Strong Normalization for Guarded Types

Andreas Abel   Andrea Vezzosi

Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

PLS Seminar
ITU, Copenhagen, Denmark
20 August 2014

Introduction

- Guarded recursive types (Nakano, LICS 2000)
- (Negative) recursive types in type theory
- Applications in semantics (abstracting step-indexing)
- Applications in FRP (causality)
- This talk: strong normalization

Guarded types

- Types and terms.

$$\begin{array}{lcl} A, B & ::= & A \to B \mid \blacktriangleright A \mid X \mid \mu X A \\ t, u & ::= & x \mid \lambda x t \mid t\ u \mid \mathsf{next}\ t \mid t * u \end{array}$$

- Type equality: congruence closure of $\vdash \mu X A = A[\mu X A / X]$.
- Typing $\Gamma \vdash t : A$.

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{next}\ t : \blacktriangleright A} \qquad \frac{\Gamma \vdash t : \blacktriangleright (A \to B) \qquad \Gamma \vdash u : \blacktriangleright A}{\Gamma \vdash t * u : \blacktriangleright B}$$

$$\frac{\Gamma \vdash t : A \qquad \vdash A = B}{\Gamma \vdash t : B}$$

Reduction

- Redex contraction $t \mapsto t'$.

$$\begin{array}{lcl} (\lambda x t)\, u & \mapsto & t[u/x] \\ \mathsf{next}\, t * \mathsf{next}\, u & \mapsto & \mathsf{next}\, (t\ u) \end{array}$$

- Full one-step reduction $t \longrightarrow t'$: Compatible closure of $\mapsto$.

Recursion from recursive types

Guarded recursion combinator can be encoded.

$$
\begin{array}{rclcclcl}
 & & f & : & \blacktriangleright A \to A & & \\
 & & B & := & \mu X.\, \blacktriangleright X \to A & = & \blacktriangleright B \to A \\
 & & x & : & \blacktriangleright B & = & \blacktriangleright (\blacktriangleright B \to A) \\
 & & x * \mathsf{next}\, x & : & \blacktriangleright A & & \\
 & & f\,(x * \mathsf{next}\, x) & : & A & & \\
\omega & := & \lambda x.\, f\,(x * \mathsf{next}\, x) & : & \blacktriangleright B \to A & = & B \\
\mathsf{Y} & := & \omega\,(\mathsf{next}\,\omega) & : & A & &
\end{array}
$$

$$
\mathsf{Y} \longrightarrow f\,(\mathsf{next}\,\omega * \mathsf{next}\,(\mathsf{next}\,\omega)) \longrightarrow f\,(\mathsf{next}\,(\omega\,(\mathsf{next}\,\omega))) = f\,(\mathsf{next}\,\mathsf{Y})
$$

Full reduction $\longrightarrow$ diverges.

Restricted reduction

- Restore normalization: do not reduce under next.
- Relaxed: reduce only under next up to a certain depth.
- Family $\longrightarrow_n$ of reduction relations.

$$\frac{t \mapsto t'}{t \longrightarrow_n t'} \qquad \frac{t \longrightarrow_n t'}{\mathsf{next}\, t \longrightarrow_{n+1} \mathsf{next}\, t'}$$

- Plus compatibility rules for all other term constructors.
- $\longrightarrow_n$ is monotone in n (more fuel gets you further).
- Goal: each $\longrightarrow_n$ is strongly normalizing.

Strong normalization as well-foundedness

- $t \in \mathsf{sn}_n$ if $\longrightarrow_n$ reduction starting with t terminates.

$$\frac{\forall t'.\, t \longrightarrow_n t' \implies t' \in \mathsf{sn}_n}{t \in \mathsf{sn}_n}$$

- sn_n is antitone in n, since $\longrightarrow_n$ occurs negatively.
- More reductions $\implies$ less termination.

Inductive SN

- Lambda-calculus:

$$\frac{\vec{u} \in \mathsf{SN}}{x\,\vec{u} \in \mathsf{SN}} \qquad \frac{t \in \mathsf{SN}}{\lambda x t \in \mathsf{SN}} \qquad \frac{t[u/x]\,\vec{u} \in \mathsf{SN} \qquad u \in \mathsf{SN}}{(\lambda x t)\,u\,\vec{u} \in \mathsf{SN}}$$

- With evaluation contexts $E ::= \_ \mid E\,u$:

$$\frac{}{\_ \in \mathsf{SN}} \qquad \frac{E \in \mathsf{SN} \qquad u \in \mathsf{SN}}{E\,u \in \mathsf{SN}}$$

$$\frac{E \in \mathsf{SN}}{E[x] \in \mathsf{SN}} \qquad \frac{t \in \mathsf{SN}}{\lambda x t \in \mathsf{SN}} \qquad \frac{E[t[u/x]] \in \mathsf{SN} \qquad u \in \mathsf{SN}}{E[(\lambda x t)\,u] \in \mathsf{SN}}$$

Inductive SN (ctd.)

- Strong contraction $t \mapsto^{\mathsf{SN}} t'$.

$$\frac{u \in \mathsf{SN}}{(\lambda x t)\, u \mapsto^{\mathsf{SN}} t[u/x]}$$

- "Strong head reduction" $t \longrightarrow^{\mathsf{SN}} t'$.

$$\frac{t \mapsto^{\mathsf{SN}} t'}{E[t] \longrightarrow^{\mathsf{SN}} E[t']}$$

- SN with strong head reduction.

$$\frac{t \longrightarrow^{\mathsf{SN}} t' \qquad t' \in \mathsf{SN}}{t \in \mathsf{SN}}$$

SN with guarded types

- Extending evaluation contexts: $E ::= \cdots \mid E * u \mid \mathsf{next}\, t * E$
- Extending strong contraction:

$$\frac{u \in \mathsf{SN}_n}{(\lambda x t)\, u \mapsto^{\mathsf{SN}}_n t[u/x]} \qquad \frac{}{\mathsf{next}\, t * \mathsf{next}\, u \mapsto^{\mathsf{SN}}_n \mathsf{next}\,(t\, u)}$$

- Adding index to strong head reduction:

$$\frac{t \mapsto^{\mathsf{SN}}_n t'}{E[t] \longrightarrow^{\mathsf{SN}}_n E[t']} \qquad \frac{t \longrightarrow^{\mathsf{SN}}_n t' \qquad t' \in \mathsf{SN}_n}{t \in \mathsf{SN}_n}$$

- Adding rule for introduction:

$$\frac{}{\mathsf{next}\, t \in \mathsf{SN}_0} \qquad \frac{t \in \mathsf{SN}_n}{\mathsf{next}\, t \in \mathsf{SN}_{n+1}}$$

- SN_n is antitone in n.

Notions of s.n. coincide?

- Rules for SN_n are closure properties of sn_n.
- $\mathsf{SN}_n \subseteq \mathsf{sn}_n$ follows by induction on SN_n.
- Converse $\mathsf{sn}_n \subseteq \mathsf{SN}_n$ does not hold!
- Counterexamples are ill-typed s.n. terms, e.g.,

$$(\lambda x.\, x) * y \qquad \text{or} \qquad (\mathsf{next}\, x)\, y.$$

- Solution: consider only well-typed terms.
- Proof of $t \in \mathsf{sn}_n \implies t \in \mathsf{SN}_n$ by case distinction on t: neutral ($E[x]$), introduction ($\lambda xt, \mathsf{next}\, t$), or weak head redex.

Saturated sets (semantic types)

- Types are modeled by sets $\mathcal{A}$ of s.n. terms.
- Semantic function space should contain λs and terms that weak head reduce to λs.
- n-closure $\overline{\mathcal{A}}_n$ of $\mathcal{A}$ inductively:

$$\frac{t \in \mathcal{A}}{t \in \overline{\mathcal{A}}_n} \qquad \frac{E \in \mathsf{SN}_n}{E[x] \in \overline{\mathcal{A}}_n} \qquad \frac{t \longrightarrow_n^{\mathsf{SN}} t' \qquad t' \in \overline{\mathcal{A}}_n}{t \in \overline{\mathcal{A}}_n}$$

- $\mathcal{A}$ is n-saturated ($\mathcal{A} \in \mathsf{SAT}_n$) if $\overline{\mathcal{A}}_n \subseteq \mathcal{A}$.
- Saturated sets are non-empty (contain e.g. the variables).

Constructions on semantic types

- Function space and "later":

$$\begin{array}{lcl} \mathcal{A} \to \mathcal{B} & = & \{t \mid t\, u \in \mathcal{B} \text{ for all } u \in \mathcal{A}\} \\ \blacktriangleright_n \mathcal{A} & = & \overline{\{\mathsf{next}\, t \mid t \in \mathcal{A} \text{ if } n > 0\}}_n \end{array}$$

- If $\mathcal{A}, \mathcal{B} \in \mathsf{SAT}_n$ then $\mathcal{A} \to \mathcal{B} \in \mathsf{SAT}_n$.
- $\blacktriangleright_0 \mathcal{A} \in \mathsf{SAT}_0$.
- If $\mathcal{A} \in \mathsf{SAT}_n$ then $\blacktriangleright_{n+1} \mathcal{A} \in \mathsf{SAT}_{n+1}$.

Type interpretation

- Type interpretation $[\![A]\!]_n \in \mathsf{SAT}_n$

$$
\begin{array}{lcl}
[\![A \to B]\!]_n & = & \bigcap_{n' \leq n}([\![A]\!]_{n'} \to [\![B]\!]_{n'}) \\
[\![\blacktriangleright A]\!]_0 & = & \blacktriangleright_0 \mathsf{SN}_0 = \overline{\{\mathsf{next}\ t\}}_0 \\
[\![\blacktriangleright A]\!]_{n+1} & = & \blacktriangleright_{n+1} [\![A]\!]_n \\
[\![\mu X A]\!]_n & = & [\![A[\mu X A/X]]\!]_n
\end{array}
$$

- By lex. induction on $(n, \mathsf{size}(A))$ where $\mathsf{size}(\blacktriangleright A) = 0$.
- Requires recursive occurrences of X to be guarded by a $\blacktriangleright$.

Type soundness

- Context interpretation:

$$\rho \in [\![\Gamma]\!]_n \iff \rho(x) \in [\![A]\!]_n \text{ for all } (x{:}A) \in \Gamma$$

- Identity substitution $\mathsf{id} \in [\![\Gamma]\!]_n$ since $x \in [\![A]\!]_n$.
- Type soundness: if $\Gamma \vdash t : A$ then $t\rho \in [\![A]\!]_n$ for all n and $\rho \in [\![\Gamma]\!]_n$.
- Corollary: $t \in \mathsf{SN}_n$ for all n.

Formalization in Agda

- Intensional type theory does not support quotients well: in our case, types modulo type equality.
- $\Longrightarrow$ use infinite type expressions instead (coinduction).
- Only guarded types admit an interpretation.
- Typing judgement needs to be restricted to guarded types.
- $\Longrightarrow$ use mixed inductive-coinductive representation of types to express guard condition.

$$\begin{array}{lll} A, B & ::= & A \to B \mid \blacktriangleright A' \\ A', B' & ::=^{\mathrm{co}} & A \end{array}$$

- Intensional (propositional) equality too weak for coinductive types.
- $\Longrightarrow$ add an extensionality axiom for our coinductive type.

Well-typed terms

- We used intrinsically well-typed terms (data structure indexed by typing context and type expression).
- Second Kripke dimension (context) required “everywhere”, e.g., in SN and $[\![A]\!]$.

Conclusions

- Strong normalization is a new result, albeit expected.
- Main focus: Agda formalization.
- Needed dedication (mostly Andrea's).
- Forthcoming APLAS 2014 paper (literate Agda, fully machine-checked).
- Fuzzy hope that HoTT will improve equality situation for coinductive types.

Acknowledgments

- Rasmus Møgelberg, for discussions and the present invitation.
- Lars Birkedal, for a previous invitation that initiated this research.
- Neel Krishnaswami, for email input.
- The (other) Agda developers, especially Stevan Andjelkovic for the LaTeXbackend.