The Recursive Polarized Dual Calculus

Aaron Stump
Computer Science
The University of Iowa
Iowa City, Iowa, USA

Golden Age of Intuitionistic Type Theory

- All-time high interest in tools like Coq, Agda
- Many exciting applications:
  - Software: Quark verified web-browser kernel [Jang et al. 2012]
  - Mathematics: Feit-Thompson theorem [Gonthier et al. 2013]
- Important foundational developments:
  - Homotopy Type Theory [Univalent Foundations 2013]
  - Foundations of coinduction [Abel Pientka 2013, Atkey McBride 2013]

Whither Type Theory?

- More practical programming
  - Mutable state and ownership
  - General recursion
  - Control operators
- More expressive reasoning
  - Univalence: from isomorphism to equality
  - Classical logic
- Let's subsume everything

Computational Classical Type Theories

- Turning two stones into one bird: control, classicality
  - $\lambda\mu$-calculus [Parigot 1992]
  - $\bar{\lambda}\mu\tilde{\mu}$-calculus [Curien, Herbelin 2000]
  - Dual Calculus (DC) [Wadler 2003]
- Key insight [Griffin 1990]:

  Control operators have strictly classical types
- Control operators: exceptions, `call/cc`, etc.
- Important line of research in PL (e.g., [Felleisen 1988])

The Recursive Polarized Dual Calculus (RP-DC)

1. Logically minimal version of Wadler's DC
   - Just $\wedge$, $\neg$
   - Define $\vee$, $\rightarrow$ as usual
   - Obtain expected typings, reductions, for term constructs
2. Simple definition of inductive types, recursion
   - cf. $\mathrm{mono}^{X.C}_{A,B,x.M} N$ in $\mathrm{DC}_{\mu\nu}$ [Kimura, Tatstuta 2013]
3. Supports mixed inductive/coinductive types
   - Inductive types $\mu X.T$
   - Define coinductive types

     $$\nu X.T \ := \ \neg \mu X. \neg [\neg X / X] T$$
   - Similar to propositional μ-calculus [Kozen 1983]

RP-DC: Propositional Fragment

Syntax

DC is based on *sequent calculus*:

- $\Gamma \vdash t : + \, T$ means term t proves type T in context Γ
- $\Gamma \vdash t : - \, T$ means t refutes T in context Γ
- Computation happens when we cut proofs against refutations

$$
\begin{array}{lcl}
\textit{types } T & ::= & X \mid T \wedge T' \mid \neg T \\
\textit{terms } t & ::= & x \mid \textbf{halt}\, T \mid (t, t') \mid \iota\, x.t \mid \textbf{not}\, t \mid \delta\, x.t \bullet t' \\
\textit{polarities } p & ::= & + \mid - \\
\textit{contexts } \Gamma & ::= & . \mid \Gamma, x : p\, T
\end{array}
$$

Typing

$$\frac{}{\Gamma_1, x : p\ T, \Gamma_2 \vdash x : p\ T}\ \textsc{Ax}$$

$$\frac{}{\Gamma \vdash \textbf{halt}\ T : -\ T}\ \textsc{Halt}$$

$$\frac{\Gamma \vdash t_1 : +\ T_1 \qquad \Gamma \vdash t_2 : +\ T_2}{\Gamma \vdash (t_1, t_2) : +\ T_1 \wedge T_2}\ \textsc{AndPos}$$

$$\frac{\Gamma, x : +\ T_1 \vdash t : -\ T_2}{\Gamma \vdash \iota\, x.t : -\ T_1 \wedge T_2}\ \textsc{AndNeg}$$

$$\frac{\begin{array}{c}\Gamma, x : \bar{p}\ T \vdash t_1 : +\ T' \\ \Gamma, x : \bar{p}\ T \vdash t_2 : -\ T'\end{array}}{\Gamma \vdash \delta\, x.t_1 \bullet t_2 : p\ T}\ \textsc{Cut}$$

$$\frac{\Gamma \vdash t : \bar{p}\ T}{\Gamma \vdash \textbf{not}\ t : p\, \neg T}\ \textsc{Not}$$

Reduction

Judgments: $p\ t_1 \bullet t_2 \rightsquigarrow p'\ t_1' \bullet t_2'$

Analysis rules:

$$\frac{}{p\ (t_1, t_2) \bullet \iota\, x.t \rightsquigarrow p\ t_1 \bullet \delta\, x.t_2 \cdot t}\ \textsc{AnaAnd}$$

$$\frac{}{p\ \textbf{not}\ t \bullet \textbf{not}\ t' \rightsquigarrow \bar{p}\ t' \bullet t}\ \textsc{AnaNot}$$

Cut rules with value restriction (controlled by p)

$$\frac{}{+\ v \bullet (\delta\, y.t_1 \cdot t_2) \rightsquigarrow +\ [v/y]t_1 \bullet [v/y]t_2}\ \mathrm{RP}$$

$$\frac{}{+\ (\delta\, y.t_1 \cdot t_2) \bullet t \rightsquigarrow +\ [t/y]t_1 \bullet [t/y]t_2}\ \mathrm{LP}$$

Also have *marshalling* rules

Examples

Disjunction:

$$\begin{array}{lcl} T \vee T' & := & \neg(\neg T \wedge \neg T') \\ \mathbf{in}_1\, t & := & \mathbf{not}\, \iota\, x.\delta\, y.x \bullet \mathbf{not}\, t \\ \mathbf{in}_2\, t & := & \mathbf{not}\, \iota\, x.\mathbf{not}\, t \\ [t_1, t_2] & := & \mathbf{not}\, (\mathbf{not}\, t_1, \mathbf{not}\, t_2) \end{array}$$

Derived typing:

$$\frac{\Gamma \vdash t_1 : -\, T_1 \quad \Gamma \vdash t_2 : -\, T_2}{\Gamma \vdash [t_1, t_2] : -\, T_1 \vee T_2}$$

Derived analytic reduction: $+\, \mathbf{in}_2\, t \bullet [t_1, t_2] \;\rightsquigarrow^*\; +\, t \bullet t_2$

Implication:

$$\begin{array}{lcl} T \rightarrow T' & := & \neg(T \wedge \neg T') \\ \lambda x.t & := & \mathbf{not}\, \iota\, x.\mathbf{not}\, t \\ \langle t_1, t_2 \rangle & := & \mathbf{not}\, (t_1, \mathbf{not}\, t_2) \\ t_1\ t_2 & := & \delta\, x.t_1 \bullet \langle t_2, x \rangle \end{array}$$

Strictly classical principles, control operators also derivable

RP-DC: Recursion and Corecursion

Inductive Types and Recursion

$$\begin{array}{lcl} \textit{types } T & ::= & \ldots \mid \mu X.T \\ \textit{terms } t & ::= & \ldots \mid \mathbf{rec}\, x[y = t].t' \mid x[t] \\ \textit{contexts } \Gamma & ::= & \ldots \mid \Gamma, x : p\, X \triangleright T \end{array}$$

- Accumulator y in $\mathbf{rec}\, x[y = t_1].t'$
- Updated in recursive call $x[t_2]$

$$\frac{\mathbf{OccursOnly}\ +\ X\ T \quad \Gamma \vdash t_1 : p\, T' \quad \Gamma, x : p\, X \triangleright T', y : p\, T' \vdash t_2 : -\, T}{\Gamma \vdash \mathbf{rec}\, x[y = t_1].t_2 : -\, \mu X.T}\ \textsc{MuBar}$$

$$\frac{x : p\, X \triangleright T' \in \Gamma \qquad \Gamma \vdash t : p\, T'}{\Gamma \vdash x[t] : -\, X}\ \textsc{RecCall}$$

- Special substitution $[t/x]_{\mathbf{rec}}\, t'$ updates the accumulator:

$$[\mathbf{rec}\, x[y = t].t'/x]_{\mathbf{rec}}\, (x[t'']) = \mathbf{rec}\, x[y = t''].t'$$

Example: Lists

$$\begin{array}{lll} \mathbb{L}\ A & := & \mu X.\top \vee (A \wedge X) \\ \mathbb{N} & := & \mathbb{L}\ \top \end{array} \qquad \begin{array}{lll} \bot & := & \mu X.X \\ \top & := & \neg \bot \end{array}$$

$$\begin{array}{lll} \textbf{nil} & := & \textbf{in}_1 \textbf{true} \\ \textbf{cons} & := & \lambda x.\lambda y.\textbf{in}_2(x, y) \end{array} \qquad \begin{array}{lll} \textbf{false} & := & \textbf{rec}\, x[y = t].x[t] \\ \textbf{true} & := & \textbf{not false} \end{array}$$

Definition of **append**:

$$\begin{array}{l} \lambda x.\lambda y. \\ \quad \delta\, r.x \bullet \textbf{rec}\, f[z = r]. \\ \qquad\qquad [\delta\, y'.y \bullet z, \iota\, a.f[\delta\, y'.\textbf{cons}\ a\ y' \bullet z]] \end{array}$$

- Recursively update return continuation r in accumulator z
- To match on x use a cut. $\delta r.x \bullet \ldots$
- Base case: return y. $\delta\, y'.y \bullet z$
- Step case: get element a, recurse with updated continuation. $\iota\, a.f[\delta\, y'.\textbf{cons}\ a\ y' \bullet z]$

Corecursion

$$\begin{array}{lcl} \nu X.T & := & \neg\mu X.\neg[\neg X / X]T \\ \textbf{corec}\, f[z = t_1].t_2 & := & \textbf{not}\,\textbf{rec}\, f[z = t_1].\textbf{not}\,[\neg f / f]t_2 \end{array}$$

- Essentially, defining coinductive data by **rec**
  - **rec**-terms have an infinite unfolding
  - So do coinductive data!
- Must unfold lazily during reduction
- So $\textbf{rec}\, x[y = t_1].t_2$ is considered a value

Streams

$$\mathbb{S}\ A \;:=\; \nu X.A \wedge X$$
$$\;=\; \neg\mu X.\neg(A \wedge \neg X)$$

$$\textbf{tail} \;:=\; \lambda x.\delta\, y.x \bullet \textbf{not}\,\textbf{not}\,\iota\, y'.y$$
$$\textbf{head} \;:=\; \lambda x.\delta\, y.x \bullet \textbf{not}\,\textbf{not}\,\iota\, y'.\delta\, z.y' \bullet y$$

Examples:

$$\textbf{repeat} \;:=\; \lambda x.\textbf{corec}\, f[z = \textbf{true}].(x, f[\textbf{true}])$$
$$\;=\; \lambda x.\textbf{not}\,\textbf{rec}\, f[z = \textbf{true}].\textbf{not}\,(x, \textbf{not}\, f[\textbf{true}])$$

$$\textbf{nats} \;:=\; \lambda n.\textbf{corec}\, f[x = n].(n, f[\textbf{Suc}\ n])$$

$$\textbf{map} \;:=\; \lambda f.\lambda x.\textbf{corec}\, h[y = x].(f\ (\textbf{head}\ y), h[\textbf{tail}\ y])$$

Mixed inductive/coinductive types (see paper)

RP-DC: Metatheoretic Results

Logical Consistency

Theorem

The type $T \wedge \neg T$ is not provable by any **halt**-*free term in the empty context, for any type T.*

Canonical Inhabitants

- Q. What makes RP-DC nonconstructive?
- A. Closed normal forms need not be canonical values
- One proposal **Canon** $t : p\ T$ for when t is canonical of type T

$$\frac{\textbf{Canon}\ t_1 : +\ T_1 \quad \textbf{Canon}\ t_2 : +\ T_2}{\textbf{Canon}\ (t_1, t_2) : +\ T_1 \wedge T_2}\ \textsc{CanAndP}$$

$$\frac{\textbf{Canon}\ t : -\ T_2}{\textbf{Canon}\ \iota\, x.t : -\ T_1 \wedge T_2}\ \textsc{CanAndN2}$$

$$\frac{\textbf{Canon}\ t : p\ T}{\textbf{Canon}\ \textbf{not}\ t : \bar{p}\ T}\ \textsc{CanNot}$$

$$\frac{\textbf{Canon}\ t : -\ T_1}{\textbf{Canon}\ \iota\, x.\delta\, y.x \bullet t : -\ T_1 \wedge T_2}\ \textsc{CanAndN1}$$

$$\frac{\textbf{OccursOnly}\ +\ X\ T \quad \textbf{Canon}\ t : +\ [\mu\, X.T / X]\, T}{\textbf{Canon}\ t : +\ \mu\, X.T}\ \textsc{CanMu}$$

$$\frac{}{\textbf{Canon}\ \textbf{halt}\ T : -\ T}\ \textsc{CanHalt}$$

A Canonicity Theorem

Define the following (additionally, $S \neq X$ in $\mu X.S$):

$$\begin{array}{lcl} \textit{positive canonical } S & ::= & X \mid S \wedge S' \mid \neg R \mid \mu X.S \\ \textit{negative canonical } R & ::= & R \wedge R' \mid \neg S \mid \perp \end{array}$$

Theorem (Canonicity)

Suppose that t is a value, and the only **halt***-subterms it contains are of the form* **halt** S'. *Also, suppose every declaration in* Γ *is of the form* $x : -\, S_1$ *or* $x : +\, R_1$. *Then:*

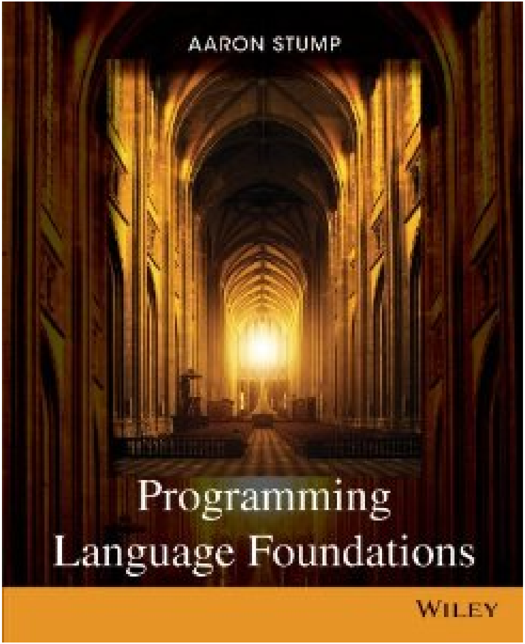
- *If* $\Gamma \vdash t : +\, S$, *then* **Canon** $t : +\, S$
- *If* $\Gamma \vdash t : -\, R$, *then* **Canon** $t : -\, R$

Conclusion

- Recursive Polarized Dual Calculus (RP-DC)
  - Version of DC with just $\wedge$, $\neg$, and μ types
  - Others definable, like $\nu X.T = \neg\mu X.\neg[\neg X/X]T$
  - Mixed recursion/corecursion supported
  - Logical consistency, canonicity
- Future work:
  - More metatheory: normalization (cf. Krivine's *classical realizability*)
  - Dependent types:

$$\frac{\Gamma \vdash t_1 : + T_1 \qquad \Gamma \vdash t_2 : + [t_1/x]T_2}{\Gamma \vdash (t_1, t_2) : + x : T \wedge T'} \qquad \frac{\Gamma, x : + T_1 \vdash t : - T_2}{\Gamma \vdash \iota x.t : - x : T_1 \wedge T_2}$$

Acknowledgments: NSF (Trellys project), Ott [Sewell et al. 2010]


AARON STUMP
Programming
Language Foundations
WILEY

Typing Rules for Inductive Types

$$\frac{\begin{array}{l}\textbf{OccursOnly}\ +\ X\ T \\ \Gamma \vdash t_1 : p\ T' \\ \Gamma, x : p\ X \triangleright T', y : p\ T' \vdash t_2 : -\ T\end{array}}{\Gamma \vdash \textbf{rec}\ x[y = t_1].t_2 : -\ \mu X.T}\ \textsc{MuBar} \qquad \frac{\begin{array}{l}\textbf{OccursOnly}\ +\ X\ T \\ \Gamma \vdash t : +[\mu X.T/X]T\end{array}}{\Gamma \vdash t : +\ \mu X.T}\ \textsc{Mu}$$

$$\frac{x : p\ X \triangleright T' \in \Gamma \qquad \Gamma \vdash t : p\ T'}{\Gamma \vdash x[t] : -X}\ \textsc{RecCall}$$