A Metric Model of Lambda Calculus with Guarded Recursion

Jan Schwinghammer

Programming Systems Lab
Saarland University

joint work with Lars Birkedal and Kristian Støvring

Metric semantics of lambda calculus

Metric semantics of simply typed lambda calculus

(Complete, bounded, non-empty) ultrametric spaces and non-expansive functions, *CBUlt*, forms a CCC.

Metric semantics of call-by-name PCF

Counting clock ticks to determine the similarity of programs, ensuring that recursive definitions are contractive [Escardo, 1998].

Metric semantics of Functional Reactive Programming

Streams of events, and "causal" stream transformers as non-expansive functions [Krishnaswami & Benton, 2010].

Metric semantics of lambda calculus

Two questions:

1. Are there good syntactic criteria for determining when a term denotes a contractive function in *CBUlt*?
2. Which recursive types can be interpreted?

Nakano's modal type system [Nakano, LICS'00] gives an answer.

Talk outline

1. Reasoning about streams and stream transformers
2. Nakano's calculus
3. Metric semantics

Recursively defined streams

Integer streams

$$\texttt{data Stream = Cons of Int} \times \texttt{Stream}$$

$$\texttt{nats}_1 \texttt{ = iterate succ 0}$$
$$\texttt{nats}_2 \texttt{ = Cons (0, (map succ nats}_2\texttt{))}$$

Well-definedness

Are these "good" recursive definitions? For instance,

- `xs = Cons (1,xs)` is,
- `ys = tail (Cons (1,ys))` is not,
- `zs = f (Cons (1,zs))` may or may not be, depending on `f`.

Guarded recursion

Recursive occurrences should be guarded by constructors.

Guardedness

$\mathtt{f} : \mathtt{Stream} \to \mathtt{Stream}$ is guarded if

$$\forall\, \mathtt{xs}, \mathtt{ys}, n.\quad \lfloor \mathtt{xs} \rfloor_n = \lfloor \mathtt{ys} \rfloor_n \quad \Rightarrow \quad \lfloor \mathtt{f}\,(\mathtt{xs}) \rfloor_{n+1} = \lfloor \mathtt{f}\,(\mathtt{ys}) \rfloor_{n+1}$$

i.e., $\mathtt{f}$ is "productive."

Unique fixed points

A consequence of guardedness is the following proof principle:

[Ralf Hinze, ICFP'08 pearl:
Streams and unique fixed points]

$$\begin{array}{rl} \text{To prove} & xs = ys : \mathtt{Stream} \\ \text{find} & f : \mathtt{Stream} \to \mathtt{Stream},\ \text{guarded}, \\ \text{such that} & f(xs) = xs \ \wedge\ f(ys) = ys \end{array}$$

For instance, let $f(s) = \mathtt{Cons\ 0\ (map\ succ\ } s\mathtt{)}$.

- f is guarded
- $f(\mathtt{nats}_2) = \mathtt{nats}_2$ by definition
- $f(\mathtt{nats}_1) = \mathtt{nats}_1$ by equational reasoning

The principle yields $\mathtt{nats}_1 = \mathtt{nats}_2$.

Nakano's typed lambda calculus

[Hiroshi Nakano, LICS'00:
A modality for recursion]

A simply typed cbn lambda calculus with a unary type constructor:

$$\tau \ ::= \ Int \mid \tau \times \tau' \mid \tau \to \tau' \mid \underbrace{ty}_{\text{type names}} \mid \underbrace{\bullet\tau}_{\text{later } \tau}$$

Each type name associated with a declaration:

$$data\ ty \ = \ In_1\ of\ \tau_1 \mid \ldots \mid In_k\ of\ \tau_k$$

For instance,

$$data\ S = Cons\ of\ Int \times S$$
$$data\ U = Fold\ of\ U \to \tau$$

Typing guardedness

Intuition:

- $\bullet\tau$: values of type τ that can only be used in guarded positions
- $\bullet S \to S$ consists of guarded stream functions

Constructor applications are guarded:

$$\frac{\Gamma \vdash t : \bullet\tau_j}{\Gamma \vdash In_j(t) : ty}$$

Pattern matching adds guardedness constraint:

$$\frac{\Gamma \vdash t : ty \qquad \Gamma, x_1{:}\bullet\tau_1 \vdash t_1 : \tau \qquad \ldots \qquad \Gamma, x_k{:}\bullet\tau_k \vdash t_k : \tau}{\Gamma \vdash case\ t\ of\ In_1(x_1) \Rightarrow t_1 \mid \ldots \mid In_k(x_k) \Rightarrow t_k : \tau}$$

Typing guardedness

Fixed points are restricted to guarded recursion:

$$\frac{\Gamma \vdash t : \bullet\tau \to \tau}{\Gamma \vdash \mathit{fix}\ t : \tau}$$

Function application is generalized:

$$\frac{\Gamma \vdash t_1 : \bullet^n(\tau \to \sigma) \qquad \Gamma \vdash t_2 : \bullet^n\tau}{\Gamma \vdash t_1\, t_2 : \bullet^n\sigma}$$

Subtyping relation with axioms like:

$$\tau \leq \bullet\tau \qquad \bullet\tau \to \bullet\sigma \leq \bullet(\tau \to \sigma) \qquad \ldots$$

Recursively defined streams, revisited

- We have $\vdash \lambda xs.\, Cons\,(1, xs) : \bullet\, S \to S$
  thus $\vdash fix\ \lambda xs.\, Cons\,(1, xs) : S$
- We have $\vdash tail : S \to \bullet\, S$
  thus, only $\vdash \lambda ys.\, tail\,(Cons\,(1, ys)) : \bullet\, S \to \bullet\, S$
- However, $f{:}\,\bullet S \to S \vdash \lambda zs.\, f\,(Cons\,(1, zs)) : S$
  thus $f{:}\,\bullet S \to S \vdash fix\ \lambda zs.\, f\,(Cons\,(1, zs)) : S$

Using $data\ U = Fold\ of\ U \to \tau$ one can type the Y combinator:

- We have $\vdash Y : (\bullet\, \tau \to \tau) \to \tau$

Recursively defined streams, revisited

- We have $\vdash \lambda xs.\, Cons(1, xs) : \bullet S \to S$
  thus $\vdash fix\ \lambda xs.\, Cons(1, xs) : S$
- We have $\vdash tail : S \to \bullet S$
  thus, only $\vdash \lambda ys.\, tail(Cons(1, ys)) : \bullet S \to \bullet S$
- However, $f : \bullet S \to S \vdash \lambda zs.\, f(Cons(1, zs)) : S$
  thus $f : \bullet S \to S \vdash fix\ \lambda zs.\, f(Cons(1, zs)) : S$

Using $data\ U = Fold\ of\ U \to \tau$ one can type the Y combinator:

- We have $\vdash Y : (\bullet\tau \to \tau) \to \tau$

Recursively defined streams, revisited

- We have $\vdash \lambda xs.\, Cons(1, xs) : \bullet S \to S$
  thus $\vdash fix\ \lambda xs.\, Cons(1, xs) : S$
- We have $\vdash tail : S \to \bullet S$
  thus, only $\vdash \lambda ys.\, tail(Cons(1, ys)) : \bullet S \to \bullet S$
- However, $f : \bullet S \to S \vdash \lambda zs.\, f(Cons(1, zs)) : S$
  thus $f : \bullet S \to S \vdash fix\ \lambda zs.\, f(Cons(1, zs)) : S$

Using $data\ U = Fold\ of\ U \to \tau$ one can type the Y combinator:

- We have $\vdash Y : (\bullet \tau \to \tau) \to \tau$

Recursively defined streams, revisited

- We have $\vdash \lambda xs.\, Cons\,(1, xs) : \bullet\, S \to S$
  thus $\vdash fix\ \lambda xs.\, Cons\,(1, xs) : S$
- We have $\vdash tail : S \to \bullet\, S$
  thus, only $\vdash \lambda ys.\, tail\,(Cons\,(1, ys)) : \bullet\, S \to \bullet\, S$
- However, $f : \bullet S \to S \vdash \lambda zs.\, f\,(Cons\,(1, zs)) : S$
  thus $f : \bullet S \to S \vdash fix\ \lambda zs.\, f\,(Cons\,(1, zs)) : S$

Using $data\ U = Fold\ of\ U \to \tau$ one can type the Y combinator:

- We have $\vdash Y : (\bullet\,\tau \to \tau) \to \tau$

Unique fixed point principle, revisited

$$
\begin{array}{rl}
\text{To prove} & s \simeq t : \tau \\
\text{find} & f : \bullet\tau \to \tau \\
\text{such that} & f(s) \overset{*}{\leftrightarrow} s \ \wedge \ f(t) \overset{*}{\leftrightarrow} t
\end{array}
$$

- $s \simeq t : \tau$ denotes contextual equivalence:

$$\forall C[\cdot] : Int.\ C[s] \overset{*}{\to} \underline{n} \ \Leftrightarrow \ C[t] \overset{*}{\to} \underline{n}$$

- guardedness is expressed abstractly using $\bullet\tau \to \tau$.
- τ need not be S,

Ultrametric spaces

Complete ultrametric spaces *CBUlt*

Complete metric spaces (X, d) satisfying ultrametric inequality:

$$d(x, y) \;\leq\; \max\{d(x, z), d(z, y)\}$$

Products

Cartesian product $X_1 \times X_2$ equipped with max distance:

$$d((x_1, x_2), (y_1, y_2)) \;=\; \max\{d_1(x_1, y_1), d_2(x_2, y_2)\}$$

Exponentials

Non-expansive functions $X_1 \to_{\mathrm{ne}} X_2$ equipped with sup distance:

$$d(f, g) \;=\; \sup\{d_2(f\,x, g\,x) \mid x \in X_1\}$$

Semantics of $\bullet\tau$

Key idea of the semantics:
Later modality as scaling functor $\frac{1}{2}\cdot(-) : \mathit{CBUlt} \longrightarrow \mathit{CBUlt}$

$$\frac{1}{2}\cdot(X,d) \stackrel{\text{def}}{=} (X,d') \quad \text{where } d'(x,y) = 1/2\cdot d(x,y)$$

Guardedness as contractiveness
$\bullet\tau \to \sigma$ denotes contractive functions from τ to σ:

- Assume $d(x,y) = c$ in $[\![\tau]\!]$.
- Then $d(x,y) = c/2$ in $[\![\bullet\tau]\!]$.
- Thus, $d(f\,x, f\,y) \leq c/2$ in $[\![\sigma]\!]$.

In particular, interpretation $[\![\mathit{fix}\ t]\!]$ qua Banach fixed point theorem.

Semantics of recursive types

By separating positive and negative occurrences of *ty* in τ define

$$F_\tau : CBUlt^{\mathrm{op}} \times CBUlt \longrightarrow CBUlt$$

For each *data* $ty = In_1 \; of \; \tau_1 \mid \ldots \mid In_k \; of \; \tau_k$ define

$$F(X^-, X^+) \stackrel{\mathrm{def}}{=} \tfrac{1}{2} \cdot F_{\tau_1}(X^-, X^+) + \ldots + \tfrac{1}{2} \cdot F_{\tau_k}(X^-, X^+)$$

For instance,

- for *data* $S = Cons \; of \; Int \times S$:
  $F(X^-, X^+) = \frac{1}{2} \cdot (\mathbb{Z} \times X^+)$
- for *data* $U = Fold \; of \; U \to \tau$:
  $F(X^-, X^+) = \frac{1}{2} \cdot (X^- \to_{\mathrm{ne}} F_\tau(X^-, X^+))$

Semantics of recursive types, cnt'd

Then F is locally contractive:

$$d(F(f_1, g_1), F(f_2, g_2)) \leq 1/2 \cdot \max\{d(f_1, g_1), d(f_2, g_2)\}$$

Theorem (America & Rutten, 1989)

Let $F : CBUlt^{op} \times CBUlt \longrightarrow CBUlt$ be locally contractive. Then there exists a unique (X, d) such that $F(X, X) \cong X$.

For instance, for *data* $S = \textit{Cons of Int} \times S$ have $X \cong \frac{1}{2} \cdot (\mathbb{Z} \times X)$:

- $d(s, s') \leq 2^{-n}$ iff $\lfloor s \rfloor_{n-1} = \lfloor s' \rfloor_{n-1}$

Adequacy

Theorem

The model is sound and adequate for the operational semantics:

1. $s \to t \;\Rightarrow\; [\![s]\!] = [\![t]\!]$
2. $[\![s]\!] = [\![t]\!] \;\Rightarrow\; s \simeq t$

Proof sketch.

1. By an easy induction.
2. By a logical relation between semantics and syntax.

Kripke logical relation

Relations $R^k_\tau \subseteq [\![\tau]\!] \times \mathrm{Tm}(\tau)$, given by:]

$$
\begin{aligned}
n\ R^k_{Int}\ t &\Leftrightarrow t \xrightarrow{*} \underline{n} \\
f\ R^k_{\tau_1 \to \tau_2}\ t &\Leftrightarrow \forall j \leq k, a_1, t_1.\ a_1\ R^j_{\tau_1}\ t_1 \Rightarrow f a_1\ R^j_{\tau_2}\ t\ t_1 \\
a\ R^k_{\bullet\tau}\ t &\Leftrightarrow k > 0 \Rightarrow a\ R^{k-1}_\tau\ t \\
a\ R^k_{ty}\ t &\Leftrightarrow a = (\iota \circ \mathrm{in}_j)(a') \wedge t \xrightarrow{*} In_j(t') \wedge a'\ R^k_{\bullet\tau_j}\ t'
\end{aligned}
$$

Theorem (fundamental property)

$\forall t \in Tm(\tau)\ \forall k.\ [\![t]\!]\ R^k_\tau\ t.$

Proving the unique fixed point principle

- Suppose $f : \bullet\tau \to \tau$.
  Then $[\![f]\!]$ is a contractive function on $[\![\tau]\!]$
- Suppose $f(s) \overset{*}{\leftrightarrow} s$ and $f(t) \overset{*}{\leftrightarrow} t$.
  By soundness, $[\![s]\!]$ and $[\![t]\!]$ are both fixed points of $[\![f]\!]$.
  By the contractiveness of $[\![f]\!]$ and Banach theorem, $[\![s]\!] = [\![t]\!]$.
- By adequacy, $s \simeq t$.

Summary

Nakano's system is a calculus of total functions.

Ultrametric spaces form a model of Nakano's lambda calculus:

- $\bullet\tau$ is interpreted as a scaling functor on *CBUlt*,
- $\bullet\tau \to \sigma$ denotes the contractive functions on τ, and
- syntactic notion of guardedness becomes contractiveness.

Banach's fixed point theorem explains

- the typing of the recursion operators, as well as
- the unique fixed point principle.

An open question is full abstraction of the model.

Outlook

F. Pottier proposed a variant of System F with recursive kinds:

- used as target language for translation of ML-like languages,
- Nakano's calculus on the *kind* level, and
- unique fixed point principle used to prove *type equivalences*.

It should be possible to give a model by indexing over *CBUlt*.

Recursively defined properties on recursive types:

- step-indexed logics [Dreyer et al., LICS'09]
- semantics of expressive type systems tomorrow

Explore semantics for $\bullet$ on type *and* logical level.

Thank you.