Using the Selection Monad

.

Gordon Plotkin

.

Types and Topology
A workshop in honour of Martín Escardó's 60th birthday

Smart binary search

$$
\begin{array}{l}
\texttt{let}\,\texttt{BinIndexSearch}(x:\texttt{Int}, a:\texttt{Array}[\texttt{Int}], l:\texttt{Int}, r:\texttt{Int}) = \\
\quad \texttt{let}\, m:\texttt{Int} = \texttt{choose-from}\ [l, r] \\
\quad\quad \texttt{if}\ a[m] = x\ \texttt{then}\ m \\
\quad\quad\quad \texttt{else}\ \texttt{cost}(1); //\text{pay-to-recurse} \\
\quad\quad\quad\quad \texttt{if}\ a[m] < x\ \texttt{then}\,\texttt{BinIndexSearch}(x, a, m+1, r) \\
\quad\quad\quad\quad\quad \texttt{else}\,\texttt{BinIndexSearch}(x, a, l, m-1)
\end{array}
$$

A language

Types

$$\sigma \ ::= \ \texttt{Bool} \mid \texttt{Rew} \mid \ldots\text{other basic types } \ldots \mid$$
$$\texttt{Unit} \mid \sigma \times \sigma \mid \sigma \to \sigma$$

Terms

$$M_1 \ ::= \ x \mid \texttt{tt} \mid \texttt{ff} \mid r \mid \ldots\text{other basic type constants } \ldots \mid$$
$$M_1 = M_2 \mid M_1 \leq M_2 \mid M_1 + M_2 \mid \texttt{con}_p(M_1, M_2) \mid \ldots \mid$$
$$\texttt{if } M \texttt{ then } M_1 \texttt{ else } M_2 \mid$$
$$M_1 \,\texttt{or}\, M_2 \mid M_1 \cdot M_2 \mid M_1 +_p M_2 \mid$$
$$* \mid \langle M_1, M_2 \rangle \mid \texttt{fst}(M) \mid \texttt{snd}(M) \mid (\lambda x : \sigma . M) \mid M_1 M_2$$

Typing

Rewards

$$0 : \texttt{Rew} \qquad \frac{M_1 : \texttt{Rew} \quad M_2 : \texttt{Rew}}{M_1 \leq M_2 : \texttt{Bool}}$$

$$\frac{M_1 : \texttt{Rew} \quad M_2 : \texttt{Rew}}{M_1 + M_2 : \texttt{Rew}} \qquad \frac{M_1 : \texttt{Rew} \quad M_2 : \texttt{Rew}}{\texttt{con}_p(M_1, M_2) : \texttt{Rew}}$$

Effects: decision, reward, probability

$$\frac{M_1 : \sigma \quad M_2 : \sigma}{M_1 \,\texttt{or}\, M_2 : \sigma} \qquad \frac{M_1 : \texttt{Rew} \quad M_2 : \sigma}{M_1 \cdot M_2 : \sigma} \qquad \frac{M_1 : \sigma \quad M_2 : \sigma}{M_1 +_p M_2 : \sigma}$$

The selection monad

The monad (due to Martín Escardó and Paulo Oliva)

$$\mathrm{S}(X) \;=_{\text{def}}\; (X \to \mathbb{R}) \to X \;=\; X^{(\mathbb{R}^X)}$$

The elements of $\mathrm{S}(X)$ are selection functions $F : (X \to \mathbb{R}) \to X$.
Given a reward function $\gamma : X \to \mathbb{R}$, F selects $F\gamma \in X$

This selection may be optimal:

$$F\gamma = \mathrm{argmax}\; x \in X.\, \gamma(x)$$

Unit Effect-free computations:

$$\eta_{\mathrm{T}}(x) = \lambda\gamma.x$$

The reward associated to the selection $F\gamma \in X$ is:

$$\mathbf{R}(F|\gamma) \;=_{\text{def}}\; \gamma(F\gamma)$$

Kleisli extension

$$\frac{X \xrightarrow{f} \mathrm{S}(Y)}{\mathrm{S}(X) \xrightarrow{f_{\mathrm{S}}^{\dagger}} \mathrm{S}(Y)}$$

Equivalently:

GIVEN $f : X \to \mathrm{S}(Y) \quad F \in \mathrm{S}(X) \quad \gamma : Y \to \mathbb{R}$ FIND $y \in Y$

Step 1 Find (optimal) reward for a given choice of $x \in X$:

$$\mathbf{R}(fx|\gamma)$$

Step 2 Find (best) choice of X for the resulting (optimal) reward continuation $x \mapsto \mathbf{R}(fx|\gamma)$:

$$x_{\text{best}} \;=\; F(x \mapsto \mathbf{R}(fx|\gamma)) : X$$

Step 3 Using γ, proceed with f with that choice of $x_{\text{best}} \in X$:

$$fx_{\text{best}}\gamma : Y$$

Which is to say:

$$f^{\dagger_{\mathrm{S}}}(F)(\gamma) = fx_{\text{best}}\gamma = f(F(x \mapsto \mathbf{R}(fx|\gamma)))\gamma$$

Making binary choices

We can define a binary algebraic operation or for (binary) decision-making by making the choice giving the greatest reward. For $F, G \in \mathrm{S}(X) = (\mathbb{R}^X \to X)$ set

$$(F \,\mathrm{or}_X\, G)\gamma = \begin{cases} F\gamma & (\text{if } \mathbf{R}(F|\gamma) \geq \mathbf{R}(G|\gamma)) \\ G\gamma & (\text{otherwise}) \end{cases}$$

This operation is associative and left-biased, meaning that each component or_X is, ie for all X:

$$\begin{aligned} (F \,\mathrm{or}_X\, G) \,\mathrm{or}_X\, H &= F \,\mathrm{or}_X\, (G \,\mathrm{or}_X\, H) \\ F \,\mathrm{or}_X\, (G \,\mathrm{or}_X\, F) &= F \,\mathrm{or}_X\, G \end{aligned}$$

It is not commutative.

The selection monad transformer: adding other effects

Taking an auxiliary monad T to model other effects, one can combine it with the selection monad to obtain a monad

$$\mathrm{S}(X) = (X \to \mathbb{R}) \to \mathrm{T}(X)$$

One needs an algebra $\alpha : \mathrm{T}(\mathbb{R}) \to \mathbb{R}$ to do this.

As auxiliary monad we take

$$\mathrm{DW}(X) = \mathscr{D}_{\mathrm{f}}(\mathbb{R} \times X)$$

Denotational semantics

- We combine DW with the selection monad to obtain:

$$\mathrm{S}(X) = (X \to \mathbb{R}) \to \mathrm{DW}(X)$$

- We then use Moggi's interpretation of the computational λ-calculus, to obtain a denotational semantics of our language:

$$\frac{\Gamma \vdash M : \sigma}{\mathscr{S}[\![\Gamma]\!] \xrightarrow{\mathscr{S}[\![M]\!]} \mathrm{S}(\mathscr{S}[\![\sigma]\!])}$$

- To interpret binary decisions $M \,\texttt{or}\, N$ we use the binary algebraic operator or_{S} on S, as (almost) defined above.

- To interpret rewards $r \cdot M$ and probabilistic choice $M +_p N$ we extend the corresponding DW-algebraic operations $(r \cdot -)_{\mathrm{DW}}$ and $(+_p)_{\mathrm{DW}}$ pointwise to S.

Choice (more detail)

- As before

$$(F \,\mathrm{or}_X\, G)\gamma = \begin{cases} F\gamma & (\text{if } \mathbf{R}(F|\gamma) \geq \mathbf{R}(G|\gamma)) \\ G\gamma & (\text{otherwise}) \end{cases}$$

- but now

$$\frac{F\gamma = \Sigma_i p_i(r_i, x_i)}{\mathbf{R}(F|\gamma) = \Sigma_i p_i(r_i + \gamma x_i)}$$

Rewards and probabilistic choice

- Reward at DW level

$$r.\Sigma_i p_i(r_i, x_i) = \Sigma_i p_i(r + r_i, x_i)$$

- Reward at selection monad level

$$(r.F)\gamma = r.(F\gamma)$$

- Probabilistic choice at selection monad level

$$(F +_p G)\gamma = F\gamma +_p G\gamma$$

Ordinary operational semantics

Values

$$V ::= \mathtt{tt} \mid \mathtt{ff} \mid 0 \mid \ldots \mid * \mid \langle V_1, V_2 \rangle \mid \lambda x : \sigma . M$$

Some redex transitions

$$r \leq r' \rightarrow \mathtt{tt} \quad (\text{if } [\![r]\!] \leq [\![r']\!]) \qquad (\lambda x : \sigma . M[x])V \rightarrow M[V]$$

$$M_1 \,\mathtt{or}\, M_2 \xrightarrow[\mathtt{or}_i]{} M_i, \text{for } i = 1, 2$$

$$r \cdot M_1 \xrightarrow[\cdot 1]{r} M_1, \text{for } i = 1$$

$$M_1 +_p M_2 \xrightarrow[(+_p)_i]{} M_i, \text{for } i = 1, 2$$

Program evaluation to effect terms

Effect values (= effect trees = interaction trees)

$$E ::= V \mid E_1 \,\mathtt{or}\, E_2 \mid r \cdot E \mid E_1 +_p E_2$$

Evaluation of programs to effect values

$$\mathrm{Op}(M) = \begin{cases} V & (\text{if } M = V) \\ \mathrm{Op}(M') & (\text{if } M \to M') \\ \mathrm{Op}(M_1) \,\mathtt{or}\, \mathrm{Op}(M_2) & (\text{if } M \xrightarrow[\mathtt{or}_i]{} M_i, \text{for } i = 1, 2) \\ r \cdot \mathrm{Op}(M_1) & (\text{if } M \xrightarrow[\cdot 1]{r} M_1) \\ \mathrm{Op}(M_1) +_p \mathrm{Op}(M_2) & (\text{if } M \xrightarrow[(+_p)_i]{} M_i, \text{for } i = 1, 2) \end{cases}$$

Effect terms E as games against nature

The positions of a game E are its subterms P:

- if P is a value V, then P is a final position, returning V;
- if $P = \mathtt{or}(P_1, P_2)$, Player can choose whether to move to position P_1 or P_2;
- if $P = r \cdot P_1 : \sigma$, Player moves to P_1, and r is added to the accumulated reward (initially 0);
- if $P = P_1 +_p P_2$, Nature picks P_1 with probability p, and P_2 with probability $1 - p$.

Strategies, their outcomes, and their expected rewards

Game strategies

$$* : V \qquad \frac{s : E_1}{1s : \mathtt{or}(E_1, E_2)} \qquad \frac{s : E_2}{2s : \mathtt{or}(E_1, E_2)}$$

$$\frac{s : E}{s : r \cdot E} \qquad \frac{s_1 : E_1 \quad s_2 : E_2}{\langle s_1, s_2 \rangle : E_1 +_p E_2}$$

Outcome distributions $\mathrm{Out}(s, E) \in \mathscr{D}_{\mathrm{f}}(\mathbb{R} \times \mathrm{Val}_\sigma)$ for $E : \sigma$:

$$\begin{array}{rcl}
\mathrm{Out}(*, V) & = & \delta_{\langle 0, V \rangle} \\
\mathrm{Out}(0s, E_1 \,\mathtt{or}\, E_2) & = & \mathrm{Out}(s, E_1) \\
\mathrm{Out}(1s, E_1 \,\mathtt{or}\, E_2) & = & \mathrm{Out}(s, E_2) \\
\mathrm{Out}(s, r \cdot E) & = & r \cdot \mathrm{Out}(s, E) \\
\mathrm{Out}(\langle s_1, s_2 \rangle, E_1 +_p E_2) & = & \mathrm{Out}(s_1, E_1) +_p \mathrm{Out}(s_2, E_2)
\end{array}$$

Expected rewards

$$\mathbf{E}\left(\sum p_i \langle r_i, V_i \rangle\right) = \sum p_i r_i$$

The optimising operational semantics

Optimal strategy for a program $M : \sigma$

$$s_{\mathrm{opt}} = \mathrm{argmax}(\lambda s : \mathrm{Op}(M). \mathbf{E}(\mathrm{Out}(s, \mathrm{Op}(M))))$$

Optimizing operational semantics $\mathrm{Op}_{\mathrm{opt}}(M) \in \mathrm{DW}(\mathrm{Val}_\sigma)$, for $M : \sigma$:

$$\mathrm{Op}_{\mathrm{opt}}(M) = \mathrm{Out}(s_{\mathrm{opt}}, \mathrm{Op}(M))$$

Two theorems

Adequacy

Theorem

For any program $M : b$ of basic type:

$$\mathscr{S}[\![M]\!](0) = \mathrm{Op}_{\mathrm{opt}}(M)$$

Full Abstraction For $M, N : \sigma$ define observational equivalence by

$$M \approx_\sigma N \iff \forall C[\,] : \sigma \to \texttt{Bool}.\mathrm{Op}_{\mathrm{opt}}(M) = \mathrm{Op}_{\mathrm{opt}}(N)$$

Theorem

For any programs $M, N : b$ of basic type:

$$M \approx_b N \iff \mathscr{S}[\![M]\!] = \mathscr{S}[\![N]\!]$$

The problem

Smart Choices example: searching an ordered array a for a value x

$$\mathbf{BinIndexSearch}(x : \mathtt{Int}, a : \mathbf{Array}[\mathtt{Int}], l : \mathtt{Int}, r : \mathtt{Int}) =$$

$$\quad \mathtt{let}\, m : \mathtt{Int} = \mathtt{choose\text{-}from}\,[l, r]\,\mathtt{using}\,[x, a[l], a[r]]$$

$$\quad\quad \mathtt{if}\ a[m] = x\ \mathtt{then}\ m$$

$$\quad\quad\quad \mathtt{else}\ \mathtt{cost}(1); //\text{pay-to-recurse}$$

$$\quad\quad\quad\quad \mathtt{if}\ a[m] < x\ \mathtt{then}\ \mathbf{BinIndexSearch}(x, a, m+1, r)$$

$$\quad\quad\quad\quad\quad \mathtt{else}\ \mathbf{BinIndexSearch}(x, a, l, m-1)$$

An approach

Combine two ideas

Handlers

$+$

Loss Continuations

(aka the selection monad)

$=$

Smart Handlers

Denotational Semantics - monads and types

Our interaction trees: used to model unhandled operations

$$T_\varepsilon(X) = \left(\sum_{\substack{\ell \in \varepsilon \\ \mathsf{op} : \mathbf{out} \xrightarrow{\ell} \mathbf{in} \\ 0 < i \leq \varepsilon(\ell)}} \mathscr{S}[\![\mathbf{out}]\!] \times T_\varepsilon(X)^{\mathscr{S}[\![\mathbf{in}]\!]} \right) + X$$

Our selection monads

$$S_\varepsilon(X) = (X \to T_\varepsilon(\mathbb{R})) \to T_\varepsilon(\mathbb{R} \times X)$$

Semantics of types

$$\mathscr{S}[\![A \to B \,!\, \varepsilon]\!] = \mathscr{S}[\![A]\!] \to S_\varepsilon(\mathscr{S}[\![B]\!])$$

Bibliography

Escardó and Oliva
Selection functions, bar recursion and backward induction
Mathematical Structures in Computer Science 20.2 (2010): 127-168.

Escardó and Oliva
The Herbrand functional interpretation of the double negation shift
The Journal of Symbolic Logic 82.2 (2017): 590-607.

Abadi and Plotkin
Smart choices and the selection monad
Logical Methods in Computer Science 19 (2023).

Plotkin and Xie
Handling the selection monad
PLDI 2025
Full version: arXiv preprint arXiv:2504.03890 (2025).