On Shape Irrelevance and Polymorphism in Type Theory

Andreas Abel
(joint work in progress with Gabriel Scherer)

Department of Computer Science
Ludwig-Maximilians-University Munich

14th Agda Intensive Meeting (AIM XIV)
Shonan Village Center
near Tokyo, Japan
9 September 2011
**Type systems for computational irrelevance**

- Separate computationally relevant parts from “administrative” (computationally irrelevant) parts.
- Used for:
  1. Extracting programs
  2. Strengthening equational theory (ignore irrelevant parts during equality checking)
  3. Pruning terms, reducing memory footprint
- Kinds of irrelevance:
  1. Proof arguments (like $x \neq 0$ in division)
  2. Type arguments (like $A$ in `append A xs ys`)
  3. “Forced” arguments (like $n$ in `vcons A n x xs`)
  4. Termination evidence: universe levels and sizes

Andreas Abel (joint work in progress with Gabriel Scherer) (LMU)
The Agda Pipeline

**Frontend**
- Parsing
- Scope checking
- Abstract syntax
- Type reconstruction
- Type checker
- Internal erasure
- Termination inference
- Internal syntax
- Extraction
- Optimizations
- Backend
- Code generation

Andreas Abel (joint work in progress with Gabriel Scherer) (LMU)

Shape Irrelevance
### Internal Erasure vs. Extraction

<table>
<thead>
<tr>
<th></th>
<th>Erasure</th>
<th>Extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free vars</td>
<td>yes (terms)</td>
<td>no (programs)</td>
</tr>
<tr>
<td>Evaluation</td>
<td>under binder</td>
<td>funs = black boxes</td>
</tr>
<tr>
<td>Equality</td>
<td>intensional ($\beta\eta$), decid.</td>
<td>observational, undecid.</td>
</tr>
<tr>
<td>Empty types</td>
<td>hypothetical inhabitants</td>
<td>no closed inhabitants</td>
</tr>
<tr>
<td>Identity proofs</td>
<td>relevant</td>
<td>irrelevant ($=\text{refl}$)</td>
</tr>
<tr>
<td>Accessibility proofs</td>
<td>relevant (termination!)</td>
<td>irrelevant</td>
</tr>
</tbody>
</table>

Extraction can erase all inhabitants of *propositional types*, i.e., with at most one closed inhabitant.
ICC* and EPTS

- Barras and Bernardo (FoSSaCS 2008) and Sheard and Mishra-Linger (FoSSaCS 2008)

\[
\Gamma ⊢ A : \text{Set} \quad \Gamma, \ x : A ⊢ B : \text{Set} \\
\Gamma ⊢ [x : A] → B : \text{Set}
\]

no rule for \([x : A] ∈ \Gamma\)

\[
\Gamma, \ [x : A] ⊢ t : B \\
\Gamma ⊢ λxt : [x : A] → B
\]

\[
\Gamma ⊢ r : [x : A] → B \quad \Gamma ⊢ s : A \\
\Gamma ⊢ rs : B[s/x]
\]

- Resurrection \((-)⊕\) (Pfenning 2001) turns irrelevant assumptions \([x : A]\) into relevant ones \((x : A)\).

- Irrelevant function argument can be relevant in function codomain.

\[
\lambda A \lambda x. x : [A : \text{Set}] → A ⊸ A
\]
Equality in ICC* 

- Equality in ICC* is untyped $\beta\eta$ after erasure.
- Does not scale to typed $\eta$-equality with unit type $\top$ in the presence of large eliminations.
- Given $h : \texttt{[A : Set]} \rightarrow (A \rightarrow A) \rightarrow \texttt{Bool}$, then?

$$h (\mathbb{N} \rightarrow \mathbb{N}) (\lambda x \lambda y. x y) = h \top (\lambda x. ()) : \texttt{Bool}$$

- Algorithm would check heterogeneous $\lambda y. x y : \mathbb{N} \rightarrow \mathbb{N} = () : \top$?
- But then $t : A = () : \top = t' : A'$, inconsistent!
Irrelevance in Agda 2.2.10

- Irrelevant function arguments need to be irrelevant in codomain.

\[
\Gamma \vdash A : \text{Set} \quad \Gamma, .(x : A) \vdash B : \text{Set} \\
\Gamma \vdash .(x : A) \rightarrow B : \text{Set}
\]

- Type \( .(A : \text{Set}) \rightarrow A \rightarrow A \) ill-formed.
- Equality is typed \( \beta \eta \), ignoring irrelevant arguments.
- No need for heterogeneous equality.
Shape-directed $\eta$-equality

- $\eta$-laws are applied according to the type shape: function type, record type (e.g., unit type), other type.
- Exact type not necessary.
- Exploited by Harper/Pfenning’s simply-typed equality check for LF.
- More subtle with large eliminations!

\[
T : \text{Bool} \to \text{Set} \\
T \text{true} = \top \\
T \text{false} = \mathbb{N} \to \mathbb{N}
\]

- Shape of $T u$ depends on value of $u$. 

Andreas Abel (joint work in progress with Gabriel Scherer) (LMU)
Shape-Irrelevance

- A function is **shape-irrelevant** if the value of the argument does not influence the (deep) shape of the result.
- Prime example: Data type constructors.

\[
\begin{align*}
\text{List} & \quad : \quad ^{(A: \text{Set})} \rightarrow \text{Set} \\
\text{Vec} & \quad : \quad ^{(A: \text{Set})} \rightarrow ^{(n: \mathbb{N})} \rightarrow \text{Set} \\
\Sigma & \quad : \quad ^{(A: \text{Set})} \rightarrow ^{(B: A \rightarrow \text{Set})} \rightarrow \text{Set}
\end{align*}
\]

- Parameters in data constructors and projections are **irrelevant**!

\[
\begin{align*}
\text{nil} & \quad : \quad .(A: \text{Set}) \rightarrow \text{List } A \\
\text{vcons} & \quad : \quad .(A: \text{Set}) \rightarrow .(n: \mathbb{N}) \rightarrow A \rightarrow \text{Vec } A \ n \rightarrow \text{Vec } A \ (\text{suc } n) \\
\_ & , \_ & \quad : \quad .(A: \text{Set}) \rightarrow .(B: A \rightarrow \text{Set}) \rightarrow (a: A) \rightarrow B \ a \rightarrow \Sigma \ A \ B
\end{align*}
\]
Typing rules for irrelevance and shape-irrelevance

- Function classifier \( p \ ::= . \) irrelevant function
  \( \hat{\,} \) shape-irrelevant function
  \( \epsilon \) ordinary (relevant) function

- “Going types” \( p^\oplus \) turns “.” into “\( \hat{\,} \) ”.

\[
\begin{align*}
\Gamma \vdash A : \text{Set} & \quad \Gamma, p^\oplus(x:A) \vdash B : \text{Set} \\
\Gamma \vdash p(x:A) \to B : \text{Set} & \quad \Gamma, p(x:A) \vdash t : B \\
\Gamma \vdash \lambda x t : p(x:A) \to B & \\
\Gamma \vdash \lambda x t : p(x:A) \to B \\
\end{align*}
\]

\[
\begin{align*}
(x:A) \in \Gamma & \quad \Gamma \vdash r : p(x:A) \to B \\
\Gamma \vdash r s : B[s/x] & \\
\Gamma \vdash x : A & \quad \Gamma^p \vdash s : A \\
\Gamma \vdash r s : B[s/x] & \\
\end{align*}
\]

\[
\begin{align*}
\Gamma^p \vdash A = B : \text{Set} & \quad \Gamma^\oplus \vdash A = B : \text{Set} \\
\Gamma \vdash t : A & \quad \Gamma \vdash t : B \\
\end{align*}
\]

- \( \Gamma^p \) makes assumptions available that are at least as relevant as \( p \).
Examples

- Type of \( \text{nil} : (A : \text{Set}) \rightarrow \text{List} A \) well-formed

\[
\text{List} : \hat{(A : \text{Set})} \rightarrow \text{Set} \quad A : \text{Set} \vdash A : \text{Set} \\
\hat{(A : \text{Set})} \vdash \text{List} A \\
\vdash (A : \text{Set}) \rightarrow \text{List} A
\]

- Irrelevantly quantified variables may appear shape-irrelevantly in codomain. Then, \( \text{List} \)'s argument is shape-irrelevant.

- Universe-polymorphic lists \( \text{UList} : (i : \text{Level}) \rightarrow \hat{(A : \text{Set} i)} \rightarrow \text{Set} i \).
Shape-directed equality

- Equality judgement $\Gamma \vdash t = t' : A$ relaxed: $A$ is the common shape of $t$ and $t'$.

\[
\Gamma \vdash t = t' : (x : A) \rightarrow B \\
\Gamma \vdash t u = t' u' : B[u/x]
\]

- Note: $\Gamma, \hat{(x : A)} \vdash B$, hence $B[u/x]$ and $B[u'/x]$ same shape.

\[
\Gamma \vdash t = t' : \hat{(x : A)} \rightarrow B \\
\Gamma \hat{\vdash} u = u' : A \\
\Gamma \vdash t u = t' u' : B[u/x]
\]
Unification

- Unification finds parameters.

\[
\begin{align*}
\text{nil }_1 \Rightarrow & \text{ List }_1 \\
\text{List }_1 = & A \\
\text{List }_1 = & \text{List } A \\
\text{nil }_1 \iff & \text{List } A
\end{align*}
\]

- Irrelevant parameters are not uniquely determined

\[
\begin{align*}
\text{nil }_1 \_2 \Rightarrow & \text{UList }_1 \_2 \\
\text{UList }_1 \_2 = & A : \text{Set } i \\
\text{no eq. for }_1 \\
\text{UList }_1 \_2 = & \text{UList } i A \\
\text{nil }_1 \_2 \iff & \text{UList } i A
\end{align*}
\]
Related Work

1. Proof Irrelevance in LF (Pfenning, Reed)
2. Bracket Types (Awodey, Bauer)
3. Uniform quantification (Berger, Schwichtenberg)
4. Program extraction in Coq (Paulin-Mohring, Letouzey)
5. Implicit Calculus of Constructions (Miquel, Barras, Bernardo)
6. Erasure Pure Type Systems (Mishra-Linger, Sheard)
7. Lightning (Brady, McBride)