Parametric Polymorphism, Strictness and Recursion

Alex Simpson

LFCS, School of Informatics
University of Edinburgh, UK

(Joint work with Pino Rosolini)
Girard’s decomposition:

\[ \sigma \to \tau = !\sigma \to \tau \]

Second-order linear type theory . . .

\[ \sigma \to \tau \quad \text{— Linear function space} \]
\[ !\sigma \quad \text{— Exponentials ("thunks")} \]
\[ \forall \alpha. \sigma \quad \text{— Polymorphic types} \]

. . . plus recursion . . .

\[ Y : \forall \alpha. !(!\alpha \to \alpha) \to \alpha \]
\[ \forall \alpha. (\alpha \to \alpha) \to \alpha \]
• Simple language based on very few primitives

• Nevertheless, astonishingly rich range of definable datatypes (partly following Freyd)

• Relational parametricity (Reynolds) is crucial for establishing correctness properties

• Relationally-parametric model based on admissible partial-equivalence relations over a domain-theoretic model of the untyped $\lambda$-calculus.

[Plotkin 1993]
...considered as a programming language (Lily)

- Simple operational semantics
- Establish operational properties of language (operational extensionality, correctness of datatype encodings) ...
- ...using advanced operational techniques (Howe’s method, Mason/Talcott “ciu” theorem, syntactic form of relational parametricity using $(\cdot)^{TT}$-closed relations)

[Bierman, Pitts & Russo 2000]
This talk

1. Extension of Lily, based on interpreting $\to$ as “strict” rather than “linear” function space
   (Motivations: type-theoretically, operationally and denotationally natural; possible target language for strictness analysis?)

2. Operational theorems about this extension (cf. [BPR 2000])

3. Simple (!?) denotational proofs of the operational theorems
DILL [Barber & Plotkin]

Main judgment form:

\[
\Gamma \mid \Delta \vdash t : \sigma
\]

\(t\) has type \(\sigma\) in context \(\Gamma, \Delta\) and each variable in \(\Delta\) is “used” exactly once by \(t\).

The notion of “usage” is subtle.

\[
\Gamma \mid \Delta \vdash t : \sigma \\
\Gamma, \Delta \mid - \vdash !t : !\sigma
\]

Variables “used” by \(t\) are “mentioned” but not “used” by \(!t\).
Adaptation for strictness

\( \Gamma | \Delta \vdash t : \sigma \) now means: \( t \) has type \( \sigma \) in context \( \Gamma, \Delta \) and each variable in \( \Delta \) is “used” (but possibly more than once) by \( t \)

Implemented by (non-standard) “contraction” rule

\[
\frac{\Gamma, x : \sigma | \Delta, y : \sigma \vdash t : \sigma}{\Gamma | \Delta, y : \sigma \vdash t[y/x] : \sigma}
\]

cf. [Momigliano & Pfenning 2003]

(In conventional linear logic terms, this amounts to having “diagonal” maps \( \sigma \rightarrow (\!\sigma) \otimes \sigma \).)
Syntax and operational semantics: $\sigma \rightarrow^* \tau$

\[
\begin{array}{c}
\lambda x: \sigma. \ t \Downarrow \ \lambda x: \sigma. \ t \\
\hline
\lambda x: \sigma. \ s \Downarrow^v \ \lambda x: \sigma. \ s' \quad t \Downarrow^v \ v' \quad s'[v'/x] \Downarrow^v \ v \quad s(t) \Downarrow^v \ v \\
\hline
s \Downarrow^n \ \lambda x: \sigma. \ s' \quad s'[t/x] \Downarrow^n \ v \quad s(t) \Downarrow^n \ v
\end{array}
\]
Syntax and operational semantics: $!\sigma$

\[
\begin{align*}
\text{let } !x = s \text{ in } t & \Downarrow v \\
!t & \Downarrow !t \\
\end{align*}
\]

Syntax and operational semantics: $\forall\alpha. \sigma$

\[
\begin{align*}
\text{let } !x = s \text{ in } t & \Downarrow v \\
\Lambda\alpha. t & \Downarrow \Lambda\alpha. t \\
\end{align*}
\]

\[
\begin{align*}
t & \Downarrow \Lambda\alpha. t' \\
t'[\sigma/\alpha] & \Downarrow v \\
t(\sigma) & \Downarrow v
\end{align*}
\]

MFPS XX, May 2004
Syntax and operational semantics: recursion

\[
\frac{t[\text{rec } x : \sigma. \ t / x] \Downarrow v}{\text{rec } x : \sigma. \ t \Downarrow v}
\]

Under strictness interpretation of judgements, the natural typing rule for \( \text{rec } x : \sigma. \ t \) is:

\[
\frac{\Gamma, x : \sigma | \Delta \vdash t : \sigma}{\Gamma | \Delta \vdash \text{rec } x : \sigma. \ t : \sigma}
\]

This is equivalent to having a polymorphic fixed-point combinator

\[
Y : \forall \alpha. (\forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha) \rightarrow \alpha
\]

\[
\forall \alpha. (\alpha \rightarrow \alpha) \rightarrow \alpha
\]
Observational preorder

Crucial idea [BPR 2000]: observe termination at !-types only.

Equivalent to extending language with primitive “ground” type(s) (e.g. unit, booleans, natural numbers) and observing termination at ground type(s) only.

For closed $\sigma$, an observational $\sigma$-context is a term of the form

$x : \sigma \mid - \vdash C[x] : ! \tau$

$t \sqsubseteq_{\text{obs}} t' : \sigma \iff C[t] \Downarrow \text{ implies } C[t'] \Downarrow$, for all obs. $\sigma$-ctxts $C[\cdot]$
Theorem (Strictness) For $s : ! \tau$, $s \Downarrow^v$ iff $s \Downarrow^n$.

Thus the definition of $\subseteq_{\text{obs}}$ is insensitive to the difference between $\Downarrow^v$ and $\Downarrow^n$.

The restriction to terms of $!$-type is crucial. E.g. consider

$$M = (\lambda f : \text{unit} \to \text{unit}. \lambda x : \text{unit}. f(x))(\Omega_{\text{unit} \to \text{unit}})$$

Then $M \Downarrow^n$, but $M \Uparrow^v$.

The analogous result for Lily is proved in [BPR 2000] using operational techniques.

We give a denotational proof.
Applicative simulation

The relation $\sqsubseteq_{\text{app}}$ is the largest relation between closed terms of identical closed type satisfying:

1. if $s \sqsubseteq_{\text{app}} s' : \sigma \rightarrow \tau$ then, for all $t : \sigma$, it holds that $s(t) \sqsubseteq_{\text{app}} s'(t) : \tau$

2. if $s \sqsubseteq_{\text{app}} s' : !\sigma$ then $s \Downarrow !t$ implies $s' \Downarrow !t'$ where $t \sqsubseteq_{\text{app}} t' : \sigma$

3. if $s \sqsubseteq_{\text{app}} s' : \forall \alpha. \sigma$ then, for all closed $\tau$, it holds that $s(\tau) \sqsubseteq_{\text{app}} s'(\tau) : \sigma[\tau/\alpha]$

Using strictness theorem, can show definition robust. (Can replace “all $t : \sigma$” with “all $v : \sigma$” in (1). Can use either $\Downarrow^v$ or $\Downarrow^n$ in (2).)
Theorem (Operational extensionality) \( t \sqsubseteq_{\text{obs}} t' \iff t \sqsubseteq_{\text{app}} t' \)

The proof of the \( \Rightarrow \) direction is straightforward. (One shows that \( \sqsubseteq_{\text{obs}} \) is a relation satisfying the closure conditions of \( \sqsubseteq_{\text{app}} \).)

The proof of the \( \Leftarrow \) implication is non-trivial.

The analogous result for Lily is proved in [BPR 2000] using operational techniques.

We give a denotational proof.
Plotkin’s Datatype Encodings

Function space \( \sigma \rightarrow \tau = !\sigma \rightarrow \tau \)

Smash product \( \sigma \otimes \tau = \forall \alpha. (\sigma \rightarrow \tau \rightarrow \alpha) \rightarrow \alpha \)

Coproduct \( \sigma \oplus \tau = \forall \alpha. !(\sigma \rightarrow \alpha) \rightarrow !(\tau \rightarrow \alpha) \rightarrow \alpha \)

Product \( \sigma \times \tau = \forall \alpha. ((\sigma \rightarrow \alpha) \oplus (\tau \rightarrow \alpha)) \rightarrow \alpha \)

Existential \( \exists \alpha. \sigma = \forall \beta. (\forall \alpha. (\sigma \rightarrow \beta)) \rightarrow \beta \)

(Co)inductive \( \mu \alpha. \sigma = \forall \alpha. !(\sigma \rightarrow \alpha) \rightarrow \alpha \quad (\alpha \text{ +ve in } \sigma) \)

\[ \equiv \exists \alpha. !(\alpha \rightarrow \sigma) \otimes \alpha \]

Recursive \( \mu \alpha. \sigma(\alpha, \alpha) = \mu \beta. \sigma(\mu \alpha. \sigma(\beta, \alpha), \beta) \quad (\alpha, \beta \text{ resp. } -, +\text{ve in } \sigma(\alpha, \beta)) \)

Unit type \( \text{unit} = \forall \alpha. \alpha \rightarrow \alpha \)
Theorem (Datatype correctness) The datatype encodings are operationally correct.

Main ingredient needed to prove the theorem is an account of relational parametricity

Relational parametricity (for Lily) addressed in [BPR 2000] using $(\cdot)^{TT}$-closed binary relations between terms

We give simple denotational account of relational parametricity
Outline of denotational approach

Closed type $\sigma$ interpreted as “domain” $[\sigma]$

Closed term $t : \sigma$ interpreted as $[t] \in [\sigma]$

Proposition (Computational adequacy)  For $t : !\sigma$,

1. $t \Downarrow^\nu$ iff $[t] \Downarrow$

2. $t \Downarrow^n$ iff $[t] \Downarrow$

Strictness theorem is immediate consequence.

Obtain operational extensionality as by-product of proof of computational adequacy (following Jung, Pitts)
Outline of relational parametricity

Open type $\sigma(\Theta)$ interpreted as:

- A “domain” $[[\sigma]]_\gamma$
  where $\gamma$ is an environment mapping $\Theta$ to domains

- And as an “admissible relation”

$$[[\sigma]]_{\gamma_R} \in R([[\sigma]]_{\gamma_1}, [[\sigma]]_{\gamma_2})$$

where $\gamma_R$ maps each $\alpha \in \Theta$ to $\gamma_R(\alpha) \in R(\gamma_1(\alpha), \gamma_2(\alpha))$

Proposition (Identity extension) For closed $\sigma$ and $d, d' \in [[\sigma]]$,

$$[[\sigma]]^R(d, d') \text{ iff } d = d'$$
Possible instantiations of denotational approach

- Ordinary domain-theoretic models of polymorphic lambda-calculus
  However, no known model of relational parametricity

- PER models over domain-theoretic model of untyped $\lambda$-calculus
  [Plotkin 1993]
  However, PER models are fiddly to work with

- Domains as sets (“synthetic domain theory”)
Synthetic domain theory (Scott, Rosolini, Hyland, …)

Consider domains as (special) sets, and morphisms of domains as arbitrary functions

Consistent with intuitionistic set theory

Allows (relatively complex) domain-theoretic constructions to be replaced by (relatively simple) set-theoretic constructions

Some overhead in getting things to work. In particular, pointedness, lifting and strictness are subtle

However, polymorphism and relational parametricity can be handled very straightforwardly
Overview of our approach to SDT

Working in intuitionistic set theory (IZF), we implement the equation

\[
domain = \text{pointed predomain}
\]

where:

- Predomains are special sets.
- Pointed sets are algebras for a simple equational theory

Need axioms to ensure that predomains and domains support desired domain-theoretic constructions
Axioms given in this morning’s talk: “Axioms for Synthetic Domain Theory”. Here recall only main points.

There is a dominance $\Sigma \subseteq \mathcal{P}(\emptyset)$.

For any set $X$, define its lifting by:

$$\mathcal{L}X = \{ e \in \mathcal{P}(X) \mid (\forall x, x' \in e. x = x') \land ((\exists x \in e) \in \Sigma) \}$$

Singleton $\{ \cdot \}: X \rightarrow \mathcal{L}X$ exhibits $\mathcal{L}X$ as the free pointed set over $X$, where a pointed set is an equational algebra $(A, \{r_p : A^p \rightarrow A\}_{p \in \Sigma})$.

A function $f: A \rightarrow B$ between pointed sets $(A, \{r_p\}_p)$ and $(B, \{s_p\}_p)$ is strict if it is an algebra homomorphism.
There is a class of special sets, called predomains, closed under isomorphism, (set-indexed) products, equalizers, finite coproducts, lifting and containing \( \mathbb{N} \).

A domain is a pointed set \((A, \{r_p\}_p)\) where \( A \) is a predomain.

There is a set \( D \) of domains such that, for every domain \( D \), there exists \( E \in D \) with \( D \cong E \).

For every domain \( D \) there is a fixed-point operator \( \text{fix}_D : (D \to D) \to D \) satisfying uniformity.
Interpretation of types as domains

If $D, E$ are domains then an admissible relation between $D$ and $E$ is a subset $R \subseteq D \times E$ such that $R$ is a subdomain of $D \times E$.

We write $\mathcal{R}(D, E)$ for the set of all admissible relations.

\[
[\alpha]_\gamma = \gamma(\alpha)
\]

\[
[\sigma \rightarrow \tau]_\gamma = [\sigma]_\gamma \rightarrow [\tau]_\gamma
\]

\[
[! \sigma]_\gamma = L[\sigma]_\gamma
\]

\[
[\forall \alpha. \sigma]_\gamma = \{ \pi \in \prod_{D \in \mathcal{D}} [\sigma]_{\gamma[D/\alpha]} \mid \forall D_1, D_2 \in \mathcal{D}, \forall R \in \mathcal{R}(D_1, D_2), \]
\[
[\sigma]_{\Delta_\gamma[R/\alpha]}(\pi_{D_1}, \pi_{D_2}) \}
\]

where $\Delta_\gamma$ maps each $\alpha \in \Theta$ to the identity relation on $\gamma(\alpha)$.
Relational interpretation of types

\[
\lbrack \alpha \rbrack^R_\gamma (d_1, d_2) \Leftrightarrow \gamma^R (\alpha)(d_1, d_2)
\]

\[
\lbrack \sigma \rightarrow \tau \rbrack^R_\gamma (f_1, f_2) \Leftrightarrow \forall d_1 \in \lbrack \sigma \rbrack_\gamma_1, d_2 \in \lbrack \sigma \rbrack_\gamma_2.
\]

\[
\lbrack \sigma \rbrack^R_\gamma (d_1, d_2) \rightarrow \lbrack \tau \rbrack^R_\gamma (f_1(d_1), f_2(d_2))
\]

\[
\lbrack ! \sigma \rbrack^R_\gamma (e_1, e_2) \Leftrightarrow (\forall d_1 \in \lbrack \sigma \rbrack_\gamma_1 \cdot e_1 = \{d_1\} \rightarrow \exists d_2 \in \lbrack e_2 \rbrack_\gamma_2 \cdot \lbrack \sigma \rbrack^R_\gamma (d_1, d_2))
\]

\[
\wedge (\forall d_2 \in \lbrack \sigma \rbrack_\gamma_2 \cdot e_2 = \{d_2\} \rightarrow \exists d_1 \in \lbrack e_1 \rbrack_\gamma_1 \cdot \lbrack \sigma \rbrack^R_\gamma (d_1, d_2))
\]

\[
\lbrack \forall \alpha. \sigma \rbrack^R_\gamma (\pi_1, \pi_2) \Leftrightarrow \forall D_1, D_2 \in \mathbf{D}, R \in \mathcal{R}(D_1, D_2).
\]

\[
\lbrack \sigma \rbrack^R_\gamma [R/\alpha]((\pi_1)_{D_1}, (\pi_2)_{D_2}).
\]

Identity extension holds!
Interpretation of terms

A term $\Gamma \mid \Delta \vdash \Theta \ t : \tau$ is interpreted as a function

$$\llbracket t \rrbracket_\gamma : \left( \prod_{x:\sigma_x \in \Gamma} \llbracket \sigma_x \rrbracket_\gamma \right) \times \left( \prod_{y:\sigma_y \in \Delta} \llbracket \sigma_y \rrbracket_\gamma \right) \to \llbracket \tau \rrbracket_\gamma$$

This function is strict in each argument $\llbracket \sigma_y \rrbracket_\gamma$ for $y : \sigma_y \in \Delta$

The interpretation preserves relations:

if $\llbracket \sigma_z \rrbracket_{\gamma \mathcal{R}} (d_z, e_z)$ for all $z : \sigma_z \in \Gamma, \Delta$ then $\llbracket \tau \rrbracket_{\gamma \mathcal{R}} (\llbracket t \rrbracket_{\gamma_1} (\vec{d}), \llbracket t \rrbracket_{\gamma_2} (\vec{e}))$
Computational Adequacy

For \( e \in LD \) define:

\[
e \downarrow \iff \exists d \in D. e = \{d\}
\]

We prove statement (2) of computational adequacy:

for all \( t : ! \sigma \), \( t \downarrow^n \) iff \( \llbracket t \rrbracket \downarrow \)

\( \Rightarrow \) implication by straightforward induction on \( \downarrow^n \) relation

\( \Leftarrow \) implication by defining “approximation relations” between syntax and semantics
Approximation relations

For a domain $D$ and closed type $\sigma$, an approximation relation between $D$ and $\sigma$ is a relation $\preceq$ between elements of $D$ and closed terms of type $\sigma$ satisfying:

1. $\{d \mid d \preceq t\}$ is a subdomain of $D$
2. If $d \downarrow$ implies $d \preceq t$ then $d \preceq t$
   (where $d \downarrow \iff$ there exists $t : D \rightarrow \Sigma$ such that $t(d)$)
3. If $d \preceq t$ and $t \equiv_{\text{app}} t'$ then $d \preceq t'$

Define $\mathcal{A}(D, \sigma)$ to be set of all approximation relations between $D$ and $\sigma$. 

MFPS XX, May 2004
Given $\zeta$ mapping each $\alpha \in \Theta$ to an approximation relation

$$\lessapprox_\alpha \in \mathcal{A}(\gamma(\alpha), \tau_\alpha),$$

interpret open type $\sigma(\Theta)$ as approximation relation

$$\lessapprox_\zeta \in \mathcal{A}([\sigma]_\gamma, \sigma[\vec{\tau}/\Theta])$$

defined by:

$$d \lessapprox_\zeta^\alpha t \iff d \lessapprox_\alpha t$$

$$f \lessapprox_\zeta^{\sigma_1^{-o}\sigma_2} s \iff \forall d \in [\sigma_1]_\gamma, \forall t : \sigma_1[\vec{\tau}/\Theta], d \lessapprox_\zeta^{\sigma_1} t \rightarrow f(d) \lessapprox_\zeta^{\sigma_2} s(t)$$

$$e \lessapprox_\zeta^{!\sigma} t \iff \forall d \in [\sigma]_\gamma, e = \{d\} \rightarrow (\exists t' : \sigma. t \Downarrow^n t' \text{ and } d \lessapprox_\zeta^{\sigma} t')$$

$$\pi \lessapprox_\zeta^{\forall\alpha.\sigma} t \iff \forall D \in D, \forall \tau, \forall \lessapprox \in \mathcal{A}(D, \tau). \pi_D \lessapprox_\zeta^{\sigma[\lessapprox/\alpha]} t(\tau).$$
Lemma  For closed types $\sigma, \tau$, term $x : \sigma \vdash t : \tau$, element $d \in \llbracket \sigma \rrbracket$ and closed $s : \sigma$, it holds that $d \preceq^\sigma s$ implies $\llbracket t \rrbracket_{(x := d)} \preceq^\tau t[s/x]$.

Corollary 1  For closed $t : \tau$, it holds that $\llbracket t \rrbracket \preceq^\tau t$.

Corollary 2 (Computational adequacy)  For closed $t : !\tau$, it holds that $t \Downarrow$ implies $t \Downarrow^\eta$.

Corollary 3  For closed $t, t' : \tau$, it holds that $t \sqsubseteq_{\text{app}} t'$ iff $\llbracket t \rrbracket \preceq^\tau t'$.

Corollary 4 (Operational extensionality)  For closed $t, t' : \tau$, it holds that $t \sqsubseteq_{\text{app}} t'$ iff $t \sqsubseteq_{\text{obs}} t'$.
Summary

Working in intuitionistic set theory, have constructed relationally parametric model of language.

Have used model to establish operational properties (strictness theorem, operational extensionality)

Can exploit relational parametricity to infer datatype correctness

Synthetic setting supports simple set-theoretic style of proof

Consistency of axioms assured by existence of (many) realizability models of IZF in which axioms hold.

Such realizability models are all $\Pi^0_2$-absolute, so operational consequences derived from our axioms are indeed true in reality.
Topological domains

In one realizability model (related to Dana Scott’s “equilogical spaces”) can satisfy axioms with the resulting categories of predomains and domains having the following simple external descriptions.

Predomains are topological quotients of countably-based topological spaces that are monotone convergence spaces. (A mon. conv. space is a topological space satisfying: it is a dcpo under its specialization order; and every open set is Scott-open in the specialization order.)

Domains are predomains with a least element (in the specialization order)

(Strict) maps are continuous functions (that preserve least element)