How to Represent It in Agda
On Proof-Relevant Relations and Evidence-Aware Programming

Andreas Abel¹

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

29th Agda Implementors’ Meeting
Ochanomizu University, Tokyo, Japan
13 March 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 maintainance 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:
  
  \[
  \begin{aligned}
  \text{zero} & \quad \frac{}{a \in (a :: as)} \\
  \text{suc} & \quad \frac{a \in as}{a \in (b :: as)}
  \end{aligned}
  \]

- 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 $3$:
  
  \[
  \begin{aligned}
  \text{zero} & : \quad 3 \in (3 :: 7 :: 3 :: []) \\
  \text{suc (suc zero)} & : \quad 3 \in (3 :: 7 :: 3 :: [])
  \end{aligned}
  \]
Sublists

- Inductive sublist relation $as \subseteq bs$:

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

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

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

- $\subseteq$ is a category.

  \[
  \begin{align*}
  \text{id} & \quad : \quad as \subseteq as \\
  \circ & \quad : \quad (as \subseteq bs) \to (bs \subseteq cs) \to (as \subseteq cs)
  \end{align*}
  \]

  reflexivity

  transitivity

- Single extension

  \[
  \begin{align*}
  \text{sgw} & \quad : \quad as \subseteq (a :: as)
  \end{align*}
  \]
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 \( \subseteq^{\text{op}} \).
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 Voewodsky†, 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 intuitionstic implicational logic $\Gamma \vdash A$:

$$
\begin{align*}
\text{var} & \quad A \in \Gamma \\
\Gamma & \vdash A \\
\text{app} & \quad \Gamma \vdash A \Rightarrow B \\
\Gamma & \vdash B \\
\text{abs} & \quad (A :: \Gamma) \vdash B \\
\Gamma & \vdash A \Rightarrow B
\end{align*}
$$

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

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

- Inferences stay valid under additional hypotheses (monotonicity):

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

  adjust indices of hypotheses (\textit{var})

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

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

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

- 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:

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

- Universal truth is passed down to sublists:

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

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

  replaces hypotheses $A \in \Delta$ by derivations of $\Gamma \vdash A$.
- $\text{Subst } \Gamma \Delta := \text{All } (\Gamma \vdash _) \Delta$ is a category:

  $\text{id} : \text{Subst } \Gamma \Gamma$

  $\text{comp} : \text{Subst } \Gamma \Delta \rightarrow \text{Subst } \Delta \Phi \rightarrow \text{Subst } \Gamma \Phi$

- Singleton substitution

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

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

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

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

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

\[
\begin{align*}
\text{var } \quad & x : A \in \Gamma \\
\text{Ne (var } x \text{)}
\end{align*}
\]

\[
\begin{align*}
\text{app } \quad & \text{Ne } t \quad \text{Nf } u \\
\text{Ne (app } t \text{ } u \text{)}
\end{align*}
\]

\[
\begin{align*}
\text{ne } \quad & \text{Ne } t \\
\text{Nf } t
\end{align*}
\]

\[
\begin{align*}
\text{abs } \quad & \text{Nf } t \\
\text{Nf (abs } t \text{)}
\end{align*}
\]
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:

\[
\llbracket A \rrbracket_{\Gamma} : \Gamma \vdash A \rightarrow \text{Type}
\]

\[
\llbracket o \rrbracket_{\Gamma} t = \text{NE } t
\]

\[
\llbracket A \Rightarrow B \rrbracket_{\Gamma} t = \forall \Delta \ (w : \Gamma \subseteq \Delta)(u : \Delta \vdash A) \\
\rightarrow \llbracket A \rrbracket_{\Delta} u \rightarrow \llbracket B \rrbracket_{\Delta} (\text{app (weak } w \text{ } t) \ u)
\]

- Soundness and completeness (combine to normalization):

\[
\text{sound} : (t : \Gamma \vdash A)(\sigma : \text{Subst } \Delta \Gamma) \rightarrow \llbracket \Gamma \rrbracket_{\Delta} \sigma \rightarrow \llbracket A \rrbracket_{\Delta} (\text{subst } \sigma \ t)
\]

\[
\text{complete} : \llbracket A \rrbracket_{\Gamma} t \rightarrow \text{NF } t
\]
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 \( w \in \alpha \):

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

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

- Judgement \( u.X \leadsto v.\beta \)

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

uv.Y \leadsto \epsilon.\alpha \\
Y ::= \alpha

- To parse \( w \in S \) derive \( \epsilon.S \leadsto w.\epsilon \).
- Soundness: If \( u.X \leadsto v.\beta \) and \( w \in \beta \) then \( vw \in X \).
- Completeness: If \( u.X \leadsto v.\alpha\beta \) and \( w \in \alpha \) then \( u.X \leadsto vw.\beta \).
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.