In what 2-category do PCAs most naturally live?

John Longley
Laboratory for Foundations of Computer Science
University of Edinburgh

PCARC workshop, 9 July 2010

Background

In 1992 (PSSL 50), I introduced a theory of PCAs and applicative morphisms, as a framework for investigating, *e.g.*,

> Which PCAs can be 'simulated' in which other PCAs, and in what ways?

Mathematically pleasing, but . . .

- most 'models of computation' aren't (naturally) PCAs,
- the category of PCAs doesn't have much good structure.

In 1999 (FLoC, Trento), I gave a generalization to typed PCAs. Admitted a lot more examples, but still excluded many important 'models' (e.g. process calculi, labelled transition systems).

> How far can the mathematical theory be generalized?

Goal of this talk

Generalize the ideas of 'model' and 'simulation' still further, in such a way that

- the nice mathematical theory still goes through,
- a wide range of models from across CS are admitted,
- the class of models has better structure / closure properties

Key idea: PCAs and TPCAs naturally model higher order flavours of computation.

Here we 'flatten' everything out to first order, and later show how higher order models fit in.

The original theory for PCAs (quick review)

A PCA is a partial applicative structure $(A,\ \cdot : A \times A \rightharpoonup A)$ containing elements k, s such that

$$k \cdot x \cdot y \ = \ x \qquad s \cdot x \cdot y \downarrow \qquad s \cdot x \cdot y \cdot z \ = \ x \cdot z \cdot (y \cdot z)$$

An applicative morphism $\gamma : A \longrightarrow\!\!\!\triangleright\, B$ is a total relation such that for some $r \in B$ we have

$$\gamma(a, b) \wedge \gamma(a', b') \wedge a \cdot a' \downarrow \ \Rightarrow \ \gamma(a \cdot a',\ r \cdot b \cdot b')$$

Given $\gamma, \delta : A \longrightarrow\!\!\!\triangleright\, B$, we write $\gamma \preceq \delta$ if for some $t \in B$ we have

$$\gamma(a, b) \ \Rightarrow \ \delta(a,\ t \cdot b)$$

All this defines a preorder-enriched category $\mathcal{PCA}$.

Connection with realizability models (review)

For any PCA A, we can build a category of assemblies $\mathcal{A}sm(A)$.

An applicative morphism $\gamma : A \longrightarrow\!\!\triangleright B$ then induces a functor $\mathcal{A}sm(\gamma) : \mathcal{A}sm(A) \to \mathcal{A}sm(B)$.

Theorem: The functors so arising are (up to isomorphism) precisely the regular functors $\mathcal{A}sm(A) \to \mathcal{A}sm(B)$ that commute with the forgetful functors Γ_A, Γ_B to $\mathcal{S}et$ and the inclusions ∇_A, ∇_B from $\mathcal{S}et$.

In fact, the $\mathcal{A}sm$ construction extends to a 2-functor $\mathcal{PCA} \to \Gamma\nabla\mathcal{REG}$ which is locally an equivalence.

Corollary: $\mathcal{A}sm(A) \simeq \mathcal{A}sm(B)$ (as categories) iff $A \simeq B$ (in $\mathcal{PCA}$).

Typed PCAs (brief sketch)

Instead of a single carrier set A, we may allow a whole family of carrier sets corresponding to different 'datatypes'.

By definition, typed PCAs are higher order: for any types A, B, there's a type $[A \Rightarrow B]$ with an application $\cdot : [A \Rightarrow B] \times A \to B$.

Ordinary 'untyped' PCAs arise as a special case: $[A \Rightarrow A] = A$.

Modulo a few type decorations, everything on the last two slides still works.

Sample results and applications

1. Any PCA A admits a boolean-respecting applicative morphism $K_1 \longrightarrow\!\!\triangleright A$, unique up to $\preceq\succeq$.
2. Let C be the typed PCA of (Kleene-Kreisel) total continuous functionals over N, and P that of (Scott-Ershov) partial continuous functionals. There is essentially just one N-respecting applicative morphism C $\longrightarrow\!\!\triangleright K_2$. Similarly for P $\longrightarrow\!\!\triangleright K_2$, though not e.g. for C $\longrightarrow\!\!\triangleright$ P.
3. The total extensional collapses of P and K_2 are isomorphic (both yield C). Quite hard to prove 'directly', but routine by induction on types if we strengthen claim to 'isomorphic realizably over K_2'.

Can one obtain results in this spirit for a wider range of 'models of computation'?

Main definition I: C-structures. (New stuff starts here)

A C-structure $\mathbf{C}$ consists of:

- a family $|\mathbf{C}|$ of inhabited sets (think datatypes)
- for each $A, B \in |\mathbf{C}|$, a set $\mathbf{C}[A, B]$ of relations from A to B (think computable operations, which may be partial and/or non-deterministic)

such that

- for each $A \in |\mathbf{C}|$ we have $\mathrm{id}_A \in \mathbf{C}[A, A]$
- for any $r \in \mathbf{C}[A, B]$, $s \in \mathbf{C}[B, C]$ there exists $t \in \mathbf{C}[A, C]$ such that $r(a, b) \wedge s(b, c) \Rightarrow t(a, c)$ (call any such t a supercomposite of r and s).

Examples of C-structures (sketch)

1. Any typed PCA: let $|\mathbf{C}|$ be its collection of types, and $\mathbf{C}[A, B]$ the set of partial functions represented by an element of $[A \Rightarrow B]$.

2. Let $\mathcal{L}$ be your favourite programming language or process calculus. Let $|\mathbf{C}|$ be some class of 'values' in $\mathcal{L}$ (e.g. whnf's) sorted by type. For any 'evaluation context' $K[-]$ of $\mathcal{L}$, let r_K be the relation $\{(t, u) \mid K[t] \rightsquigarrow^* u\}$ on $|\mathbf{C}|$-terms, and let $\mathbf{C}[A, B]$ be the set of r_K for suitably typed K.

3. Given any labelled transition system, let $|\mathbf{C}| = \{S\}$ where S is the set of states. For w any finite sequence of labels, let r_w be the relation $\{(x, y) \mid x \xrightarrow{w} y\}$ on S, and let $\mathbf{C}[S, S]$ be the set of such r_w.

Main definition II: Realizations

Let $\mathbf{C}, \mathbf{D}$ be C-structures. A realization $\gamma : \mathbf{C} \longrightarrow\!\!\triangleright \mathbf{D}$ consists of:

- a function $\gamma : |\mathbf{C}| \to |\mathbf{D}|$,
- for each $A \in |\mathbf{C}|$, a total relation γ_A from A to γA

such that every $r \in \mathbf{C}[A, B]$ is tracked by some $r' \in \mathbf{D}[\gamma A, \gamma B]$:

$$r(a,b) \wedge \gamma_A(a,a') \Rightarrow \exists b'.\, r'(a',b') \wedge \gamma_B(b,b')$$

(Choice here re non-determinism: will revisit later.)

If $\gamma, \delta : \mathbf{C} \longrightarrow\!\!\triangleright \mathbf{D}$ are realizations, we say γ is transformable to δ ($\gamma \preceq \delta$) if for each $A \in |\mathbf{C}|$ there exists $t \in \mathbf{D}[\gamma A, \delta A]$ such that

$$\gamma_A(a,a') \Rightarrow \exists a''.\, t(a',a'') \wedge \delta_A(a,a'')$$

Fact: All this defines a preorder-enriched category $\mathcal{CSTRUCT}$.

The $\mathcal{A}sm$ construction on C-structures

Given a C-structure $\mathbf{C}$, define a category $\mathcal{A}sm(\mathbf{C})$ as follows.

- Objects X are triples $(|X|, A_X, \Vdash_X)$, where $|X|$ is a set, $A_X \in |\mathbf{C}|$, and $\Vdash_X \subseteq A_X \times |X|$ satisfies $\forall x.\exists a.\, a \Vdash_X x$.

- Morphisms $f : X \to Y$ are functions $f : |X| \to |Y|$ that are 'tracked' by some $r \in \mathbf{C}[A_X, A_Y]$ (again, choice here):

$$a \Vdash_X x \wedge f(x) = y \Rightarrow \exists b.\, b \Vdash_Y y \wedge r(a, b)$$

N.B. By the realizability model on $\mathbf{C}$, we shall mean $\mathcal{A}sm(\mathbf{C})$ equipped with its forgetful functor $\Gamma_{\mathbf{C}} : \mathcal{A}sm(\mathbf{C}) \to \mathcal{S}et$.

Structure in $(\mathcal{A}sm(\mathbf{C}), \Gamma_C)$

- Subobjects: given $X \in \mathcal{A}sm(\mathbf{C})$, any subset of $\Gamma(X)$ lifts to a subobject of X with the expected universal property.

- Quotients: given $X \in \mathcal{A}sm(\mathbf{C})$, any quotient of $\Gamma(X)$ lifts to a quotient of X with the expected universal property.

- 'Copies': given $X \in \mathcal{A}sm(\mathbf{C})$ and $S \in \mathcal{S}et$, there is an object $X \propto S \in \mathcal{A}sm(\mathbf{C})$ equipped with morphisms

$$\pi : X \propto S \to X \qquad \rho : \Gamma(X \propto S) \to S$$

  satisfying an obvious universal property.

In general, we say $(\mathcal{C},\ \Gamma : \mathcal{C} \to \mathcal{S}et)$ is a quasi-regular Γ-category if it possesses this structure.

Extending $\mathcal{A}sm$ to realizations

A realization $\gamma : \mathbf{C} \longrightarrow\!\!\!\triangleright\, \mathbf{D}$ induces a quasi-regular Γ-functor

$$\mathcal{A}sm(\gamma) : \mathcal{A}sm(\mathbf{C}) \to \mathcal{A}sm(\mathbf{D})$$

Indeed, up to iso, every such functor arises in this way.

Theorem: $\mathcal{A}sm$ extends to a 2-functor $\mathcal{CSTRUCT} \to \Gamma\mathcal{QREG}$ which is locally an equivalence.

Corollary: $\mathcal{A}sm(\mathbf{C}) \simeq \mathcal{A}sm(\mathbf{D})$ as Γ-categories iff $\mathbf{C} \simeq \mathbf{D}$ as C-structures.

This validates the definition of $\mathcal{CSTRUCT}$ to some extent.

Subcategories of $\mathcal{CSTRUCT}$

Many interesting classes of C-structures and/or realizations can be identified.

E.g. C-structures can be deterministic, be total, have booleans, have natural numbers, . . .

Realizations can be discrete, be projective, respect booleans, respect natural numbers, . . .

Several of these properties are reflected in properties of the corresponding categories/functors (much as in PCA setting).

Let's look at a less familiar property (recall the choice re non-determinism).

Tight C-structures and realizations

Call a C-structure tight if for all $r \in \mathbf{C}[A,B]$, $s \in \mathbf{C}[B,C]$ there exists $t \in \mathbf{C}[A,C]$ such that

$$r(a,b) \wedge s(b,c) \wedge t(a,c') \Rightarrow \exists b'.\, r(a,b') \wedge s(b',c')$$

Call a realization γ tight if every $r \in \mathbf{C}[A,B]$ is 'tightly tracked' by some $r' \in \mathbf{D}[\gamma A, \gamma B]$: that is, r' tracks r, and

$$r(a,b) \wedge \gamma(a,a') \wedge r'(a',b') \Rightarrow \gamma(b,b')$$

Similarly define a tight morphism in $\mathcal{A}sm(\mathbf{C})$.

If $\mathbf{C}$ is tight, the tight morphisms form a subcategory $\mathcal{A}sm_t(\mathbf{C})$ of $\mathcal{A}sm(\mathbf{C})$. Moreover, the quasi-regular Γ-functors $\mathcal{A}sm(\mathbf{C}) \to \mathcal{A}sm(\mathbf{D})$ corresponding to tight realizations are precisely those that restrict to $\mathcal{A}sm_t(\mathbf{C}) \to \mathcal{A}sm_t(\mathbf{D})$.

Another subclass: $\mathbb{C}$-structures with products

Say $\mathbf{C}$ has finite (monoidal) products if $|\mathbf{C}|$ contains 1 and is closed under binary products, pairings of computable relations exist, and moreover the associativity and left/right unit mappings are present in $\mathbf{C}$ (in both directions).

This makes $\mathcal{A}sm(\mathbf{C})$ a monoidal category.

Say $\gamma : \mathbf{C} \longrightarrow\!\!\triangleright \mathbf{D}$ is monoidal if suitable relations are present in $\mathbf{D}[\gamma A \times \gamma B, \gamma(A \times B)]$ and $\mathbf{D}[1, \gamma 1]$.

Then $\mathcal{A}sm(\gamma)$ is a monoidal functor iff γ is monoidal.

Higher order C-structures

Assume $\mathbf{C}$ has finite products.
Say $\mathbf{C}$ is higher order if for any $A, B \in |\mathbf{C}|$ there exist $[A \Rightarrow B] \in |\mathbf{C}|$ and $ev_{A,B} \in \mathbf{C}[[A \Rightarrow B] \times A,\ B]$ such that

$$\forall r \in \mathbf{C}[C \times A, B].\, \exists \tilde{r} \in \mathbf{C}[C, [A \Rightarrow B]].\, r = (\tilde{r} \times \mathsf{id}_A); ev$$

(Uniqueness not required.)

Now, a realization $\gamma : \mathbf{C} \longrightarrow\!\!\triangleright\, \mathbf{D}$ is precisely a family of relations such that pairing and application in $\mathbf{C}$ are tracked in $\mathbf{D}$. So PCA-style applicative morphisms are simply monoidal realizations.

Philosophical point: ‘Equivalence’ for notions of higher order computation is nothing more than their equivalence as first order notions.

Structure in $\mathcal{CSTRUCT}$

Early indications suggest that $\mathcal{CSTRUCT}$ has a respectable amount of categorical structure. E.g.

- Products (no surprise)
- Sums via disjoint union (not available in $\mathcal{PCA}$).
- Curiosity: $\mathcal{CSTRUCT}$ is almost cartesian closed!

  Specifically, given $\mathbf{C}$ and $\mathbf{D}$, there exists a realization $eval : \mathbf{D}^{\mathbf{C}} \times \mathbf{C} \longrightarrow\!\!\triangleright \mathbf{D}$ such that for any $\alpha : \mathbf{E} \times \mathbf{C} \longrightarrow\!\!\triangleright \mathbf{D}$ there's an $\tilde{\alpha} : \mathbf{E} \longrightarrow\!\!\triangleright \mathbf{D}^{\mathbf{C}}$ making the usual diagram commute, and moreover $\tilde{\alpha}$ is unique up to $\preceq\succeq$ among single-valued realizations with this property.

  This is enough to characterize $\mathbf{D}^{\mathbf{C}}$ up to equivalence in $\mathcal{CSTRUCT}$.

  No idea what this 'means', but it's an encouraging sign!

Construction of $\mathbf{D}^{\mathbf{C}}$ (sketch)

A family $\mathcal{F}$ of realizations $\mathbf{C} \longrightarrow\!\triangleright\, \mathbf{D}$ is uniformly tracked if

- all members of $\mathcal{F}$ agree at the level of types: $\gamma A = \gamma' A$ for all $\gamma, \gamma' \in \mathcal{F}$, $A \in |\mathbf{C}|$
- for all $A, B \in |\mathbf{C}|$ and $r \in \mathbf{C}[A, B]$ there exists some r' in $\mathbf{D}$ that tracks r w.r.t. every $\gamma \in \mathcal{F}$.

If $\mathcal{F}, \mathcal{G}$ are uniformly tracked families, a relation $\mathcal{R} \subseteq \mathcal{F} \times \mathcal{G}$ is uniformly transformable if for all $A \in |\mathbf{C}|$ there exists t in $\mathbf{D}$ such that for all $(\gamma, \delta) \in \mathcal{R}$, t witnesses $\gamma \preceq \delta$ at A.

The C-structure $\mathbf{D}^{\mathbf{C}}$ is now defined as follows:

- $|\mathbf{D}^{\mathbf{C}}|$ is the set of inhabited, uniformly tracked families
- $\mathbf{D}^{\mathbf{C}}[\mathcal{F}, \mathcal{G}]$ is the set of uniformly transformable $\mathcal{R} \subseteq \mathcal{F} \times \mathcal{G}$.

Some scattered remarks

$K_1^{K_1}$ is vast and complicated (probably worse than the lattice of Turing degrees).

However, the analogue for boolean-respecting realizations is just the one-element C-structure.

Let $L = \Lambda^0/\beta$. It's amusing to see how many inequivalent boolean-respecting realizations $L \longrightarrow\!\!\triangleright L$ one can find. So the boolean-respecting analogue of L^L might be interesting.

Crazy idea: 'homotopy theory' for notions of computability?

Conclusions and further work

C-structures give us a much larger and more 'rounded' class of models of computation than typed PCAs. The switch from higher order to first order seems crucial.

(Moral: perhaps classifying higher order computability notions is somehow a less 'natural' goal than I thought?)

It would be nice to have some examples of interesting results involving realizations for process calculi etc. (E.g. that two existing process calculi are non-trivially equivalent in $\mathcal{CSTRUCT}$?)

Could also be interesting to think about examples arising from physical systems, where 'computable' could mean 'physically realizable' in some sense.