Another look at countable products of searchable sets

Martín Escardó
26 Feb 2008
Theory Lab Lunch

Martín Escardó — Lab lunch talk 26 Feb 2008
(Another look at my LICS'2007 paper.)

searchable subset of a type X

$A \subseteq X$ $\quad\quad$ $\varepsilon : (X \to B) \to X$

$\quad\quad p \longmapsto a_0 \in A$ s.t.
if $\exists a \in A .\ p(a)$ then $p(a_0)$

$$SX \stackrel{\text{def}}{=} (X \to B) \to X$$

Exhaustible subset

$$QX = (X \to B) \to B$$

$\exists : (X \to B) \to B$

$p \longmapsto$ truth value $\exists a \in A .\ p(a)$

Proposition searchable $\Rightarrow$ exhaustible

Proof
$$\begin{aligned} SX &\longrightarrow QX \\ \varepsilon &\longmapsto \lambda p . p(\varepsilon\, p) \end{aligned}$$
$\square$

Proposition computable images of exhaustible sets are exhaustible.
$\quad\quad\quad\quad\quad\quad\quad\quad$ searchable $\quad\quad\quad\quad$ searchable

Proof (1) let $f : X \to Y$. Then $\exists y \in f(A) .\ q(y) \Leftrightarrow \exists x \in A .\ q(f(x))$

$$\begin{aligned} (X \to Y) &\longrightarrow (QX \longrightarrow QY) \\ f &\longmapsto \lambda \exists . \quad \lambda p .\ \exists(\lambda x . q(f(x))) \end{aligned}$$

(2) To find $y \in f(A)$ s.t. $q(y)$, find $x \in A$ s.t. $q(f(a))$
and apply f to it.

$$\begin{aligned} (X \to Y) &\longrightarrow (SX \longrightarrow SY) \\ f &\longmapsto \lambda \varepsilon . \quad \lambda q .\ f(\varepsilon(\lambda x . p(f\, x))). \end{aligned}$$

$\square$

(This part from Barbados notes 2004.)

<u>Proposition</u> Singletons (consisting of a computable element) are exhaustible & searchable.

<u>Proof</u> (1)
$$\begin{aligned} X &\longrightarrow QX \\ x &\longmapsto \lambda p.\, p(x) \end{aligned}$$

(2)
$$\begin{aligned} X &\longrightarrow SX \\ x &\longmapsto \lambda p.\, x \end{aligned}$$
$\square$

<u>Proposition</u> A union of exhaustible sets is exhaustible.
searchable searchable

<u>Proof</u> (1) $x \in \bigcup \mathcal{A}$ iff $\exists A \in \mathcal{A}$ s.t. $x \in A$

$$\begin{aligned} QQX &\longrightarrow QX \\ \exists &\longmapsto \lambda p.\, \exists(\lambda \exists.\, \exists(p)) \end{aligned}$$

(2) To find $x \in \bigcup \mathcal{A}$ s.t. $p(x)$, find $A \in \mathcal{A}$ s.t. $\exists a \in A$. $x \in A$ s.t. $p(x)$.
& then find

$$\begin{aligned} SSX &\longrightarrow SX \\ E &\longmapsto \lambda p.\, E(\lambda \varepsilon.\, p(\varepsilon\, p))(p) \end{aligned}$$

$\square$

We have two monads and a monad morphism.

Functoriality
Naturality
&
unit and
associativity laws.

$$\begin{tikzcd} SX \arrow[r, "\eta_{SX}"] \arrow[dr, equal] & SSX \arrow[d, "\mu_X"] & SX \arrow[l, "S\eta_X"'] \arrow[dl, equal] \\ & SX & \end{tikzcd} \qquad \begin{tikzcd} SSSX \arrow[r, "S\mu_X"] \arrow[d, "\mu_{SX}"'] & SSX \arrow[d, "\mu_X"] \\ SSX \arrow[r, "\mu_X"'] & SX \end{tikzcd}$$

$$\begin{tikzcd} SX \arrow[rr, "\alpha_X"] & & QX \\ & X \arrow[ul, "\eta_X"] \arrow[ur, "\eta_X"'] & \end{tikzcd} \qquad \begin{tikzcd} SSX \arrow[r, "S\alpha_X \cdot \alpha_{QX} = \alpha_{SX} \cdot Q\alpha_X"] \arrow[d, "\mu_X"'] & QQX \arrow[d, "\mu_X"] \\ SX \arrow[r, "\alpha_X"'] & QX \end{tikzcd}$$

So what?

Want to derive <u>other</u> search algorithms from this.

These monads are automatically strong (we are in a ccc & the functor, unit, multiplication are internalized)

<u>Extension operator</u>

$f : X \longrightarrow SY$

$$\begin{tikzcd} SX \arrow[rr, dashed, "f^{\#}"] \arrow[dr, "Sf"'] & & SY \\ & SSY \arrow[ur, "\mu_Y"'] & \end{tikzcd}$$

This realizes the following proposition.

<u>Proposition</u> If $\{B_x \subseteq Y \mid x \in X\}$ is a family of sets s.t. B_x is searchable uniformly in $x \in X$ and $A \subseteq X$ is searchable, then $\bigcup_{x \in A} B_x$ is searchable.

<u>Proof</u> let $f : X \longrightarrow SY$ s.t $f(x) = \varepsilon_{B_x}$.

Then $\varepsilon_{\bigcup\{B_x \mid x \in A\}} = f^{\#}(\varepsilon_A)$. $\square$

In any ccc-monad the strength can be defined as

$$X \times TY \longrightarrow T(X \times Y)$$

$$(x, \varepsilon) \longmapsto \underbrace{(\lambda y.\, \eta(x,y))}_{Y \longrightarrow T(X \times Y)}{}^{\#}(\varepsilon)$$

This realizes:

<u>Proposition</u> If $x \in X$ and $B \subseteq Y$ is searchable, then $\{x\} \times B$ is searchable.

<u>co-strength</u>: $TX \times Y \longrightarrow T(X \times Y)$

$$(\varepsilon, y) \longmapsto (\lambda x.\, \eta(x,y))^{\#}(\varepsilon)$$

<u>Proposition</u> If $A \subseteq X$ and $B \subseteq Y$ are searchable, then so is $A \times B$.

<u>Proof</u> This can be realized in two different ways because the monad is not commutative.

(i) Extend strength

$$TX \times TY \longrightarrow T(X \times Y)$$

$$(\varepsilon_A, \varepsilon_B) \longmapsto (\lambda x.\, \mathrm{strength}(x, \varepsilon_B))^{\#}(\varepsilon_A)$$

(ii) Extend co-strength. $\square$

Expanding all the definitions, (i) gives the same algorithm given in LICS'2007

$$TX \times TY \longrightarrow T(X \times Y)$$

$$\begin{aligned}(\varepsilon_A, \varepsilon_B) \longmapsto\ & \text{let } a_0 = \varepsilon_A(\lambda a.\, \exists_{b\in B}\, p(a,b)) \\ & \phantom{\text{let }} b_0 = \varepsilon_B(\lambda b.\, p(a_0, b)) \\ & \text{in } (a_0, b_0)\end{aligned}$$

Curiosity (?)

$$s = \text{(co) strength}$$

$$TX \times TY \longrightarrow T(X \times Y)$$
$$\longleftarrow$$
$$r = \langle T\pi_X, T\pi_Y \rangle$$

In general, $r \circ s \neq$ identity. E.g $T = Q$

$$QX = (X \to B) \to B$$

But for $T = S$, $r \circ s =$ identity.

Is that "important"?

LICS'2007 infinite product

final coalgebra

$$
\begin{tikzcd}
X \times X^{\omega} \arrow[r, shift right, "\mathrm{cons}"'] & X^{\omega} \arrow[l, shift right, "\langle \mathrm{head}, \mathrm{tail} \rangle"']
\end{tikzcd}
$$

$$
\begin{tikzcd}
SX \times SX^{\omega} \arrow[rr, dashed, "\mathrm{cons}'"] \arrow[dr, "\mathrm{strength}"'] & & SX^{\omega} \\
& S(X \times X^{\omega}) \arrow[ur, "S\,\mathrm{cons}"'] &
\end{tikzcd}
$$

this also has a retraction

$$
\begin{tikzcd}
(SX)^{\omega} \arrow[r, "\text{LICS functional } \Pi"] & SX^{\omega} \\
SX \times (SX)^{\omega} \arrow[u, "\mathrm{cons}"] \arrow[r, "\mathrm{Id} \times \Pi"'] & SX \times SX^{\omega} \arrow[u, "\mathrm{cons}'"']
\end{tikzcd}
$$

OR:

$$\Pi\left(\mathrm{cons}(\varepsilon_0, \vec{\varepsilon})\right) = \mathrm{cons}'(\varepsilon_0, \Pi(\vec{\varepsilon}))$$

It wasn't written like this in LICS'2007, but this is equivalent.