How to Represent It in Agda

On Proof-Relevant Relations and Evidence-Aware Programming

Andreas Abel[1]

[1]Department 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):
  - propsitions-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:

$$\mathsf{zero}\ \frac{}{a \in (a :: as)} \qquad \mathsf{suc}\ \frac{a \in as}{a \in (b :: as)}$$

- 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:

$$\begin{array}{rcl} \mathsf{zero} & : & 3 \in (3 :: 7 :: 3 :: []) \\ \mathsf{suc}\,(\mathsf{suc}\,\mathsf{zero}) & : & 3 \in (3 :: 7 :: 3 :: []) \end{array}$$

Sublists

- Inductive sublist relation $as \subseteq bs$:

$$\text{skip}\ \frac{as \subseteq bs}{as \subseteq (b :: bs)} \qquad \text{keep}\ \frac{as \subseteq bs}{(a :: as) \subseteq (a :: bs)} \qquad \text{done}\ \frac{}{[] \subseteq []}$$

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

$$\begin{array}{lcl} \text{skip}\,(\text{keep}\,\text{done}) & : & (a :: []) \subseteq (a :: a :: []) \\ \text{keep}\,(\text{skip}\,\text{done}) & : & (a :: []) \subseteq (a :: a :: []) \end{array}$$

- $\subseteq$ is a category.

$$\begin{array}{lcll} \text{id} & : & as \subseteq as & \text{reflexivity} \\ \_\circ\_ & : & (as \subseteq bs) \to (bs \subseteq cs) \to (as \subseteq cs) & \text{transitivity} \end{array}$$

- Single extension

$$\text{sgw} \quad : \quad as \subseteq (a :: as)$$

Membership in sublists

- Membership is inherited from sublists:

$$\mathsf{reindex} \quad : \quad (as \subseteq bs) \to (a \in as) \to (a \in bs)$$

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

- Trivium: reindex is a functor from $\_ \subseteq \_$ to $(a \in \_) \to (a \in \_)$.
- In category speak: reindex is a presheaf on $\subseteq^{\mathsf{op}}$.

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 $\Gamma \vdash A$:

$$\text{var}\ \frac{A \in \Gamma}{\Gamma \vdash A} \qquad \text{app}\ \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \qquad \text{abs}\ \frac{(A :: \Gamma) \vdash B}{\Gamma \vdash A \Rightarrow B}$$

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

$$\begin{array}{lcl}
t := \mathsf{app}\,(\mathsf{var\,zero})\,(\mathsf{var}\,(\mathsf{suc\,zero})) & : & (A \Rightarrow B :: A :: [] \vdash B) \\
\mathsf{abs}\,(\mathsf{abs}\,t) & : & ([] \vdash A \Rightarrow (A \Rightarrow B) \Rightarrow B) \\
\mathsf{abs}\,(\mathsf{abs}\,(\mathsf{var}\,(\mathsf{suc\,zero}))) & : & A \Rightarrow (A \Rightarrow A) \\
\mathsf{abs}\,(\mathsf{abs}\,(\mathsf{var\,zero})) & : & A \Rightarrow (A \Rightarrow A)
\end{array}$$

Weakening

- Inferences stay valid under additional hypotheses (monotonicity):

$$\mathsf{weak} \quad : \quad (\Gamma \subseteq \Delta) \to (\Gamma \vdash A) \to (\Delta \vdash A)$$

  adjust indices of hypotheses (var)

- weak is a functor from $\_ \subseteq \_$ to $(\_ \vdash A) \to (\_ \vdash A)$.

List.All: true on every element

- $\mathsf{All}\ P\ as$: Predicate P holds on all elements of list as.

$$[\,]\ \frac{}{\mathsf{All}\ P\ [\,]} \qquad (\_ :: \_)\ \frac{P\ a \qquad \mathsf{All}\ P\ as}{\mathsf{All}\ P\ (a :: as)}$$

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

$$\begin{array}{lcl} \mathsf{lookup} & : & \mathsf{All}\ P\ as \to a \in as \to P\ a \\ \mathsf{tabulate} & : & (\forall a.\ a \in as \to P\ a) \to \mathsf{All}\ P\ as \end{array}$$

- Universal truth is passed down to sublists:

$$\mathsf{select} \quad : \quad as \subseteq bs \to \mathsf{All}\ P\ bs \to \mathsf{All}\ P\ as$$

Substitution

- Inhabitants of $\mathsf{All}\,(\Gamma \vdash \_)\,\Delta$ are
  - proofs that all formulas in Δ are derivable from hypotheses Γ
  - substitutions from Δ to Γ
- Parallel substitution

$$\mathsf{subst} \quad : \quad \mathsf{All}\,(\Gamma \vdash \_)\,\Delta \to \Delta \vdash A \to \Gamma \vdash A$$

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

$$\begin{array}{lcl} \mathsf{id} & : & \mathsf{Subst}\,\Gamma\,\Gamma \\ \mathsf{comp} & : & \mathsf{Subst}\,\Gamma\,\Delta \to \mathsf{Subst}\,\Delta\,\Phi \to \mathsf{Subst}\,\Gamma\,\Phi \end{array}$$

- Singleton substitution

$$\mathsf{sg} \quad : \quad \Gamma \vdash A \to \mathsf{Subst}\,\Gamma\,(A :: \Gamma)$$

Term equality and normal forms

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

$$\beta\ \frac{t : (A :: \Gamma \vdash B) \qquad u : \Gamma \vdash A}{\mathsf{app}\,(\mathsf{abs}\,t)\,u =_{\beta\eta} \mathsf{subst}\,(\mathsf{sg}\,u)\,t}$$

$$\eta\ \frac{t : (\Gamma \vdash A \Rightarrow B)}{t =_{\beta\eta} \mathsf{abs}\,(\mathsf{app}\,(\mathsf{weak}\,\mathsf{sgw}\,t)\,(\mathsf{var}\,\mathsf{zero}))}$$

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

$$\mathsf{var}\ \frac{x : A \in \Gamma}{\mathsf{Ne}\,(\mathsf{var}\,x)} \qquad \mathsf{app}\ \frac{\mathsf{Ne}\ t \qquad \mathsf{Nf}\ u}{\mathsf{Ne}\,(\mathsf{app}\,t\,u)}$$

$$\mathsf{ne}\ \frac{\mathsf{Ne}\ t}{\mathsf{Nf}\ t}\ t : (\Gamma \vdash o) \qquad \mathsf{abs}\ \frac{\mathsf{Nf}\ t}{\mathsf{Nf}\,(\mathsf{abs}\,t)}$$

Normalization

- Having a normal/neutral form:

$$\begin{array}{lcl} \mathsf{NF}\, t & = & \exists t' =_{\beta\eta} t.\ \mathsf{Nf}\ t' \\ \mathsf{NE}\, t & = & \exists t' =_{\beta\eta} t.\ \mathsf{Ne}\ t' \end{array}$$

- Interpretation of formulas as types:

$$\begin{array}{lcl} [\![A]\!]_\Gamma & : & \Gamma \vdash A \to \mathsf{Type} \\ [\![o]\!]_\Gamma t & = & \mathsf{NE}\, t \\ [\![A \Rightarrow B]\!]_\Gamma t & = & \forall \Delta\ (w : \Gamma \subseteq \Delta)(u : \Delta \vdash A) \\ & & \to [\![A]\!]_\Delta u \\ & & \to [\![B]\!]_\Delta (\mathsf{app}\, (\mathsf{weak}\, w\ t)\ u) \end{array}$$

- Soundness and completeness (combine to normalization):

$$\begin{array}{lcl} \mathsf{sound} & : & (t : \Gamma \vdash A)(\sigma : \mathsf{Subst}\, \Delta\, \Gamma) \to [\![\Gamma]\!]_\Delta \sigma \to [\![A]\!]_\Delta (\mathsf{subst}\, \sigma\ t) \\ \mathsf{complete} & : & [\![A]\!]_\Gamma t \to \mathsf{NF}\, t \end{array}$$

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 α, β, e.g. $XabY$
  - rules r given by a type family $\_ ::= \_$. We write $r : (X ::= \alpha)$ if $X \to \alpha$ is a rule of the CFG.
- Word membership $w \in \alpha$:

$$\text{red}\ \frac{X ::= \alpha \qquad w \in \alpha}{w \in X}$$

$$\varepsilon\ \frac{}{\varepsilon \in \varepsilon} \qquad \text{tm}\ \frac{w \in \beta}{aw \in a\beta} \qquad \text{nt}\ \frac{u \in X \qquad v \in \beta}{uv \in X\beta}$$

- Proofs of $w \in \alpha$ are parse trees.

Earley parser

- Judgement $u.X \rightsquigarrow v.\beta$

$$\text{init}\ \frac{}{\varepsilon.S \rightsquigarrow \varepsilon.S} \qquad \text{predict}\ \frac{u.X \rightsquigarrow v.Y\beta \qquad Y ::= \alpha}{uv.Y \rightsquigarrow \varepsilon.\alpha}$$

$$\text{scan}\ \frac{u.X \rightsquigarrow v.a\beta}{u.X \rightsquigarrow va.\beta} \qquad \text{combine}\ \frac{u.X \rightsquigarrow v.Y\beta \qquad uv.Y \rightsquigarrow w.\varepsilon}{u.X \rightsquigarrow vw.\beta}$$

- 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.