Strong Normalization for Guarded Recursive Types

Andreas Abel   Andrea Vezzosi

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

Theory Seminar
Institute of Cybernetics, Tallinn, Estonia
18 December 2014

Introduction

- Guarded recursive types (Nakano, LICS 2000)
- Negative recursive types while maintaining consistency
  - $\mu X. \blacktriangleright X \to A$
  - $\mathsf{fix} : (\blacktriangleright A \to A) \to A$
- Applications
  - Semantics (abstracting step-indexing)
  - Functional Reactive Programming (causality)
  - Coinduction (productivity, with a "Globally"/"$\Box$" modality)

- This talk: Strong Normalization.

Guarded types

- Types and terms.

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

- Occurrences of X in $\mu X. A$ must be under a $\blacktriangleright$ "guard".
- Good:
  - $\mu X. \blacktriangleright X$
  - $\mu X. A \times \blacktriangleright X$ and $\mu X. \blacktriangleright (A \times X)$
  - $\mu X. (\blacktriangleright X) \to A$ and $\mu X. \blacktriangleright (X \to A)$.
- Bad:
  - $\mu X. X$ and $\mu X. A \times X$
  - $\mu X. X \to A$ and $\mu X. X \to \blacktriangleright A$
  - $\mu X. \blacktriangleright \mu X. X$.

Typing

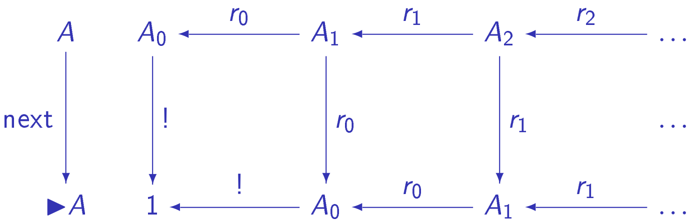
- Type equality: congruence closure of $\vdash \mu X.A = A[\mu X.A/X]$.
- Typing $\Gamma \vdash t : A$.

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

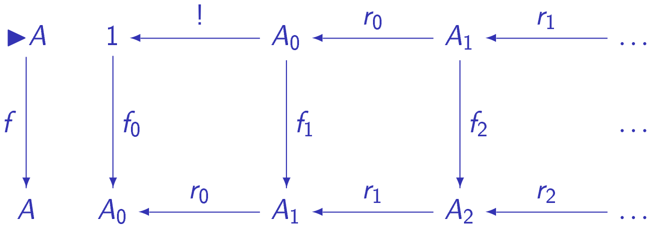
$$\frac{\Gamma \vdash t : A \qquad \vdash A = B}{\Gamma \vdash t : B}$$

Denotational Semantics

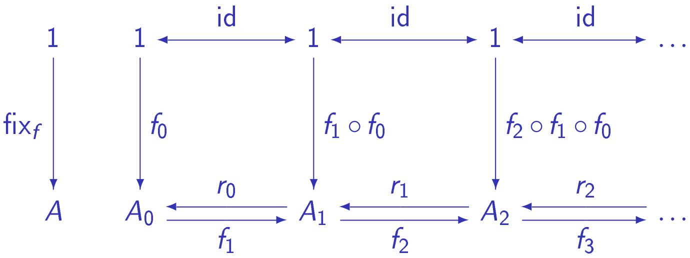
Types as streams of sets:
$A : \mathbb{N} \to Set$ with restriction maps.



Fixed-point construction (intuition)



Any map $f : \blacktriangleright A \to A$ has a fixed-point $\mathrm{fix}_f : 1 \to A$:



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.
The standard Y combinator would need a type T such that

$$T = T \to A$$

to typecheck the self applications of x and ω:

$$\begin{array}{lclcll} & & f & : & A \to A & \\ & & x\,x & : & A & \text{if } x : T \\ \omega & := & \lambda(x : T).\, f\,(x\,x) & : & T \to A & \\ \mathsf{Y} & := & \omega\,\omega & : & A & \end{array}$$

Recursion from recursive types

We can solve $T = \blacktriangleright T \to A$:

$$T = \mu X. \blacktriangleright X \to A$$

So we get a guarded fixpoint combinator:

$$\begin{array}{lllll}
 & & f & : & \blacktriangleright A \to A \\
 & & x & : & \blacktriangleright(\blacktriangleright T \to A) \quad \text{if } x : \blacktriangleright T \\
 & & x * \mathsf{next}\, x & : & \blacktriangleright A \quad \text{if } x : \blacktriangleright T \\
\omega & := & \lambda(x : \blacktriangleright T).\, f\,(x * \mathsf{next}\, x) & : & \blacktriangleright T \to A \\
\mathsf{Y}_f & := & \omega\,(\mathsf{next}\,\omega) & : & A
\end{array}$$

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

Note: Full reduction $\longrightarrow$ of Y_f diverges.

More Examples

- Streams!?
- RepMin: One pass through binary tree, replacing all labels by their minimum.
- Attribute grammars!?

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.

Restricted reduction (Example)

$$\mathsf{Y} \longrightarrow_0^* f\,(\mathsf{next}\,\mathsf{Y}) \not\longrightarrow_0$$

$$\mathsf{Y} \longrightarrow_1^* f\,(\mathsf{next}\,(f\,(\mathsf{next}\,\mathsf{Y}))) \not\longrightarrow_1$$

$$\mathsf{Y} \longrightarrow_2^* f\,(\mathsf{next}\,(f\,(\mathsf{next}\,(f\,(\mathsf{next}\,\mathsf{Y}))))) \not\longrightarrow_2$$

$$\vdots$$

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_n

- Take the inductively defined normal forms:

$$E ::= \_ \mid E\,u \mid E * u \mid \mathsf{next}\,t * E$$

$$\frac{E \in \mathsf{SN}_n}{E[x] \in \mathsf{SN}_n} \qquad \frac{t \in \mathsf{SN}_n}{\lambda x.t \in \mathsf{SN}_n} \qquad \frac{}{\mathsf{next}\,t \in \mathsf{SN}_0} \qquad \frac{t \in \mathsf{SN}_n}{\mathsf{next}\,t \in \mathsf{SN}_{n+1}}$$

- And close them under "Strong head reduction" $t \longrightarrow_n^{\mathsf{SN}} t'$

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

- $t \longrightarrow_n^{\mathsf{SN}} t'$ is like weak head reduction but erased terms must be s.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 ($\lambda x.t, \mathsf{next}\, t$), or weak head redex.

Saturated sets (semantic types)

- Types are modeled by sets $\mathscr{A} \subseteq \mathsf{SN}_n$.
- n-closure $\overline{\mathscr{A}}_n$ of $\mathscr{A}$ inductively:

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

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

Constructions on semantic types

- Function space and "later":

$$\mathscr{A} \to \mathscr{B} = \{t \mid t\,u \in \mathscr{B} \text{ for all } u \in \mathscr{A}\}$$
$$\blacktriangleright_n \mathscr{A} = \overline{\{\text{next}\,t \mid t \in \mathscr{A} \text{ if } n > 0\}}_n$$

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

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

Syntax of types as a mixed inductive-coinductive datatype:

$$\mathsf{Ty} = \nu X \mu Y.\,(Y \times Y) + X$$

```
mutual
  data Ty : Set where
    _→̂_ : (a b : Ty)   → Ty
    ▸̂_  : (a∞ : ∞Ty)   → Ty

  record ∞Ty : Set where
    coinductive
    constructor delay_
    field       force_ : Ty
```

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

Well-typed terms

```
data Tm (Γ : Cxt) : (a : Ty) → Set where
  var   : ∀{a}       (x : Var Γ a)                                  → Tm Γ a
  abs   : ∀{a b}     (t : Tm (a :: Γ) b)                            → Tm Γ (a →̂ b)
  app   : ∀{a b}     (t : Tm Γ (a →̂ b))  (u : Tm Γ a)               → Tm Γ b
  next  : ∀{a∞}      (t : Tm Γ (force a∞))                          → Tm Γ (▸̂ a∞)
  _*_   : ∀{a∞ b∞}   (t : Tm Γ (▸̂(a∞ ⇒ b∞))) (u : Tm Γ (▸̂ a∞))      → Tm Γ (▸̂ b∞)
```

- 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 & Further work

- Strong normalization is a new result, albeit expected for the restricted reduction.
- Agda formalization (ca. 3kLoc, 170kB) useful as basis for further research.
- Add modalities to handle (co)inductive types.
- Integrate into Intensional Type Theory.