Extraction of Programs from Proofs using Postulated Axioms
or
Ideal and Concrete Objects in Type Theory

Anton Setzer

Swansea University, Swansea UK
(Joint work with Chi Ming Chuang)

10 October 2011

Main Type Theoretic Setting

- We use a variant of Martin-Löf Type Theory based essentially on the syntax of the theorem prover Agda.
  - Abstract formulation work in progress.
- In Martin-Löf Type Theory types denoted by keyword Set.
- Three main constructs:
  - dependent function types,
  - algebraic data types,
  - coalgebraic data types.

Dependent Function Types

- $$(x : A) \to B$$

  type of functions mapping $a : A$ to an element of type $B[x := a]$.
- Essentially $\Pi x : A.B$ (subtle differences).
- Formula $\forall x : A.B$ represented by $(x : A) \to B$.

Algebraic data types

$$\begin{array}{l} \mathrm{data}\ \mathbb{N} : \mathrm{Set} \\ \quad \begin{array}{lcl} 0 & : & \mathbb{N} \\ \mathrm{S} & : & \mathbb{N} \to \mathbb{N} \end{array} \end{array}$$

Functions defined by pattern matching

$$\begin{array}{l} f : \mathbb{N} \to \mathbb{N} \\ \begin{array}{lcccl} f & 0 & = & 5 \\ f & (\mathrm{S}\ 0) & = & 12 \\ f & (\mathrm{S}\ (\mathrm{S}\ n)) & = & (f\ \ n) * 20 \end{array} \end{array}$$

Coalgebraic data types

$$
\begin{array}{l}
\text{coalg Stream : Set where} \\
\quad \begin{array}{lcl}
\text{head} & : & \text{Stream} \to \mathbb{N} \\
\text{tail} & : & \text{Stream} \to \text{Stream}
\end{array}
\end{array}
$$

$$
\begin{array}{l}
\text{inc} : \mathbb{N} \to \text{Stream} \\
\begin{array}{lcl}
\text{head } (\text{inc } n) & = & n \\
\text{tail } \; (\text{inc } n) & = & \text{inc } (n+1)
\end{array}
\end{array}
$$

(Non-Agda syntax)

Postulates

$$
\begin{array}{llcl}
\text{postulate} & A & : & \mathbb{N} \to \mathrm{Set} \\
\text{postulate} & f & : & \mathbb{N} \to \mathbb{N} \\
\text{postulate} & \mathit{lem} & : & f\ 1 == f\ (f\ 0)
\end{array}
$$

Standard Approach to Formulation of $\mathbb{R}$

- Define
$$\mathbb{N}, \mathbb{Z}, \mathbb{Q}$$
  in a standard way.
- Define using these constructs
  - the Cauchy Reals
  $$\Sigma a : \mathbb{N} \to \mathbb{Q}.\mathrm{Cauchy}(a)$$
  - or any other representation of constructive real numbers.
- Problem: Need to prove theorems constructively, even if they have no computational content.

Ideal Objects

- Use of ideal and concrete objects.
- Use approach by Berger transferred to type theory:
  Axiomatize the real numbers abstractly. E.g.

$$
\begin{array}{lll}
\text{postulate} & \mathbb{R} & : & \text{Set} \\
\text{postulate} & 0_{\mathbb{R}}, 1_{\mathbb{R}} & : & \mathbb{R} \\
\text{postulate} & \_ == \_ & : & \mathbb{R} \to \mathbb{R} \to \text{Set} \\
\text{postulate} & \_ + \_ & : & \mathbb{R} \to \mathbb{R} \to \mathbb{R} \\
\text{postulate} & \text{commutative} & : & (r\ s : \mathbb{R}) \to r + s == s + r \\
\ldots & & &
\end{array}
$$

Concrete Objects

- Formulate $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ as standard computational data types.

$$
\begin{aligned}
&\text{data } \mathbb{N} : \text{Set where}\\
&\quad 0 \;:\; \mathbb{N}\\
&\quad \mathrm{S} \;:\; \mathbb{N} \to \mathbb{N}
\end{aligned}
$$

$$
\begin{aligned}
&\_+\_ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}\\
&n + 0 = n\\
&n + \mathrm{S}\, m = \mathrm{S}\,(n + m)
\end{aligned}
$$

$$
\begin{aligned}
&\_*\_ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}\\
&\ldots
\end{aligned}
$$

$$
\begin{aligned}
&\text{data } \mathbb{Z} : \text{Set where}\\
&\quad \ldots
\end{aligned}
$$

$$
\text{data } \mathbb{Q} : \text{Set where}
$$

Embedding of $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ into $\mathbb{R}$

- Embed $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ into $\mathbb{R}$:

$$
\begin{array}{llcl}
\multicolumn{4}{l}{\mathrm{embed}_{\mathbb{N}} : \mathbb{N} \to \mathbb{R}} \\
\mathrm{embed}_{\mathbb{N}} & 0 & = & 0_{\mathbb{R}} \\
\mathrm{embed}_{\mathbb{N}} & (\mathrm{S}\ n) & = & \mathrm{embed}_{\mathbb{N}}\ n + 1_{\mathbb{R}}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{embed}_{\mathbb{Z}} : \mathbb{Z} \to \mathbb{R} \\
\ldots
\end{array}
$$

$$
\begin{array}{l}
\mathrm{embed}_{\mathbb{Q}} : \mathbb{Q} \to \mathbb{R} \\
\ldots
\end{array}
$$

Signed Digit Representations

- Signed digit (SD) representable real numbers in $[-1,1]$ are of the form

$$r = 0.\underbrace{1}_{d}\underbrace{11(-1)0(-1)01(-1)\cdots}_{2*r-d}$$

- So

$$r \in \mathrm{SD} \Leftrightarrow r \in [-1,1] \wedge \exists d \in \{-1,0,1\}.2*r-d \in \mathrm{SD}$$

  SD = largest fixed point fulfilling that equation.
- Formulation in type theory:

$$\begin{array}{l}
\text{coalg } \mathrm{SD} : \mathbb{R} \to \mathrm{Set} \text{ where} \\
\quad \begin{array}{lclclcl}
\in[-1,1] & : & (r:\mathbb{R}) & \to & \mathrm{SD}\ r & \to & r \in [-1,1] \\
\text{digit} & : & (r:\mathbb{R}) & \to & \mathrm{SD}\ r & \to & \{-1,0,1\} \\
\text{tail} & : & (r:\mathbb{R}) & \to & (p:\mathrm{SD}\ r) & & \\
 & & & \to & \multicolumn{3}{l}{\mathrm{SD}\ (2_{\mathbb{R}} * r - \text{digit } r\ p)}
\end{array}
\end{array}$$

Extraction of Programs

- From
$$p : \mathrm{SD}\ r$$
one can obtain the first n digits of r.
- Show e.g. closure of SD under $\mathbb{Q} \cap [-1, 1]$, $+ \cap [-1, 1]$, $*$, $\frac{\pi}{10} \cdots$
- Then we extract the first n digits of any real number formed using these operations.
- Has been done (excluding $\frac{\pi}{10}$) in Agda, program extraction can be executed feasibly.

Problem with Program Extraction

- We don't want that
$$d : \mathrm{Digit}$$
$$d = \cdots$$
and evaluation of d to normal form has result
$$\mathrm{ax1}\ (\mathrm{ax2}\ (\mathrm{ax3}\ \cdots))$$
- We want that d evaluates to -1 or 0 or 1.

Example 1

$\mathrm{postulate\ ax} : B \vee C$

$f : B \vee C \to \mathbb{B}$
$f\ (\mathrm{inl}\ x) = \mathrm{tt}$
$f\ (\mathrm{inr}\ x) = \mathrm{ff}$

$(f\ \mathrm{ax})$ in normal form, doesn't start with a constructor

Example 2

$\text{postulate ax} : A \wedge B$

$f : A \wedge B \to \mathbb{B}$
$f \; \langle a, b \rangle = \cdots$

$(f \; \text{ax})$ in normal form doesn't start with a constructor

Example 3

$\text{postulate ax} : r0 == r1$

$\text{transfer} : (r\ s : \mathbb{R}) \to r == s \to \text{SD}\ r \to \text{SD}\ s$
$\text{transfer}\ r\ r\ \text{refl}\ p = p$

$\text{firstdigit} : (r : \mathbb{R}) \to \text{SD}\ r \to \text{Digit}$
$\text{firstdigit}\ r\ a = \cdots$

$p : \text{SD}\ r_0$
$p = \cdots$

$q : \text{SD}\ r_1$
$q = \text{transfer}\ r_0\ r_1\ \text{ax}$

$q' : \text{Digit}$
$q' = \text{firstdigit}\ r_1\ q$

NF of q' doesn't start with a constructor

Solution

- Problem occurs because an element of an algebraic data type was
  - introduced by a postulate
  - eliminated by an elimination rule for that type
- Restriction needed: If A is a postulated constant then either
  - $A : (x_1 : B_1) \to \cdots \to (x_n : B_n) \to \mathrm{Set}$ or
  - $A : (x_1 : B_1) \to \cdots \to (x_n : B_n) \to A' \ t_1 \cdots t_n$ where A' is a postulated constant.
- Essentially: postulated constants have result type a postulated type.

Theorem

- Assume some healthy conditions (e.g. strong normalisation, confluence, elements starting with different constructors are different).
- Assume result type of postulated axioms is always a postulated type.
- Then every closed term in normal form which is an element of an algebraic data type is in canonical normal form (starts with a constructor).

Proof Assuming Simple Pattern Matching

- Assume $t : A$, t closed in normal form, A algebraic data type.
- Show by induction on $\mathrm{length}(t)$ that t starts with a constructor.
- Let $t = f\ t_1 \cdots t_n$, f function symbol or constructor.
- f cannot be postulated or directly defined.
- If f is defined by pattern matching on say t_i.
    - By IH t_i starts with a constructor.
    - t has a reduction, wasn't in NF
- So f is a constructor.

Reduction of Nested Pattern Matching to Simple Pattern Matching

Difficult proof in the thesis of Chi Ming Chuang.

Logic for Ideal Objects

- The following fulfils our conditions:

$$
\begin{array}{lll}
\text{postulate} & \_ \vee' \_ & : \mathrm{Set} \to \mathrm{Set} \to \mathrm{Set} \\
\text{postulate} & \text{excluded\_middle} & : (X : \mathrm{Set}) \to X \vee' \neg X \\
\text{postulate} & \vee'_{\mathrm{elim}} & : (A \to C) \to (B \to C) \to A \vee' B \to C \\
 & & \quad \text{(for postulated } C\text{)}
\end{array}
$$

Negated Axioms

- Not allowed (using $\neg A = A \to \bot$

$$\begin{aligned} &\text{postulate } a : \neg A \\ &\text{postulate } b : A \\ \\ &c : \bot \to \mathbb{N} \\ &c\ () \\ \\ &d : \mathbb{N} \\ &d = c\ (a\ b) \end{aligned}$$

- However: If the type theory used $\not\vdash p : \bot$ and every postulated type has result type postulate type or $\bot$, then conclusions of theorem fulfilled.

Conclusion

Conclusion

- If result types of postulated constants are postulated types, then closed elements of algebraic types evaluate to constructor normal form.
- Makes develop of programs much easier (by postulating axioms or proving them using ATP).
- Axiomatic treatment of $\mathbb{R}$.
- Program extraction for proofs with real number computations works very well.
- Possible application to type theory with partial and total objects.