this intuition by giving a categorical axiomatization of parametricity and showing that it subsumes naturality as well as dinaturality. The treatment is based on the notion of *reflexive graphs* proposed by O'Hearn and Tennent [19] and used in [32, 22]. We add an additional axiom to reflexive graphs to the effect that morphisms can be embedded into "relations" (in an abstract sense) and obtain what are called *subsumptive reflexive graphs*. The desired results hold for these structures. This work complements the results of Plotkin and Abadi [23] where they show that parametricity subsumes dinaturality within a logic for polymorphic lambda calculus. Our results are more general in that they are stated at the level of abstraction of categories and, so, can be applied in a wide variety of contexts where other programming language phenomena are modeled (such as recursion, side effects, continuations etc.).

We were led to this study in considering the semantics of Algol-like languages where both parametricity and naturality are essential components [29, 21, 19]. Naturality is involved in the fact that denotations of terms act the "same way" at various possible expansions of state sets. An example of this phenomenon is the equivalence [14]:

$$(\mathbf{new} \ x.c) \equiv c \quad \forall c : \mathbf{comm}$$

Parametricity is involved in arguing that local variables are hidden from external client programs. For example, the equivalence [19]:

$$(\mathbf{new} \ x.x := 0; p(x, x := x + 1)) \equiv (\mathbf{new} \ x.x := 0; p(-x, x := x - 1)) \\ \forall p: \mathbf{exp} \times \mathbf{comm} \rightarrow \mathbf{comm}$$

follows from parametricity because the client procedure p cannot access the local variable x except via the provided operations.

In the semantics described in [19], parametricity does not subsume naturality. The issue was left open in [17]. This leads one to wonder if the notion of parametricity used in these models is the right one (because our intuition suggests that the right notion of parametricity should subsume naturality). In this paper, we give new models for Idealized Algol where parametricity subsumes naturality. In addition to giving a stronger notion of parametricity, our models also capture the idea of irreversible state change implicit in Algol-like languages. (See [19, Sec. 11] and [26, 16, 17] for a discussion of irreversibility.) In contrast to O’Hearn and Reynolds’s use of linear functions [17], our model uses pure parametricity to model irreversibility by making states sufficiently abstract so that only commands are allowed to manipulate them.

2 Parametricity and Naturality

Parametricity for set-theoretic types, as described in [31], involves statements of the form:

$$\forall x \in A, x' \in A'. \quad x R x' \implies f(x) S f'(x')$$

In short, this states that the functions f and f' map R -related pairs to S -related pairs. We also use the following diagrammatic notation to represent this preservation property:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ R \updownarrow & & \updownarrow S \\ A' & \xrightarrow{f'} & B' \end{array} \tag{1}$$

The notion of reflexive graphs, proposed in [19] abstracts such diagrams to a categorical level. The abstraction involves four kinds of entities:

- “vertices” such as A and A' (sets in the special case),
- “vertex morphisms” or “arrows” such as f and f' (functions in the special case),
- “edges” such as R and S (relations in the special case), and

- “edge morphisms” such as the above square regarded as a morphism $(f, f') : R \rightarrow S$ (relation preservation property in the special case).

In the horizontal dimension, we have categories (one for vertices and one for edges). In the vertical dimension, we do not need composition, but we need a notion of an identity edge I_A for each vertex A . This leads to the following definition:

Definition 1 *A reflexive graph (of categories) \mathbf{G} consists of two categories \mathbf{G}_v and \mathbf{G}_e , and three functors:*

$$\delta_0, \delta_1 : \mathbf{G}_e \rightarrow \mathbf{G}_v \quad I : \mathbf{G}_v \rightarrow \mathbf{G}_e$$

such that

$$\delta_i \circ I = \text{id}_{\mathbf{G}_v}$$

(The functors δ_0 and δ_1 pick out the source and target of edges and edge morphisms and I assigns the identities in the vertical dimension.)

The canonical example of reflexive graphs is **Set** whose vertex category is the category of sets and the edge category is that of relations and relation-preservation squares. The identity edge I_A is the diagonal relation on A .

A reflexive graph is said to be *relational* if for any pair of vertex arrows $f: A \rightarrow B$, $f': A' \rightarrow B'$ and edges $R : A \leftrightarrow A'$, $S : B \leftrightarrow B'$, there is at most one edge morphism $\phi: R \rightarrow S$ such that $\delta_0(\phi) = f$ and $\delta_1(\phi) = f'$. A diagrammatic representation of ϕ such as (1) is called a *parametricity square*. The terminology is motivated by the fact that edges R and S determine a relation $[R \rightarrow S] \subseteq \text{hom}(A, B) \times \text{hom}(A', B')$. We assume that all our reflexive graphs are relational.

A reflexive graph has two kinds of duals: \mathbf{G}^{op} obtained by inverting the arrows and \mathbf{G}^{co} obtained by inverting the edges. We say that \mathbf{G} is *symmetric* if $\mathbf{G}^{\text{co}} \cong \mathbf{G}$ and use the notation R^\smile and φ^\smile for the images under the isomorphism.

Definition 2 *A subsumptive reflexive graph (SRG) is a symmetric reflexive graph equipped with a family of “subsumption mappings” $\langle - \rangle_{A, B}$ which associate to each arrow $g: A \rightarrow B$ an edge $\langle g \rangle : A \leftrightarrow B$ such that*

1. $\langle \text{id}_X \rangle = I_X$, and

2. the left hand side diagram below is a parametricity square (the shape of an edge morphism) iff the right hand side diagram is a commutative square.¹

$$\begin{array}{ccc}
 A & \xrightarrow{f} & B \\
 \langle g \rangle \downarrow & & \downarrow \langle h \rangle \\
 A' & \xrightarrow{f'} & B'
 \end{array}
 \iff
 \begin{array}{ccc}
 A & \xrightarrow{f} & B \\
 g \downarrow & & \downarrow h \\
 A' & \xrightarrow{f'} & B'
 \end{array}
 \quad (2)$$

Note that $\langle - \rangle$ is injective. If $\langle g_1 \rangle = \langle g_2 \rangle = R : A \leftrightarrow A'$ then $\text{id}_R : \langle g_1 \rangle \rightarrow \langle g_2 \rangle$ which implies $g_1 = g_2$ by (2).

Here are some examples.

- The reflexive graph **Set** of sets, functions and binary relations can be made subsumptive by taking $\langle g \rangle$ to be graph of the function g . We refer to this as the “standard” subsumption mapping.
- The reflexive graph **Cpo** of ω -cpo’s, continuous functions and ω -complete relations can be made subsumptive with the standard subsumption mapping. Similarly, the reflexive graph **Cpo_⊥** of pointed cpo’s, strict continuous functions and complete relations² can be made subsumptive. However the reflexive graph **Cppo** of pointed cpo’s where the morphisms are *all* continuous functions and edges are complete relations cannot be made subsumptive. Since a continuous function need not be strict, its graph is not a complete relation in general.
- The reflexive graph **Per**(D) has per’s over a partial combinatory algebra D as vertices, per-morphisms as arrows, and “saturated” relations as edges [3, 4]. A saturated relation $R : A \leftrightarrow B$ is a relation $R \subseteq D \times D$ such that $A; R; B = R$. **Per**(D) is made subsumptive by taking $\langle g \rangle$ to be the saturation of the graph of an arbitrary realizer of g .
- For any category **C**, the arrow category **C[→]** can be regarded as a reflexive graph. This is trivially a SRG with $\langle g \rangle = g$.
- For any category **C**, the category of spans **Span**(**C**) can be regarded as a (non-relational) reflexive graph. It can be made a SRG with the subsumption $\langle g \rangle = (\text{id}, g)$.

¹For non-relational reflexive graphs, the left hand side of the equivalence involves existential quantification.

²A complete relation is an ω -complete relation that is strict in the sense that $\perp_A R \perp_{A'}$.

All these examples are symmetric, with the exception of \mathbf{C}^\rightarrow . Note that the subsumption mapping $\langle - \rangle$ of a relational reflexive graph \mathbf{G} is nothing but a reflexive graph-functor $\langle - \rangle : \mathbf{G}_v^\rightarrow \rightarrow \mathbf{G}$ that is identity on the vertex categories and full and faithful on edge categories. (This characterization only works when \mathbf{G} is a relational reflexive graph.)

If \mathbf{G} and \mathbf{H} are SRG's, their product $\mathbf{G} \times \mathbf{H}$ is a SRG with the subsumption mapping $\langle (f, g) \rangle = (\langle f \rangle, \langle g \rangle)$. The dual reflexive graph \mathbf{G}^{coop} is a SRG with the subsumption mapping $\langle f^{\text{op}} \rangle = \langle f \rangle^{\text{co}}$. If \mathbf{G} is, in addition, symmetric, then \mathbf{G}^{op} is a SRG with subsumption $\langle g^{\text{op}} \rangle = \langle g \rangle^\sim$.

2.1 Functors and transformations

Functors and natural transformations for reflexive graphs are special cases of those for indexed categories [2, 7]. A *RG-functor* is a pair of functors $(F_v : \mathbf{G}_v \rightarrow \mathbf{H}_v, F_e : \mathbf{G}_e \rightarrow \mathbf{H}_e)$ preserving the structure of reflexive graphs. A *parametric natural transformation* $\tau : F \rightarrow G$ is similarly a pair of national transformations $(\tau_v : F_v \rightarrow G_v, \tau_e : F_e \rightarrow G_e)$ that commutes with this structure. For relational reflexive graphs, all this means is that τ_v is natural, and, for every edge $R : A \leftrightarrow A'$ of \mathbf{G} , there is a parametricity square:

$$\begin{array}{ccc}
 F_v A & \xrightarrow{(\tau_v)_A} & G_v A \\
 F_e R \uparrow & & \uparrow G_e R \\
 F_v A' & \xrightarrow{(\tau_v)_{A'}} & G_v A'
 \end{array} \tag{3}$$

We call this the “parametricity” property of τ . (The naturality of τ_e is trivial.) We normally drop the subscripts v and e and write FA , FR , τ_A etc.

A *SRG-functor* $F : \mathbf{G} \rightarrow \mathbf{H}$ between SRG's is a RG-functor that preserves the subsumption mapping:

$$F \langle g \rangle = \langle Fg \rangle$$

We are interested in parametric natural transformations between SRG-functors.

The terminology of “covariant” and “contravariant” functors for functors of types $\mathbf{G} \rightarrow \mathbf{H}$ and $\mathbf{G}^{\text{op}} \rightarrow \mathbf{H}$ respectively is standard. Similarly, we can define “non-variant” functors to be functors of type $\mathbf{G}^\circ \rightarrow \mathbf{H}$, where \mathbf{G}° is the graph with all the non-identity arrows of \mathbf{G} erased. For any functor $F : \mathbf{G} \rightarrow \mathbf{H}$, there is a corresponding non-variant functor $F^\circ : \mathbf{G}^\circ \rightarrow \mathbf{H}$ which maps the vertices and edges of \mathbf{G} , but ignore the morphisms. A parametric

natural transformation $\tau : F^\circ \rightarrow G^\circ$ has only a parametricity requirement. The naturality requirement is trivial. We think of such a family τ as a “parametric transformation” from F to G . Parametric transformations need not be natural in general. But, for SRG-functors, they are.

Theorem 3 *If $F, G : \mathbf{G} \rightarrow \mathbf{H}$ are SRG-functors, a parametric transformation $\tau : F^\circ \rightarrow G^\circ$ is in fact a parametric natural transformation $\tau : F \rightarrow G$.*

Proof: Since \mathbf{G} is a SRG, for every arrow $g : A \leftrightarrow A'$, there is an edge $\langle g \rangle : A \leftrightarrow A'$. Since \mathbf{H} is a SRG, we have the parametricity square on the left, which is equivalent to the commutative square on the right:

$$\begin{array}{ccc}
 FA & \xrightarrow{\tau_A} & GA \\
 \uparrow F\langle g \rangle & & \uparrow G\langle g \rangle \\
 FA' & \xrightarrow{\tau_{A'}} & GA'
 \end{array}
 \Leftrightarrow
 \begin{array}{ccc}
 FA & \xrightarrow{\tau_A} & GA \\
 \downarrow Fg & & \downarrow Gg \\
 FA' & \xrightarrow{\tau_{A'}} & GA'
 \end{array}$$

Hence τ is natural. □

When higher-order types are involved, one encounters functors with mixed variance, and dinatural transformations [13, 3] give an approximate notion of uniform families for such functors. We show that parametricity subsumes dinaturality whenever the reflexive graphs are symmetric. Suppose $F : \mathbf{G}^{\text{op}} \times \mathbf{G} \rightarrow \mathbf{H}$ is a “multi-variant” functor where \mathbf{G} is a symmetric SRG. Since $(\mathbf{G}^{\text{op}})^\circ = \mathbf{G}^\circ$, we have $F^\circ : \mathbf{G}^\circ \times \mathbf{G}^\circ \rightarrow \mathbf{H}$ and, hence, $F^\circ \Delta : \mathbf{G}^\circ \rightarrow \mathbf{H}$.

Theorem 4 *If $F, G : \mathbf{G}^{\text{op}} \times \mathbf{G} \rightarrow \mathbf{H}$ are SRG-functors where \mathbf{G} is a symmetric SRG, then a parametric transformation $\tau : F^\circ \Delta \rightarrow G^\circ \Delta$ is in fact a parametric dinatural transformation $\tau : F \rightarrow G$.*

Proof: For τ to be dinatural, we need that, for every $g : A \rightarrow A'$ in \mathbf{G}_v , the following hexagon commutes in \mathbf{H}_v :

$$\begin{array}{ccccc}
 & & FAA & \xrightarrow{\tau_A} & GAA \\
 & \nearrow FgA & & & \searrow GA_g \\
 FA'A & & & & GAA' \\
 & \searrow FA'g & & & \nearrow GgA' \\
 & & FA'A' & \xrightarrow{\tau_{A'}} & GA'A'
 \end{array}$$

But, since \mathbf{H} is a SRG, the hexagon is equivalent to the outer parametricity square below:

$$\begin{array}{ccccccc}
 FA'A & \xrightarrow{FgA} & FAA & \xrightarrow{\tau_A} & GAA & \xrightarrow{GA_g} & GAA' \\
 \uparrow I_{FA'A} & & \uparrow F^\circ\langle g \rangle\langle g \rangle & & \uparrow G^\circ\langle g \rangle\langle g \rangle & & \uparrow I_{GAA'} \\
 FA'A & \xrightarrow{FA'g} & FA'A' & \xrightarrow{\tau_{A'}} & GA'A' & \xrightarrow{GgA'} & GAA'
 \end{array}$$

The outer square is the composite of the three small squares, of which the middle square follows from the parametricity of τ and the first and last squares are symmetric. So, it is enough to show that the first square is a parametricity square. Note that it can be obtained by applying the F functor to the following squares in \mathbf{G}^{op} and \mathbf{G} respectively (since $\langle g^{\text{op}} \rangle^\sim$ of \mathbf{G}^{op} is the same as $\langle g \rangle$ of \mathbf{G}):

$$\begin{array}{ccc}
 A' & \xrightarrow{g^{\text{op}}} & A \\
 \uparrow I_{A'} & & \uparrow \langle g^{\text{op}} \rangle^\sim \\
 A' & \xrightarrow{\text{id}_{A'}} & A'
 \end{array}
 \qquad
 \begin{array}{ccc}
 A & \xrightarrow{\text{id}_A} & A \\
 \uparrow I_A & & \uparrow \langle g \rangle \\
 A & \xrightarrow{g} & A'
 \end{array}$$

These parametricity squares follow from evident commutative squares. \square

The conclusion is that whenever we work with SRG-functors over symmetric SRG's, parametricity is all that is needed. We get naturality and dinaturality for free. Recall that dinatural transformations do not normally compose [3]. On the other hand, parametric transformations compose. Thus, we should think of parametricity as a strengthening of the dinaturality criterion which allows composition. (The price is that we give up composition in the vertical dimension, but that is no loss.)

2.2 Parametric functor categories

In this section, we briefly sketch the implications of the SRG concept to functor categories, which will be of use in the semantics of the next section. We will concentrate on SRG-functors of type $\mathbf{W} \rightarrow \mathbf{Set}$ where \mathbf{W} is a small symmetric SRG. From the programming language point of view, it will be convenient to think of \mathbf{W} as a category of “worlds.” So, we use the notation $f : X \leq Y$ to denote the types of arrows of \mathbf{W} (instead of the standard $f : X \rightarrow Y$). The intuition is that Y is a future world of X and f is a witness to this fact.

Recall that a SRG-functor $F : \mathbf{W} \rightarrow \mathbf{Set}$ induces a non-variant functor $F^\circ : \mathbf{W} \rightarrow \mathbf{Set}$. We can recover F from F° .

Fact 5 *If $G : \mathbf{W}^\circ \rightarrow \mathbf{Set}$ is a non-variant functor and $G = F^\circ$ for some SRG-functor $F : \mathbf{W} \rightarrow \mathbf{Set}$, then this SRG-functor F is uniquely determined.*

This is because the action of F on morphisms can be recovered from the action of G on edges: $\langle Fg \rangle = F\langle g \rangle = G\langle g \rangle$ and there is a unique arrow Fg such that $\langle Fg \rangle = G\langle g \rangle$. Thus, to define SRG-functors, we only need to define the corresponding non-variant functors and verify that there is an action on the arrows.

The functor category $[\mathbf{W} \rightarrow \mathbf{Set}]$ is given by the following data:

- objects — SRG-functors $F : \mathbf{W} \rightarrow \mathbf{Set}$,
- arrows — parametric transformations $\tau : F^\circ \rightarrow G^\circ$.

Let $F, G : \mathbf{W} \rightarrow \mathbf{Set}$ be SRG-functors and W an object of \mathbf{W} . Then we define the set $[\forall X \geq W. FX \rightarrow GX]$ as follows. Its elements are families of functions $\tau = \{\tau_f : FX \rightarrow GX\}_{f: X \geq W}$ indexed by arrows from W such that, for all edges R , we have the following implication of parametricity squares:

$$\begin{array}{ccc} W & \xrightarrow{f} & X \\ \uparrow I_W & & \uparrow R \\ W & \xrightarrow{f'} & X' \end{array} \implies \begin{array}{ccc} FX & \xrightarrow{\tau_f} & GX \\ \uparrow FR & & \uparrow GR \\ FX' & \xrightarrow{\tau_{f'}} & GX' \end{array}$$

Such families are “natural” in the following sense: for every arrow $g : X \leq Y$, we have a commutative square

$$\begin{array}{ccc} FX & \xrightarrow{\tau_f} & GX \\ \downarrow Fg & & \downarrow Gg \\ FY & \xrightarrow{\tau_{f;g}} & GY \end{array}$$

The bounded parametric family construction $[\forall X \geq - : FX \rightarrow GX]$ forms an SRG-functor. Its action on an edge $R : W \leftrightarrow W'$ is a relation

$[\forall S \geq R. FS \rightarrow GS]$ defined as follows: $\tau [\forall S \geq R. FS \rightarrow GS] \tau'$ iff, for every edge $S : X \leftrightarrow X'$, we have the implication:

$$\begin{array}{ccc}
 W & \xrightarrow{f} & X \\
 R \uparrow & & \uparrow S \\
 W' & \xrightarrow{f'} & X'
 \end{array}
 \implies
 \begin{array}{ccc}
 FX & \xrightarrow{\tau f} & GX \\
 FS \uparrow & & \uparrow GS \\
 FX' & \xrightarrow{\tau' f'} & GX'
 \end{array}$$

Its action on arrows is now uniquely determined. If $g : W \leq W'$ is an arrow,

$$[\forall X \geq g. FX \rightarrow GX](\tau) = \{\tau_{f \circ g}\}_{f : X \geq W'}$$

Theorem 6 *The functor category $[\mathbf{W} \rightarrow \mathbf{Set}]$ is cartesian-closed.*

Products are given pointwise:

$$\begin{aligned}
 (F \times G)X &= FX \times GX \\
 (F \times G)R &= FR \times GR
 \end{aligned}$$

and the exponential $F \Rightarrow G$ is $[\forall X \geq -. FX \rightarrow GX]$. The details are as in [19, Theorem 7].

3 Semantics of Algol-like languages

An Algol-like language is a typed lambda calculus whose base types support state-manipulation [29]. The typical base types are:

- `comm` — the type of “commands” or state-transformers,
- `expδ` — the type of “expressions” or state-readers that give results of type δ (where δ ranges over primitive data types like `int` and `bool`), and
- `varδ` — the type of mutable “variables” that hold δ -typed values.

The typical primitive operations of Algol-like languages are shown in Table 1. The crucial primitive is the local variable declaration: `newδ λv. C` creates a new local variable and binds it to v during the execution of the command C . To interpret this operation, one must enlarge the state set to contain the new variable and interpret the command C in the enlarged state set. However, any non-local identifiers used in C do not have direct access to

skip	:	comm
–; –	:	comm × comm → comm
deref	:	var _δ → exp _δ
–:=–	:	var _δ × exp _δ → comm
new _δ	:	(var _δ → comm) → comm

Table 1: Typical constants of Algol-like languages

the newly created variable. This enforces a form of data abstraction which is the key issue in the examples of the Introduction.

We give two semantic models for this language, both of which interpret Algol types as SRG-functors of type $[\mathbf{W} \rightarrow \mathbf{Set}]$ for a small SRG \mathbf{W} .

3.1 A subsumptive model

For the first model, the vertices of \mathbf{W} will be state sets. The types of Algol will be interpreted as follows:

$$\begin{aligned}
\text{comm}(Q) &= \forall X \geq Q. [X \rightarrow X] \\
\text{exp}_\delta(Q) &= \forall X \geq Q. [X \rightarrow \delta] \\
\text{var}_\delta(Q) &= \text{exp}_\delta(Q) \times [\delta \rightarrow \text{comm}(Q)] \\
(\theta_1 \times \theta_2)(Q) &= \theta_1(Q) \times \theta_2(Q) \\
(\theta_1 \Rightarrow \theta_2)(Q) &= \forall X \geq Q. [\theta_1(X) \rightarrow \theta_2(X)]
\end{aligned} \tag{4}$$

(For simplicity, we use the same notation for both type terms and semantic types.) We now describe the SRG of worlds \mathbf{W} .

For any set X , let $T(X)$ denote the monoid of transformations $[X \rightarrow X]$. The unit and multiplication of the monoid are id_X and sequential composition “;”. In addition, our monoids are equipped with a so-called “diagonal operation”:

$$\begin{aligned}
D &: [X \rightarrow T(X)] \rightarrow T(X) \\
Dg &= \lambda x. g(x)(x)
\end{aligned}$$

The intuition behind the diagonal operation is that it represents a dinatural family of “assignment operations”:

$$\begin{aligned}
\text{asgn}_A &: [A \rightarrow T(X)] \times [X \rightarrow A] \rightarrow T(X) \\
\text{asgn}_A(a, e) &= \lambda x. a(e(x))(x)
\end{aligned}$$

Note that two operations are inter-definable:

$$\text{asgn}_A(a, e) = D(a \circ e) \quad Dg = \text{asgn}_X(g, \text{id}_X)$$

(In fact, $[X \rightarrow T(X)]$ is the coend $\int^A [A \rightarrow T(X)] \times [X \rightarrow A]$. So, D is merely a compact representation of the assignment family.)

The vertex category \mathbf{W}_v has sets (of some limited cardinality) as objects. Its morphisms $f : Q \leq X$ are pairs of functions:

$$(\varphi_f : X \rightarrow Q, \tau_f : T(Q) \rightarrow T(X))$$

such that

1. $\tau_f(\text{id}_Q) = \text{id}_X$
2. $\tau_f(a; b) = \tau_f(a); \tau_f(b)$
3. $\tau_f(D_Q g) = D_X(\lambda x. \tau_f(g(\varphi_f(x))))$
4. $\varphi_f(\tau_f(a)(x)) = a(\varphi_f(x))$
5. $\tau(a)(s) = x' \wedge \varphi(x) = \varphi(x') \implies x = x'$.

The first three axioms state that the operations of the monoids are preserved, and the fourth axiom states that the monoid action on the states is preserved. The last axiom states that the expansions of commands to larger state sets leave the new state components unchanged. For any sets Q and Z , there is an “expansion morphism” $\text{expnd}_{Q,Z} : Q \leq Q \times Z$, with the components:

$$\begin{aligned} \varphi : Q \times Z \rightarrow Q &= \lambda(q, z). q \\ \tau : T(Q) \rightarrow T(Q \times Z) &= \lambda a. \lambda(q, z). (a(q), z) \end{aligned}$$

In fact, every morphism of \mathbf{W}_v is essentially an expansion. This category of worlds is very similar to the original category used by Reynolds [29], but it seems to have been ignored in later work.

The edge category \mathbf{W}_e has objects $R : Q \leftrightarrow Q'$ which are pairs of relations

$$(R_s : Q \leftrightarrow Q', R_t : T(Q) \leftrightarrow T(Q'))$$

such that

1. $\text{id}_Q R_t \text{id}'_Q$
2. $a R_t a' \wedge b R_t b' \implies (a; b) R_t (a'; b')$

skip_Q	$=$	$\Lambda f : X \geq Q. \text{id}_X$
$c_1;_Q c_2$	$=$	$\Lambda f : X \geq Q. c_1[f]; c_2[f]$
$\text{deref}_Q(v)$	$=$	$\text{fst}(v)$
$v :=_Q e$	$=$	$\Lambda f : X \geq Q. \text{asgn}_\delta(\lambda n. \text{snd}(v)(n)[f], e[f])$
$\text{new}_{\delta,Q}(k)$	$=$	$\Lambda f : X \geq Q. \text{fst}(k[\text{expnd}_{X,\delta}](\text{top})(x, \text{init}_\delta))$
		where $\text{top} = (\Lambda g : Y \geq X \times \delta. \lambda y. \text{snd}(\varphi_g(y)),$
		$\lambda n. \Lambda g : Y \geq X \times \delta. \tau_g(\lambda(x, m). (x, n)))$
		and $\text{init}_\delta =$ some fixed element of δ

Table 2: Interpretation of Algol constants

3. $g[R_s \rightarrow R_t]g' \implies D_Q g R_t D_{Q'} g'$
4. $\text{apply}_{Q,Q}[R_t \times R_s \rightarrow R_s] \text{apply}_{Q',Q'}$, or, equivalently, $R_t \subseteq [R_s \rightarrow R_s]$.

An edge morphism $(f, f') : R \rightarrow S$ exists iff $\varphi_f[S_s \rightarrow R_s] \varphi_{f'}$ and $\tau_f[R_t \rightarrow S_t] \tau_{f'}$. Note that the definition of edges precisely parallels that of morphisms. It is easy to show \mathbf{W} is a SRG, with the subsumption mapping $\langle(\varphi_f, \tau_f)\rangle = (\langle\varphi_f\rangle^\smile, \langle\tau_f\rangle)$.

The interpretation of Algol now proceeds along familiar lines. Algol types are interpreted by (4) with the obvious action on edges. The interpretation of primitive operations is shown in Table 2.

To give some insight into what this represents, we calculate the interpretation of base types:

Theorem 7 *There are bijections: $\text{exp}_\delta(Q) \cong [Q \rightarrow \delta]$ and $\text{comm}(Q) \cong [Q \rightarrow Q]$.*

Proof:

1. Suppose $e \in \text{exp}_\delta(Q)$. By parametricity, whenever $(f, f') : I_Q \leq R$, $e_f[R_s \rightarrow I_\delta]e_{f'}$. In particular, taking $f = \text{id}_Q$ and $R = \langle f' \rangle$, we have $e_{\text{id}}[\langle\varphi_f\rangle^\smile \rightarrow I_\delta]e_{f'}$ which means $e_{f'} = e_{\text{id}} \circ \varphi_{f'}$. Hence, $e_{f'}$ is uniquely determined by e_{id} .
2. Suppose $c \in \text{comm}(Q)$. Whenever $(f, f') : I_Q \leq R$, $c_f[R_s \rightarrow R_s]c_{f'}$. Let $f = \text{id}_Q$ and $f' = (\varphi, \tau) : Q \leq X$. We show that $c_{f'} = \tau(c_{\text{id}})$. Let $x_0 \in X$ be some element and $q_0 = \varphi(x_0)$. Consider the edge $R = (R_s, \langle\tau\rangle) : Q \leftrightarrow X$ where $R_s = \{((c_{\text{id}})^n(q_0), (\tau(c_{\text{id}}))^n(x_0)) \mid n \geq 0\}$.

It may be seen that $(\text{id}_Q, f') : I_Q \leq R$. Since $q_0 R_s x_0$, we have $c_{\text{id}}(q_0) R_s c_{f'}(x_0)$. By definition of R_s , this means that, for some $n \geq 0$, $(c_{\text{id}})^n(q_0) = c_{\text{id}}(q_0)$ and $\tau(c_{\text{id}})^n(x_0) = c_{f'}(x_0)$. By the axiom 5 of morphisms, we have $\tau(c_{\text{id}})(x_0) = c_{f'}(x_0)$. By quantifying over x_0 , we conclude that $c_{f'} = \tau(c_{\text{id}})$. Hence, $c_{f'}$ is uniquely determined by c_{id} . \square

Note that the second isomorphism is nontrivial. As recognized in the work on subtyping, types of the form $\forall X \geq Q. X \rightarrow X$ often do not have enough elements [33, 11]. The structure of the category of worlds is a key to getting this type to be meaningful.

This semantics improves on the one given in [19] in that it uses a stronger parametricity criterion that subsumes naturality. To get some appreciation for this, consider the type.

$$\begin{aligned} \forall Q. (\text{comm} \rightarrow \text{comm})(Q) &= \forall Q. \forall X \geq Q. [\text{comm}(X) \rightarrow \text{comm}(X)] \\ &\cong \forall Q. \forall X \geq Q. [T(X) \rightarrow T(X)] \end{aligned}$$

The following family of this type is parametric in O’Hearn and Tennent’s sense:

$$p_Q = \Lambda f : X \geq Q. \lambda a. \lambda x. x[Q \rightarrow \varphi_f(a(x))]$$

The notation $x[Q \rightarrow \varphi_f(a(x))]$ denotes the value that is the same as x , but with its Q -part updated to $\varphi_f(a(x))$. Since $X \geq Q$ means that $X \cong Q \times Z$ for some Z , this is a reasonable operation. But is p_Q parametric? Consider three state sets $Q \leq Q \times Y \leq Q \times Y \times Z$. When used at world $X = Q \times Y \times Z$, $p_{Q \times Y}$ updates the $Q \times Y$ part of X , but p_Q updates only the Q -part of X . In fact, the family p_Q is not natural for this reason. In our setting, this kind of a family is not parametric because there is an edge $\langle \text{expnd}_{Q,Y} \rangle$ whose preservation requires that p_Q and $p_{Q \times Y}$ must act the same way.

3.2 Modeling State Change

The first model has the defect that it does not model state changes faithfully. In Algol-like languages, state changes, once carried out, cannot be undone. There are no “snap back” operations. However, since state changes are modeled as ordinary functions in the above model, this irreversibility is not captured. For example, consider the function

$$\begin{aligned} t : \text{comm} \times \text{exp} &\rightarrow \text{exp} \\ t_Q(c, e) &= \Lambda f : X \geq Q. \lambda x. e[f](c[f](x)) \end{aligned}$$

It is parametric. But, the expression $t_Q(c, e)$ has the effect of executing c and then undoing it. Snap back operations of this kind invalidate the usual reasoning principles of imperative programs such as Hoare Logic [10].

We can rectify this defect by using a more sophisticated category of worlds \mathbf{W} . The vertices of \mathbf{W} are now pairs $W = (Q_W, T_W)$ where Q_W is a set and $T_W \subseteq [Q_W \rightarrow Q_W]$ is a submonoid of the monoid of transformations that is also closed under the diagonal operation $D : [Q_W \rightarrow T_W] \rightarrow T_W$. Pairs of this kind are called *transformation monoids* in automata theory [8]. They model resources that are “active” in the sense that they have operations for state transformation. The intuition in adding the T_W components to the worlds is that there are contexts in which the allowed state transformations are restricted. For example, a state component corresponding to an output stream can only be extended, not truncated or modified in other arbitrary ways. As another example, expressions are not allowed any state transformations at all.

The morphisms and edges of \mathbf{W} are as before, but only the axioms 1 through 3 are used. The fourth axiom which requires that the monoid action be preserved is completely dropped. The consequence of this change is that states become “abstract.” They can be modified by the transformations in T_W and these modifications can be observed by other transformation via the diagonal operation. But, arbitrary functions cannot read and compute states as in the example t_Q above.

Examples of morphisms in this category are the following:

- For every world $W = (Q_W, T_W)$, there is a world $W_0 = (Q_W, 1)$ where $1 = \{\text{id}_{Q_W}\}$ is the one-element monoid. There is an evident morphism $i_W : W_0 \leq W$ which sends id_{Q_W} to itself. There is also a morphism $j_W : W \leq W_0$ in the other direction, which sends every $a \in T_W$ to id_{Q_W} . As a result, we have a “passification morphism” $p_W = (j_W; i_W) : W \leq W_0 \leq W$ which sends every $a \in T_W$ to id_{Q_W} . The effect of this morphism is to prohibit all state changes.
- If $W_1 = (Q_1, T_1)$ and $W_2 = (Q_2, T_2)$ are worlds, there is a world $W_1 \otimes W_2 = (Q_1 \times Q_2, T_1 \otimes T_2)$ where $T_1 \otimes T_2 = \{a \times b \mid a \in T_1 \wedge b \in T_2\}$. There are morphisms $f_i : W_i \leq W_1 \otimes W_2$ with components

$$\begin{array}{ll} \varphi_i & : Q_1 \times Q_2 \rightarrow Q_i & \varphi_i(q_1, q_2) = q_i \\ \tau_i & : T_i \rightarrow T_1 \otimes T_2 & \tau_1(a) = a \times \text{id} \quad \tau_2(b) = \text{id} \times b \end{array}$$

The idea of $W_1 \otimes W_2$ is that it combines the resources represented by W_1 and W_2 .

The types of Algol are now interpreted as follows:

$$\begin{aligned} \text{comm}(W) &= \forall X \geq W. T_X \cong T_W \\ \text{exp}_\delta(W) &= \forall X \geq W. [Q_X \rightarrow \delta] \cong [Q_W \rightarrow \delta] \\ \text{var}_\delta(W) &= \text{exp}_\delta(W) \times [\delta \rightarrow \text{comm}(W)] \end{aligned}$$

The constants of Algol are interpreted in the same manner as before.

We summarize the effects achieved in this model:

- There is a notion of *passive* values which cause no state changes. These are the values that are invariant under the passification morphisms p_W . For example, all values of $\text{exp}_\delta(W)$ are passive. But, $\text{comm}(W)$ has only one passive element, viz., skip_W .
- Similarly, one can define a notion of *passive types* which form a reflective subcategory of $[\mathbf{W} \rightarrow \mathbf{Set}]$ as in [25, 15]. This means that snapback operations, like the function t_Q above, do not exist.
- There is a notion of non-interference: two values $a, b \in F(W)$ are non-interfering if they come from separate factors of W (with respect to \otimes). Thus, we can model logics for program reasoning such as Specification Logic [30, 27] as well as interference control type systems [28, 15].

Evidently, the effects of this model are very similar to the model of Tennent studied in [34, 18, 15]. But, two points are worth noting. Our model includes parametricity, whereas it is not clear how to add parametricity to the Tennent's model. In fact, all the aspects of state change are modeled here using only parametricity (which includes naturality). Secondly, we are able to model these aspects without involving any notion of divergence. Thus, our model can be used with logics of total correctness. In contrast, Tennent's model uses divergence in a crucial fashion to model passivity. Intuitively, the notions of passivity and non-interference should be orthogonal to divergence because even total programs have these properties. Our model shows that this is so.

4 Conclusion

The idea that naturality is a special case of parametricity underlies many pieces of work including [9, 23, 35]. We have given an axiomatization of parametricity where this is generally the case.

Since the structures used for this purpose, subsumptive reflexive graphs, are so much like categories (and, in fact, form a 2-category), we hope that this paves the way for restating much of the programming theory to account for parametricity. Examples include domain theory, linear functions, call-by-value, continuations, monads, abstract data types, data refinement etc. On the other side, many categorical concepts can also benefit from generalizing from naturality to parametricity. A good example is the concept of ends and coends.

As one example of the application of these ideas, we considered the semantics of Algol-like languages. Here, our criteria for a strong notion of parametricity showed up deficiencies in the existing theory. Finding a better theory of parametricity led to rich dividends in terms of addressing hard issues like passivity and non-interference. These ideas should also be applicable in other situations, e.g., in the presence of nondeterminism [1] and concurrency [6].

A further application is for polymorphic calculi with subtyping such as $F_{<}$. There is not yet a convincing theory of parametricity for these calculi. Modeling record-updates in a parametric setting has been an unsolved problem [33, 24]. Our modeling of Algol points to a possible direction for addressing these issues.

Acknowledgements This work considerably benefited from several discussions with Peter O’Hearn, John Reynolds, Bob Tennent, Edmund Robinson and Hongseok Yang. It was supported by NSF grant CCR-96-33737 and a travel grant from EPSRC.

References

- [1] M. Abadi and L. Lamport. The existence of refinement mappings. *Theoretical Comput. Sci.*, 82(2):253–284, May 1991.
- [2] A. Asperti and G. Longo. *Categories, Types and Structures: An Introduction to Category Theory for the Working Computer Scientist*. MIT Press, 1991.
- [3] E. S. Bainbridge, P. Freyd, A. Scedrov, and P. J. Scott. Functorial polymorphism. *Theoretical Comput. Sci.*, 70:35–64, 1990.
- [4] R. Belucci, M. Abadi, and P.-L. Curien. A model for formal parametric polymorphism: A PER interpretation for System R. In *Typed Lambda*

- Calculi and Applications - TLCA '95*, LNCS, pages 32–46. Springer-Verlag, 1995. (Expanded version in a Manuscript, 1995).
- [5] S. Brookes, M. Main, A. Melton, and M. Mislove, editors. *Mathematical Foundations of Programming Semantics: Eleventh Annual Conference*, volume 1 of *Electronic Notes in Theor. Comput. Sci.* Elsevier, 1995.
- [6] S. D. Brookes. The essence of Parallel Algol. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 164–174. IEEE Computer Society Press, July 1996. Reprinted as Chapter 21 of [20].
- [7] R. L. Crole. *Categories for Types*. Cambridge University Press, Cambridge, 1993.
- [8] S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1974. (Volumes A and B).
- [9] R. Hasegawa. Categorical data types in parametric polymorphism. *Math. Struct. Comput. Sci.*, 4(1):71–109, Mar 1994.
- [10] C. A. R. Hoare. An axiomatic basis for computer programming. *Comm. ACM*, 12:576–583, 1969.
- [11] M. Hoffman and B. C. Pierce. Positive subtyping. *Information and Computation*, 126:11–33, 1996.
- [12] IEEE Computer Society Press. *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, July 1994.
- [13] S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.
- [14] A. R. Meyer and K. Sieber. Towards fully abstract semantics for local variables. In *Fifteenth Ann. ACM Symp. on Princ. of Program. Lang.*, pages 191–203. ACM, 1988. (Reprinted as Chapter 7 of [20]).
- [15] P. W. O’Hearn, A. J. Power, M. Takeyama, and R. D. Tennent. Syntactic control of interference revisited. In Brookes et al. [5]. (Reprinted as Chapter 18 of [20]).
- [16] P. W. O’Hearn and U. S. Reddy. Objects, interference and Yoneda embedding. In Brookes et al. [5].

- [17] P. W. O’Hearn and J. C. Reynolds. From Algol to polymorphic linear lambda-calculus. Electronic manuscript, Queen Mary and Westfield, London, April 1997. URL <http://www.qmw.ac.uk/ohearn>.
- [18] P. W. O’Hearn and R. D. Tennent. Semantical analysis of specification logic, Part 2. *Inf. Comput.*, 107(1):25–57, 1993. (Reprinted as Chapter 14 of [20]).
- [19] P. W. O’Hearn and R. D. Tennent. Parametricity and local variables. *J. ACM*, 42(3):658–709, 1995. (Reprinted as Chapter 16 of [20]).
- [20] P. W. O’Hearn and R. D. Tennent. *Algol-like Languages (Two volumes)*. Birkhäuser, Boston, 1997.
- [21] F. J. Oles. *A Category-Theoretic Approach to the Semantics of Programming Languages*. PhD thesis, Syracuse University, 1982.
- [22] A. M. Pitts. Relational properties of domains. *Inf. Comput.*, 15(2):66, Jun 1996.
- [23] G. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Typed Lambda Calculi and Applications - TLCA '93*, LNCS, pages 361–375. Springer-Verlag, 1993.
- [24] G. Plotkin, M. Abadi, and L. Cardelli. Subtyping and parametricity. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science* [12], pages 310–319.
- [25] U. S. Reddy. Passivity and independence. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science* [12], pages 342–352.
- [26] U. S. Reddy. Global state considered unnecessary: An introduction to object-based semantics. *J. Lisp and Symbolic Computation*, 9:7–76, 1996. (Reprinted as Chapter 19 of [20]).
- [27] U. S. Reddy. Objects and classes in Algol-like languages. In *Fifth Intern. Workshop on Foundations of Object-oriented Languages*. electronic proceedings at <http://pauillac.inria.fr/~remy/fool/proceedings.html>, Jan 1998.
- [28] J. C. Reynolds. Syntactic control of interference. In *ACM Symp. on Princ. of Program. Lang.*, pages 39–46. ACM, 1978. (Reprinted as Chapter 10 of [20]).

- [29] J. C. Reynolds. The essence of Algol. In J. W. de Bakker and J. C. van Vliet, editors, *Algorithmic Languages*, pages 345–372. North-Holland, 1981. (Reprinted as Chapter 3 of [20]).
- [30] J. C. Reynolds. Idealized Algol and its specification logic. In D. Neel, editor, *Tools and Notions for Program Construction*, pages 121–161. Cambridge Univ. Press, 1982. (Reprinted as Chapter 6 of [20]).
- [31] J. C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing '83*, pages 513–523. North-Holland, Amsterdam, 1983.
- [32] E. Robinson and G. Rosolini. Reflexive graphs and parametric polymorphism. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science* [12].
- [33] E. Robinson and R. D. Tennent. Bounded quantification and record-update problem. Message to **Types** electronic mailing list, 1988.
- [34] R. D. Tennent. Semantical analysis of specification logic. *Inf. Comput.*, 85(2):135–162, 1990. (Reprinted as Chapter 13 of [20]).
- [35] P. Wadler. Theorems for free! In *Fourth Intern. Conf. Functional Program. Lang. and Computer Architecture*, pages 347–359. ACM, 1989.