Towards a Total Dependently Typed Core Language

Andreas Abel

Department of Computer Science
Ludwig-Maximilians-University Munich

Dependently Typed Programming Workshop (DTP 2011)
Shonan Village Center
near Tokyo, Japan
14 September 2011

Core Language for Agda

- Small language which can express Agda's features
  1. (Co)inductive families
  2. Dependent pattern matching
  3. Coverage
  4. Universe stratification
  5. Termination
- Small Trusted Code Base
- Explaining induction-recursion, induction-induction, ...
- Input for backend (compiler)

From PiSigma to TypeCore

- Dependent functions Π
- Dependent pairs Σ
- Atoms (labels) $\mathsf{Bool} = \{'\mathsf{true}, '\mathsf{false}\}$
- ~~Case~~ Non-dependent pattern matching
- Equality
- ~~General recursion~~ Well-founded recursion
- ~~Type : Type~~ Stratified universes with polymorphism
- Sizes and measures
- Higher-order subtyping
- Bounded size quantification $\forall i < j$, $\exists i < j$

Simple Types

- Unit type $1 = \{'\mathsf{unit}\}$
- Strict product $A \otimes B = \Sigma\_ : A.\ B$
- Lazy product $A \times B$ (using left application $a \triangleright f := f\ a$)

$$\begin{array}{lcl} A \times B & = & (x : \mathsf{Bool}) \to x \triangleright \lambda \{\ '\mathsf{true} \mapsto A \\ & & \qquad ;\ '\mathsf{false} \mapsto B\ \} \\ \mathsf{pair}\ a\ b & = & \lambda \{\ '\mathsf{true} \mapsto a \\ & & \qquad ;\ '\mathsf{false} \mapsto b\ \} \end{array}$$

- Disjoint sum $A + B$

$$\begin{array}{lcl} A + B & = & \Sigma x : \mathsf{Bool}.\ x \triangleright \lambda \{\ '\mathsf{true} \mapsto A \\ & & \qquad ;\ '\mathsf{false} \mapsto B\ \} \\ \mathsf{inl}\ a & = & ('\mathsf{true}, a) \\ \mathsf{inr}\ b & = & ('\mathsf{false}, b) \end{array}$$

Least Fixpoints

- Example (natural numbers): $F\,X = 1 + X$
- Inflationary iteration:

$$\mu^i F = \bigcup_{j<i} F^j(\mu^j F)$$

- Least fixed-point:

$$\mu F = \bigcup_{i} \mu^i F$$

- Special cases for monotone F:

$$\begin{array}{lcl} \mu^0 \;\; F & = & \{\} \\ \mu^{i+1}\, F & = & F^i\,(\mu^i F) \end{array}$$

Sizes and Bounded Quantification

- $0, 1, \cdots : \mathsf{Size}$ and successor $\_ + 1$
- $\Gamma \vdash a < b$ size comparison
- Bounded universal

$$\frac{\Gamma, i < b \vdash t : C}{\Gamma \vdash \lambda i \mapsto t : \forall i < b.\ C}\ \forall^{<} I \qquad \frac{\Gamma \vdash t : \forall i < b.\ C \qquad \Gamma \vdash a < b}{\Gamma \vdash t\, a : C[a/i]}\ \forall^{<} E$$

- Bounded existential

$$\frac{\Gamma, i < b, x : A \vdash t : C}{\Gamma, (i, x) : \exists i < b.\ A \vdash t : C}\ \exists^{<} L \qquad \frac{\Gamma \vdash a < b \qquad \Gamma \vdash t : C[a/i]}{\Gamma \vdash (a, t) : \exists i < b.\ C}\ \exists^{<} R$$

Higher-Order Subtyping

- Union is covariant:

$$\frac{\Gamma, i : \mathsf{Size} \vdash C : \mathsf{Set}}{\Gamma \vdash \exists i < \_.\ C : +\mathsf{Size} \to \mathsf{Set}}$$

$$\frac{\Gamma \vdash F : +\mathsf{Size} \to C \qquad \Gamma \vdash^{a \leq b}}{\Gamma \vdash F\, a \leq F\, b : C}$$

- Intersection is contravariant:

$$\frac{\Gamma, i : \mathsf{Size} \vdash C : \mathsf{Set}}{\Gamma \vdash \forall i < \_.\ C : -\mathsf{Size} \to \mathsf{Set}}$$

$$\frac{\Gamma \vdash F : -\mathsf{Size} \to C \qquad \Gamma \vdash^{a \leq b}}{\Gamma \vdash F\, b \leq F\, a : C}$$

Natural numbers

$$
\begin{array}{lcl}
\hat{\mathbb{N}}\mathrm{at} & : & +\mathsf{Size} \to \mathsf{Set} \\
\hat{\mathbb{N}}\mathrm{at}\, i & = & \exists j < i.\ 1 + \hat{\mathbb{N}}\mathrm{at}\, j \\
\hat{z}\mathrm{ero} & : & \forall i.\ \hat{\mathbb{N}}\mathrm{at}\,(i+1) \\
\hat{z}\mathrm{ero}\, i & = & (i, \mathsf{inl}\,'\mathsf{unit}) \\
\hat{s}\mathrm{ucc} & : & \forall i.\ \hat{\mathbb{N}}\mathrm{at}\, i \to \hat{\mathbb{N}}\mathrm{at}\,(i+1) \\
\hat{s}\mathrm{ucc}\, i\, n & = & (i, \mathsf{inr}\, n)
\end{array}
\qquad
\begin{array}{lcl}
\mathbb{N}\mathrm{at} & : & \mathsf{Set} \\
\mathbb{N}\mathrm{at} & = & \exists i.\ \hat{\mathbb{N}}\mathrm{at}\, i \\
\mathrm{zero} & : & \mathbb{N}\mathrm{at} \\
\mathrm{zero} & = & (1, \hat{z}\mathrm{ero}\, 0) \\
\mathrm{succ} & : & \mathbb{N}\mathrm{at} \to \mathbb{N}\mathrm{at} \\
\mathrm{succ}\,(i, n) & = & (i+1, \hat{s}\mathrm{ucc}\, i\, n)
\end{array}
$$

$$
\begin{array}{lcl}
\mathrm{double} & : & \forall i.\ \hat{\mathbb{N}}\mathrm{at}\, i \to \mathbb{N}\mathrm{at} \\
\mathrm{double}\, i\, (j^{<i}, \mathsf{inl}\ \_) & = & \mathrm{zero} \\
\mathrm{double}\, i\, (j^{<i}, \mathsf{inr}\, n) & = & \mathrm{succ}\,(\mathrm{succ}\,(\mathrm{double}\, j\, n))
\end{array}
$$

Termination Measures

- Making termination argument explicit:

$$\begin{array}{lcl} \mathsf{double} & : & \forall i.\ |i| \Rightarrow \hat{\mathsf{N}}\mathsf{at}\ i \to \mathsf{Nat} \\ \mathsf{double}\ i\ (j^{<i}, \mathsf{inl}\ \_) & = & \mathsf{zero} \\ \mathsf{double}\ i\ (j^{<i}, \mathsf{inr}\ n) & = & \mathsf{succ}\ (\mathsf{succ}\ (\mathsf{double}\ j\ n)) \end{array}$$

- Termination problem is type-checking problem:

$$\frac{\begin{array}{lcl} i & : & \mathsf{Size} \\ \mathsf{double} & : & \forall j < i.\ \hat{\mathsf{N}}\mathsf{at}\ j \to \mathsf{Nat} \\ (j, \mathsf{inr}\ n) & : & \hat{\mathsf{N}}\mathsf{at}\ i \end{array}}{\mathsf{succ}\ (\mathsf{succ}\ (\mathsf{double}\ j\ n) : \mathsf{Nat}}$$

- Extends to lexicographic measures.

Induction-recursion is just mutual recursion

- A universe of Nat and Π.

$$
\begin{array}{lcl}
\hat{\mathsf{U}} & : & +\mathsf{Size} \to \mathsf{Set} \\
\hat{\mathsf{U}}\, i & = & \exists j < i.\ 1 + \Sigma A : \hat{\mathsf{U}}\, j.\ \hat{\mathsf{El}}\, j\, A \to \hat{\mathsf{U}}\, j \\
 & & \\
\hat{\mathsf{El}} & : & \forall i.\ \hat{\mathsf{U}}\, i \to \mathsf{Set} \\
\hat{\mathsf{El}}\, i\, (j^{<i}, \mathsf{inl}\, \mathit{'unit}) & = & \mathsf{Nat} \\
\hat{\mathsf{El}}\, i\, (j^{<i}, \mathsf{inr}\, (A, B)) & = & (x : \hat{\mathsf{El}}\, j\, A) \to \hat{\mathsf{El}}\, j\, (B\, x)
\end{array}
$$

Greatest Fixpoints

- Example (streams): $F\,X = A \times X$
- Deflationary iteration:

$$\nu^i F = \bigcap_{j<i} F^j(\nu^j F)$$

- Greatest fixed-point:

$$\nu F = \bigcap_i \nu^i F$$

- Special cases for monotone F:

$$\begin{array}{lcl} \nu^0 \quad F & = & \top \\ \nu^{i+1}\,F & = & F^i\,(\nu^i F) \end{array}$$

Streams

$$
\begin{array}{lcl}
\hat{\mathsf{Stream}} & : & -\mathsf{Size} \to +\mathsf{Set} \to \mathsf{Set} \\
\hat{\mathsf{Stream}}\, i\, A & = & \forall j < i.\ A \times \hat{\mathsf{Stream}}\, j\, A \\
\hat{\mathsf{head}} & : & \hat{\mathsf{Stream}}\, i\, A \to \forall j < i.\ A \\
\hat{\mathsf{head}}\, s\, j & = & s\, j\, \mathsf{true} \\
\hat{\mathsf{tail}} & : & \hat{\mathsf{Stream}}\, i\, A \to \forall j < i.\ \hat{\mathsf{Stream}}\, j\, A \\
\hat{\mathsf{tail}}\, s\, j & = & s\, j\, \mathsf{false} \\
\hat{\mathsf{cons}} & : & A \to \hat{\mathsf{Stream}}\, i\, A \to \hat{\mathsf{Stream}}\, (i+1)\, A \\
\hat{\mathsf{cons}}\, a\, s\, j^{<i+1} & = & \mathsf{pair}\, a\, s
\end{array}
$$

Streams (Limit)

$$
\begin{array}{lcl}
\mathsf{Stream} & : & +\mathsf{Set} \to \mathsf{Set} \\
\mathsf{Stream}\ A & = & \forall i.\ \hat{\mathsf{S}}\mathsf{tream}\ i\ A \\
\\
\mathsf{head} & : & \mathsf{Stream}\ A \to A \\
\mathsf{head}\ s & = & \hat{\mathsf{h}}\mathsf{ead}\ (s\ 1)\ 0 \\
\\
\mathsf{tail} & : & \mathsf{Stream}\ A \to \mathsf{Stream}\ A \\
\mathsf{tail}\ s\ i & = & \hat{\mathsf{t}}\mathsf{ail}\ (s\ (i+1))\ i \\
\\
\mathsf{cons} & : & A \to \mathsf{Stream}\ A \to \mathsf{Stream}\ A \\
\mathsf{cons}\ a\ s\ i & = & \mathsf{cons}\ a\ (s\ i)
\end{array}
$$

Programming Streams

- Define streams by their projections:

$$
\begin{array}{lcl}
\mathsf{map} & : & (A \to B) \to \forall i.\ \hat{\mathsf{S}}\mathsf{tream}\ i\ A \to \hat{\mathsf{S}}\mathsf{tream}\ i\ B \\
\mathsf{map}\ f\ i\ s\ j\ '\mathsf{true} & = & f\ (s\ j\ '\mathsf{true}) \\
\mathsf{map}\ f\ i\ s\ j\ '\mathsf{false} & = & \mathsf{map}\ f\ j\ (s\ j\ '\mathsf{false})
\end{array}
$$

Conclusions

- Expressive language with small implementation:
  1. Type checking.
  2. Size comparison.
  3. Higher-order subtyping and equality checking.
  4. No extra termination/positivity checker.
  5. No semi-coninuity check.
- Consistent, but not strongly normalizing.
- TODO:
  - Reduction strategy (cf. PiSigma).
  - Implementation: Evolve MiniAgda into TypeCore.
  - Correctness by model construction.

References

- PiSigma: Altenkirch, Danielsson, Loeh, Oury (FLOPS 2010)
- Sized types in typed programming:
  - Hughes, Pareto, and Sabry (POPL 1996)
  - Barthe et. al. (MSCS 2004, LPAR 2006)
  - Blanqui (RTA 2004, CSL 2005)
  - Abel (LMCS 2008, **PAR-10**)
- Approximations in μ-calculus:
  - Schöpp (FoSSaCS 2002)
  - Dam and Sprenger (FoSSaCS 2003)
  - Brotherston (PhD 2007)