Termination and Guardedness Checking with Continuous Types

Towards a Higher-Order Polymorphic Lambda-Calculus With Sized Types

Andreas Abel

Slide 1

TLCA 2003
Valencia, Spain
June 10, 2003

— Work in progress —

Work supported by: GKLI (DFG), TYPES and APPSEM-II

Setting the stage...

- Curry-Howard-Isomorphism:
  proofs by induction = programs with recursion
- Only *terminating* programs constitute valid proofs.
- Design issue: How to integrate terminating recursion into proof/programming language?

Slide 2

One approach: special forms of recursion

Slide 3

- Tame recursion by restricting to special patterns.
- Iteration/catamorphisms
  e.g. Haskell's `List.fold`
- Primitive recursion/paramorphisms
- Problems:
  - Non-trivial operational semantics makes it harder to understand programs.
  - I do not want to write all of my list-processing functions using `fold`.

Another approach: recursion with termination checking

Slide 4

- Use *general recursion*: `letrec`.
- Has "intuitive" meaning through simple operational semantics.
- In general not normalizing, need termination checking.
- Here we used the *sized types* approach [Hughes et al. 1996] [Barthe et al. 2003?].
- View data as trees.
- *Size* = height = # constructors in longest path of tree.
- Height of input data must decrease in each recursive call.
- Termination is ensured by type-checker.

Sized types in a nutshell

Slide 5

- Sizes are *upper bounds*.
- List^{a} denotes lists of length $< a$.
- List^{∞} denotes list of arbitrary (but finite) length.
- Sizes induce *subtyping*: $\mathsf{List}^{a} \leq \mathsf{List}^{b}$ if $a \leq b$.
- In general, sizes are *ordinal numbers*, needed e.g. for infinitely branching trees.
- Size expressions:

$$
\begin{array}{rcll}
a & ::= & i & \text{variable} \\
  & | & a+1 & \text{sucessor} \\
  & | & \infty & \text{ultimate limit, denoting } \Omega \text{ (first uncountable)}
\end{array}
$$

Example: list splitting

Slide 6

$$
\begin{array}{ll}
\mathsf{split} : & \forall A\!:\!*.\ \mathsf{List}\ A \to \mathsf{List}\ A \times \mathsf{List}\ A \\
\mathsf{split}\ [\,] & = \langle [\,] \quad , [\,] \quad \rangle \\
\mathsf{split}\ (y :: l\ ) & = \mathsf{let}\ \langle xs\ , ys\ \rangle = \mathsf{split}\ l\ \ \mathsf{in} \\
 & \quad \langle (y :: ys) \quad , xs \quad\quad \rangle
\end{array}
$$

- Sized types allow us to express that split denotes a non-size increasing function.

Example: list splitting

$\mathsf{split} : \forall i\!:\!\mathsf{ord}.\, \forall A\!:\!*.\ \mathsf{List}^i A \to \mathsf{List}\ A \times \mathsf{List}\ A$

$$\begin{aligned}
\mathsf{split}\ [\,] &= \langle [\,]\quad, [\,]\quad \rangle \\
\mathsf{split}\ (y :: l^i)^{i+1} &= \mathsf{let}\ \langle xs\ , ys\ \rangle = \mathsf{split}\, l^i\ \mathsf{in} \\
&\quad \langle (y :: ys)\quad , xs\quad \rangle
\end{aligned}$$

Slide 7

- To compute split at stage $i+1$, split is only used at stage i.
- Hence, split is terminating.

Example: list splitting

$\mathsf{split} : \forall i\!:\!\mathsf{ord}.\, \forall A\!:\!*.\ \mathsf{List}^i A \to \mathsf{List}^i A \times \mathsf{List}^i A$

$$\begin{aligned}
\mathsf{split}\ [\,]^{i+1} &= \langle [\,]^{i+1}, [\,]^{i+1} \rangle \\
\mathsf{split}\ (y :: l^i)^{i+1} &= \mathsf{let}\ \langle xs^i, ys^i \rangle = \mathsf{split}\, l^i\ \mathsf{in} \\
&\quad \langle (y :: ys)^{i+1}, xs^{i \le i+1} \rangle
\end{aligned}$$

Slide 8

- We additionally can infer that split is non-size increasing.
- Using split, we can define merge sort...

Example: merge sort

Slide 9

$$\begin{aligned}
&\mathsf{merge} : \quad \mathsf{List}\ \mathsf{Int} \to \quad \mathsf{List}\ \mathsf{Int} \to \mathsf{List}\ \ \mathsf{Int} \\
&\mathsf{msort} : \quad \mathsf{List}\ \mathsf{Int} \to \mathsf{List}\ \ \mathsf{Int} \\
&\mathsf{msort}\ [\,] = [\,] \\
&\mathsf{msort}\ (x :: k) = \mathsf{case}\ k\ \mathsf{of} \\
&\qquad [\,] \to x :: [\,] \\
&\qquad \mid (y :: l) \to \mathsf{let}\ (xs, ys) = \mathsf{split}\ l\ \mathsf{in} \\
&\qquad\qquad \mathsf{merge}\ (\mathsf{msort}\ (x :: xs)) \\
&\qquad\qquad\qquad (\mathsf{msort}\ (y :: ys))
\end{aligned}$$

Example: merge sort

Slide 10

$$\begin{aligned}
&\mathsf{merge} : \forall i : \mathsf{ord}.\ \mathsf{List}^i\ \mathsf{Int} \to \forall j : \mathsf{ord}.\ \mathsf{List}^j\ \mathsf{Int} \to \mathsf{List}^\infty\ \mathsf{Int} \\
&\mathsf{msort} : \forall i : \mathsf{ord}.\ \mathsf{List}^i\ \mathsf{Int} \to \mathsf{List}^\infty\ \mathsf{Int} \\
&\mathsf{msort}\ [\,]^{i+1} = [\,] \\
&\mathsf{msort}\ (x :: k^i) = \mathsf{case}\ k^{j+1=i}\ \mathsf{of} \\
&\qquad [\,] \to x :: [\,] \\
&\qquad \mid (y :: l^j) \to \mathsf{let}\ (xs^j, ys^j) = \mathsf{split}\ l^j\ \mathsf{in} \\
&\qquad\qquad \mathsf{merge}\ (\mathsf{msort}\ (x :: xs)^{j+1=i}) \\
&\qquad\qquad\qquad (\mathsf{msort}\ (y :: ys)^{j+1=i})
\end{aligned}$$

Example: addition for tree ordinals

Slide 11

Datatype Ord of tree ordinals.

$$\begin{array}{lll} \mathsf{Zero} & : & \mathsf{Ord} \\ \mathsf{Succ} & : & \mathsf{Ord} \to \mathsf{Ord} \\ \mathsf{Lim} & : & (\mathsf{Nat} \to \mathsf{Ord}\ ) \to \mathsf{Ord} \end{array}$$

Addition for tree ordinals.

$$\mathsf{add} : \mathsf{Ord} \to \mathsf{Ord} \to \mathsf{Ord}$$

$$\begin{array}{llcl} \mathsf{add}\ x & \mathsf{Zero} & = & x \\ \mathsf{add}\ x & (\mathsf{Succ}\ y\ ) & = & \mathsf{Succ}\ (\mathsf{add}\ x\ y\ ) \\ \mathsf{add}\ x & (\mathsf{Lim}\ f\ \ ) & = & (\mathsf{Lim}\ (\lambda n.\ \mathsf{add}\ x\ (f\ n)\ )) \end{array}$$

Example: addition for tree ordinals

Slide 12

Datatype Ord of tree ordinals.

$$\begin{array}{lll} \mathsf{Zero} & : & \forall i{:}\mathsf{ord}.\ \mathsf{Ord}^{i} \\ \mathsf{Succ} & : & \forall i{:}\mathsf{ord}.\ \mathsf{Ord}^{i} \to \mathsf{Ord}^{i+1} \\ \mathsf{Lim} & : & \forall i{:}\mathsf{ord}.\ (\mathsf{Nat} \to \mathsf{Ord}^{i}) \to \mathsf{Ord}^{i+1} \end{array}$$

Addition for tree ordinals.

$$\mathsf{add} : \mathsf{Ord} \to \forall i{:}\mathsf{ord}.\ \mathsf{Ord}^{i} \to \mathsf{Ord}$$

$$\begin{array}{llcl} \mathsf{add}\ x & \mathsf{Zero} & = & x \\ \mathsf{add}\ x & (\mathsf{Succ}\ y^{i})^{i+1} & = & \mathsf{Succ}\ (\mathsf{add}\ x\ y^{i}) \\ \mathsf{add}\ x & (\mathsf{Lim}\ f^{\cdot \to i})^{i+1} & = & (\mathsf{Lim}\ (\lambda n.\ \mathsf{add}\ x\ (f\ n)^{i})) \end{array}$$

Example: addition for tree ordinals

Slide 13

Datatype Ord of tree ordinals.

$$\begin{aligned} \mathsf{Zero} &: \forall i\,{:}\,\mathsf{ord}.\ \mathsf{Ord}^i \\ \mathsf{Succ} &: \forall i\,{:}\,\mathsf{ord}.\ \mathsf{Ord}^i \to \mathsf{Ord}^{i+1} \\ \mathsf{Lim} &: \forall i\,{:}\,\mathsf{ord}.\ (\mathsf{Nat} \to \mathsf{Ord}^i) \to \mathsf{Ord}^{i+1} \end{aligned}$$

Addition for tree ordinals.

$$\mathsf{add} : \mathsf{Ord}^\infty \to \forall i\,{:}\,\mathsf{ord}.\ \mathsf{Ord}^i \to \mathsf{Ord}^\infty$$

$$\begin{aligned} \mathsf{add}\ x^\infty\ \mathsf{Zero} &= x^\infty \\ \mathsf{add}\ x^\infty\ (\mathsf{Succ}\ y^i)^{i+1} &= \mathsf{Succ}\ (\mathsf{add}\ x\ y^i)^\infty \\ \mathsf{add}\ x^\infty\ (\mathsf{Lim}\ f^{\cdot \to i})^{i+1} &= (\mathsf{Lim}\ (\lambda n.\ \mathsf{add}\ x\ (f\ n)^i))^\infty \end{aligned}$$

Lambda-calculus with subtyping

Slide 14

- Types

$$A, B, C ::= 1 \mid A \times B \mid A + B \mid A \to B$$

- Terms

$$\begin{aligned} r, s, t \ ::= \ & \langle\rangle \mid \langle s, t\rangle \mid \mathsf{fst}\ r \mid \mathsf{snd}\ r \\ & \mid \mathsf{inl}\,t \mid \mathsf{inr}\,t \mid \mathsf{case}\,r\ \mathsf{of}\ \mathsf{inl}\,x \Rightarrow s \mid \mathsf{inr}\,y \Rightarrow t \\ & \mid x \mid \lambda x.t \mid r\ s \mid \mathsf{let}\ x\,{=}\,r\ \mathsf{in}\ t \end{aligned}$$

- Subtyping

$$\frac{}{1 \le 1} \qquad \frac{A_1 \le A_2 \qquad B_1 \le B_2}{A_1 \times B_1 \le A_2 \times B_2} \qquad \frac{A_1 \le A_2 \qquad B_1 \le B_2}{A_1 + B_1 \le A_2 + B_2}$$

$$\frac{A_2 \le A_1 \qquad B_1 \le B_2}{A_1 \to B_1 \le A_2 \to B_2}$$

Polymorphism

- Types

$$A, B, C ::= \dots \mid X \mid \forall X.\, A$$

- Typing

$$\frac{\Gamma, X \vdash t : A}{\Gamma \vdash t : \forall X.\, A} \qquad \frac{\Gamma \vdash t : \forall X.\, A}{\Gamma \vdash t : [B/X]A}$$

Slide 15

- Subtyping

$$\frac{}{X \leq X} \qquad \frac{\Gamma \vdash [C/X]A \leq B}{\Gamma \vdash (\forall X.A) \leq B} \qquad \frac{\Gamma, X \vdash A \leq B}{\Gamma \vdash A \leq \forall X.B}$$

Size polymorphism

- Sizes

$$a, b, c ::= i \mid a + 1 \mid \infty$$

- Ordering

$$\frac{}{i \leq i} \qquad \frac{a \leq b}{a + 1 \leq b + 1} \qquad \frac{a \leq b}{a \leq b + 1} \qquad \frac{}{\infty \leq \infty}$$

Slide 16

- Types

$$A, B, C ::= \dots \mid \forall i.\, A$$

- Typing

$$\frac{\Gamma, i \vdash t : A}{\Gamma \vdash t : \forall i.\, A} \qquad \frac{\Gamma \vdash t : \forall i.\, A}{\Gamma \vdash t : [a/X]A}$$

- Subtyping

$$\frac{\Gamma \vdash [a/X]A \leq B}{\Gamma \vdash (\forall i.A) \leq B} \qquad \frac{\Gamma, i \vdash A \leq B}{\Gamma \vdash A \leq \forall i.B}$$

Inductive types

Slide 17

- Types and terms

$$A, B, C \quad ::= \quad \dots \mid \mu^a X.\, A \qquad \text{where } X \text{ only pos in } A$$
$$r, s, t \quad ::= \quad \dots \mid \mathsf{in}\, t \mid \mathsf{out}\, t$$

- Typing

$$\frac{\Gamma \vdash t : [\mu^a X.A/X]A}{\Gamma \vdash \mathsf{in}\, t : \mu^{a+1} X.A} \qquad \frac{\Gamma \vdash r : \mu^{a+1} X.A}{\Gamma \vdash \mathsf{out}\, r : [\mu^a X.A/X]A}$$

- Subtyping

$$\frac{a \leq b \text{ or } b = \infty \qquad \Gamma, X \vdash A \leq B}{\Gamma \vdash \mu^a X.A \leq \mu^b X.B}$$

- Admissible typing rules

$$\frac{\Gamma \vdash t : [\mu^\infty X.A/X]A}{\Gamma \vdash \mathsf{in}\, t : \mu^\infty X.A} \qquad \frac{\Gamma \vdash r : \mu^\infty X.A}{\Gamma \vdash \mathsf{out}\, r : [\mu^\infty X.A/X]A}$$

Inductive types – example

Slide 18

$$\mathsf{List}^i(A) \quad := \quad \mu^i X.\, 1 + A \times X$$

$$\mathsf{nil} \quad : \quad \forall A \forall i.\, \mathsf{List}^i A$$
$$:= \quad \mathsf{in}(\mathsf{inl}\langle\rangle)$$

$$\mathsf{cons} \quad : \quad \forall A.\, A \to \forall i.\, \mathsf{List}^i(A) \to \mathsf{List}^{i+1}(A)$$
$$:= \quad \lambda a \lambda as.\, \mathsf{in}(\mathsf{inr}\langle a, as\rangle)$$

$$\mathsf{head} \quad : \quad \forall A \forall i.\, \mathsf{List}^{i+1}(A) \to (1 + A)$$
$$\mathsf{head} \quad := \quad \lambda l.\, \mathsf{case}\, (\mathsf{out}\, l)\ \mathsf{of}\ \mathsf{inl}\ \_ \Rightarrow \mathsf{inl}\langle\rangle \mid \mathsf{inr}\, p \Rightarrow \mathsf{inr}(\mathsf{fst}\ p)$$

Could we also type $\mathsf{head} : \forall A \forall i.\, \mathsf{List}^i(A) \to (1 + A)$?

Case distinction for inductive types

Slide 19

- Case distinction on ordinal i:

$$\frac{\Gamma, i, \Gamma' \vdash r : \mu^{i} X.A \qquad \Gamma, j, x : \mu^{j+1} X.A \vdash t : C(j+1)}{\Gamma, i, \Gamma' \vdash \mathsf{let}\ x = r\ \mathsf{in}\ t : C(i)}$$

  where i only pos in $C(i)$.

- Better typing for head:

$$\begin{array}{lcl} \mathsf{head} & : & \forall A \forall i.\, \mathsf{List}^{i}(A) \to (1 + A) \\ \mathsf{head} & := & \lambda l.\, \mathsf{let}\ x^{j+1} = l^{i}\ \mathsf{in} \\ & & \qquad \mathsf{case}\,(\mathsf{out}\, x)\ \mathsf{of}\ \mathsf{inl}\,\_ \Rightarrow \mathsf{inl}\langle\rangle\ \mid\ \mathsf{inr}\, p \Rightarrow \mathsf{inr}(\mathsf{fst}\ p) \end{array}$$

- Case distinction on ordinal could be integrated into case construct.

Infinite structures

Slide 20

- On infinite objects like streams, we are interested in the *definedness* rather than the size.
- $s : \mathsf{Stream}^{a}(A)$ means s is defined upto depth a.
- Objects which are defined upto depth ∞ are called *productive*.
- Subtyping for streams:

$$\mathsf{Stream}^{\infty}(A) \leq \ldots \mathsf{Stream}^{i+1}(A) \leq \mathsf{Stream}^{i}(A)$$

Coinductive types

Slide 21

- Types

$$A, B, C ::= \ldots \mid \nu^a X.\, A \qquad \text{where } X \text{ only pos in } A$$

- Typing

$$\frac{\Gamma \vdash t : [\nu^a X.A/X]A}{\Gamma \vdash \mathsf{in}\, t : \nu^{a+1} X.A} \qquad \frac{\Gamma \vdash r : \nu^{a+1} X.A}{\Gamma \vdash \mathsf{out}\, r : [\nu^a X.A/X]A}$$

- Subtyping

$$\frac{a \leq b \text{ or } b = \infty \qquad \Gamma, X \vdash A \leq B}{\Gamma \vdash \nu^b X.A \leq \nu^a X.B}$$

- Admissible typing rules

$$\frac{\Gamma \vdash t : [\nu^\infty X.A/X]A}{\Gamma \vdash \mathsf{in}\, t : \nu^\infty X.A} \qquad \frac{\Gamma \vdash r : \nu^\infty X.A}{\Gamma \vdash \mathsf{out}\, r : [\nu^\infty X.A/X]A}$$

Coinductive types – example

Slide 22

$\mathsf{Stream}^i(A)$	$:=$	$\nu^i X.\, A \times X$
s_cons	$:$	$\forall A.\, A \to \forall i.\, \mathsf{Stream}^i(A) \to \mathsf{Stream}^{i+1}(A)$
	$:=$	$\lambda a \lambda as.\, \mathsf{in}\langle a, as\rangle$
s_head	$:$	$\forall A \forall i.\, \mathsf{Stream}^{i+1}(A) \to A$
s_head	$:=$	$\lambda s.\ \mathsf{fst}\,(\mathsf{out}\, s)$
s_tail	$:$	$\forall A \forall i.\, \mathsf{Stream}^{i+1}(A) \to \mathsf{Stream}^i(A)$
s_tail	$:=$	$\lambda s.\ \mathsf{snd}\,(\mathsf{out}\, s)$

Recursion and corecursion

Slide 23

- Terms

$$r, s, t ::= \ldots \mid \mathsf{fix}^{\mu}\ s \mid \mathsf{fix}^{\nu}\ s$$

- Typing

$$\frac{\Gamma, i \vdash s : ((\mu^{i} X.A) \to B(i)) \to (\mu^{i+1} X.A) \to B(i+1)}{\Gamma \vdash \mathsf{fix}^{\mu}\ s : \forall i.\, (\mu^{i} X.A) \to B(i)} \quad i\ \cap\text{-}\mathsf{cont}\ B(i)$$

$$\frac{\Gamma, i \vdash s : A(i) \to A(i+1)}{\Gamma \vdash \mathsf{fix}^{\nu}\ s : \forall i.\, A(i)} \quad i\ \mathsf{legal}^{\nu}\ A(i)$$

- Legal types for corecursion

$$\frac{}{i\ \mathsf{legal}^{\nu}\ \nu^{i} X.A}\ (i \notin A) \qquad \frac{i\ \mathsf{legal}^{\nu}\ A \qquad i\ \mathsf{legal}^{\nu}\ B}{i\ \mathsf{legal}^{\nu}\ A \times B}$$

$$\frac{i\ \mathsf{only\ pos\ in}\ A \qquad i\ \mathsf{legal}^{\nu}\ B}{i\ \mathsf{legal}^{\nu}\ A \to B}$$

Corecursion example: sequence of natural numbers

Slide 24

- Map for streams in sugared recursion syntax:

  $\mathsf{s\_map} : \forall X \forall Y.\, (X \to Y) \to \forall i.\, \mathsf{Stream}^{i}(X) \to \mathsf{Stream}^{i}(Y)$

  $\mathsf{s\_map}\ f\ (x :: xs^{i})^{i+1} = ((f\ x) :: \mathsf{s\_map}\ f\ xs^{i})^{i+1}$

- Stream of natural numbers in orginal recursion syntax:

  $\mathsf{nats} : \forall i.\, \mathsf{Stream}^{i}(\mathsf{Int})$

  $\mathsf{nats} = \mathsf{fix}^{\nu}\ \lambda nats.\, (0 :: (\mathsf{s\_map}\ +1\ nats^{i})^{i})^{i+1}$

Reduction

Slide 25

- Special evaluation contexts:

$$E ::= \bullet \mid E\, s \mid \mathsf{fst}\ E \mid \mathsf{snd}\ E$$

- Rules for (co)inductive data and (co)recursion:

$$\begin{array}{lll} \mathsf{out}\,(\mathsf{in}\,t) & \longrightarrow_\beta & t \\ \mathsf{fix}^\mu\ s\,(\mathsf{in}\,t) & \longrightarrow_\beta & s\,(\mathsf{fix}^\mu\ s)\,(\mathsf{in}t) \\ \mathsf{out}\,(E[\mathsf{fix}^\nu\ s]) & \longrightarrow_\beta & \mathsf{out}\,(E[s\,(\mathsf{fix}^\nu\ s)]) \end{array}$$

- Add β-rules for the lambda-calculus with sums and products plus congruence rules.
- For the resulting reduction relation we can show strong normalization.

Semantics of size expressions

Slide 26

Let θ be a mapping of size variables into ordinals $< \Omega + \omega$.

$$\begin{array}{lll} [\![i]\!]\theta & = & \theta(i) \\ [\![a+1]\!]\theta & = & [\![a]\!]+1 \\ [\![\infty]\!]\theta & = & \Omega \end{array}$$

Semantics of types

- Let $\Gamma \vdash A : *$ and $\theta : \Gamma$ a valuation of the free type and size variables in A.
- Semantics $[\![A]\!]\theta \subseteq \mathsf{SN}$ (saturated set):

Slide 27

$$
\begin{array}{lcl}
[\![A \to B]\!]\theta & = & \{r \mid r\,s \in [\![B]\!]\theta \text{ for all } s \in [\![A]\!]\theta\} \\
[\![\mathsf{List}^{i}(A)]\!]\theta & = & \Phi^{\alpha}_{\mathsf{List}(A),\theta}(\emptyset) \\
[\![\mathsf{Stream}^{i}(A)]\!]\theta & = & \Phi^{\alpha}_{\mathsf{Stream}(A),\theta}(\mathsf{SN})
\end{array}
$$

with ordinal $\alpha = [\![i]\!]\theta$ and

$$
\begin{array}{lcl}
\Phi_{\mathsf{List}(A),\theta}(Q) & = & \{\mathsf{nil}, \mathsf{cons}\,s\,t \mid s \in [\![A]\!]\theta,\ t \in Q\} \\
\Phi_{\mathsf{Stream}(A),\theta}(Q) & = & \{r \mid \mathsf{s\_head}\,r \in [\![A]\!]\theta,\ \mathsf{s\_tail}\,r \in Q\}
\end{array}
$$

Uniform Operation Iteration

- Iterates Φ^{α} can be defined uniformly for least and greatest fixed-points using limes inferior. Let $P : \mathsf{On} \to \mathcal{P}(\mathsf{SN})$.

$$
\begin{array}{lcl}
\underline{\lim}_{\alpha \to \lambda} P(\alpha) & = & \bigcup_{\alpha_0 < \lambda} \bigcap_{\alpha_0 \leq \alpha < \lambda} P(\alpha) \\
\Phi^{0}(Q) & = & Q \\
\Phi^{\alpha+1}(Q) & = & \Phi(\Phi^{\alpha}(Q)) \\
\Phi^{\lambda}(Q) & = & \underline{\lim}_{\alpha \to \lambda} \Phi^{\alpha}(Q)
\end{array}
$$

Slide 28

- Lemma: Assume Φ increasing, i.e., $\Phi^{\alpha}(Q) \subseteq \Phi^{\beta}(Q)$ for $\alpha \leq \beta$.
$$\Phi^{\lambda}(Q) = \bigcup_{\alpha < \lambda} \Phi^{\alpha}(Q)$$
- Lemma: Assume Φ decreasing, i.e., $\Phi^{\alpha}(Q) \supseteq \Phi^{\beta}(Q)$ for $\alpha \leq \beta$.
$$\Phi^{\lambda}(Q) = \bigcap_{\alpha < \lambda} \Phi^{\alpha}(Q)$$

On the side condition on recursion

Slide 29

- Recall the recursion rule:

$$\frac{\Gamma, i \vdash s : ((\mu^{i} X.A) \to B(i)) \to (\mu^{i+1} X.A) \to B(i+1)}{\Gamma \vdash \mathsf{fix}^{\mu}\, s : \forall i.\, (\mu^{i} X.A) \to B(i)} \quad i\ \cap\text{-}\mathsf{cont}\ B(i)$$

- Soundness is proven by transfinite induction on the ordinal $[\![i]\!]$. For the limit case to hold, B must admit

$$\begin{array}{rl} \underline{\lim}_{\alpha\to\lambda} & [\![(\mu^{i} X.A) \to B(i)]\!](i \mapsto \alpha) \\ \subseteq & [\![(\mu^{i} X.A) \to B(i)]\!](i \mapsto \lambda) \end{array}$$

- Thus, result type of this fix^{μ}-construction must be *continuous* in i.
- We distinguish two kinds of continuity.

$\cup$-Continuity

Slide 30

- A set-valued function $P : \mathsf{On} \to \mathcal{P}(\mathsf{SN})$ is called $\cup$-continuous if

$$P(\lambda) \subseteq \underline{\lim}_{\alpha\to\lambda} P(\alpha)$$

  Hughes, Pareto & Sabry (1996) call P *overshooting*.

- Grammar for $\cup$-continuous types $i\ \cup\text{-}\mathsf{cont}\ A$:

$$\frac{i\ \mathsf{only\ neg\ in}\ A}{i\ \cup\text{-}\mathsf{cont}\ A} \qquad \frac{i\ \cup\text{-}\mathsf{cont}\ A, B}{i\ \cup\text{-}\mathsf{cont}\ A + B, A \times B} \qquad \frac{i\ \cup\text{-}\mathsf{cont}\ A}{i\ \cup\text{-}\mathsf{cont}\ \mathsf{List}^{a}(A)}$$

- Theorem: If $i\ \cup\text{-}\mathsf{cont}\ A$ then $[\![A]\!](i)$ is $\cup$-continuous.

$\cap$-Continuity

- A set-valued function $P : \mathsf{On} \to \mathcal{P}(\mathsf{SN})$ is called $\cap$-continuous if

$$\varliminf_{\alpha \to \lambda} P(\alpha) \subseteq P(\lambda)$$

Hughes, Pareto & Sabry (1996) call P *undershooting*.

Slide 31

- Grammar for $\cap$-continuous types $i\ \cap\text{-}\mathsf{cont}\ A$:

$$\frac{i\ \mathsf{only\ pos\ in}\ A}{i\ \cap\text{-}\mathsf{cont}\ A} \qquad \frac{i\ \cap\text{-}\mathsf{cont}\ A, B}{i\ \cap\text{-}\mathsf{cont}\ A + B, A \times B} \qquad \frac{i\ \cap\text{-}\mathsf{cont}\ A}{i\ \cap\text{-}\mathsf{cont}\ \mathsf{List}^{a}(A)}$$

$$\frac{i\ \cup\text{-}\mathsf{cont}\ A \qquad i\ \cap\text{-}\mathsf{cont}\ B}{i\ \cap\text{-}\mathsf{cont}\ A \to B} \qquad \frac{i\ \cap\text{-}\mathsf{cont}\ A}{i\ \cap\text{-}\mathsf{cont}\ \mathsf{Stream}^{a}(A)}$$

- Theorem: If $i\ \cap\text{-}\mathsf{cont}\ A$ then $[\![A]\!](i)$ is $\cap$-continuous.

Work in progress: F^{ω} with sized types

Slide 32

- Kinds.

κ	$::=$	$*$	types
	$\mid$	ord	ordinal sizes
	$\mid$	$\kappa \xrightarrow{+} \kappa'$	covariant type constructors
	$\mid$	$\kappa \xrightarrow{-} \kappa'$	contravariant type constructors
	$\mid$	$\kappa \xrightarrow{0} \kappa'$	invariant type constructors

- "Subconstructors" $F \leq G : \kappa$. E.g.,

$$\frac{X \leq Y : \kappa \vdash F\,X \leq G\,Y : \kappa'}{F \leq G : \kappa \xrightarrow{+} \kappa'}$$

- Well-kindedness definable by $F : \kappa \iff F \leq F : \kappa$

Inductive constructors

- Inductive constructors.

$$\mu_\kappa : \mathsf{ord} \xrightarrow{+} (\kappa \xrightarrow{+} \kappa) \xrightarrow{+} \kappa$$

- Example for an inductive type:

Slide 33

$$\mathsf{List} = \lambda i \lambda A.\ \mu_* i\,(\lambda X.\, 1 + A \times X)$$

- Inductive functors: μ_κ for $\kappa = * \to *$.
- E.g., $\mathsf{Term}\,A$, de Bruijn terms with free variables in A:

$$\mathsf{Term} = \mu_{*\to*} \infty \lambda T \lambda A.\, A + T(1 + A) + TA \times TA$$

Conclusions

Sized types:

- Conceptually *lean* way of ensuring termination.
- Well-typedness ensures termination.
- No external static analysis required.

Slide 34

System F^ω:

- Size expressions can be integrated into constructors.
- Sized types scale to higher-order polymorphism.

Goal: extend to dependent types.

Related Work

- Hughes, Pareto, Sabry (1996)
  Proving the correctness of reactive system using sized types
- Barthe, Frade, Giménez, Pinto, Uustalu (2003?)
  Type-based termination of recursive definitions
- Buchholz (2003?)
  Recursion on nonwellfounded trees

Slide 35