Strong Normalization for Equi-(Co-)Inductive Types

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

Typed Lambda Calculi and Applications, TLCA'07
27 June 2007
Paris, France

Introduction

- Type-based termination: Each well-typed program terminates.
- Applications:
  - Type-theoretic theorem provers
  - Dependently-typed programming!?
- Mixed inductive/coinductive types and mixed recursive/corecursive programs.

Example: Stream Processors

- Modelling I/O in purely functional languages.
- `SP a b` contains codes for stream processors,
- i.e., functions from streams over `a` to streams over `b`.

```
data SP a b where
  get :: (a -> SP a b) -> SP a b
  put :: b -> SP a b -> SP a b

map :: (a -> b) -> SP a b
map f = get (\ a -> put (f a) (map f))
```

- Similar in FUDGETS library (GUI in Haskell).
- Theoretical treatment: Ghani, Hancock, Pattinson (ENTCS 2006).

Stream Processors as Mixed Inductive/Coinductive Type

- Haskell type:

```
data SP a b where
  get :: (a -> SP a b) -> SP a b
  put :: b -> SP a b -> SP a b
```

- Productivity: only finitely many `get`s before each `put`.
- Model `SP` by a least fixed-point nested (inductive type) inside a greatest fixed-point (coinductive type).

$$\mathsf{SP}\ A\ B := \nu X \mu Y.\,(B \times X) + (A \to Y)$$

Executing Stream Processors

- Stream eating: Execute `SP`-code.

```
eat :: SP a b -> [a] -> [b]
eat (get f) (a:as) = eat (f a) as
eat (put b t)   as  = b : eat t as
```

- Is `eat` total?
- 1st call to `eat` not guarded-by-constructor.
- This work: a type system ensuring totality.

Inductive Types

- Least fixed-points μF of monotone type constructors F.
- E.g. $\mathsf{List}\,A = \mu F$ with $F\,X = 1 + A \times X$.
- Iso-inductive types: Explicit folding and unfolding.

$$F\,(\mu F) \stackrel{\mathsf{in}}{\to} \mu F \stackrel{\mathsf{out}}{\to} F\,(\mu F)$$

$$\begin{array}{lcccl} \mathsf{nil} & := & \mathsf{in} \circ \mathsf{inl} & : & 1 \to \mathsf{List}\,A \\ \mathsf{cons} & := & \mathsf{in} \circ \mathsf{inr} & : & A \times \mathsf{List}\,A \to \mathsf{List}\,A \end{array}$$

- Equi-inductive types: Implicit (deep) folding via type equality.

$$F\,(\mu F) = \mu F$$

$$\begin{array}{lcl} \mathsf{nil} & := & \mathsf{inl} \\ \mathsf{cons} & := & \mathsf{inr} \end{array}$$

Motivation for Equi-Style

- In normalization proofs, mostly iso-types are chosen (Altenkirch [93–99], Barthe et al.[01–06], Geuvers [92], Giménez, Matthes [98], Mendler [87-91]; CIC).
- Notable exceptions: Parigot [92], Raffalli [93–94].
- Iso-types can be trivially simulated by equi-types, normalization results can be inherited.
- Equi-types in iso-types only by translation of typing derivations.
- Normalization for equi-types not implied by norm. for iso-types.
- *Loss of structure on terms requires compensating structures on types.*

Inductive Types: Construction From Below

- Least fixed-points can be reached by ordinal iteration:

$$\begin{array}{lcl} \mu^0 \; F & = & \emptyset \\ \mu^{\alpha+1} \; F & = & F\,(\mu^\alpha\, F) \\ \mu^\lambda \; F & = & \bigcup_{\alpha<\lambda} \mu^\alpha F \end{array}$$

- Size expressions $a ::= \imath \mid 0 \mid a+1 \mid \infty$.
- Sized inductive types $\mu^a F$.
- Laws: β, η, and

$$\begin{array}{lcl} \infty + 1 & = & \infty \\ \mu^{a+1} F & = & F\,(\mu^a F). \end{array}$$

- $\mathrm{List}^a A$ contains list of length $< a$.

Recursion

- General recursion (partial):

$$\frac{f : A \rightarrow C \vdash t : A \rightarrow C}{\mathsf{fix}\,(\lambda f.t) : A \rightarrow C}$$

- Recursion on size (total):

$$\frac{f : \mu^{\imath} F \rightarrow C \vdash t : \mu^{\imath+1} F \rightarrow C}{\mathsf{fix}^{\mu}\,(\lambda f.t) : \mu^{\infty} F \rightarrow C}$$

Sized Coinductive Types

- Greatest fixed-points $\nu^{\infty} F$ of monotone F.
- Approximation from above.
- E.g. $\mathsf{Stream}^{a} A = \nu^{a} \lambda X.\, A \times X$ contains streams of depth $\geq a$.
- Corecursion on depth (total):

$$\frac{f : \nu^{\imath} F \vdash t : \nu^{\imath+1} F}{\mathsf{fix}^{\nu}\, (\lambda f.t) : \nu^{\infty} F}$$

- E.g., $\mathsf{repeat}\, x = \mathsf{fix}^{\nu}(\lambda y.\, (x, y))$.

Terminating Reduction for Recursion

- Naive reduction $\mathsf{fix}^\mu s \longrightarrow s\,(\mathsf{fix}^\mu s)$ diverges.
- Lazy (weak head) values $v ::= (r, s) \mid \cdots \mid \lambda xt \mid \mathsf{fix}^\mu s \mid \mathsf{fix}^\nu s$.
- Only expand recursive functions applied to a value.

$$\mathsf{fix}^\mu s\, v \longrightarrow s\,(\mathsf{fix}^\mu s)\, v$$

- Shallow evaluation contexts $e(\_) := \mathsf{fst}\,\_ \mid \cdots \mid \_\,s \mid \mathsf{fix}^\mu\, s\,\_$.
- Deep evaluation contexts $E(\_) = e_1(\ldots e_n(\_))$ for $n \geq 0$.

Termination Reduction for Corecursion

- Only expand corecursive objects whose value is demanded.

$$e(\mathsf{fix}^{\nu}\, s) \longrightarrow e(s\,(\mathsf{fix}^{\nu}\, s))$$

- Non-confluence. Critical pair: $s = \lambda z \lambda x.x$ and

$$\mathsf{fix}^{\mu} s\,(\mathsf{fix}^{\nu}\, s)$$

$$\swarrow \qquad \searrow$$

$$s\,(\mathsf{fix}^{\mu} s)\,(\mathsf{fix}^{\nu}\, s) \qquad \mathsf{fix}^{\mu} s\,(s\,(\mathsf{fix}^{\nu}\, s))$$

$$\downarrow \qquad \downarrow$$

$$(\lambda x.x)\,(\mathsf{fix}^{\nu}\, s) \qquad \mathsf{fix}^{\mu} s\,(\lambda xx)$$

$$\downarrow \qquad \downarrow$$

$$\mathsf{fix}^{\nu}\, s \qquad s\,(\mathsf{fix}^{\mu} s)\,(\lambda xx)$$

$$\downarrow$$

$$\lambda xx$$

Breaking the Symmetry

- Do not unfold corecursive arguments of recursive functions.

$$e(\mathsf{fix}^\nu\, s) \longrightarrow e(s\,(\mathsf{fix}^\nu\, s)) \qquad e(\_) \neq \mathsf{fix}^\mu\, s'\, \_$$

- Confluence regained.
- Strong normalization provable.

Proving Strong Normalization

- $\mathcal{S}$ set of strongly normalizing terms.
- Safe (weak head) reduction, preserves s.n. in both directions.

$$
\begin{array}{lcll}
E((\lambda x t)\, s) & \rhd & E([s/x]t) & \text{if } s \in \mathsf{SN} \\
E(\mathsf{fix}^{\mu} s\, v) & \rhd & E(s\,(\mathsf{fix}^{\mu} s)\, v) & \\
E(e(\mathsf{fix}^{\nu} s)) & \rhd & E(e(s\,(\mathsf{fix}^{\nu} s))) & \text{if } e(\_) \neq \mathsf{fix}^{\mu}\, s'\, \_ \\
\ldots & & & \\
\text{reflexivity, transitivity} & & &
\end{array}
$$

- $\mathcal{N} = \{t \in \mathcal{S} \mid t \rhd E(x)\}$ set of neutral terms.
- $\mathcal{A}$ saturated, $\mathcal{A} \in \mathsf{SAT}$, if $\mathcal{N} \subseteq \mathcal{A} \subseteq \mathcal{S}$ and $\mathcal{A}$ is closed under safe reduction and expansion.

Soundness of Recursion

Semantical recursion rule:

$$\frac{\forall \imath.\, s \in (\mu^{\imath}\mathcal{F} \to \mathcal{C}) \to \mu^{\imath+1}\mathcal{F} \to \mathcal{C}}{\mathsf{fix}^{\mu} s \in \mu^{\alpha}\mathcal{F} \to \mathcal{C}}$$

Show $r \in \mu^{\alpha} F$ implies $\mathsf{fix}^{\mu} s\, r \in \mathcal{C}$ by induction on ordinal α.

- Case $\alpha = 0$. Then $\mu^{0}\mathcal{F} = \mathcal{N}$ and $r \in \mathcal{N}$ implies $\mathsf{fix}^{\mu} s\, r \in \mathcal{N} \subseteq \mathcal{C}$.
- Case $\alpha = \alpha' + 1$ and $r \rhd v$.
  - $\mathsf{fix}^{\mu} s \in \mu^{\alpha'}\mathcal{F} \to \mathcal{C}$ by induction hypothesis.
  - $s\,(\mathsf{fix}^{\mu} s) \in \mu^{\alpha'+1}\mathcal{F} \to \mathcal{C}$ by assumption.
  - $\mathsf{fix}^{\mu} s\, r \rhd s\,(\mathsf{fix}^{\mu} s)\, v \in \mathcal{C}$.
- Case α limit. By induction hypothesis.

Soundness of Corecursion

Semantical corecursion rule:

$$\frac{\forall\imath. s \in \nu^{\imath}\mathcal{F} \to \nu^{\imath+1}\mathcal{F}}{\mathsf{fix}^{\nu} s \in \nu^{\alpha}\mathcal{F}}$$

By induction on α.

- Case $\alpha = 0$. Then $\nu^{0}\mathcal{F} = \mathcal{S}$ and $s \in \mathcal{S}$ implies $\mathsf{fix}^{\nu} s \in \mathcal{S}$.
- Case $\alpha = \alpha' + 1$.
  - $\mathsf{fix}^{\nu} s \in \nu^{\alpha'}\mathcal{F}$ by induction hypothesis.
  - $s\,(\mathsf{fix}^{\nu} s) \in \nu^{\alpha'+1}\mathcal{F}$ by assumption.
  - How to prove $\mathsf{fix}^{\nu} s \in \nu^{\alpha'+1}\mathcal{F}$??

Idea: make this additional closure property on saturated sets.

Guarded Saturated Sets

- Consider closure property

$$s\,(\mathsf{fix}^{\nu}\,s) \in \mathcal{A} \text{ implies } \mathsf{fix}^{\nu}\,s \in \mathcal{A}. \qquad (1)$$

- Unsound for $\mathcal{N}$: must not contain values!
- Otherwise $\mathsf{fix}^{\mu}\,s \in \mathcal{N} \to \mathcal{N}$ fails.
- Solution: define a subclass of guarded saturated sets closed under (1).

Checking Guardedness

- 1, $A \to B$, $A \times B$, ... are guarded.
- 0, $\mu^0 F$ are unguarded.
- $\nu^a F$ is guarded if $F\,1$ is or $a = 0$.
- $\mu^a F$ is guarded if $F\,0$ is and $a > 0$.
- Statically checkable through kinding system with two base kinds $*_u$ (unguarded type) and $*_g$ (guarded type).
- Guardedness is not emptiness: $1 \to 0$ is empty, but guarded.

Stream Processors Revisited

- $\mathsf{SP}\ A\ B := \nu^{\infty}\lambda X.\,\mu^{\infty}\lambda Y.\,(B \times X) + (A \to Y)$
- Sized type (I) of constructors put := inl and get := inr.

$$
\begin{array}{lcl}
\mathsf{SP}^{\imath}\ A\ B & := & \nu^{\imath}\lambda X.\,\mu^{\infty}\lambda Y.\,(B \times X) + (A \to Y) \\
\mathsf{put} & : & B \times \mathsf{SP}^{\imath}\ A\ B \to \mathsf{SP}^{\imath+1}\ A\ B \\
\mathsf{get} & : & (A \to \mathsf{SP}^{\imath+1}\ A\ B) \to \mathsf{SP}^{\imath+1}\ A\ B
\end{array}
$$

- Unfolding coinduction: $\mathsf{SP}\ A\ B = \mu^{\infty}\lambda Y.\,(B \times \mathsf{SP}\ A\ B) + (A \to Y)$
- Sized type (II).

$$
\begin{array}{lcl}
\mathsf{SP}_{\jmath}\ A\ B & := & \mu^{\jmath}\lambda Y.\,B \times \mathsf{SP}\ A\ B + (A \to Y) \\
\mathsf{get} & : & (A \to \mathsf{SP}_{\jmath}\ A\ B) \to \mathsf{SP}_{\jmath+1}\ A\ B \\
\mathsf{put} & : & B \times \mathsf{SP}_{\infty}\ A\ B \to \mathsf{SP}_{\jmath+1}\ A\ B
\end{array}
$$

Totality of Stream Eating

- eat defined by an outer coiteration into streams
- . . . and an inner iteration over stream processors.
- Expressed as a lexicographic induction over size.

$$\mathsf{eat} : \forall \imath \forall \jmath.\, \mathsf{SP}_{\jmath}\, A\, B \rightarrow \mathsf{Stream}\, A \rightarrow \mathsf{Stream}^{\imath}\, B$$

Conclusion

- Present work solves #2.005 on the Abel List of Open Problems.
- Further work: develop and verify guardedness calculus.
- Acknowledgments:

  - Stream Processor example communicated to me by Thorsten Altenkirch and Conor McBride.
  - Guardedness idea arose during invitation to LORIA by Frédéric Blanqui and Colin Riba.