Poor Man's Axi: DPL-like proofs in Type Theory

Wojciech Kołowski

Intro

In our original proposal of Poor Man's Axi, the language was split into two layers: a programming layer which consists of a strongly-typed functional programming language based on the Simply Typed Lambda Calculus, and a logical layer which consists of first-order classical logic with equality (although the "first-order" part is moot, because first-order quantifiers can quantify over functions).

The logic was presented with a bunch of judgements, the most important being the true proposition judgement. While this presentation does a good job of explaining what the logic is like, it does not address the problem of writing proofs from the perspective of the user.

Proofs

Proofs (here P are propositions, t are terms, x are variables):

e ::=

P | **assume** P **in** e | **modus-ponens** e_1 e_2 |
suppose-absurd P **in** e | **absurd** e_1 e_2 |
both e_1 e_2 | **left-and** e | **right-and** e |
left-either P e | **right-either** P e |
constructive-dilemma e_1 e_2 e_3 |
equivalence e_1 e_2 | **left-iff** e | **right-iff** e |
true | **exfalso** e
pick-any x **in** e | **specialize** e **with** t |
exists t **such that** e | **pick-witness** x **for** e_1 **in** e_2 |
double-negation e |
case e **of** $(\texttt{inl}\ a \rightarrow e_1, \texttt{inr}\ b \rightarrow e_2)$ |
refl t | **rewrite** e_1 **in** e_2 |
$e_1; e_2$

Example – propositional logic

Theorem: $(P \Rightarrow Q) \Rightarrow (Q \Rightarrow R) \Rightarrow P \Rightarrow R$.

Proof:
assume $P \Rightarrow Q$ **in**
  assume $Q \Rightarrow R$ **in**
    assume P **in**
      modus-ponens $(P \Rightarrow Q)\ P$;
      modus-ponens $(Q \Rightarrow R)\ Q$

The proof looks the same as the DPL one (page 71 in the DPL thesis), except that we don't have **begin** and **end**.

Example – first-order logic

Theorem: $(\forall x : A.\ P\ x \wedge Q\ x) \Rightarrow (\forall x : A.\ P\ x) \wedge (\forall x : A.\ Q\ x)$

Proof:

assume $\forall x : A.\ P\ x \wedge Q\ x$ **in**
  pick-any y **in**
    specialize $\forall x : A.\ P\ x \wedge Q\ x$ **with** y;
    left-and $P\ y \wedge Q\ y$;
  pick-any y **in**
    specialize $\forall x : A.\ P\ x \wedge Q\ x$ **with** y;
    right-and $P\ y \wedge Q\ y$;
  both $(\forall y : A.\ P\ y)$ $(\forall y : A.\ Q\ y)$

Again, the proof looks the same as the DPL on (page 156 in the DPL thesis), except we use indentation instead of **begin** and **end**.

Example – proof about a program

Program: $\texttt{swap} := \lambda x.\texttt{case}\ x\ \texttt{of}\ (\lambda a.\texttt{inr}\ a, \lambda b.\texttt{inl}\ b)$
Typing: $\Gamma \vdash \texttt{swap} : A + B \to B + A$

Theorem: $\forall x : A + B.\, \texttt{swap}\ (\texttt{swap}\ x) = x$

Proof:
pick-any x **in**
case x **of** $(\texttt{inl}\ a \to \textbf{refl}\ a, \texttt{inr}\ b \to \textbf{refl}\ b)$

The proof has the same structure as the proofterm you would write in Coq, except for the syntactic differences.

Proof judgement

Proof judgement:
$\Gamma \mid \Delta \vdash e : P$ – in typing context Γ and assumption context Δ, e is a proof of P.

Assumptions

$$\frac{\Gamma \ \vdash \Delta\ \texttt{valid} \qquad P \in \Delta}{\Gamma \mid \Delta \vdash P : P}\,\textsc{Ass}$$

True and false

$$\frac{\Gamma \vdash \Delta \ \texttt{valid}}{\Gamma \mid \Delta \vdash \textbf{true} : \top}\textsc{True-Intro}$$

$$\frac{\Gamma \mid \Delta \vdash e : \bot}{\Gamma \mid \Delta \vdash \textbf{exfalso}\ e : P}\textsc{False-Elim}$$

Implication

$$\frac{\Gamma \mid \Delta, P \vdash e : Q}{\Gamma \mid \Delta \vdash \textbf{assume } P \textbf{ in } e : P \Rightarrow Q}\textsc{Impl-Intro}$$

$$\frac{\Gamma \mid \Delta \vdash e_1 : P \Rightarrow Q \quad \Gamma \mid \Delta \vdash e_2 : P}{\Gamma \mid \Delta \vdash \textbf{modus-ponens } e_1\ e_2 : Q}\textsc{Impl-Elim}$$

Conjunction

$$\frac{\Gamma \mid \Delta \vdash e_1 : P \quad \Gamma \mid \Delta \vdash e_2 : Q}{\Gamma \mid \Delta \vdash \textbf{both}\ e_1\ e_2 : P \land Q}\ \textsc{And-Intro}$$

$$\frac{\Gamma \mid \Delta \vdash e : P \land Q}{\Gamma \mid \Delta \vdash \textbf{left-and}\ e : P}\ \textsc{And-Elim-L}$$

$$\frac{\Gamma \mid \Delta \vdash e : P \land Q}{\Gamma \mid \Delta \vdash \textbf{right-and}\ e : Q}\ \textsc{And-Elim-R}$$

Disjunction

$$\frac{\Gamma \mid \Delta \vdash e : P}{\Gamma \mid \Delta \vdash \textbf{left-either}\ Q\ e : P \vee Q}\textsc{Or-Intro-L}$$

$$\frac{\Gamma \mid \Delta \vdash e : Q}{\Gamma \mid \Delta \vdash \textbf{right-either}\ P\ e : P \vee Q}\textsc{Or-Intro-R}$$

$$\frac{\Gamma \mid \Delta \vdash e_1 : P \vee Q \quad \Gamma \mid \Delta \vdash e_2 : P \Rightarrow R \quad \Gamma \mid \Delta \vdash e_3 : Q \Rightarrow R}{\Gamma \mid \Delta \vdash \textbf{constructive-dilemma}\ e_1\ e_2\ e_3 : R}\textsc{Or-Elim}$$

Biconditional

$$\frac{\Gamma \mid \Delta \vdash e_1 : P \Rightarrow Q \quad \Gamma \mid \Delta \vdash e_2 : Q \Rightarrow P}{\Gamma \mid \Delta \vdash \textbf{equivalence}\ e_1\ e_2 : P \Leftrightarrow Q}\textsc{Iff-Intro}$$

$$\frac{\Gamma \mid \Delta \vdash e : P \Leftrightarrow Q}{\Gamma \mid \Delta \vdash \textbf{left-iff}\ e : P \Rightarrow Q}\textsc{Iff-Elim-L}$$

$$\frac{\Gamma \mid \Delta \vdash e : P \Leftrightarrow Q}{\Gamma \mid \Delta \vdash \textbf{right-iff}\ e : Q \Rightarrow P}\textsc{Iff-Elim-R}$$

Negation

$$\frac{\Gamma \mid \Delta, P \vdash e : \bot}{\Gamma \mid \Delta \vdash \textbf{suppose-absurd}\ P\ \textbf{in}\ e : \neg P}\textsc{Not-Intro}$$

$$\frac{\Gamma \mid \Delta \vdash e_1 : \neg P \quad \Gamma \mid \Delta \vdash e_2 : P}{\Gamma \mid \Delta \vdash \textbf{absurd}\ e_1\ e_2 : \bot}\textsc{Not-Elim}$$

Classical logic

$$\frac{\Gamma \mid \Delta \vdash e : \neg\neg P}{\Gamma \mid \Delta \vdash \textbf{double-negation}\ e : P}\textsc{Classic}$$

Proof composition (or let binding, really)

$$\frac{\Gamma \mid \Delta \vdash e_1 : P \quad \Gamma \mid \Delta, P \vdash e_2 : Q}{\Gamma \mid \Delta \vdash e_1; e_2 : Q}\textsc{Cut}$$

Universal quantifier

$$\frac{\Gamma, y : A \mid \Delta \vdash e : P\,[x := y]}{\Gamma \mid \Delta \vdash \textbf{pick-any}\ y\ \textbf{in}\ e : \forall x : A.\, P}\ \textsc{Forall-Intro}$$

$$\frac{\Gamma \mid \Delta \vdash e : \forall x : A.\, P \qquad \Gamma \vdash t : A}{\Gamma \mid \Delta \vdash \textbf{specialize}\ e\ \textbf{with}\ t : P\,[x := t]}\ \textsc{Forall-Elim}$$

Existential quantifier

$$\frac{\Gamma \vdash t : A \quad \Gamma \mid \Delta \vdash e : P\,[x := t]}{\Gamma \mid \Delta \vdash \textbf{exists } t \textbf{ such that } e : \exists x : A.\, P}\,\textsc{Exists-Intro}$$

$$\frac{{\color{green}\Gamma \vdash R\ \texttt{prop}} \quad \Gamma \mid \Delta \vdash e_1 : \exists x : A.\, P \quad \Gamma, y : A \mid \Delta, P\,[x := y] \vdash e_2 : R}{\Gamma \mid \Delta \vdash \textbf{pick-witness } y \textbf{ for } e_1 \textbf{ in } e_2 : R}\,\textsc{Exis}$$

Reasoning by cases on terms (for sums)

$$\frac{\Gamma, x : A \vdash P \ \mathtt{prop} \quad \Gamma \vdash t : A + B \quad \begin{array}{c} \Gamma, a : A \mid \Delta \vdash e_1 : P\,[x := \mathtt{inl}\ a] \\ \Gamma, b : B \mid \Delta \vdash e_2 : P\,[x := \mathtt{inr}\ b] \end{array}}{\Gamma \mid \Delta \vdash \textbf{case}\ t\ \textbf{of}\ (\mathtt{inl}\ a \to e_1, \mathtt{inr}\ b \to e_2) : P\,[x := t]}$$

Equality

$$\dfrac{\Gamma \vdash \Delta \ \texttt{valid} \quad \Gamma \vdash t : A}{\Gamma \mid \Delta \vdash \textbf{refl}\ t : t =_A t}\textsc{Eq-Intro}$$

$$\dfrac{\Gamma \mid \Delta \vdash e : t_1 =_A t_2 \quad \Gamma, x : A \vdash P\ \texttt{prop} \quad \Gamma \mid \Delta \vdash e' : P\,[x := t_1]}{\Gamma \mid \Delta \vdash \textbf{rewrite}\ e\ \textbf{in}\ e' : P\,[x := t_2]}\textsc{Eq-Elim}$$

Equality of functions

$$\frac{\Gamma \mid \Delta \vdash e : \forall x : A.\, f\ x =_B g\ x}{\Gamma \mid \Delta \vdash \textbf{funext}\ e : f =_{A \to B} g}\textsc{Funext}$$

Conversion rule

$$\frac{\Gamma, x : A \vdash P\ \texttt{prop} \quad \Gamma \mid \Delta \vdash e : P\,[x := t_1] \quad \Gamma \vdash t_1 \equiv t_2 : A}{\Gamma \mid \Delta \vdash e : P\,[x := t_2]}\ \textsc{Conv}$$