On Proof-Relevant Relations and Evidence-Aware Programming

Andreas Abel$^1$

$^1$Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

TCS Oberseminar
Ludwig-Maximilians-Universität München
11 January 2019
Proof-relevance and evidence manipulation

- Curry-Howard-Isomorphism (CHI):
  - propositions-as-types
  - proofs-as-programs
- Dependently-typed programming languages implement the CHI: e.g. Agda, Coq, Idris, Lean
- Allows maintenance and processing of evidence.
- For practical impact, we need a also programming culture; c.f. GoF, *Design Patterns: Elements of Reusable Object-Oriented Software*. 
List membership

- Membership $a \in as$ inductively definable:

\[
\begin{align*}
\text{zero} & \quad a \in (a :: as) \\
\text{suc} & \quad a \in as \\
\text{suc (suc zero)} & \quad a \in (b :: as)
\end{align*}
\]

- Proofs of $a \in as$ are indices of $a$ in $as$ (unary natural numbers).
- Two different derivations of $3 \in (3 :: 7 :: 3 :: [])$, correspond to the occurrences of $3$:

\[
\begin{align*}
\text{zero} & : 3 \in (3 :: 7 :: 3 :: []) \\
\text{suc (suc zero)} & : 3 \in (3 :: 7 :: 3 :: [])
\end{align*}
\]
Sublists

- Inductive sublist relation \( as \subseteq bs \):

  \[
  \begin{align*}
  \text{skip} & \quad \frac{as \subseteq bs}{as \subseteq (b :: bs)} \\
  \text{keep} & \quad \frac{as \subseteq bs}{(a :: as) \subseteq (a :: bs)} \\
  \text{done} & \quad \frac{}{[]} \subseteq []
  \end{align*}
  \]

- A proof of \( as \subseteq bs \) describes which elements of \( bs \) should be dropped (skip) to arrive at \( as \).

  \[
  \begin{align*}
  \text{skip (keep done)} & : (a :: []) \subseteq (a :: a :: []) \\
  \text{keep (skip done)} & : (a :: []) \subseteq (a :: a :: [])
  \end{align*}
  \]

- \( \subseteq \) is a category.

  \[
  \begin{align*}
  \text{id} & \quad : \quad as \subseteq as \quad \text{reflexivity} \\
  \bigcirc \quad & \quad : \quad (as \subseteq bs) \rightarrow (bs \subseteq cs) \rightarrow (as \subseteq cs) \quad \text{transitivity}
  \end{align*}
  \]

- Single extension

  \[
  \text{sgw} \quad : \quad as \subseteq (a :: as)
  \]
Membership in sublists

- Membership is inherited from sublists:

\[
\text{reindex} : (as \subseteq bs) \rightarrow (a \in as) \rightarrow (a \in bs)
\]

adjusts the index of \(a\) in \(as\) to point to the corresponding \(a\) in \(bs\).

- Trivium: \text{reindex} is a functor from \(\_ \subseteq \_\) to \((a \in \_) \rightarrow (a \in \_)\).
- In category speak: \text{reindex} is a presheaf on \(\subseteq^{op}\).
Types, sets, propositions, singletons

- Our meta-language is (Martin-Löf) type theory: $a \in as$ and $as \subseteq bs$ are types, their proofs are inhabitants.
- Following Vladimir Voevodsky†, types are stratified by their $h$-level into singletons (0), propositions (1), sets (2), groupoids (3), . . . .
  1. A type with a unique inhabitant is a singleton ("contractible").
  2. A type with at most one inhabitant is a proposition. In other words, a type with contractible equality is a proposition.
  3. A type with propositional equality is a set.
  4. A type with a set equality is a groupoid.
- A type is of $h$-level $n + 1$ if its equality is of $h$-level $n$.
- $as \subseteq as$ is a singleton; so is $a \in (a :: [])$.
- $as \subseteq []$ is a proposition; so is $a \in (b :: [])$.
- In general $a \in as$ and $as \subseteq bs$ are sets.
Natural deduction

- Inference rules of intuitionistic implicational logic $\Gamma \vdash A$:

  \[
  \begin{array}{ccc}
  \text{var} & \frac{A \in \Gamma}{\Gamma \vdash A} \\
  \text{app} & \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \\
  \text{abs} & \frac{(A :: \Gamma) \vdash B}{\Gamma \vdash A \Rightarrow B}
  \end{array}
  \]

- Derivations of $\Gamma \vdash A$ are simply-typed lambda-terms with variables represented by de Bruijn indices $x : (A \in \Gamma)$.

  \[
  \begin{align*}
  t &:= \text{app} (\text{var} \text{ zero}) (\text{var} \text{ (suc zero)}) : (A \Rightarrow B :: A :: [] \vdash B) \\
  \text{abs} (\text{abs} t) & : ([] \vdash A \Rightarrow (A \Rightarrow B) \Rightarrow B) \\
  \text{abs} (\text{abs} (\text{var} \text{ (suc zero)))) & : A \Rightarrow (A \Rightarrow A) \\
  \text{abs} (\text{abs} (\text{var} \text{ zero})) & : A \Rightarrow (A \Rightarrow A)
  \end{align*}
  \]
Weakening

- Inferences stay valid under additional hypotheses (monotonicity):

\[
\text{weak} : (\Gamma \subseteq \Delta) \rightarrow (\Gamma \vdash A) \rightarrow (\Delta \vdash A)
\]

adjust indices of hypotheses \(\text{var}\)

- \text{weak} is a functor from \(\subseteq\) to \((\vdash A)\) → \((\vdash A)\).
List. All: true on every element

- **All $P$ as**: Predicate $P$ holds on all elements of list $as$.

\[
\begin{align*}
\text{[]} & \quad \text{All } P \text{ []} \\
(- :: -) & \quad P a \quad \text{All } P \text{ as} \\
\text{All } P \text{ (a :: as)}
\end{align*}
\]

- Proofs of **All $P$ as** are decorations of each list element $a$ with further data of type $P a$.
- Soundness is retrieval of this data, completeness tabulation:

  \[
  \begin{align*}
  \text{lookup} & : \text{All } P \text{ as} \to a \in as \to P a \\
  \text{tabulate} & : (\forall a. \ a \in as \to P a) \to \text{All } P \text{ as}
  \end{align*}
  \]

- Universal truth is passed down to sublists:

  \[
  \begin{align*}
  \text{select} & : as \subseteq bs \to \text{All } P \text{ bs} \to \text{All } P \text{ as}
  \end{align*}
  \]
Substitution

- Inhabitants of $\text{All } (\Gamma \vdash \_ ) \Delta$ are
  - proofs that all formulas in $\Delta$ are derivable from hypotheses $\Gamma$
  - substitutions from $\Delta$ to $\Gamma$

- Parallel substitution

  $$\text{subst} : \text{All } (\Gamma \vdash \_ ) \Delta \to \Delta \vdash A \to \Gamma \vdash A$$

  replaces hypotheses $A \in \Delta$ by derivations of $\Gamma \vdash A$.

- $\text{Subst } \Gamma \Delta := \text{All } (\Gamma \vdash \_ ) \Delta$ is a category:

  $$\text{id} : \text{Subst } \Gamma \Gamma$$
  $$\text{comp} : \text{Subst } \Gamma \Delta \to \text{Subst } \Delta \Phi \to \text{Subst } \Gamma \Phi$$

- Singleton substitution

  $$\text{sg} : \Gamma \vdash A \to \text{Subst } \Gamma (A :: \Gamma)$$
Term equality and normal forms

- For \( t, t' : (\Gamma \vdash A) \) define \( \beta\eta \)-equality \( t =_{\beta\eta} t' \) as the least congruence over

\[
\begin{align*}
\beta & \quad t : (A :: \Gamma \vdash B) \quad u : \Gamma \vdash A \\
\text{app (abs } t \text{) } u =_{\beta\eta} \text{ subst (sg } u \text{) } t \\
\end{align*}
\]

\[
\eta \quad t : (\Gamma \vdash A \Rightarrow B) \\
\text{abs (app (weak sgw } t \text{) (var zero))}
\]

- \( \beta\eta \)-normality \( \text{Nf } t \) and neutrality \( \text{Ne } t \) (where \( o \) base formula):

\[
\begin{align*}
\text{var } x : A \in \Gamma \\
\text{Ne (var } x \text{)} \\
\text{app } t & \quad \text{Nf } u \\
\text{Ne (app } t \text{ u) } \\
\text{ne } t & \quad t : (\Gamma \vdash o) \\
\text{Nf } t & \quad \text{abs } t \\
\text{Nf (abs } t \text{) } \\
\end{align*}
\]
Normalization

- Having a normal/neutral form:

  \[ \text{NF } t = \exists t' =_{\beta\eta} t. \text{Nf } t' \]
  \[ \text{NE } t = \exists t' =_{\beta\eta} t. \text{Ne } t' \]

- Interpretation of formulas as types:

  \[ \langle A \rangle_{\Gamma} : \Gamma \vdash A \to \text{Type} \]
  \[ \langle o \rangle_{\Gamma} t = \text{NE } t \]
  \[ \langle A \Rightarrow B \rangle_{\Gamma} t = \forall \Delta (w : \Gamma \subseteq \Delta)(u : \Delta \vdash A) \]
  \[ \rightarrow \langle A \rangle_{\Delta} u \]
  \[ \rightarrow \langle B \rangle_{\Delta} (\text{app (weak } w t) u) \]

- Soundness and completeness (combine to normalization):

  sound : (t : \Gamma \vdash A)(\sigma : \text{Subst } \Delta \Gamma) \rightarrow \langle \Gamma \rangle_{\Delta} \sigma \rightarrow \langle A \rangle_{\Delta} (\text{subst } \sigma t)
  
  complete : \langle A \rangle_{\Gamma} t \rightarrow \text{NF } t
Formal languages

- A context-free grammar (CFG) be given by
  - terminals \( a, b, c, \ldots \) (words \( u, v, w, \ldots \))
  - non-terminals \( X, Y, Z, \ldots \)
  - sentential forms \( \alpha, \beta \), e.g. \( XabY \)
  - rules \( r \) given by a type family \( \_ ::= \_ \). We write \( r : (X ::= \alpha) \) if \( X \rightarrow \alpha \) is a rule of the CFG.

- Word membership \( w \in \alpha \):

  \[
  \begin{align*}
  \text{red} & \quad X ::= \alpha \quad \text{w} \in \alpha \\
  & \quad \frac{}{w \in X}
  
  \varepsilon & \quad \text{tm} \quad w \in \beta \\
  & \quad \frac{}{\varepsilon \in \varepsilon} \\
  & \quad \frac{}{aw \in a\beta}
  
  nt & \quad \frac{u \in X \quad v \in \beta}{uv \in X\beta}
  
  \end{align*}
  \]

- Proofs of \( w \in \alpha \) are parse trees.
Earley parser

- **Judgement** \( u.X \rightsquigarrow v.\beta \)

\[
\begin{align*}
\text{init} & \quad \varepsilon.S \rightsquigarrow \varepsilon.S \\
\text{predict} & \quad u.X \rightsquigarrow v.Y\beta \\
\text{scan} & \quad u.X \rightsquigarrow v.a\beta \\
\text{combine} & \quad uv.Y \rightsquigarrow \varepsilon.\alpha \\
\end{align*}
\]

- To parse \( w \in S \) derive \( \varepsilon.S \rightsquigarrow w.\varepsilon \).
- **Soundness**: If \( u.X \rightsquigarrow v.\beta \) and \( w \in \beta \) then \( vw \in X \).
- **Completeness**: If \( u.X \rightsquigarrow v.\alpha\beta \) and \( w \in \alpha \) then \( u.X \rightsquigarrow vw.\beta \).
Conclusion

- Many CHI design patterns to discover!
- Current trend: revisit parsing theory from a type-theoretic perspective.
- Edwin Brady: bootstrapping Blodwen in Idris.
- Large project: bootstrap Agda.