Denotational semantics for lazy initialization of letrec

black holes as exceptions rather than divergence

Keiko Nakata
Institute of Cybernetics, Tallinn

Estonian-Latvian Theory Days, Rakari, 2 October 2010

Lazy evaluation

Lazy evaluation implements

- on-demand computation — evaluate when necessary
- memorization of computation — evaluate just once

Lazy evaluation in practice

Lazy evaluation is useful in practice.

- Dynamically linked shared libraries, plugins
- Lazy class initialization in Java and F#
- Lazy file initialization in F#
- Alice ML
- OSGi, NetBeans (through bundles)
- Eclipse

Syntax

$$
\begin{array}{llll}
\textit{Expressions} & M, N & ::= & n \mid s \mid x \mid \lambda x.M \mid M\ N \mid \bullet \\
 & & \mid & \textsf{let rec } x_1 = M_1 \textsf{ and } \ldots \textsf{ and } x_n = M_n \textsf{ in } M \\
 & & \mid & M; N \mid \textsf{print } M \\
\textit{Results} & V & ::= & n \mid s \mid \lambda x.M \mid \bullet \\
\textit{Types} & \tau & ::= & \textsf{nat} \mid \textsf{string} \mid \tau_1 \to \tau_2
\end{array}
$$

Lazy evaluation for letrec

Lazy evaluation provides a useful measure to initialize (unrestricted) recursive bindings

$$\text{let rec } x_1 = M_1 \text{ and } \ldots \text{ and } x_n = M_n \text{ in } N$$

where M_i's are arbitrary expressions.

- On-demand computation to find a most successful initialization order.
  - the initialization succeeds if and only if there is a non-circular order in which the bindings can be initialized.
- Memorization for value recursion
  - initialization may perform side-effects which are produced just once

Examples

let rec $x = \text{print}\,''hi''$ and $y = \text{print}\,''bye''$ in x
$\Rightarrow$ hi

let rec $x = \text{print}\,''hi''$ and $y = \text{print}\,''bye''$ in $x; x$
$\Rightarrow$ hi

let rec $x = y$ and $y = \text{print}\,''bye''$ and $z = \text{print}\,''hi''$ in x
$\Rightarrow$ bye

Black holes as exceptions

Circular initialization, or black holes, signal a runtime exception.

$\text{let rec } x = x \text{ in } x$	$\Rightarrow$ exception
$\text{let rec } x = (\lambda y.y)\ x \text{ in } x$	$\Rightarrow$ exception
$\text{let rec } f = \lambda x.f \text{ in } f$	$\Rightarrow$ termination
$\text{let rec } f = \lambda x.f\ x \text{ in } f\ 0$	$\Rightarrow$ divergence

NB. Traditionally black holes denote divergence. (But many programming languages implement black holes as exceptions.)

Natural semantics

Judgment form

$\langle\Psi\rangle\, M \Downarrow \langle\Phi\rangle\, V$ expresses that an expression M in an initial heap Ψ evaluates to a result V with the heap being Φ.

Inference rules of the Natural semantics

Result

$$\langle \Psi \rangle\ V \Downarrow \langle \Psi \rangle\ V$$

Application

$$\frac{\langle \Psi \rangle\ M_1 \Downarrow \langle \Phi \rangle\ \lambda x.N \quad \langle \Phi[x' \mapsto M_2] \rangle\ N[x'/x] \Downarrow \langle \Psi' \rangle\ V \quad x' \text{ fresh}}{\langle \Psi \rangle\ M_1\ M_2 \Downarrow \langle \Psi' \rangle\ V}$$

Variable

$$\frac{\langle \Psi[x \mapsto \bullet] \rangle\ \Psi(x) \Downarrow \langle \Phi \rangle\ V}{\langle \Psi \rangle\ x \Downarrow \langle \Phi[x \mapsto V] \rangle\ V}$$

Letrec

$$\frac{\langle \Psi[x'_1 \mapsto M'_1, \ldots, x'_n \mapsto M'_n] \rangle\ N' \Downarrow \langle \Phi \rangle\ V \quad x'_1, \ldots, x'_n \text{ fresh}}{\langle \Psi \rangle\ \mathsf{let\ rec}\ x_1 = M_1, \ldots, x_n = M_n\ \mathsf{in}\ N \Downarrow \langle \Phi \rangle\ V}$$

where $M'_i = M_i[x'_1/x_1] \ldots [x'_n/x_n]$

Error$_\beta$

$$\frac{\langle \Psi \rangle\ M_1 \Downarrow \langle \Phi \rangle\ \bullet}{\langle \Psi \rangle\ M_1\ M_2 \Downarrow \langle \Phi \rangle\ \bullet}$$

How to detect black holes

$$\dfrac{\dfrac{\langle x' \mapsto \bullet \rangle\, \bullet \Downarrow \langle x' \mapsto \bullet \rangle\, \bullet}{\langle x' \mapsto x' \rangle\, x' \Downarrow \langle x' \mapsto \bullet \rangle\, \bullet}}{\langle\rangle\ \text{let rec}\ x = x\ \text{in}\ x \Downarrow \langle x' \mapsto \bullet \rangle\, \bullet}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x' \mapsto \bullet, y' \mapsto \bullet \rangle\, y' \Downarrow \langle x' \mapsto \bullet\ \text{and}\ y' = \bullet \rangle\, \bullet}}{\langle x' \mapsto y', y' \mapsto \bullet \rangle\, x' \Downarrow \langle x' \mapsto \bullet\ \text{and}\ y' = \bullet \rangle\, \bullet}}{\langle x' \mapsto y', y' \mapsto x' \rangle\, y' \Downarrow \langle x' \mapsto \bullet\ \text{and}\ y' = \bullet \rangle\, \bullet}}{\langle\rangle\ \text{let rec}\ x = y\ \text{and}\ y = x\ \text{in}\ y \Downarrow \langle x' \mapsto \bullet\ \text{and}\ y' = \bullet \rangle\, \bullet}$$

Lazy evaluation as a most successful initialization strategy of recursive bindings

The initialization succeeds if and only if there is a non-circular order in which the bindings can be initialized.

- The operational semantics searches such an order by on-demand computation.
- The denotational semantics searches such one by
  - initializing recursive bindings in parallel and
  - choosing the most successful result as the denotation.

  (Denotational semantics does not have evaluation order.)

Typing

$$n : \mathrm{nat} \qquad s : \mathrm{string} \qquad x : \mathrm{type}(x) \qquad \bullet : \tau$$

$$\frac{x : \tau_1 \quad M : \tau_2}{\lambda x.M : \tau_1 \to \tau_2} \qquad \frac{M : \tau_1 \to \tau_2 \quad N : \tau_1}{M\ N : \tau_2}$$

$$\frac{x_1 : \tau_1 \quad \ldots \quad x_n : \tau_n \quad M_1 : \tau_1 \quad \ldots \quad M_n : \tau_n \quad N : \tau}{\text{let rec } x_1 = M_1 \text{ and } \ldots \text{ and } x_n = M_n \text{ in } N : \tau}$$

Denotational semantics

An expression M of type τ denotes an element of $(V_\tau + \mathrm{Err}_\tau)_\perp$.

Err_τ is a singleton, whose only element is $\bullet_\tau$.

V_τ denotes proper values of type τ and is defined by

$$V_{\mathrm{nat}} = N \qquad V_{\tau_0 \to \tau_1} = [(V_{\tau_0} + \mathrm{Err}_{\tau_0})_\perp \to (V_{\tau_1} + \mathrm{Err}_{\tau_1})_\perp]$$

Notations

Denotational semantics

For $d \in (V_{\tau_0 \to \tau_1} + \mathrm{Err}_{\tau_0 \to \tau_1})_\perp$ and $d' \in (V_{\tau_0} + \mathrm{Err}_{\tau_0})_\perp$, application of d to d' is defined by

$$d(d') = \begin{cases} \perp_{\tau_1} & \text{when } d = \perp_{\tau_0 \to \tau_1} \\ \bullet_{\tau_1} & \text{when } d = \bullet_{\tau_0 \to \tau_1} \\ \varphi(d') & \text{when } d = \varphi \in V_{\tau_0 \to \tau_1} \end{cases}$$

Moreover we write $(d)^*$ to denote the strict version of d on both $\perp$ and $\bullet$, i.e.,

$$(d)^*(d') = \begin{cases} \perp_{\tau_1} & \text{when } d = \varphi \text{ and } d' = \perp_{\tau_0} \\ \bullet_{\tau_1} & \text{when } d = \varphi \text{ and } d' = \bullet_{\tau_0} \\ d(d') & \text{otherwise} \end{cases}$$

An environment, ρ, maps variables to denotations: $\rho(x) \in (V_\tau + \mathrm{Err}_\tau)_\perp$ where $x : \tau$.
The least environment, $\rho_\perp$, maps all variables to bottom elements.

Semantic function

Denotational semantics

The semantic function $[\![M : \tau]\!]_\rho$ assigns a denotation to a typing derivation $M : \tau$ under an environment ρ.

$$
\begin{array}{rcl}
[\![n : \tau]\!]_\rho & = & n \\
[\![x : \tau]\!]_\rho & = & \rho(x) \\
[\![\bullet : \tau]\!]_\rho & = & \bullet_\tau \\
[\![\lambda x.M : \tau_0 \to \tau_1]\!]_\rho & = & \lambda \nu.[\![M : \tau_1]\!]_{\rho[x \mapsto \nu]} \\
[\![M^{\tau_0 \to \tau_1}\ N^{\tau_0} : \tau_1]\!]_\rho & = & ([\![M : \tau_0 \to \tau_1]\!]_\rho)([\![N : \tau_0]\!]_\rho) \\
[\![\text{let rec } x_1 = M_1^{\tau_1}, \ldots, x_n = M_n^{\tau_n} \text{ in } N : \tau]\!]_\rho & = & [\![N : \tau]\!]_{\{\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\}_\rho^{(n)}}
\end{array}
$$

Semantic function for heaps

Denotational semantics

$$\{\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\}_\rho^{(0)} = \rho[x_1 \mapsto \bullet_{\tau_1}, \ldots, x_n \mapsto \bullet_{\tau_n}]$$

$$\{\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\}_\rho^{(m+1)} =$$
$$\mu\rho'.\rho[x_1 \mapsto [\![M_1 : \tau_1]\!]_{\rho_m} \cdot [\![M_1 : \tau_1]\!]_{\rho'}, \ldots, x_n \mapsto [\![M_n : \tau_n]\!]_{\rho_m} \cdot [\![M_n : \tau_n]\!]_{\rho'}]$$
$$\textit{where } \rho_m = \{\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\}_\rho^{(m)}$$

$d \cdot d'$ abbreviates $((\lambda y.\lambda x.x)^*(d))(d')$

Denotation of heaps

The denotation of a heap $\Psi = x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}$ under an environment ρ is computed as follows.

1. Pre-initialize to black holes.
   $\rho_0 = \rho[x_1 \mapsto \bullet_{\tau_1}, \ldots, x_n \mapsto \bullet_{\tau_n}]$.
2. Compute the denotation of $M_i : \tau_i$ under ρ_0.
3. Compute the fixed-point semantics for M_i's whose evaluation was successful under ρ_0.

   $\rho_1 = \mu\rho'.\rho[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n]$ where

$$d_i = \begin{cases} \bullet_{\tau_i} & \textit{when } [\![M_i : \tau_i]\!]_{\rho_0} = \bullet_{\tau_i} \\ [\![M_i : \tau_i]\!]_{\rho'} & \textit{otherwise} \end{cases}$$

4. Compute the denotation of $M_i : \tau_i$ under ρ_1.
5. Compute the fixed-point semantics for M_i's whose evaluation was successful under ρ_1.
6. ...

Denotation of heaps (cont.)

Generally, ρ_{m+1} is given by taking the fixed-point semantics for the recursive bindings whose initialization is successful under the environment ρ_m

$$\rho_{m+1} = \mu\rho'.\rho[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n]$$

$$\text{where } \; d_i = \begin{cases} \bullet_{\tau_i} & \textit{when } \; [\![M_i : \tau_i]\!]_{\rho_m} = \bullet_{\tau_i} \\ [\![M_i : \tau_i]\!]_{\rho'} & \textit{otherwise} \end{cases}$$

This process is iterated for n times; it converges by then:

$$\forall m,\ \{\!\!\{\Psi\}\!\!\}^{(n)}_{\rho} = \{\!\!\{\Psi\}\!\!\}^{(n+m)}_{\rho}$$

Semantic function for heaps

Denotational semantics

$$\{\!\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\!\}_\rho^{(0)} = \rho[x_1 \mapsto \bullet_{\tau_1}, \ldots, x_n \mapsto \bullet_{\tau_n}]$$

$$\{\!\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\!\}_\rho^{(m+1)} =$$
$$\mu\rho'.\rho[x_1 \mapsto [\![M_1 : \tau_1]\!]_{\rho_m} \cdot [\![M_1 : \tau_1]\!]_{\rho'}, \ldots, x_n \mapsto [\![M_n : \tau_n]\!]_{\rho_m} \cdot [\![M_n : \tau_n]\!]_{\rho'}]$$
$$\textit{where } \rho_m = \{\!\!\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\!\!\}_\rho^{(m)}$$

$d \cdot d'$ abbreviates $((\lambda y.\lambda x.x)^*(d))(d')$

Evaluations preserve the denotations of expressions.

Proposition

For any typed expression $M : \tau$*, if* $\langle\rangle\, M \Downarrow \langle\Psi\rangle\, V$*, then* $V : \tau$ *and* $[\![M : \tau]\!]_{\rho_\perp} = [\![V : \tau]\!]_{\{\!\{\Psi\}\!\}_{\rho_\perp}}$*.*

An expression evaluates to a result if and only if its denotation is non-bottom.

Proposition

For any typed expression $M : \tau$*,* $[\![M : \tau]\!]_{\rho_\perp} \neq \perp_\tau$ *iff there are* Φ *and* V *such that* $\langle\rangle\, M \Downarrow \langle\Phi\rangle\, V$*.*

Operational soundness of equational laws for letrec

β_{need}
$(\lambda x.M)\ N = \text{let rec}\ x = N\ \text{in}\ M$
lift
$(\text{let rec}\ D\ \text{in}\ M)\ N = \text{let rec}\ D\ \text{in}\ M\ N$
deref
$\text{let rec}\ x = V, D\ \text{in}\ C[x] = \text{let rec}\ x = V, D\ \text{in}\ C[V]$
deref_{env}
$\text{let rec}\ x = C[x'], x' = V, D\ \text{in}\ M = \text{let rec}\ x = C[V], x' = V, D\ \text{in}\ M$
assoc
$\text{let rec}\ x = (\text{let rec}\ D\ \text{in}\ M), D'\ \text{in}\ N = \text{let rec}\ D, x = M, D'\ \text{in}\ N$

where D abbreviates $x_1 = M_1 \ldots x_n = M_n$.

Monadic framework
for effectful unrestricted value recursion

Joint work with Masahito Hasegawa

$$\frac{\Gamma \vdash L : A \to T\ B}{\Gamma \vdash L^* : A \to T\ B} \qquad \frac{}{\Gamma \vdash \eta_A : A \to T\ A} \qquad \frac{}{\Gamma \vdash \bullet_A : T\ A}$$

$$\frac{\begin{array}{c}\Gamma, x_1 : T\ A_1, \ldots, x_n : T\ A_n \vdash L_1 : T\ A_1 \\ \cdots \\ \Gamma, x_1 : T\ A_1, \ldots, x_n : T\ A_n \vdash L_n : T\ A_n\end{array}}{\Gamma \vdash \mu(x_1^{T\ A_1}, \ldots, x_n^{T\ A_n}).(L_1, \ldots, L_n) : T\ A_1 \times \ldots T\ A_n}$$

Modeled in a target language given by a cartsian closed category equipped with a strong monad and a uniform T-fixed point operator and a family of black hole constants.

Black holes are exceptions!