Sized (Co-)Inductive Types

With Applications to Generic Programming

Andreas Abel

ProgLog Seminar, April 5, 2006

1 Example: Generic Finite Maps

Finite Maps for List-shaped Keys

- Key type: `[a]`, value type: `v`

```
data MapList f v
   = Leaf
   | Node (Maybe v) (f (MapList f v))
```

- If `f w` $\cong$ `a` $\to_{\mathsf{fin}}$ `w`, then `MapList f v` $\cong$ `[a]` $\to_{\mathsf{fin}}$ `v`.
- Looking up a list-shaped key:

```
lookupList :: (forall w. a -> f w -> Maybe w) ->
              [a] -> MapList f v -> Maybe v
lookupList lo l      Leaf        = Nothing
lookupList lo []     (Node mv m) = mv
lookupList lo (a:as) (Node mv m)
    = lo a m >>= lookupList as
```

Merging Finite Maps

- Merging (possibly undefined) values.

```
comb :: (v -> v -> v)
     -> Maybe v -> Maybe v -> Maybe v
```

- Merging finite maps:

```
mergeList ::
  (forall w. (w -> w -> w) -> f w -> f w -> f w)
  -> (v -> v -> v)
```

```
  -> MapList f v -> MapList f v -> MapList f v
mergeList mergeF c Leaf t = t
mergeList mergeF c t Leaf = t
mergeList mergeF c (Node m1 t1) (Node m2 t2) =
  Node (comb c m1 m2)
       (mergeF (mergeList mergeF c) t1 t2)
```

Termination of Merging

- Is `mergeList` terminating on all inputs?

```
mergeList mergeF c (Node m1 t1) (Node m2 t2) =
  Node (comb c m1 m2)
       (mergeF (mergeList mergeF c) t1 t2)
```

- Consider:

```
mf m t1 t2 = m (Node Nothing t1) (Node Nothing t2)

run = mergeList mf fst (Node Nothing t1)
                       (Node Nothing t2)
```

- But: `mf` cannot be assigned the type

```
  forall w. (w -> w -> w) -> f w -> f w -> f w
```

Sized Inductive Types

- Type constructor $\mu : \mathsf{ord} \xrightarrow{+} (* \xrightarrow{+} *) \xrightarrow{+} *$.
- $\mu^a F$ contains trees of height $< a$.
- Finite maps:

$$\begin{array}{lcl} \mathsf{Map}\langle\mathsf{List}\rangle & : & \mathsf{ord} \xrightarrow{+} (* \xrightarrow{+} *) \xrightarrow{+} * \xrightarrow{+} * \\ \mathsf{Map}\langle\mathsf{List}\rangle^a\, F\, V & = & \mu^a\, (\lambda X.\, 1 + (1 + V) \times F\, X) \end{array}$$

- Terminating recursion

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

Analysis of Merging

- Auxiliary definitions:

$$\begin{array}{lcl} \mathsf{Map}\langle\mathsf{List}\rangle^a\, F\, V & = & \mu^a\, (\lambda X.\, 1 + (1+V) \times F\, X) \\ \\ \mathsf{Bin} & : & * \xrightarrow{\circ} * \\ \mathsf{Bin} & := & \lambda V.\, V \to V \to V \\ \\ \mathsf{comb} & : & \forall V.\, \mathsf{Bin}\, V \to \mathsf{Bin}\, (\mathbf{1} + V) \end{array}$$

- Merging, formalized:

$$\begin{array}{lcl} \mathsf{merge}\langle\mathsf{List}\rangle & : & \forall F.\, (\forall V.\, \mathsf{Bin}\, V \to \mathsf{Bin}\, (F\, V)) \to \\ & & \quad \forall W.\, \mathsf{Bin}\, W \to \forall \imath \mathsf{Bin}\, (\mathsf{Map}\langle\mathsf{List}\rangle^\imath\, F\, W) \\ \mathsf{merge}\langle\mathsf{List}\rangle & := & \lambda merge_F \lambda c.\, \mathsf{fix}_0^\mu\, \lambda merge. \\ & & \quad \mathsf{comb}\, (\lambda(mv_1, t_1)\lambda(mv_2, t_2). \\ & & \qquad (\mathsf{comb}\ c\ mv_1\ mv_2,\ merge_F\ merge\, t_1\, t_2)) \end{array}$$

2 Iso- and Equi-(Co)inductive Types

Iso-Inductive Types

- Principle: $\mu^{a+1}\, F$ is *isomorphic* to $F\, (\mu^a\, F)$.
- Size-level equality:

$$\infty + 1 = \infty$$

- Introduction:

$$\frac{t : F\, (\mu^a\, F)}{\mathsf{in}\, t : \mu^{a+1} F}$$

- Elimination:

$$\frac{t : \mu^{a+1} F}{\mathsf{out}\, t : F\, (\mu^a\, F)} \qquad \frac{s : \forall \imath.\, (\mu^\imath F \to C) \to \mu^{\imath+1} F \to C}{\mathsf{fix}^\mu\, s : \mu^a F \to C}$$

- Reductions:

$$\begin{array}{lcl} \mathsf{out}\, (\mathsf{in}\, t) & \longrightarrow & t \\ \mathsf{fix}^\mu\, s\, (\mathsf{in}\, t) & \longrightarrow & s\, (\mathsf{fix}^\mu\, s)\, (\mathsf{in}\, t) \end{array}$$

Iso-Coinductive Types

- Principle: $\nu^{a+1}\, F$ is *isomorphic* to $F\, (\nu^a\, F)$.
- Size-level equality:

$$\infty + 1 = \infty$$

- Introduction:

$$\frac{t : F\,(\nu^a\,F)}{\mathsf{in}\,t : \nu^{a+1}F} \qquad \frac{s : \forall\imath.\,(A_{1..n} \to \nu^\imath\,F) \to A_{1..n} \to \nu^{\imath+1}F}{\mathsf{fix}^\nu_n\,s : A_{1..n} \to \nu^a F}$$

- Elimination:

$$\frac{t : \nu^{a+1}F}{\mathsf{out}\,t : F\,(\nu^a\,F)}$$

- Reductions:

$$\begin{array}{lcl} \mathsf{out}\,(\mathsf{in}\,t) & \longrightarrow & t \\ \mathsf{out}\,(\mathsf{fix}^\nu_n\,s\,t_{1..n}) & \longrightarrow & \mathsf{out}\,(s\,(\mathsf{fix}^\nu_n\,s)\,t_{1..n}) \end{array}$$

Equi-Inductive Types

- Principle: $\mu^{a+1}\,F = F\,(\mu^a\,F)$ as type-level equality.
- *Deep* (un)folding.
- No tagging on the term level.
- No native introduction/elimination principles.
- Recursion:

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

- Reduction:

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

- Values: $v ::= \lambda xt \mid (r, s) \mid \mathsf{inl}\,r \mid \mathsf{inr}\,r \mid \ldots$

Equi-Coinductive Types

- Principle: $\nu^{a+1}\,F = F\,(\nu^a\,F)$ as type-level equality.
- Corecursion:

$$\frac{s : \forall\imath.\,(A_{1..n} \to \nu^\imath\,F) \to A_{1..n} \to \nu^{\imath+1}F}{\mathsf{fix}^\nu_n\,s : A_{1..n} \to \nu^a F}$$

- Reduction:

$$e(\mathsf{fix}^\nu_n\,s\,t_{1..n}) \quad \longrightarrow \quad e(s\,(\mathsf{fix}^\nu_n\,s)\,t_{1..n})$$

- Elimination frames:

$$e(\_) ::= \_\,s \mid \mathsf{fst}\,\_ \mid \mathsf{snd}\,\_ \mid \mathsf{case}\,\_\ldots$$

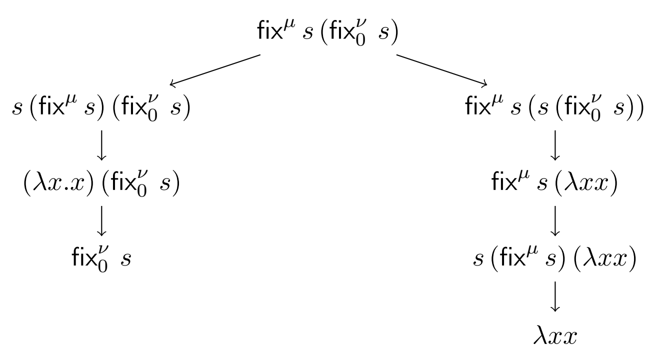
Embedding Iso into Equi

- $\mu^a_{\mathsf{iso}}\,(\lambda X.A) := \mu^a\,(\lambda X.\,1 \to A)$.
- $\mathsf{in}\,t := \lambda\_.t$
- $\mathsf{out}\,t := t\,()$
- Note: $\mathsf{in}\,t$ is translated into a value
- $\mathsf{out}\,\_$ is translated into an evaluation frame
- Reductions are simulated:

$$\begin{array}{lcl} \mathsf{out}\,(\mathsf{in}\,t) & \longrightarrow & t \\ \mathsf{fix}^\mu\,s\,(\mathsf{in}\,t) & \longrightarrow & s\,(\mathsf{fix}^\mu\,s)\,(\mathsf{in}\,t) \\ \mathsf{out}\,(\mathsf{fix}^\nu_n\,s\,t_{1..n}) & \longrightarrow & \mathsf{out}\,(s\,(\mathsf{fix}^\nu_n\,s)\,t_{1..n}) \end{array}$$

Non-Confluence

- Recursive evaluation frame $e(\_) = \mathsf{fix}^\mu\,s\,\_$.
- Corecursive value $v = \mathsf{fix}^\nu_n\,s\,t_{1..n}$



- Critical term: $s = \lambda z\lambda x.x$ and [2ex]

Avoiding Non-Confluence

- Consider $\mathsf{fix}^\mu\,s\,(\mathsf{fix}^\nu_0\;s)$ blocked (unfold neither recursion nor corecursion).
- Drawback: there are closed, blocked terms.
- Consolidation: arise only for unguarded types like $\mu(\lambda X.\nu(\lambda Y.A))$.
- Speculative: unfold recursion and corecursion simultaneously.

3 Models and Strong Normalization

Modelling Sized Types (Iso)

- $\mathcal{A} := [\![A]\!]$ is a saturated set of strongly normalizing terms.
- $[\![A \to B]\!] = \{r \mid r\, s \in [\![B]\!]$ for all $s \in [\![A]\!]$.
- $\mathcal{A}^\mu = \overline{\{\mathsf{in}\, t \mid t \in \mathcal{A}\}}$
- $\mathcal{A}^\nu = \{t \mid \mathsf{out}\, t \in \mathcal{A}\}$
- $\bot$ be least saturated set
- $\top = \mathsf{SN}$, greatest saturated set
- $[\![\mu^a F]\!] = \boldsymbol{\mu}^{[a]}\mathcal{F}$ with $\mathcal{F}(\mathcal{A}) = ([\![F]\!](\mathcal{A}))^\mu$

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

Soundness of Recursion (Iso)

- Term model
- Recursion rule: $$\frac{s : \forall\imath.\,(\mu^\imath F \to C) \to \mu^{\imath+1} F \to C}{\mathsf{fix}^\mu\; s : \mu^a F \to C}$$
- Soundness by transfinite induction on $[\![a]\!]$.
- Step case
  - Show $\mathsf{fix}^\mu\, s\, r \in [\![C]\!]$ for $r \in [\![\mu^{a+1} F]\!]$.
  - Iso-system: we can assume $r \longrightarrow \mathsf{in}\, t$.
  - Have $\mathsf{fix}^\mu\, s\,(\mathsf{in}\, t) \longrightarrow s\,(\mathsf{fix}^\mu\; s)\,(\mathsf{in}\, t)$.
  - Show $s\,(\mathsf{fix}^\mu\; s)\,(\mathsf{in}\, t) \in [\![C]\!]$.
  - Use assumption for s on induction hypothesis.

Soundness of Corecursion (Iso)

- Corecursion rule: $$\frac{s : \forall\imath.\,\nu^\imath\, F \to \nu^{\imath+1} F}{\mathsf{fix}^\nu_n\, s : \nu^a F}$$
- Soundness by transfinite induction on $[\![a]\!]$.
- Step case

- Show $\mathsf{fix}^\nu\ s \in [\![\nu^{a+1} F]\!]$.
- Iso-system: Show $\mathsf{out}\,(\mathsf{fix}^\nu\ s) \in [\![F\,(\nu^a F)]\!]$.
- Have $\mathsf{out}\,(\mathsf{fix}^\nu\ s) \longrightarrow \mathsf{out}\,(s\,(\mathsf{fix}^\nu\ s))$.
- Show $s\,(\mathsf{fix}^\mu\ s) \in [\![\nu^{a+1} F]\!]$.
- Use assumption for s on induction hypothesis.

Modelling Types (Equi)

- Semantic terms: set of terms that pass a certain number of test
- t passes $E \;:\Longleftrightarrow\; t \perp E$
- $t \perp E \;:\Longleftrightarrow\; E(t) \in \mathsf{SN}$
- $\mathcal{A} = \mathcal{E}^\perp = \{t \mid t \perp E \text{ for all } E \in \mathcal{E}\}$
- $\mathcal{A} \to \mathcal{E}^\perp = \{E \circ (\_\,s) \mid E \in \mathcal{E}, s \in \mathcal{A}\}$
- Galois connection: $\mathcal{A}^\perp \supseteq \mathcal{E} \iff \mathcal{A} \subseteq \mathcal{E}^\perp$.
- Closure: $\overline{\mathcal{A}} = \mathcal{A}^{\perp\perp}$.
- Infimum of saturated sets: $\inf_{i\in I} \mathcal{A}_i = \bigcap_{i\in I} \mathcal{A}_i$
- Supremum of saturated sets: $\sup_{i\in I} \mathcal{A}_i = \overline{\bigcup_{i\in I} \mathcal{A}_i}$

Soundness of Corecursion (Equi)

- Corecursion rule:
$$\frac{s : \forall\imath.\, \nu^\imath\, F \to \nu^{\imath+1} F}{\mathsf{fix}^\nu_n\ s : \nu^a F}$$
- Soundness by transfinite induction on $[\![a]\!]$.
- Step case
  - Show $\mathsf{fix}^\nu\ s \in [\![\nu^{a+1} F]\!] = \mathcal{E}^\perp$.
  - Assume arbitrary $E \in \mathcal{E}$
  - Show $E\,(\mathsf{fix}^\nu\ s) \in \mathsf{SN}$.
  - Have $E\,(\mathsf{fix}^\nu\ s) \longrightarrow E\,(s\,(\mathsf{fix}^\nu\ s))$.
  - Show $s\,(\mathsf{fix}^\mu\ s) \in [\![\nu^{a+1} F]\!]$.
  - Use assumption for s on induction hypothesis.

4 Generic Programming with Sized Types

Generic Programming: Type-Indexed Types

- Type-Indexed Types

$$
\begin{array}{lcll}
\mathsf{Type}\langle C\rangle & = & \textit{user-defined} & \text{for } C \in \{1, +, \times, \mathsf{Int}, \mathsf{Char}, \dots\} \\
\mathsf{Type}\langle X\rangle & = & X & \\
\mathsf{Type}\langle \lambda X F\rangle & = & \lambda X.\, \mathsf{Type}\langle F\rangle & \\
\mathsf{Type}\langle F\, G\rangle & = & \mathsf{Type}\langle F\rangle\ \mathsf{Type}\langle G\rangle & \\
\mathsf{Type}\langle \mu\rangle & = & \mu &
\end{array}
$$

Generic Programming: Type-Indexed Values

- Type-Indexed Values

$$
\begin{array}{lcll}
\mathsf{poly}\langle C\rangle & = & \textit{user-defined} & \\
\mathsf{poly}\langle X\rangle & = & x & \\
\mathsf{poly}\langle \lambda \imath F : \mathsf{ord} \to \kappa\rangle & = & \mathsf{poly}\langle F\rangle & \\
\mathsf{poly}\langle \lambda X F : \kappa_1 \to \kappa_2\rangle & = & \lambda x.\, \mathsf{poly}\langle F\rangle & \text{where } \kappa_1 \neq \mathsf{ord} \\
\mathsf{poly}\langle F\, G\rangle & = & \mathsf{poly}\langle F\rangle\ \mathsf{poly}\langle G\rangle & \\
\mathsf{poly}\langle \mu^a_\kappa\rangle & = & \mathsf{fix}^\mu_n & \text{for some } n
\end{array}
$$

Generic Programming: Results

- Extended gen. prog. to sized types.
- Termination of generic finite map operations (Hinze) through typing.