Free-Algebra Models for the π-Calculus

Ian Stark

Laboratory for Foundations of Computer Science
School of Informatics
The University of Edinburgh

Tuesday 14 June 2005

http://www.ed.ac.uk/~stark/freamp.html

Summary

The finite π-calculus:

$$P \ ::= \ \bar{x}y.P \ | \ x(y).P \ | \ \nu x.P \ | \ P + Q \ | \ P|Q \ | \ 0$$

has an explicit set-theoretic model, fully-abstract for strong late bisimulation congruence. [Fiore, Moggi, Sangiorgi; Stark]

We characterise this as the minimal free algebra for certain operations and equations, in the setting of Power and Plotkin's enriched Lawvere theories.

Summary

The finite π-calculus:

$$P ::= \bar{x}y.P \mid x(y).P \mid \nu x.P \mid P + Q \mid P|Q \mid 0$$

has an explicit set-theoretic model, fully-abstract for strong late bisimulation congruence. [Fiore, Moggi, Sangiorgi; Stark]

We characterise this as the minimal free algebra for certain operations and equations, in the setting of Power and Plotkin's enriched Lawvere theories.

This combines separate theories of nondeterminism, I/O and name creation in a modular fashion. As a bonus, we get a whole category of models, a modal logic and a computational monad. The tricky part is that everything has to happen inside the functor category $Set^{\mathcal{I}}$.

Overview

- Equational theories for different features of computation
- Using the functor category $Set^{\mathcal{I}}$
- A theory of π
- Free-algebra models and full abstraction

Notions of computation

Moggi: Computational monads for programming language features

- Nondeterminism $TX = \mathcal{P}_{\mathtt{fin}}X$
- Mutable state $TX = (S \times X)^S$
- Interactive I/O $TX = \mu Y.(X + V \times Y + Y^V)$
- Exceptions $TX = X + E$

Power and Plotkin:

Monad $\longleftrightarrow$ Algebraic theory

Use correspondence to characterize each T as free model for appropriate "notion of computation"

Algebras for nondeterministic computation

An object of nondeterministic computation A in cartesian $\mathcal{C}$ needs ...

Operations

$$\mathrm{choice} : A^2 \longrightarrow A$$
$$\mathrm{nil} : 1 \longrightarrow A$$

Equations

$$\mathrm{choice}(p, q) = \mathrm{choice}(q, p)$$
$$\mathrm{choice}(\mathrm{nil}, p) = \mathrm{choice}(p, p) = p$$
$$\mathrm{choice}(p, (\mathrm{choice}(q, r)) = \mathrm{choice}(\mathrm{choice}(p, q), r)$$

...giving a category $\mathcal{ND}(\mathcal{C})$ of algebras $(A, \mathrm{choice}, \mathrm{nil})$

Free algebras

Free $\mathcal{ND}$-algebras over sets give a computational monad:

$$\mathcal{ND}(\mathit{Set})$$

$$\text{free} \qquad F \uparrow \;\dashv\; \downarrow U \qquad \text{forgetful}$$

$$\mathit{Set}$$

$$T = (U \circ F) = \mathcal{P}_{\mathit{fin}}$$

Operations induce generic effects in the Kleisli category:

$$\left.\begin{aligned} \mathit{choice} : A^2 &\longrightarrow A^1 \\ \mathit{nil} : A^0 &\longrightarrow A^1 \end{aligned}\right\} \implies \left\{\begin{aligned} \mathit{arb} : 1 &\longrightarrow T\,2 \\ \mathit{deadlock} : 1 &\longrightarrow T\,0 \end{aligned}\right.$$

Notions of computation determine monads

Power/Plotkin

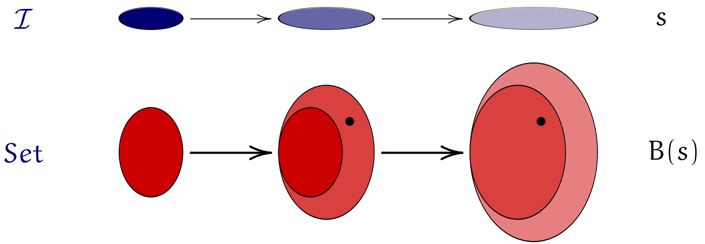
Operations + Equations $\longrightarrow$ Free-algebra models of computational features

$\longrightarrow$ Monads + generic effects

- Characterisation of known computational monads *and* effects
- Simple and flexible combination of theories
- Enriched models and arities: countably infinite, posets, $\omega\mathrm{Cpo}$

Varying sets

Functor category $Set^{\mathcal{I}}$: structures that vary with the available names
where $\mathcal{I}$ = finite name sets and injections



Object $B \in Set^{\mathcal{I}}$ is a varying set: for finite name set s it gives a set $B(s)$ of values using names from s, and says how they change with renaming.

Structure in $Set^{\mathcal{I}}$

$Set^{\mathcal{I}}$ has two jobs:

- Arena for building name-aware algebras and monads
- Source of arities for operations

Relevant structure:

- Pairs $A \times B$ and function space $A \to B$
- Separated pairs $A \otimes B$ and fresh function space $A \multimap B$
- Object of names N
- Shift endofunctor $\delta A = A(\_ + 1)$, with $\delta A \cong N \multimap A$

In particular, object N serves as a varying arity.

Constructions in $Set^{\mathcal{I}}$

Cartesian closed

$$(A \times B)(k) = A(k) \times B(k)$$
$$B^A(k) = [A(k + \_), B(k + \_)]$$

Monoidal closed

$$(A \otimes B)(k) = \int^{k' + k'' \hookrightarrow k} A(k') \times B(k'')$$
$$(A \multimap B)(k) = [A(\_), B(k + \_)]$$

More constructions in $Set^{\mathcal{I}}$

Object of names, shift operator

$$N(k) = k$$
$$\delta A(k) = A(k+1)$$

Connections

$$A \otimes B \longrightarrow A \times B \qquad \delta A \cong N \multimap A$$
$$(A \to B) \longrightarrow (A \multimap B) \qquad \delta N \cong N + 1$$

When A and B are pullback-preserving, the two maps are injective and surjective respectively.

Structure in $Set^{\mathcal{I}}$

$Set^{\mathcal{I}}$ has two jobs:

- Arena for building name-aware algebras and monads
- Source of arities for operations

Relevant structure:

- Pairs $A \times B$ and function space $A \to B$
- Separated pairs $A \otimes B$ and fresh function space $A \multimap B$
- Object of names N
- Shift endofunctor $\delta A = A(\_ + 1)$, with $\delta A \cong N \multimap A$

In particular, object N serves as a varying arity.

Theory of π: operations

Nondeterministic computation

$\mathit{nil} : 1 \longrightarrow A$	inactive process	0
$\mathit{choice} : A^2 \longrightarrow A$	process sum	$P + Q$

Input/Output

$\mathit{out} : A \longrightarrow A^{N \times N}$	output prefix	$\bar{x}y.P$
$\mathit{in} : A^N \longrightarrow A^N$	input prefix	$x(y).P$
$\mathit{tau} : A \longrightarrow A$	silent prefix	$\tau.P$

Dynamic name creation

$\mathit{new} : \delta A \longrightarrow A$	restriction	$\nu x.P$

Theory of π: component equations

Nondeterministic computation

$choice$: commutative, associative and idempotent with unit nil

Input/Output

None

Dynamic name creation

$$new(x.p) = p$$
$$new(x.new(y.p)) = new(y.new(x.p))$$

Theory of π: combining equations

Commuting component theories

$$new(x.choice(p, q)) = choice(new(x.p), new(x.q))$$

$$\begin{aligned} new(z.out_{x,y}(p)) &= out_{x,y}(new(z.p)) && z \notin \{x, y\} \\ new(z.in_x(p_y)) &= in_x(new(z.p_y)) && z \notin \{x, y\} \\ new(z.tau(p)) &= tau(new(z.p)) \end{aligned}$$

Interaction between component theories

$$\begin{aligned} new(x.out_{x,y}(p)) &= nil \\ new(x.in_x(p_y)) &= nil \end{aligned}$$

Models for the theory of π

- Category $\mathcal{PI}(Set^{\mathcal{I}})$ of π-algebras $(A \in Set^{\mathcal{I}}; in, out, \ldots, new)$
- Process P with free names in s interpreted by $[\![P]\!]_A : N^s \longrightarrow A$
- Definition by induction over the structure of P, using operations of the theory (and the expansion law for parallel composition)

Theorem

Every such π-algebra interpretation respects strong late bisimulation congruence:

$$P \approx Q \quad \Longrightarrow \quad [\![P]\!]_A = [\![Q]\!]_A$$

Of course, this doesn't yet give us any actual π-algebras to work with

Free models for π

Each component theory has a standard monad:

Nondeterminism	$\mathcal{P}_{\mathtt{fin}}(X)$
Input/Output	$\mu Y.\left(X + (N \times N \times Y) + ()N \times Y^N) + Y\right)$
Name creation	$\mathrm{Dyn}(X) = \int^k X(\_ + k)$

For the full theory of π:

$$\mathrm{Pi}(X) = \mu Y.\mathcal{P}_{\mathtt{fin}}\left(\mathrm{Dyn}(X) + (N \times N \times Y) + (N \times \delta Y) + (N \times Y^N) + Y\right)$$

... which is *not quite* an interleaving of the component monads

Results

Theorem

The category of π-algebras is monadic over $\mathrm{Set}^{\mathcal{I}}$:

$$\textit{free} \qquad \mathrm{Pi} \; \left(\mathcal{PI}(\mathrm{Set}^{\mathcal{I}}) \underset{\mathrm{U}}{\overset{\mathrm{Pi}}{\rightleftarrows}} \mathrm{Set}^{\mathcal{I}} \right) \; \dashv \qquad \mathrm{U} \qquad \textit{forgetful}$$

Monad $T_\pi = (\mathrm{U} \circ \mathrm{Pi})$ for concurrent name-passing programs:

$$\begin{aligned} \mathrm{arb} : 1 &\longrightarrow T2 & \qquad \mathrm{send} : N \times N &\longrightarrow T1 \\ \mathrm{deadlock} : 1 &\longrightarrow T0 & \mathrm{receive} : N &\longrightarrow TN \\ \mathrm{skip} : 1 &\longrightarrow T1 & \mathrm{fresh} : 1 &\longrightarrow TN \end{aligned}$$

Results

We have the following:

- A category $\mathcal{PI}(\mathcal{S}et^{\mathcal{I}})$ of π-algebras, all sound models of π-calculus bisimulation:

$$\mathrm{P} \approx \mathrm{Q} \quad \Longrightarrow \quad [\![\mathrm{P}]\!]_A = [\![\mathrm{Q}]\!]_A$$

- An explicit free-algebra construction $\mathtt{Pi} : \mathcal{S}et^{\mathcal{I}} \to \mathcal{PI}(\mathcal{S}et^{\mathcal{I}})$ such that all $\mathtt{Pi}(X)$ are fully-abstract models of π:

$$\mathrm{P} \approx \mathrm{Q} \quad \Longleftrightarrow \quad [\![\mathrm{P}]\!]_{\mathtt{Pi}(X)} = [\![\mathrm{Q}]\!]_{\mathtt{Pi}(X)}$$

- The initial free algebra $\mathtt{Pi}(0)$ is in fact the previously known fully-abstract model.

Review

- Operations + equations with enriched arities
  $\Longrightarrow$ algebraic models for features of computation
- Modular theory for π-calculus, with $Set^{\mathcal{I}}$ for both arities and algebras:

$$\pi = (\text{Nondeterminism} + \text{I/O} + \text{Name creation}) \;/\; \mathit{new} \leftrightarrow \mathrm{i/o}$$

- Explicit formulation of free algebras for this theory; all fully abstract for bisimulation congruence
- The induced computational monad is almost, but not quite, the combination of its three components.

What next?

- Use $\mathrm{FM\text{-}Cpo}$ for the full π-calculus
- Partial order arities for testing equivalences [Hennessy]
- Modal logic from the theory of π
- Modify interpretation *or* equations for early/open/weak bisimulation
- Try $\mathrm{Pi}(X)$ for applied π
- Investigate algebraic par (with effect $\mathtt{fork} : 1 \to T2$)
- Expose $\mathit{Set}^{\mathcal{I}}$ as the category of algebras for a theory of equality testing in $\mathit{Set}^{\mathcal{F}}$; and redo everything in the single cartesian closed structure of $\mathit{Set}^{\mathcal{F}}$. ($\mathcal{F}$ finite sets and all maps)

Parallel composition

Parallel composition of π-calculus processes is not algebraic, but still:

- All π-algebras can support $(P \mid Q)$ externally by expansion.
- All free π-algebras have an internally-defined map

$$par_{X,Y} : Pi(X) \times Pi(Y) \longrightarrow Pi(X \times Y) .$$

- Any multiplication $\mu : X \times X \to X$ then gives us

$$par_{\mu} : Pi(X) \times Pi(X) \longrightarrow Pi(X) .$$

- For $X = 0$, this is standard parallel composition; for $X = 1$ we get the same with an extra success process $\checkmark$.

Modal logic

Any theory gives rise to a modal logic over its algebras, with possibility and necessity modalities for every operation.

$$\begin{aligned} P \vDash \Diamond \mathrm{out}_{x,y}(\phi) &\iff \exists Q.\ P \sim \bar{x}y.Q \ \wedge\ Q \vDash \phi \\ P \vDash \Box \mathrm{out}_{x,y}(\phi) &\iff \forall Q.\ P \sim \bar{x}y.Q \ \Rightarrow\ Q \vDash \phi \\ P \vDash \Diamond \mathrm{choice}(\phi, \psi) &\iff \exists Q, R.\ P \sim (Q + R) \ \wedge\ Q \vDash \phi \ \wedge\ R \vDash \psi \end{aligned}$$

HML is definable:

$$\langle \bar{x}y \rangle \phi \ = \ \Diamond \mathrm{choice}(\Diamond \mathrm{out}_{x,y}(\phi), \mathrm{true})$$

We could also take other algebraic operations and define modalities. However, in no case is there a $(\phi \mid \psi)$ modality.