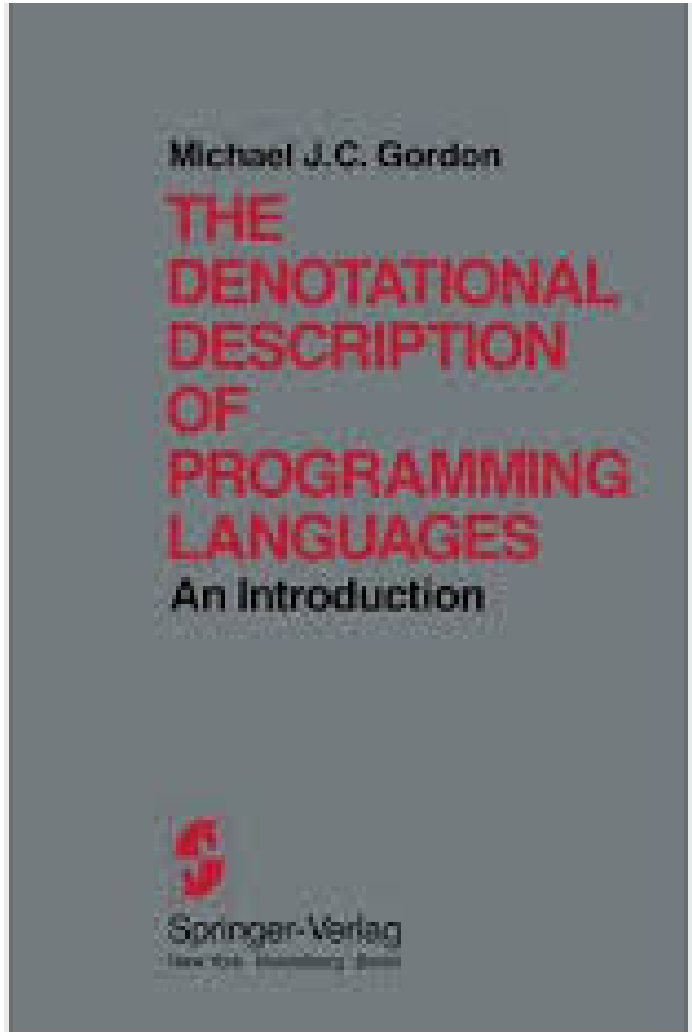
Parametric Polymorphism and Abstract Models of Storage

(In memory of Christopher Strachey, 1916-1975)

Uday S. Reddy[1]

[1]University of Birmingham

Strachey-100, Oxford



My introduction to Strachey's ideas

Section 1

Introducing the terms

Parametric polymorphism

- The term Parametric polymorphism appears in *Fundamental Concepts in Programming Languages* (1967).
- Contrasted with "*ad hoc* polymorphism" or *definition by cases* on the types involved.
- Example given:

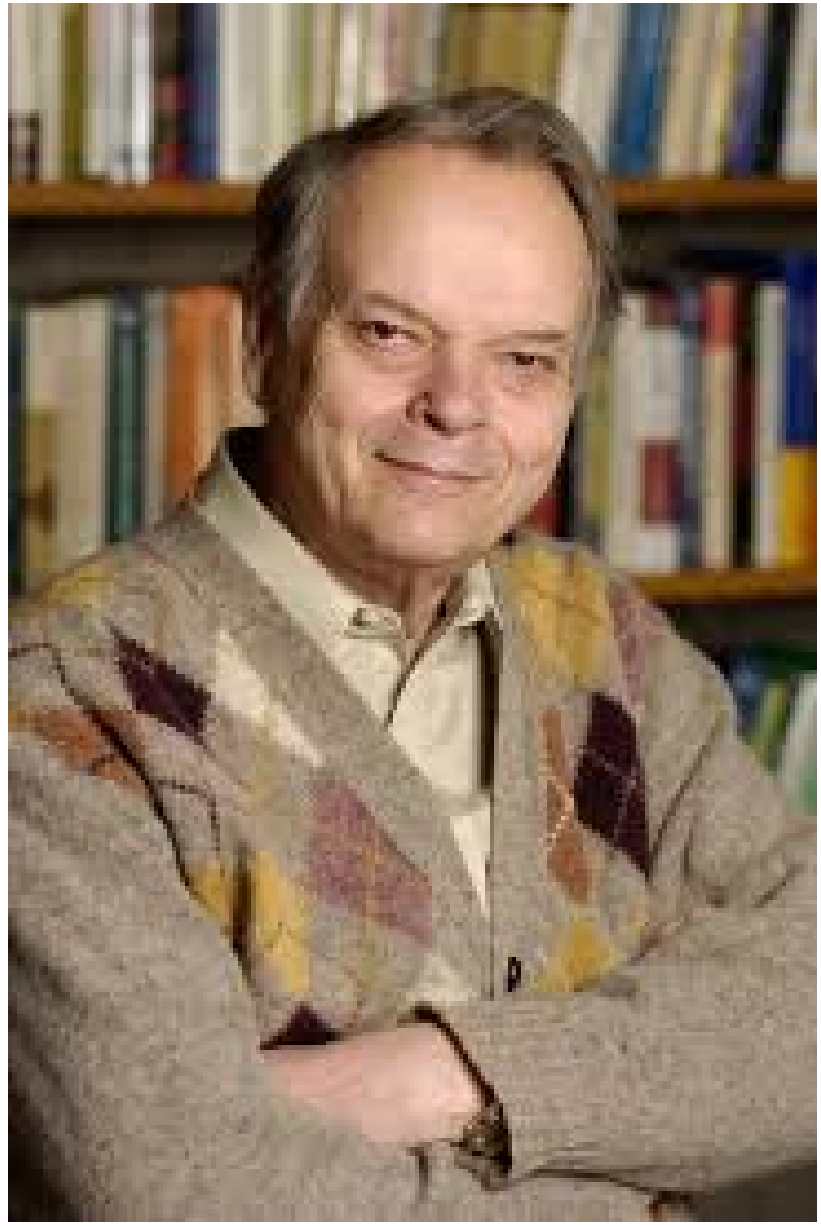
$$map_{\alpha,\beta} : (\alpha \Rightarrow \beta,\ \alpha\ \textbf{list}) \rightarrow \beta\ \textbf{list}$$

- Reynolds (1974) defines Polymorphic lambda calculus.
- Reynolds (1983) refers to Strachey. Identifies parametricity as a concept. Defines it via relation-preservation.
- The world is a different place!

Abstract models of storage

- *The Varieties of Programming Language* (1973) gives a model of store (based on locations) which is said to be "deliberately simplified".
- A more "complicated model" and a more "formalised description" is promised in a forthcoming paper titled

  An abstract model of storage
- What was Strachey planning to say in this paper?
- Once again, Reynolds steals the thunder by proposing an abstract model of storage in *The Essence of Algol* (1981).



John C. Reynolds, 1935-2013

Section 2

Parametric polymorphism

Parametric polymorphism

- In Reynolds's explanation:
  - "a parametric polymorphic function is one that behaves the same way for all types,"

  whereas an ad hoc polymorphic function may have "unrelated meanings" at different types.
- Therefore, he wants to give a definition for what it means for a polymorphic function to be parametric.

$$\mathit{map}_{\alpha,\beta} : (\alpha \Rightarrow \beta,\ \alpha\ \mathbf{list}) \to \beta\ \mathbf{list}$$

- Why? The naive models of the polymorphic lambda calculus have *ad hoc* polymorphic functions. We must "exclude" them.

Mathematicians knew parametricity

- Eilenberg and Mac Lane:
  - *Natural Isomorphisms in Group Theory* (1942)
  - *General Theory of Natural Equivalences* (1945)
  - *Algebra* (1967)
- "*... is considered natural because it furnishes for each G a unique isomorphism not dependent on any choice of generators*."
- This is representation independence or data abstraction.
- "*This exhibition of the isomorphism... is natural in that it is given simultaneously for all ... vector spaces L.*"
- This is definability without case analysis.

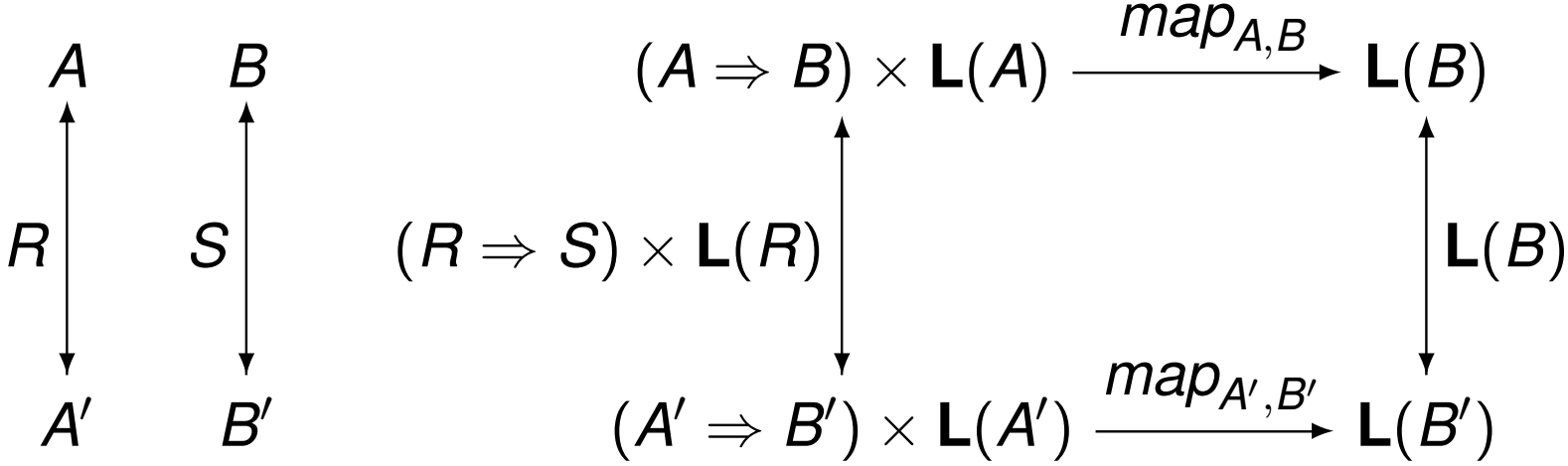
But...

- Naturality doesn't work!

$$map_{\alpha,\beta} : (\alpha \Rightarrow \beta,\ \alpha\ \mathbf{list}) \to \beta\ \mathbf{list}$$

  The input type $(\alpha \Rightarrow \beta,\ \alpha\ \mathbf{list})$ is not a functor of α.

- Reynolds notices that it is a relator.



- This is termed relational parametricity. It is a conservative extension of naturality.

How does mathematics deal with it?

- It doesn't, actually.
- Quite a lot of mathematics just deals with first-order functions.
  - For first-order functions naturality and relational parametricity are equivalent.
- Quite a lot of mathematics assumes that types are sets.
  - For sets (with equality and membership operations), the only allowed relations are isomorphisms.
- Category theorists also use dinaturality which is an approximation of relational parametricity.
  - But, dinaturals don't compose!

An example problem

- The definition of a category involves composition:

$$comp_{A,B,C} : \mathrm{Hom}(A, B) \times \mathrm{Hom}(B, C) \to \mathrm{Hom}(A, C)$$

  This should be parametric in A, B and C. If it is, I call it a parametric category.

$$\mathrm{Hom}(R, S) \times \mathrm{Hom}(S, T) \to \mathrm{Hom}(R, T)$$

- Example: **Set**, the category of sets and functions, is a parametric category.
- Counterexample: **Rel**, the category of sets and binary relations, is *not a parametric category*.

An example problem - contd.

$$
\begin{array}{ccc}
(p, q) & \mapsto & p; q \equiv a \\
\\
\mathbf{Rel}(A, B) \times \mathbf{Rel}(B, C) & \xrightarrow{comp_{A,B,C}} & \mathbf{Rel}(A, C) \\
\Big\updownarrow & & \Big\updownarrow \\
\mathbf{Rel}(R, S) \times \mathbf{Rel}(S, T) & & \mathbf{Rel}(R, T) \\
\Big\updownarrow & & \Big\updownarrow \\
\mathbf{Rel}(A', B') \times \mathbf{Rel}(B', C') & \xrightarrow{comp_{A',B',C'}} & \mathbf{Rel}(A', C') \\
\\
(p', q') & \mapsto & p'; q' \equiv a'
\end{array}
$$

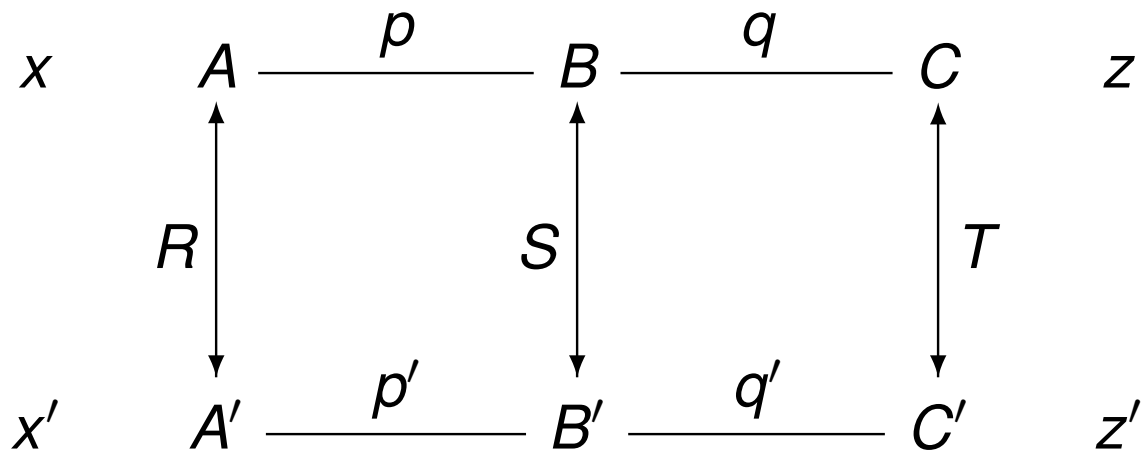
$(a, a') \in \mathbf{Rel}(R, T)$ means

$$(x, x') \in R \wedge (z, z') \in T \implies (a(x, z) \iff a'(x', z'))$$

More abstractly, we are treating $\mathbf{Rel}(A, C) \equiv [A \times C \to \mathbf{2}]$

An example problem - contd.

- To show $(p; q,\ p'; q') \in \mathbf{Rel}(R, T)$:



- Consider $(x, x') \in R$ and $(z, z') \in T$.
- We need to show $(p; q)(x, z) \iff (p'; q')(x', z')$.
- The left hand side implies there exists $y \in B$ such that $p(x, y) \wedge q(y, z)$.
- However, there may be nothing in B' related to y. The relation S could be the empty relation!
- If we turn "set-theoretic", i.e., assume that all types are sets and all relations are isomorphisms, then
- Parametricity forces us to forget that sets are sets (when we use them as types).

The upshot

- By generalising Plotkin's logical relations theorem, we can argue that anything definable in a good typed language is parametric.
- All constructive mathematics is parametric.
- For example, natural deduction rules are parametric. So are all category-theoretic adjunctions.
- The rule of excluded middle and the axiom of choice are not parametric.
- By the way, remember Abramsky's slogan:

$$\text{composition} = \text{parallel composition} + \text{hiding}$$

Parallel composition is always parametric:

$$\mathbf{Rel}_2(A, B) \times \mathbf{Rel}_2(B, C) \to \mathbf{Rel}_3(A, B, C)$$

But hiding is parametric only if we can produce the mediating witness constructively.

Parametricity is about information hiding

- We can use a polymorphic type for map:

$$\mathit{map} : \forall\alpha.\,\forall\beta.\,(\alpha \Rightarrow \beta,\ \alpha\ \textbf{list}) \to \beta\ \textbf{list}$$

- The $\forall$ quantifiers signify that the types provided as α and β are hidden from map. Those values are "black boxes" for map. This is local information hiding.
- The dual existential quantifier

$$\exists\alpha.\,T(\alpha)$$

  describes global information hiding (data abstraction).
  - We provide a type A and a suite of operations of type $T(A)$.
  - But the client programs (or client mathematicians) cannot "see" the type A.
- Reynolds explains this in terms of Descartes and Bessel teaching complex analysis.

Section 3

Abstract models of storage: Intuitionism

Mathematical semantics

- We know that Peter Landin was a strong influence on Strachey.
- Landin was a great believer in functional programming:
  - *The commonplace expressions of arithmetic and algebra have a certain simplicity that most communications to computers lack.* (1966)
- But Strachey, who was a master programmer in imperative programming languages, knew that there *must be* ways to reason about imperative programs systematically.
  - *[Declarative languages] are an interesting subset, but... inconvenient... We need them because at the moment we don't know how to construct proofs with ... imperatives and jumps.* (1966)
- Strachey seems to have thought the solution was to reduce the "imperatives" to "mathematics" (the commonplace expressions of arithmetic and algebra).
- This was perhaps too limiting.

A critique of “mathematical semantics”

- Mathematics is strongly tied into the Plato’s world of concepts, which are timeless and non-material.
- But concepts are not the only things that exist.
- Physics constructs mathematical models for physical phenomena that involve time and matter, but does not attempt to *reduce* them to mathematics.
- The point of the models is to allow us to make predictions (i.e., construct proofs), but not necessarily to state the essence of the phenomena.
- A part of Strachey’s programme was successful: Commands can be modelled as state-to-state functions.
- But the store on which the commands act is a physical object, not a Platonic concept.

Is “mathematical semantics” possible

- Today, we know that it is not possible to reduce imperative programming languages to mathemtical languages.
  - Functional languages, e.g., PCF, satisfy Milner’s context lemma, which can be thought of as an extensionality principle. But imperative programming languages do not satisfy it.
  - In Games semantics, we can see that functional languages can be interpreted by history-free strategies, but imperative languages need history-sensitive strategies.
- However, in a broad sense “mathematics” contains everything that can be logically constructed. Computer Science itself is a branch of mathematics in this sense.
- We can certainly build models for programming languages using traditional mathematical tools. But this does not amount to a reduction.

Intuitionism

- Aristotle's first syllogism:

All men are mortal
Socrates is a man

Socrates is mortal

- What did Aristotle mean by men?

Intuitionism - contd

- What would Aristotle say if you said:

All men are mortal
Christopher Strachey is a man

Christopher Strachey is mortal

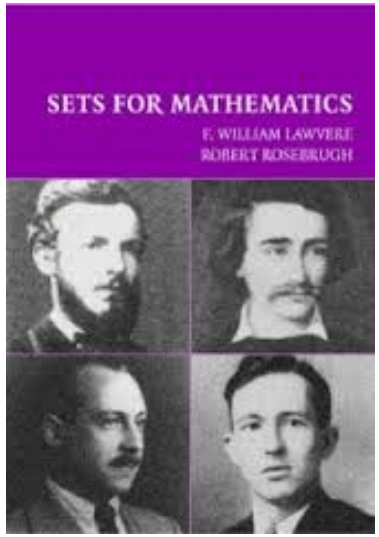
- Perhaps he would have said, “it is not a good example because there is no man called Christopher Strachey”?
- What we mean by men is different from what Aristotle would have meant by “men”. (Or is it? Did Aristotle believe in Platonic concepts?)
- But the meaning of All men are mortal is robust. Even if Aristotle didn’t know what we mean by “men”, he certainly meant that all men in our world are mortal too.

Intuitionism - contd

- I would make a case that, whenever we apply logic to physical phenomena, we have such possible worlds in mind.
- We don't reduce men to mathematics, but we can reason about them just the same.
- We have typed lambda calculus and intuitionistic logic, i.e., all *parametric* principles of reasoning still applicable to these physical objects.
- The store of imperative programs is similar. There is no fixed collection of locations just as there is no fixed collection of men.
- If we define a function of type $\mathbf{var}[\mathit{int}] \to \mathbf{com}$, the possible arguments are not just the variables that exist when the function is defined, but the variables that might exist when the function is applied.

Intuitionistic set theory

- An "intuitionistic" set (in the sense of Kripke and Lawvere) is a set parameterized by some context, which we call a "world." $A(X) = \ldots$ Lawvere also calls them "variable sets." Normal sets are called "constant sets."
- In our case, "worlds" will be store shapes.



- There is also a sense of another world being a "possible future world" of X.

$$\begin{array}{ccc} Y & & A(Y) \\ f\uparrow & & \uparrow A(f) \\ X & & A(X) \end{array}$$

If the world grows from X to Y, then every value of type A at X continues to be a value at Y via the translation $A(f)$.

Intuitionistic function space

- Functions: $(A \Rightarrow B)(X) = \forall_{Y \leftarrow X} A(Y) \to B(Y)$
- Intuitively: a "function" of type $A \Rightarrow B$ at world X can work at every future world Y, accepting arguments of type A at world Y and giving results of type B at world Y.
- In other words, functions must be stable under the growth of the world.
- In fact, all values in intuitionistic set theory must be stable under growth in this way.
- This is a fundamental idea for Computer Science: All the programs we write must be stable under the growth of the world. Examples:
  - Programs for the Internet.
  - Databases.

Parametric Intuitionistic function space

- Functions: $(A \Rightarrow B)(X) = \forall_{Y \leftarrow X} A(Y) \to B(Y)$.
- The $\forall$ type quantifier signifies parametricity. (The plain categorical construction is written as $\int_{Y \leftarrow X} A(Y) \to B(Y)$.)
- If f is a "function" of type $(A \Rightarrow B)(X)$ then it can only depend on the information available in world X.
- Even if it is called at a future world Y that has additional information, the new information available in Y would be hidden from f.
- This is a construct of "parametric category theory".
- The need for extending category-theoretic inutiotionism (toposes) with parametricity was first identified by O'Hearn and Tennent (1993): *Parametricity and Local Variables*.
- It came us a surprise to most of us at that time that category theory alone could not do the job on its own!

Example for parametric intuitionism

- Consider the equivalence (for $p : \mathbf{com} \to \mathbf{com}$):

$$\begin{aligned} &\mathbf{newvar}\ x.\{x := 0;\ p(x := !x + 1)\} \\ &\quad \equiv\ \mathbf{newvar}\ x.\{x := 0;\ p(x := !x - 1)\} \end{aligned}$$

- The argument depends on the fact:
  - "x is hidden from p."
- Equivalent to:
  - "p is parametrically polymorphic in the state space of x."
- I.e., p must preserve all possible relationships between the state spaces of x.
- For example, $R : \mathbb{Z} \leftrightarrow \mathbb{Z}$ given by:

$$n \left[R\right] n' \iff n \geq 0 \ \wedge\ n' = -n$$

- Since $x := !x + 1$ and $x := !x - 1$ preserve this relation, $p(x := x + 1)$ and $p(x := !x - 1)$ must preserve it too.

Section 4

Abstract models of storage: Automata

Towards an abstract store

- The “store” does not just contain locations that hold numbers/characters, but all kinds of data structures, like stacks, trees, or graphs, as well as databases, I/O devices, network connections etc.
- The store is everything that can change.
- When we apply functions to objects in the store, we build layers of abstractions, at each level of which we have a more abstract store.

```
let S2 = RemoveLayoutChars[
            IntcodefromFlexowriter[
              BytesfromPT]]
```

(An example from [Stoy and Strachey, 1971])

What should a store be?

- **Idea 1**: A store is a collection of locations.
- **Idea 2**: A store can be abstracted to a set of states.
- **Idea 3**: A store should be abstracted to a set of states along with its possible state transformations.
- Reynolds arrived at Idea 3 in 1981! But, perhaps, he didn't have a strong reason to pursue it.
- Oles produced a variant of the model using Idea 2 and proved that it was isomorphic to the Reynolds model. It became standard from then on.
- However, the tension between category theory and relational parametricity, which exists with the Oles model, is not present in the Reynolds model. (This problem led me to reinvent it in 1998.)

Views and lenses

- Morphisms in a category can be viewed in either direction ($\mathcal{C}$ vs. $\mathcal{C}^{\mathrm{op}}$).

- In the reverse direction, $f^\sharp$ is a way of viewing a large store Y (a database) as a small store X (a view). In the database theory, $f^\sharp$ is called a lense.
- A "lense" is a bidirectional concept: Not only can we treat the state of the database as a state of the view, but we also want to transmit the changes made to the view as changes to the database [Czarnecki et al, 2009].

Reynolds transformation monoids

- A store X is represented as a tuple

$$(\mathcal{Q}_X, \mathcal{T}_X, \alpha_X, \mathrm{read}_X)$$

(Reynolds transformation monoid) where:

- $\mathcal{Q}_X$ - a (small) set of states,
- $\mathcal{T}_X$ - a monoid of state transformations $\mathcal{T}_X \subseteq [\mathcal{Q}_X \to \mathcal{Q}_X]$,
- $\alpha_X : \mathcal{T}_X \hookrightarrow [\mathcal{Q}_X \to \mathcal{Q}_X]$ - the implicit monoid action,
- $\mathrm{read}_X : [\mathcal{Q}_X \to \mathcal{T}_X] \to \mathcal{T}_X$ - called "diagonalization":

$$\mathrm{read}_X\ p = \lambda x.\, p\ x\ x = \lambda x.\, \alpha_X(p\ x)\ x$$

allows a state transformation to be dependent on the initial state.
For example,

$$\mathit{cond}_X\ b\ c_1\ c_2 = \mathrm{read}_X\ \lambda s.\, \textbf{if}\ b(s) \neq 0\ \textbf{then}\ c_1\ \textbf{else}\ c_2$$

- **Note**: Transformation monoids in algebraic automata theory [Eilenberg, 1974] are triples $(\mathcal{Q}_X, \mathcal{T}_X, \alpha_X)$.

Logical relations for RTM's

- A logical relation $R : (\mathcal{Q}_X, \mathcal{T}_X) \leftrightarrow (\mathcal{Q}_{X'}, \mathcal{T}_{X'})$ is a pair (R_q, R_t) where

$$\begin{array}{c} X \\ \Big\updownarrow R \\ X' \end{array} \quad = \quad \left(\begin{array}{cc} \mathcal{Q}_X & \mathcal{T}_X \\ \Big\updownarrow R_q \ , & \Big\updownarrow R_t \\ \mathcal{Q}_{X'} & \mathcal{T}_{X'} \end{array} \right)$$

  - $R_q : Q_X \leftrightarrow Q_{X'}$ is a relation, and
  - $R_t : T_X \leftrightarrow T_{X'}$ is a logical relation of monoids,

  such that
  - $\alpha_X \left[R_t \to [R_q \to R_q]\right] \alpha_{X'}$, and
  - $\mathrm{read}_X \left[[R_q \to R_t] \to R_t\right] \mathrm{read}_{X'}$.

Morphisms for RTM's

- A homomorphism $f : (\mathcal{Q}_X, \mathcal{T}_X) \to (\mathcal{Q}_Y, \mathcal{T}_Y)$ is a pair (f_q, f_t)

$$\begin{array}{c} Y \\ \uparrow f \\ X \end{array} \quad = \quad \left(\begin{array}{c} \mathcal{Q}_Y \\ \downarrow f_q \\ \mathcal{Q}_X \end{array} \quad , \quad \begin{array}{c} \mathcal{T}_Y \\ \uparrow f_t \\ \mathcal{T}_X \end{array} \right)$$

  where
  - $f_q : Q_Y \to Q_X$ is a function, and
  - $f_t : T_X \to T_Y$ is a homomorphism of monoids,

  such that $(\langle f_q \rangle^{\smile}, \langle f_t \rangle)$ is a logical relation of RTM's.
- Note that f_q and f_t run in opposite directions. (Mixed variance)
- No "parametricity vs. naturality" tension with RTM's. Logical relations subsume homomorphisms.

Interpretation of Algol types

- Algol types are now functors of type **RTM** $\to$ **Set** (ignoring divergence):

$$\begin{array}{rcl} \textsc{Com}(X) & = & \mathcal{T}_X \\ \textsc{Exp}(X) & = & [\mathcal{Q}_X \to \mathit{Int}] \\ (A \Rightarrow B)(X) & = & \forall_{Y \leftarrow X} A(Y) \to B(Y) \end{array}$$

  This model does not have command snapback, i.e., models irreversible state change.

- **Fact**: $\mathrm{Hom}(\textsc{Com}, \textsc{Com}) \cong \mathbb{N}$, representable by

$$\lambda c.\, \mathbf{skip},\ \lambda c.\, c,\ \lambda c.\, (c; c),\ \ldots$$

- But it has expression snapback (does not model passivity).
- Reddy (MFPS, 2013) shows that, with an apparently minor adjustment, passivity is also obtained.

The challenge for denotational semantics

- How to capture the intensional aspects of computations in an extensional model?
- **Example of extensionality**:

$$\mathbf{gv}(x) \implies (x := !x + 1;\ x := !x + 1) \equiv (x := !x + 2)$$

- Intensional models
  - Object spaces [Reddy, O'Hearn, McCusker]
  - Action traces [Hoare, Brookes]
  - Game semantics [Abramsky, McCusker, Honda, Murawski]

  distinguish between the two sides of the equivalence.

$$\mathrm{read}_x(2),\ \mathrm{write}_x(3),\ \mathrm{read}_x(3),\ \mathrm{write}_x(4)$$
$$\mathrm{read}_x(2),\ \mathrm{write}_x(4)$$

- We are after extensional models.
- **Another example of extensionality**:

$$\mathit{stack}(s) \implies (s.\mathit{push}(v);\ s.\mathit{pop}) \equiv \mathbf{skip}$$

Naive extensional models have “junk”

- Command snapback (with divergence):

$$\textbf{try} : \textbf{com} \to \textbf{com}$$
$$\textbf{try}\ c \;=\; \lambda s.\ \begin{cases} s, & \text{if } c(s) \neq \bot \\ \bot, & \text{if } c(s) = \bot \end{cases}$$

  State changes should be irreversible. (The state is single-threaded).

- Expression snapback:

$$\textbf{do\_result\_} : \textbf{com} \times \textbf{exp} \to \textbf{exp}$$
$$\textbf{do}\ c\ \textbf{result}\ e \;=\; \lambda s.\, e(c(s))$$

  Expressions should only read the state (passivity).

- Intensional models can eliminate such “junk” relatively easily.
- Eliminating “junk” in extensional models involves inventing mathematical structure.

Example equivalences

- Irreversible state change is involved in this equivalence. Suppose $p : \mathbf{com} \to \mathbf{com}$.

$$\begin{aligned} &\mathbf{new}\ x.\ \mathbf{let}\ inc = (x := !x + 1) \\ &\qquad\qquad \mathbf{in}\ x := 0;\ p(inc); \mathbf{if}\ !x > 0\ \mathbf{then\ diverge} \\ &\equiv p(\mathbf{diverge}) \end{aligned}$$

- The first command diverges if p runs its argument command and terminates if p ignores its argument. The second command has exactly the same effect.
- The command snapback would break this equivalence. If $p = \mathbf{try}$, then it can run the argument command and set the state back to the initial state.

Example equivalences (contd)

- Passivity is involved in this equivalence (remember call-by-name):

$$\textbf{if } !x = 0 \textbf{ then } f(!x) \textbf{ else } 2 \;\equiv\; \textbf{if } !x = 0 \textbf{ then } f(0) \textbf{ else } 2$$

  where x : **var** and f : **exp** $\to$ **exp**.
- Since $f(!x)$ is a passive expression, it can only read x and it gets the value 0.
- Expression snapback would break this equivalence. Suppose:

$$f = \lambda e.\, \textbf{do } x := !x + 1 \textbf{ result } e$$

  then $f(!x)$ would have the effect of $f(1)$.

Prospective

- Strachey was a great pioneer, a founder of our discipline, an intellectual father for all of us.
- But, if he were here, he might not be satisfied with what we have done with his ideas.
- He possibly did not intend semantics to be merely a research discipline, but rather a practical tool to be used in everyday programming.
- Can we do more to propagate these ideas and to make them practical?