Proof terms for simply typed higher order logic

Stefan Berghofer and Tobias Nipkow

Institut für Informatik
TU München

Roadmap

- Motivation and background
- Proof terms: basic concepts
- Reconstruction of partial proof terms
- Compression of proof terms

Isabelle — a generic theorem prover

Meta logic

Intuitionistic higher order logic

- Terms: $t = x \mid c \mid \lambda x :: \tau.\ t \mid t\ t$

- Types: $\tau = \alpha \mid (\tau_1, \ldots, \tau_n) tc$ where $tc \in \{\rightarrow, \text{ prop}, \ \ldots\}$
  first order, Hindley-Milner polymorphism

- Logical connectives:

universal quantification	$\bigwedge$	::	$(\alpha \rightarrow \text{prop}) \rightarrow \text{prop}$
implication	$\Longrightarrow$	::	$\text{prop} \rightarrow \text{prop} \rightarrow \text{prop}$
equality	$\equiv$	::	$\alpha \rightarrow \alpha \rightarrow \text{prop}$

Object logics

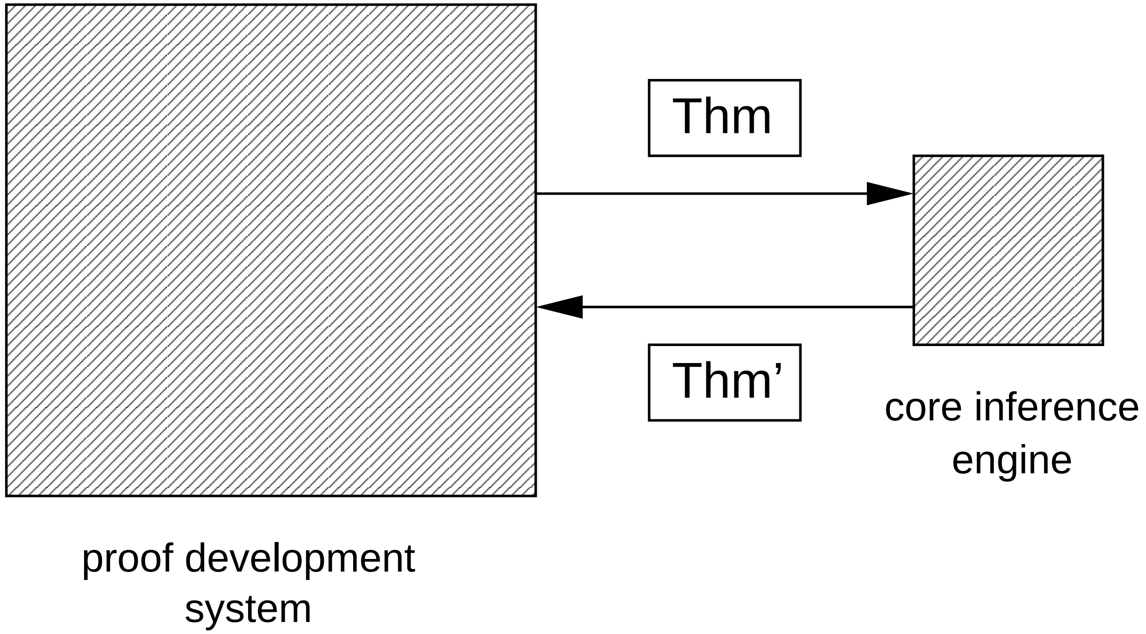
are formalized using meta logic

Formalizing object logics

$$
\begin{array}{lll}
\mathsf{Tr} & :: & \mathsf{bool} \to \mathsf{prop} \\
\longrightarrow & :: & \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \\
\forall & :: & (\alpha \to \mathsf{bool}) \to \mathsf{bool} \\
\exists & :: & (\alpha \to \mathsf{bool}) \to \mathsf{bool}
\end{array}
$$

$$
\begin{array}{llll}
\mathsf{impI} & : & \bigwedge A\ B. & (\mathsf{Tr}\ A \Longrightarrow \mathsf{Tr}\ B) \Longrightarrow \mathsf{Tr}\ (A \longrightarrow B) \\
\mathsf{impE} & : & \bigwedge P\ Q\ R. & \mathsf{Tr}\ (P \longrightarrow Q) \Longrightarrow \mathsf{Tr}\ P \Longrightarrow (\mathsf{Tr}\ Q \Longrightarrow \mathsf{Tr}\ R) \Longrightarrow \mathsf{Tr}\ R \\
\mathsf{allI} & : & \bigwedge P. & (\bigwedge x.\ \mathsf{Tr}\ (P\ x)) \Longrightarrow \mathsf{Tr}\ (\forall x.\ P\ x) \\
\mathsf{allE} & : & \bigwedge P\ x\ R. & \mathsf{Tr}\ (\forall x.\ P\ x) \Longrightarrow (\mathsf{Tr}\ (P\ x) \Longrightarrow \mathsf{Tr}\ R) \Longrightarrow \mathsf{Tr}\ R \\
\mathsf{exI} & : & \bigwedge P\ x. & \mathsf{Tr}\ (P\ x) \Longrightarrow \mathsf{Tr}\ (\exists x.\ P\ x) \\
\mathsf{exE} & : & \bigwedge P\ Q. & \mathsf{Tr}\ (\exists x.\ P\ x) \Longrightarrow (\bigwedge x.\ \mathsf{Tr}\ (P\ x) \Longrightarrow \mathsf{Tr}\ Q) \Longrightarrow \mathsf{Tr}\ Q
\end{array}
$$

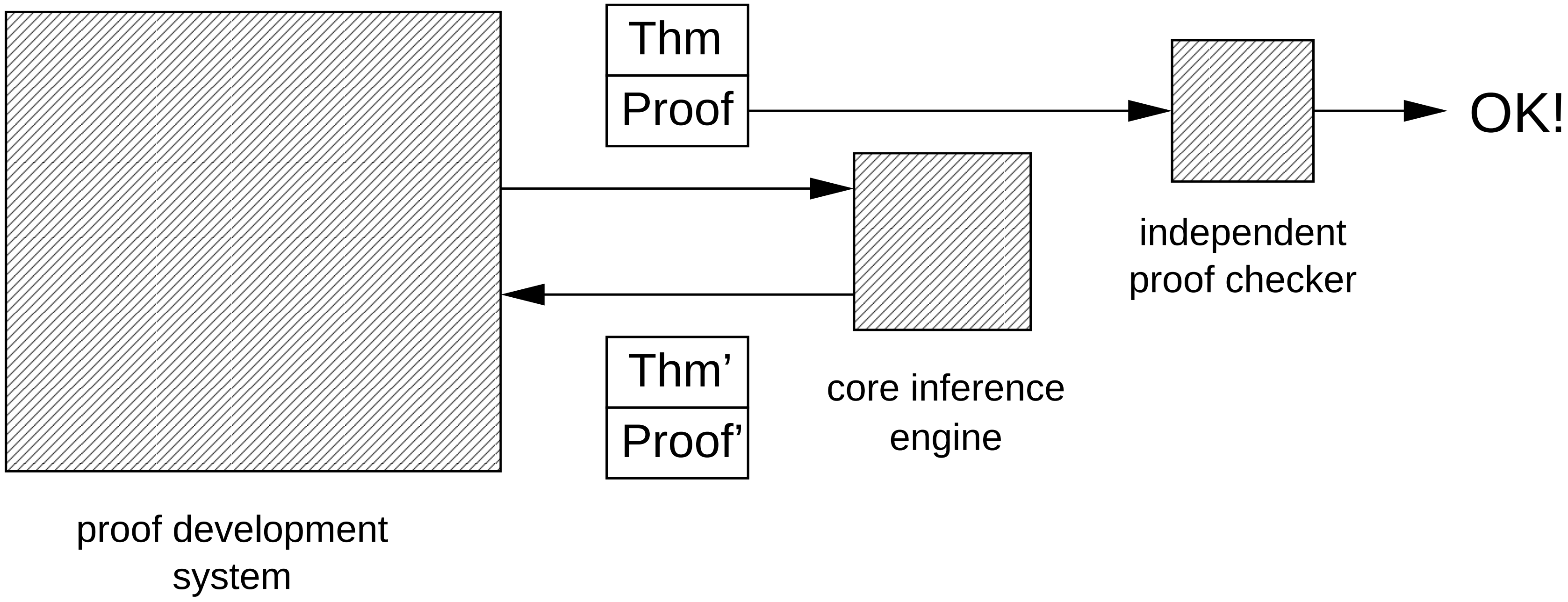
Isabelle's current architecture



- $+$ Correctness ensured by type system of implementation language (abstract data type of theorems, "LCF-approach")
- $+$ Quite efficient

- – Correctness depends on size of core inference engine (in Isabelle: > 2000 lines of code!)
- – No independent proof checking possible

Integrating proof terms into Isabelle



Proof terms – basic concepts

$$p \;=\; h \mid c_{[\overline{\tau_n/\alpha_n}]} \mid \lambda h : \varphi.\ p \mid \lambda x :: \tau.\ p \mid p\ p \mid p\ t$$

Provability

$$\frac{}{\Gamma, h : \varphi, \Gamma' \vdash h : \varphi} \qquad \frac{\Sigma(c) = \varphi}{\Gamma \vdash c_{[\overline{\tau_n/\alpha_n}]} : \varphi[\overline{\tau_n/\alpha_n}]}$$

$$\frac{\Gamma, h : \varphi \vdash p : \psi \quad \Gamma \vdash \varphi :: \mathsf{prop}}{\Gamma \vdash (\lambda h : \varphi.\ p) : \varphi \Longrightarrow \psi} \qquad \frac{\Gamma, x :: \tau \vdash p : \varphi}{\Gamma \vdash (\lambda x :: \tau.\ p) : \bigwedge x :: \tau.\ \varphi} \quad x \notin \mathsf{FV}(\Gamma)$$

$$\frac{\Gamma \vdash p : \varphi \Longrightarrow \psi \quad \Gamma \vdash q : \varphi}{\Gamma \vdash (p\ q) : \psi} \qquad \frac{\Gamma \vdash p : \bigwedge x :: \tau.\ \varphi \quad \Gamma \vdash t :: \tau}{\Gamma \vdash (p\ t) : \varphi[t/x]}$$

Proof construction in Isabelle

- Proof states are represented as theorems

$$\varphi_1 \Longrightarrow \cdots \Longrightarrow \varphi_n \Longrightarrow \varphi$$

where $\varphi_i = \bigwedge \overline{x}.\ \overline{A} \Longrightarrow P$

- To prove φ, start with trivial theorem $\varphi \Longrightarrow \varphi$

- Proof term corresponding to proof state

$$\lambda(g_1 : \varphi_1) \ldots (g_n : \varphi_n).\ \ldots\ (g_i\ \overline{x^i}\ \overline{h^i})\ \ldots$$

Representing meta inference rules

Resolution

$$\frac{P_1 \quad \dots \quad P_m}{C}\, R \qquad \frac{P'_1 \quad \dots \quad P'_{m'}}{C'}\, R' \qquad \longmapsto \qquad \frac{P'_1 \quad \dots \quad P'_{i-1} \quad P_1 \quad \dots \quad P_m \quad P'_{i+1} \quad \dots \quad P'_{m'}}{C'}$$

$$(C = P'_i)$$

proof term:

$$\lambda \overline{q_{i-1}}\ \overline{p_m}.\ R'\ \overline{q_{i-1}}\ (R\ \overline{p_m})$$

Proof by assumption

$$\frac{Q_1 \quad \dots \quad Q_{i-1} \quad \overline{P_n} \Longrightarrow P_j \quad Q_{i+1} \quad \dots \quad Q_m}{C}\, R \quad \longmapsto \quad \frac{Q_1 \quad \dots \quad Q_{i-1} \quad Q_{i+1} \quad \dots \quad Q_m}{C}$$

proof term:

$$\lambda \overline{q_{i-1}}.\ R\ \overline{q_{i-1}}\ (\lambda \overline{p_n}.\ p_j)$$

Representing meta inference rules – continued

Lifting over assumptions

$$\frac{P_1 \quad \ldots \quad P_m}{C}\, R \quad \longmapsto \quad \frac{[\![Q_1 \ \ldots \ Q_n]\!] \Longrightarrow P_1 \quad \ldots \quad [\![Q_1 \ \ldots \ Q_n]\!] \Longrightarrow P_m}{[\![Q_1 \ \ldots \ Q_n]\!] \Longrightarrow C}$$

proof term:

$$\lambda \overline{r_m}\ \overline{q_n}.\ R\ (\overline{r_m\ \overline{q_n}})$$

Lifting over parameters

$$\frac{P_1\,[\overline{a_k}] \quad \ldots \quad P_m\,[\overline{a_k}]}{C\,[\overline{a_k}]}\, R\,[\overline{a_k}] \quad \longmapsto \quad \frac{\bigwedge \overline{x_n}.\ P_1 \left[\overline{a'_k\ \overline{x_n}}\right] \quad \ldots \quad \bigwedge \overline{x_n}.\ P_m \left[\overline{a'_k\ \overline{x_n}}\right]}{\bigwedge \overline{x_n}.\ C \left[\overline{a'_k\ \overline{x_n}}\right]}$$

proof term:

$$\lambda \overline{r_m}\ \overline{x_n}.\ R \left[\overline{a'_k\ \overline{x_n}}\right]\ (\overline{r_m\ \overline{x_n}})$$

Constructing an example proof

Goal: $(\exists x.\ \forall y.\ P\ x\ y) \longrightarrow (\forall y.\ \exists x.\ P\ x\ y)$

Initial proof state: the trivial theorem

$$\lambda g : ((\exists x.\ \forall y.\ P\ x\ y) \longrightarrow (\forall y.\ \exists x.\ P\ x\ y)).\ g$$

Step 1: $\longrightarrow$ introduction

$$\begin{array}{ll}
\lambda g : (\exists x.\ \forall y.\ P\ x\ y) \Longrightarrow (\forall y.\ \exists x.\ P\ x\ y). & \\
\quad (\lambda g'.\ g') & \}\ \text{initial proof term} \\
\qquad (\mathsf{impl}\ (\exists x.\ \forall y.\ P\ x\ y)\ (\forall y.\ \exists x.\ P\ x\ y) & \quad \}\ \text{instance of impl} \\
\qquad\quad g) &
\end{array}$$

$$\longrightarrow_{\beta\eta}$$

$$\mathsf{impl}\ (\exists x.\ \forall y.\ P\ x\ y)\ (\forall y.\ \exists x.\ P\ x\ y)$$

Constructing an example proof – continued

Step 2: $\forall$ introduction

$$
\begin{array}{l}
\lambda g : (\bigwedge y.\ \exists x.\ \forall y.\ P\ x\ y \Longrightarrow \exists x.\ P\ x\ y).\\
\quad \left.\mathsf{impl}\ (\exists x.\ \forall y.\ P\ x\ y)\ (\forall y.\ \exists x.\ P\ x\ y)\right\}\ \text{proof term from step 1}\\
\qquad \left.\begin{array}{l}
((\lambda h_2 : (\exists x.\ \forall y.\ P\ x\ y \Longrightarrow \bigwedge y.\ \exists x.\ P\ x\ y).\\
\quad \lambda h_1 : (\exists x.\ \forall y.\ P\ x\ y).\\
\qquad \mathsf{allI}\ (\lambda y.\ \exists x.\ P\ x\ y)\ (h_2\ h_1))
\end{array}\right\}\ \text{lifted instance of allI}\\
\qquad\qquad\quad \left.\begin{array}{l}
(\lambda h_3 : (\exists x.\ \forall y.\ P\ x\ y)\\
\quad \lambda y :: \beta.\ g\ y\ h_3))
\end{array}\right\}\ \text{rearranging quantifiers}
\end{array}
$$

$\longrightarrow_\beta$

$$
\begin{array}{l}
\lambda g : (\bigwedge y.\ \exists x.\ \forall y.\ P\ x\ y \Longrightarrow \exists x.\ P\ x\ y).\\
\quad \mathsf{impl}\ (\exists x.\ \forall y.\ P\ x\ y)\ (\forall y.\ \exists x.\ P\ x\ y)\\
\qquad (\lambda h_1 : (\exists x.\ \forall y.\ P\ x\ y).\\
\qquad\quad \mathsf{allI}\ (\lambda y.\ \exists x.\ P\ x\ y)\ (\lambda y :: \beta.\ g\ y\ h_1))
\end{array}
$$

...

Reconstruction of partial proof terms

- What can be omitted?

$$p \;=\; h \mid c_{[\overline{\tau_n/\alpha_n}]} \mid \lambda h : \varphi.\; p \mid \lambda x :: \tau.\; p \mid p\; p \mid p\; t$$

- Model missing information by unification variables:

$\lambda x{::}\tau.\ (p\,t) \qquad \leadsto \qquad \lambda x{::}?\alpha.\ (p\,(?f\ x))$ Note: "lifting" of variables!

- Proof reconstruction judgement: $\Gamma \vdash p \rhd (\varphi,\ C)$

"p proves φ (in context Γ), provided that the constraints C are solvable"

- C contains equality constraints between terms $(t_1 =^? t_2)$ and types $(\tau_1 =^? \tau_2)$
- Similar: Type reconstruction judgement: $\Gamma \vdash t \rhd (\tau,\ D)$
- Note: τ only first order: type reconstruction decidable!

Reconstruction of partial proof terms

$$\frac{}{\Gamma', h : \varphi, \Gamma \vdash h \rhd (\varphi,\ \emptyset)} \qquad \frac{\Sigma(c) = \varphi}{\Gamma \vdash c_{[\overline{\tau_n/\alpha_n}]} \rhd (\varphi[\overline{\tau_n/\alpha_n}],\ \emptyset)}$$

$$\frac{\Gamma, h : \varphi \vdash p \rhd (\psi,\ C) \quad \Gamma \vdash \varphi \rhd (\tau,\ D)}{\Gamma \vdash (\lambda h{:}\varphi.\ p) \rhd (\varphi \Longrightarrow \psi,\ C \cup D \cup \{\tau =^? \mathsf{prop}\})}$$

$$\frac{\Gamma, x :: \tau \vdash p \rhd (\varphi,\ C)}{\Gamma \vdash (\lambda x{::}\tau.\ p) \rhd (\bigwedge x :: \tau.\ \varphi,\ \{\lambda x :: \tau.\ r =^? \lambda x :: \tau.\ s \mid (r =^? s) \in C_t\} \cup C_\tau)}$$

$$\frac{\Gamma \vdash p \rhd (\varphi,\ C) \quad \Gamma \vdash q \rhd (\psi,\ D)}{\Gamma \vdash (p\ q) \rhd (?f_{\overline{\tau_\Gamma} \to \mathsf{prop}}\ \overline{V_\Gamma},\ \{\varphi =^? (\psi \Longrightarrow ?f_{\overline{\tau_\Gamma} \to \mathsf{prop}}\ \overline{V_\Gamma})\} \cup C \cup D)}$$

$$\frac{\Gamma \vdash p \rhd (\varphi,\ C) \quad \Gamma \vdash t \rhd (\tau,\ D)}{\Gamma \vdash (p\ t) \rhd (?f_{\overline{\tau_\Gamma} \to \tau \to \mathsf{prop}}\ \overline{V_\Gamma}\ t,\ \{\varphi =^? \bigwedge x :: \tau.\ ?f_{\overline{\tau_\Gamma} \to \tau \to \mathsf{prop}}\ \overline{V_\Gamma}\ x\} \cup C \cup D)}$$

Example

Let

$$p = \lambda x :: ?\alpha_1.\ \lambda h_1 : (?f_{?\alpha_2}\ x).\ \lambda y :: ?\alpha_3.\ \lambda h_2 : (?f'_{?\alpha_4}\ x\ y).\ h_1\ h_2\ y$$

Then

$$\begin{aligned} collect([], p) \quad = \quad (& \bigwedge x :: ?\alpha_1.\ ?f_{?\alpha_2}\ x \Longrightarrow \bigwedge y :: ?\alpha_3.\ ?f'_{?\alpha_4}\ x\ y \Longrightarrow \\ & \quad ?g'_{?\alpha_1 \to ?\alpha_3 \to ?\alpha_3 \to \mathsf{prop}}\ x\ y\ y, \\ & \{\ \lambda x :: ?\alpha_1.\lambda y :: ?\alpha_3.\ ?f_{?\alpha_2}\ x\ \ =^?\\ & \quad \lambda x :: ?\alpha_1.\lambda y :: ?\alpha_3.\ ?f'_{?\alpha_4}\ x\ y \Longrightarrow ?g_{?\alpha_1 \to ?\alpha_3 \to \mathsf{prop}}\ x\ y, \\ & \quad \lambda x :: ?\alpha_1.\lambda y :: ?\alpha_3.\ ?g_{?\alpha_1 \to ?\alpha_3 \to \mathsf{prop}}\ x\ y\ \ =^? \\ & \quad \lambda x :: ?\alpha_1.\lambda y :: ?\alpha_3.\ \bigwedge z :: ?\alpha_3.\ ?g'_{?\alpha_1 \to ?\alpha_3 \to ?\alpha_3 \to \mathsf{prop}}\ x\ y\ z, \\ & \quad ?\alpha_2 =^? ?\alpha_1 \to ?\beta_1,\ ?\beta_1 =^? \mathsf{prop} \\ & \quad ?\alpha_4 =^? ?\alpha_1 \to ?\alpha_3 \to ?\beta_2,\ ?\beta_2 =^? \mathsf{prop}\}) \end{aligned}$$

Compression of proof terms

- Idea:

  1. Replace all terms / types by variables, remember original terms
  2. Collect constraints
  3. Simulate constraint solving
     If constraint is not solvable using pattern unification, replace non-pattern variables by original terms, mark these variables as needed

- Complication: unification may introduce new variables!
  Solution: record dependencies between variables

$$\begin{array}{lll} \mathit{compress}(p,\varphi) = & \texttt{let} & (p',\theta) = \mathit{varify}([],[],p) \\ & & (\varphi',C) = \mathit{collect}([],p') \\ & & V = \mathit{solves}(C \cup \{\varphi =^? \varphi'\},\theta) \\ & \texttt{in} & \theta|_V(p') \end{array}$$

Compression of proof terms

$$
\begin{array}{rcl}
\mathit{varify}(\overline{x}, \overline{\tau}, \lambda v{:}\varphi.\ p) & = & \mathtt{let}\ (p', \theta) = \mathit{varify}(\overline{x}, \overline{\tau}, p) \\
 & & \mathtt{in}\ (\lambda v{:}(?f_{\overline{\tau} \to \mathsf{prop}}\ \overline{x}).\ p',\ \theta \cup \{?f \mapsto \lambda \overline{x}.\varphi\}) \\
\mathit{varify}(\overline{x}, \overline{\tau}, \lambda y{::}\tau.\ p) & = & \mathtt{let}\ (p', \theta) = \mathit{varify}(\overline{x}y, \overline{\tau}\,?\alpha, p) \\
 & & \mathtt{in}\ (\lambda y{::}?\alpha.\ p',\ \theta) \\
\mathit{varify}(\overline{x}, \overline{\tau}, (p\ q)) & = & \mathtt{let}\ (p', \theta) = \mathit{varify}(\overline{x}, \overline{\tau}, p);\ (q', \theta') = \mathit{varify}(\overline{x}, \overline{\tau}, q) \\
 & & \mathtt{in}\ ((p'\ q'), \theta \cup \theta') \\
\mathit{varify}(\overline{x}, \overline{\tau}, (p\ t)) & = & \mathtt{let}\ (p', \theta) = \mathit{varify}(\overline{x}, \overline{\tau}, p) \\
 & & \mathtt{in}\ ((p'\ (?f_{\overline{\tau} \to ?\beta}\ \overline{x})),\ \theta \cup \{?f \mapsto \lambda \overline{x}.t\}) \\
\mathit{varify}(\overline{x}, \overline{\tau}, c_{[\overline{\tau_n/\alpha_n}]}) & = & (c_{[\overline{?\alpha_n/\alpha_n}]}, \emptyset)
\end{array}
$$

Compression of proof terms

Function $\boldsymbol{solves}(C, \theta)$

$V := \emptyset; D := \{?f \mapsto ?f \mid ?f \in \mathcal{V}ar(C)\}$

$\texttt{while}\ C \neq \emptyset\ \texttt{do}$

$\quad \texttt{if}$ there is a pattern problem $(s =^? t) \in C$

$\quad \texttt{then}\quad \sigma := mgu(s, t);\ C := \sigma(C - \{s =^? t\});$

$\quad\quad\quad \texttt{forall}\ (?f \mapsto t) \in \sigma\ \texttt{do}$

$\quad\quad\quad\quad \delta := mgu(\theta(?f), \theta(t));\ \theta := \theta \cup \delta$

$\quad\quad\quad\quad D := D \cup \{?h \mapsto D(?f) \mid ?h \in \mathcal{D}om(\delta)\}$

$\quad\quad\quad \texttt{od}$

$\quad \texttt{else}\quad$ pick some $st \in C$ such that $\mathcal{NPV}ars(st) \subseteq \mathcal{D}om(\theta);$

$\quad\quad\quad V := V \cup \{D(?f) \mid ?f \in \mathcal{NPV}ars(st)\};$

$\quad\quad\quad C := \theta|_{\mathcal{NPV}ars(st)}(C);$

$\texttt{od};$

$\texttt{return}\ V$

where e.g. $\mathcal{NPV}ars(\{\lambda x.?f\ (?g\ x) =^? \lambda x.?h\ x\}) = \{?f\}$

Constraint collection and compression — some theorems

Soundness and completeness of constraint collection

1. If $\Gamma \vdash p \triangleright (\varphi',\ C)$ and θ solves C then $\theta(\Gamma) \vdash \theta(p) : \theta(\varphi')$.

2. If $\Gamma \vdash p : \varphi$ and $\Gamma \vdash p \triangleright (\varphi',\ C)$ then $C \cup \{\varphi =^? \varphi'\}$ is solvable.

Correctness of compression

Let φ be ground. If $\vdash p : \varphi$, $q = compress(p, \varphi)$ and $(\psi, C) = collect([], q)$ then

1. $C \cup \{\varphi =^? \psi\}$ is solvable by pattern unification and

2. if θ solves $C \cup \{\varphi =^? \psi\}$ then $\vdash \theta(q) : \varphi$.

Conclusion

Achievements

- Proof terms for simply-typed higher order logic
- Integrated into theorem prover Isabelle
- Quite effective compression: over 90% of terms can be omitted

Future plans

- Special support for proofs involving equality
- Alternative compression approach: combination of static / dynamic compression?
- Applications:
  - Program extraction
  - Proof transformation / theory morphisms
  - Linking Isabelle with other provers
  - Proof-Carrying-Code