Implementing Parametricity

Jean-Philippe Bernardy

CHALMERS | GÖTEBORG UNIVERSITY

Parametricity Workshop
May 2nd, 2012

Reynolds' Parametricity

In the Polymorphic Lambda Calculus/System F:

1. Every type A can be interpreted as a relation $[\![A]\!]$.
2. Abstraction: evaluating a term in related environments yields related values.
3. Corollary: The interpretation of a term satisfies the interpretation of its type.
4. (Identity extension: The interpretation of types preserves identities)

Wadler: Free theorems

Idea: if you tell me

$$x : A$$

...then I know...

$$(x, x) \in [\![A]\!]$$

...and $[\![A]\!]$ is useful if x is polymorphic.

Example: filter

See Filter.agda

Goal

Goal: implementing parametricity in a proof assistant.

Goal

Goal: implementing parametricity in a proof assistant.
Why?

- Can already represent programs and propositions
- Seems a natural setting
  - parametricity has many applications: I want it supported in my proof assistant!
  - meta-programming (from less-typed to more-typed)
- Encode your type-system in Agda; get “specific” parametricity for free.

Goal

Goal: implementing parametricity in a proof assistant.
Why?

- Can already represent programs and propositions
- Seems a natural setting
  - parametricity has many applications: I want it supported in my proof assistant!
  - meta-programming (from less-typed to more-typed)
- Encode your type-system in Agda; get "specific" parametricity for free.

To do:

- Extend parametricity to full dependent types (CC)
- Clarify the logical status of parametricity (does $[\![A]\!]$ compute?)
- Pragmatics: make a convenient tool

Extensions

- System F (Reynolds, 1983)
- Hints of type constructors, classes (Wadler, FPCA 1989)
- Functors (Folklore; Fegaras and Sheard credit Paterson, POPL 1996)
- Seq, bottoms (Johann and Voigtländer, POPL 2004)
- Constructor classes (Voigtländer, ICFP 2009)
- Fω (Vytiniotis and Weirich, JFP 2010)
- PTSs+Inductive constructions (Bernardy, Jansson and Paterson, ICFP 2010) (Extended version in JFP: `http://publications.lib.chalmers.se/cpl/record/index.xsql?pubid=135303`)

Tool: Free Theorem Generator

Tool of choice: Voigtländer-Böhme's free theorem generator
`http://www-ps.iai.uni-bonn.de/cgi-bin/free-theorems-webui.cgi`

- specialization of relations to functions
- sublanguages of Haskell
- ...

Logical status: Types to Propositions; Programs to Proofs

x	$[\![x]\!]$
Programming Language	Logic
Types	Propositions
Programs	Proofs
STLC	LCF
Poly. LC	2nd order logic
Fω	Higher-order logic
PTS	PTS
CCω	CCω
CiC	CiC
MLTT	MLTT

Note that the logic contains the same constructions as the programming language + quantification over "programs".

Extension: Parametricity for PTSs

$$
\begin{array}{lcll}
[\![\Gamma, \quad x : A]\!] & = & [\![\Gamma]\!], \quad \overline{x : A}, \quad \dot{x} : \overline{x} \in [\![A]\!] & \\
\hline
\overline{C} \in [\![s]\!] & = & \overline{C} \rightarrow s & \\
\overline{C} \in [\![\forall x : A. \quad B]\!] & = & \forall \overline{x : A}. \quad \forall \dot{x} : \overline{x} \in [\![A]\!]. \quad \overline{z\ x} \in [\![B]\!] & \\
\overline{C} \in [\![T]\!] & = & [\![T]\!]\ \overline{C} & \underline{\text{otherwise}} \\
\hline
[\![x]\!] & = & \dot{x} & \\
[\![\lambda x : A. \quad B]\!] & = & \lambda \overline{x : A}. \quad \lambda \dot{x} : \overline{x} \in [\![A]\!]. \quad [\![B]\!] & \\
[\![A\ B]\!] & = & [\![A]\!]\ \overline{B}\ [\![B]\!] & \\
[\![T]\!] & = & \lambda \overline{z : T}. \quad \overline{z} \in [\![T]\!] & \underline{\text{otherwise}}
\end{array}
$$

Theorem (abstraction)

If $\Gamma \vdash A : B : s$, *then* $[\![\Gamma]\!] \vdash [\![A]\!] : \overline{A} \in [\![B]\!] : s$

Tool: Lightweight Free Theorems (in Agda)

`http://wiki.portal.chalmers.se/agda/agda.php?n=Libraries.LightweightFreeTheorems`

Demo

Tool: Tactics

"Parametricity in an Impredicative Sort" M. Lasson and C. Keller
`http://perso.ens-lyon.fr/marc.lasson/coqparam-draft.pdf`
Remark:

- Difficult to work with "raw" free theorems.
- Can we instantiate automatically some relations appropriately dependent on the goal?

Logical Status: open terms

What *I* really want: in Agda, use $[\![A]\!]$ on any term A.
Required:

$$\frac{\Gamma \vdash A : B}{\Gamma \vdash [\![A]\!] : A \in [\![B]\!]}$$

Ie. don't assume that every free variable in the context is parametric.

Logical Status: open terms

What *I* really want: in Agda, use $[\![A]\!]$ on any term A.
Required:

$$\frac{\Gamma \vdash A : B}{\Gamma \vdash [\![A]\!] : A \in [\![B]\!]}$$

Ie. don't assume that every free variable in the context is parametric. Actually; that's ok. Just block on free variables; wait for a concrete term to appear.
Amendment to $[\![\cdot]\!]$:

- Carry a local context of associations between variables and their relational interpretations.
- $[\![A]\!]_{x \mapsto \dot{x}, y \mapsto \dot{y}, \ldots}$
- $[\![\lambda x : A.\quad B]\!]_\rho = \lambda \overline{x : A}.\quad \lambda \dot{x} : \overline{x} \in [\![A]\!].\quad [\![B]\!]_{\rho, x \mapsto \dot{x}}$

Logical status: nesting

Why?

- binary relations obtained from unary ones.
- if $[\![\cdot]\!]$ is used in programs, you want this.

Challenge:

- evaluate $[\![[\![x]\!]]\!]_{x \mapsto \dot{x}}$
- naive approach ($[\![[\![x]\!]_{x \mapsto \dot{x}}]\!]$) does not preserve subject reduction: confuses $[\![x]\!]$ and $\dot{x}$.

Possible approaches:

- Preventing nesting: use the system of sorts presented by Lasson and Keller.
- points to higher-dimensional type-theory. (Bernardy and Moulin, LICS 2012, `http://publications.lib.chalmers.se/cpl/record/index.xsql?pubid=153094`)

Conclusions

- Extensions: Agda is ripe with exotic extensions (Co-induction (easy); Abel's irrelevance, other features?)
- Logical status: Done!
- Toolification: Room for improvement

The long term: study relationship with extensionality, induction, ...

Logical Status: Remark

Parametricity is Anti-classical