Higher order proof reconstruction from paramodulation-based refutations: the unit equality case

Andrea Asperti and Enrico Tassi

Department of Computer Science, University of Bologna

28-30 June 2007

Context

What we had:

- Matita is an ITP developed at the university of Bologna
- Lack of automation is one of the most reported issues of ITPs
- ATP are effective tools, but usually do not provide a proof object, sometimes a minimalistic trace mainly for efficiency (both space and time).
- Matita follows the independent verification principle: we need a real CIC proof
- We implemented our own first order paramodulation based automatic theorem prover (that, of course, provides a good trace), restricted to the unit equality case.

Context

What we had:

- Matita is an ITP developed at the university of Bologna
- Lack of automation is one of the most reported issues of ITPs
- ATP are effective tools, but usually do not provide a proof object, sometimes a minimalistic trace mainly for efficiency (both space and time).
- Matita follows the independent verification principle: we need a real CIC proof
- We implemented our own first order paramodulation based automatic theorem prover (that, of course, provides a good trace), restricted to the unit equality case.
- Why your own prover?

Context

What we had:

- Matita is an ITP developed at the university of Bologna
- Lack of automation is one of the most reported issues of ITPs
- ATP are effective tools, but usually do not provide a proof object, sometimes a minimalistic trace mainly for efficiency (both space and time).
- Matita follows the independent verification principle: we need a real CIC proof
- We implemented our own first order paramodulation based automatic theorem prover (that, of course, provides a good trace), restricted to the unit equality case.
- Why your own prover? For fun :-)

Aim

What we want:

- To be able to read the proofs:
  - To understand what the automatic procedure did.
  - Nice natural language rendering using MoWGLI tech.

$$\begin{aligned} a &= b \quad by \quad lemma\ 1 \\ &= c \quad by \quad lemma\ 2 \\ &= d \quad by \quad lemma\ 3 \end{aligned}$$

Aim

What we want:

- To be able to read the proofs:
  - To understand what the automatic procedure did.
  - Nice natural language rendering using MoWGLI tech.

$$\begin{aligned} a &= b \quad by \quad lemma\ 1 \\ &= c \quad by \quad lemma\ 2 \\ &= d \quad by \quad lemma\ 3 \end{aligned}$$

- Save earth from overheating
  - Fast to typecheck
  - Not re-doing proof search every time we compile a file

Plan

The plan:

1. Start from the trace of the prover
2. Transform it into a CIC object
3. Apply type preserving transformations to obtain a nice proof object suitable for point 4.
4. Render it in natural language (re-using MoWGLI/Matita rendering facility)
5. Thanks to C. Sacerdoti Coen declarative language, the printed proof is a re-executable script (PLMMS talk).

Outline

- Equality in CIC
- Superposition rules
- Proof reconstruction
- Demo
- Conclusion

Equality in CIC

- Not built in, but an inductive predicate with one constructor:

$$\mathit{refl\_eq} : x =_A x$$

- As any inductive type, comes with an eliminator in two flavours:

$$\frac{h : P\ a_1 \qquad k : a_1 =_A a_2}{(\mathrm{eq\_ind}\ A\ a_1\ P\ h\ a_2\ k) : P\ a_2}$$

$$\frac{h : P\ a_2 \qquad k : a_1 =_A a_2}{(\mathrm{eq\_ind\_r}\ A\ a_2\ P\ h\ a_1\ k) : P\ a_1}$$

Superposition rules

- Superposition left (backward reasoning)

$$\frac{\vdash l =_A r \qquad t =_B s \vdash C}{t[r]_p\sigma =_B s\sigma \vdash C\sigma}$$

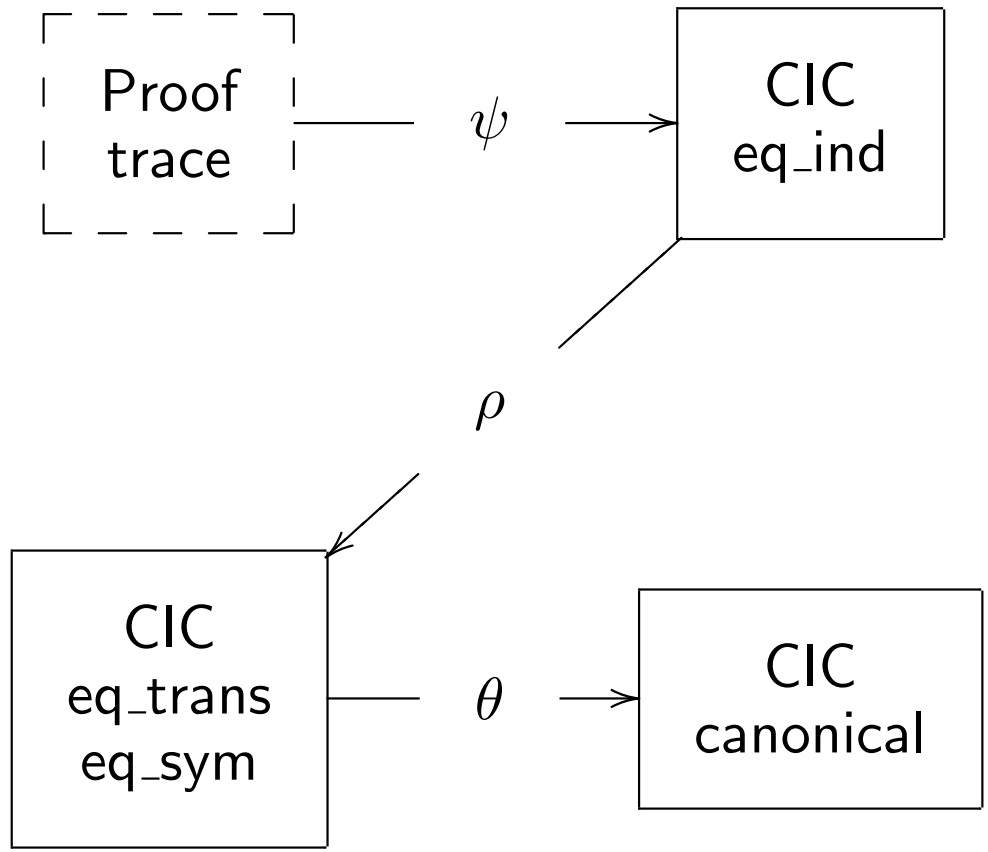
- Superposition right (forward reasoning)

$$\frac{\vdash l =_A r \qquad \vdash t =_B s}{\vdash t[r]_p\sigma =_B s\sigma}$$

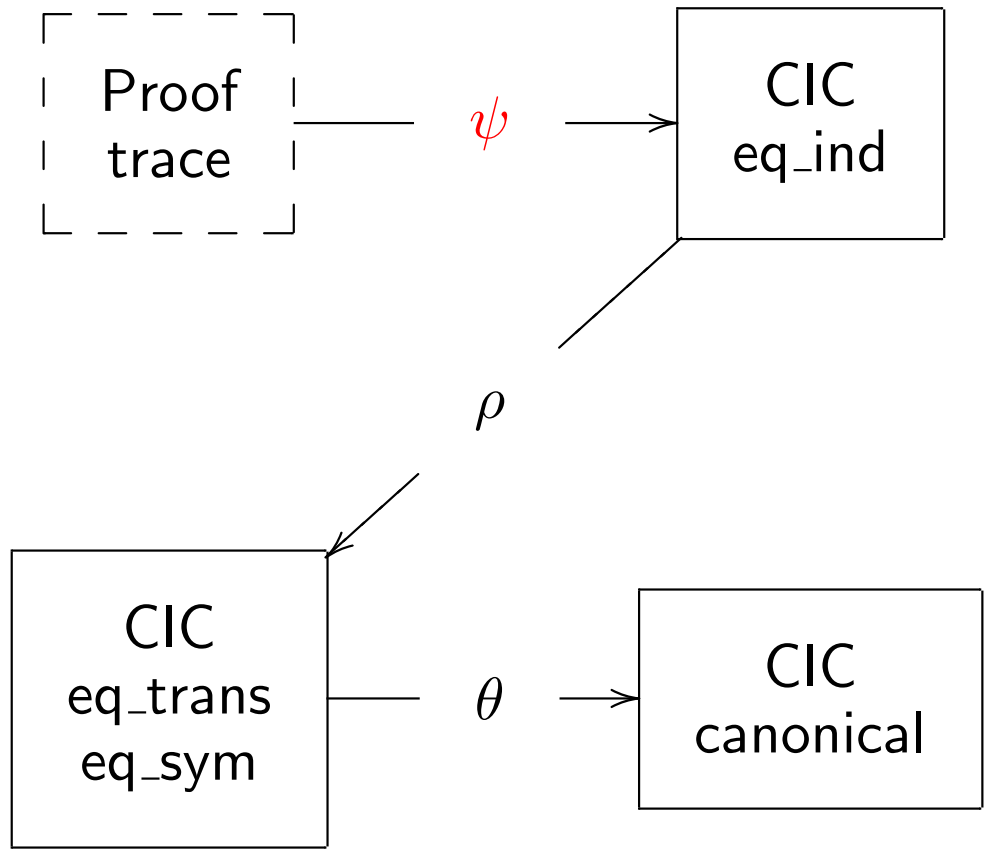
- Equality resolution

$$\frac{t =_A s \vdash C}{\vdash C\sigma}$$

Data flow (1/3)


Proof
trace
ψ
CIC
eq_ind
ρ
CIC
eq_trans
eq_sym
θ
CIC
canonical

Data flow (1/3)


Proof
trace
ψ
CIC
eq_ind
ρ
CIC
eq_trans
eq_sym
θ
CIC
canonical

Superposition rules with CIC proofs

- Superposition left (backward reasoning)

$$\frac{\vdash \quad l =_A r \qquad\qquad t =_B s \vdash \quad C}{t[r]_p\sigma =_B s\sigma \vdash \qquad\qquad C\sigma}$$

- Superposition right (forward reasoning)

$$\frac{\vdash \quad l =_A r \qquad \vdash \quad t =_B s}{\vdash \quad t[r]_p\sigma =_B s\sigma}$$

- Equality resolution

$$\frac{t =_A s \vdash \quad C}{\vdash \qquad\qquad C\sigma}$$

Superposition rules with CIC proofs

- Superposition left (backward reasoning)

$$\frac{\vdash h : l =_A r \qquad \alpha : t =_B s \vdash M : C}{\beta : t[r]_p\sigma =_B s\sigma \vdash M\sigma[R/\alpha\sigma] : C\sigma}$$

$$R = (\mathrm{eq\_ind\_r}\ A\ r\sigma\ (\lambda x : A.t[x]_p =_B s)\sigma\ \beta\ l\sigma\ h\sigma) : t\sigma =_B s\sigma$$

- Superposition right (forward reasoning)

$$\frac{\vdash \quad l =_A r \qquad \vdash \quad t =_B s}{\vdash \quad t[r]_p\sigma =_B s\sigma}$$

- Equality resolution

$$\frac{t =_A s \vdash \quad C}{\vdash \qquad\qquad C\sigma}$$

Superposition rules with CIC proofs

- Superposition left (backward reasoning)

$$\frac{\vdash h : l =_A r \qquad \alpha : t =_B s \vdash M : C}{\beta : t[r]_p\sigma =_B s\sigma \vdash M\sigma[R/\alpha\sigma] : C\sigma}$$

$$R = (\mathrm{eq\_ind\_r}\ A\ r\sigma\ (\lambda x : A.t[x]_p =_B s)\sigma\ \beta\ l\sigma\ h\sigma) : t\sigma =_B s\sigma$$

- Superposition right (forward reasoning)

$$\frac{\vdash h : l =_A r \qquad \vdash k : t =_B s}{\vdash R : t[r]_p\sigma =_B s\sigma}$$

$$R = (\mathrm{eq\_ind}\ A\ l\sigma\ (\lambda x : A.t[x]_p =_B s)\sigma\ k\sigma\ r\sigma\ h\sigma) : t[r]_p\sigma =_B s\sigma$$

- Equality resolution

$$\frac{t =_A s \vdash \qquad C}{\vdash \qquad C\sigma}$$

Superposition rules with CIC proofs

- Superposition left (backward reasoning)

$$\frac{\vdash h : l =_A r \qquad \alpha : t =_B s \vdash M : C}{\beta : t[r]_p\sigma =_B s\sigma \vdash M\sigma[R/\alpha\sigma] : C\sigma}$$

$$R = (\mathrm{eq\_ind\_r}\ A\ r\sigma\ (\lambda x : A.t[x]_p =_B s)\sigma\ \beta\ l\sigma\ h\sigma) : t\sigma =_B s\sigma$$

- Superposition right (forward reasoning)

$$\frac{\vdash h : l =_A r \qquad \vdash k : t =_B s}{\vdash R : t[r]_p\sigma =_B s\sigma}$$

$$R = (\mathrm{eq\_ind}\ A\ l\sigma\ (\lambda x : A.t[x]_p =_B s)\sigma\ k\sigma\ r\sigma\ h\sigma) : t[r]_p\sigma =_B s\sigma$$

- Equality resolution

$$\frac{\alpha : t =_A s \vdash M : C}{\vdash M[\mathrm{refl\_eq}\ A\ t\sigma/\alpha] : C\sigma}$$

Superposition rules with CIC proofs

- Superposition left (backward reasoning)

$$\frac{\vdash h : l =_A r \qquad \alpha : t =_B s \vdash M : C}{t[r]_p\sigma =_B s\sigma \vdash \qquad C\sigma}$$

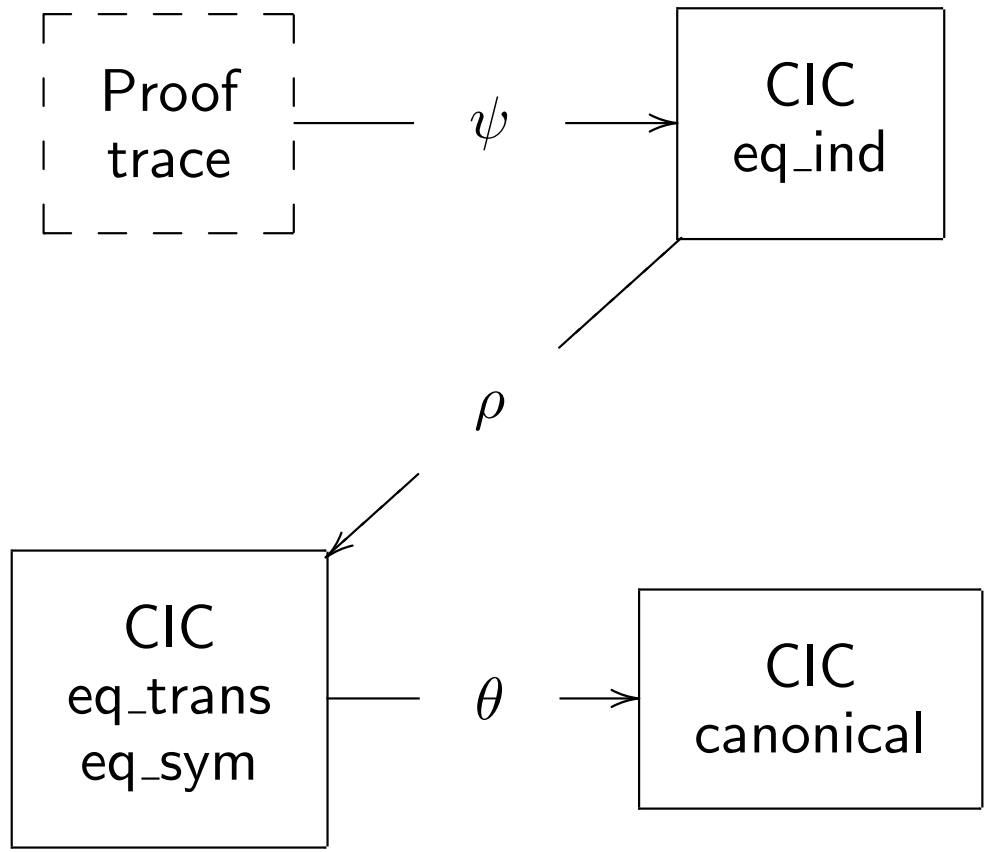
- Superposition right (forward reasoning)

$$\frac{\vdash h : l =_A r \qquad \vdash k : t =_B s}{\vdash \qquad t[r]_p\sigma =_B s\sigma}$$

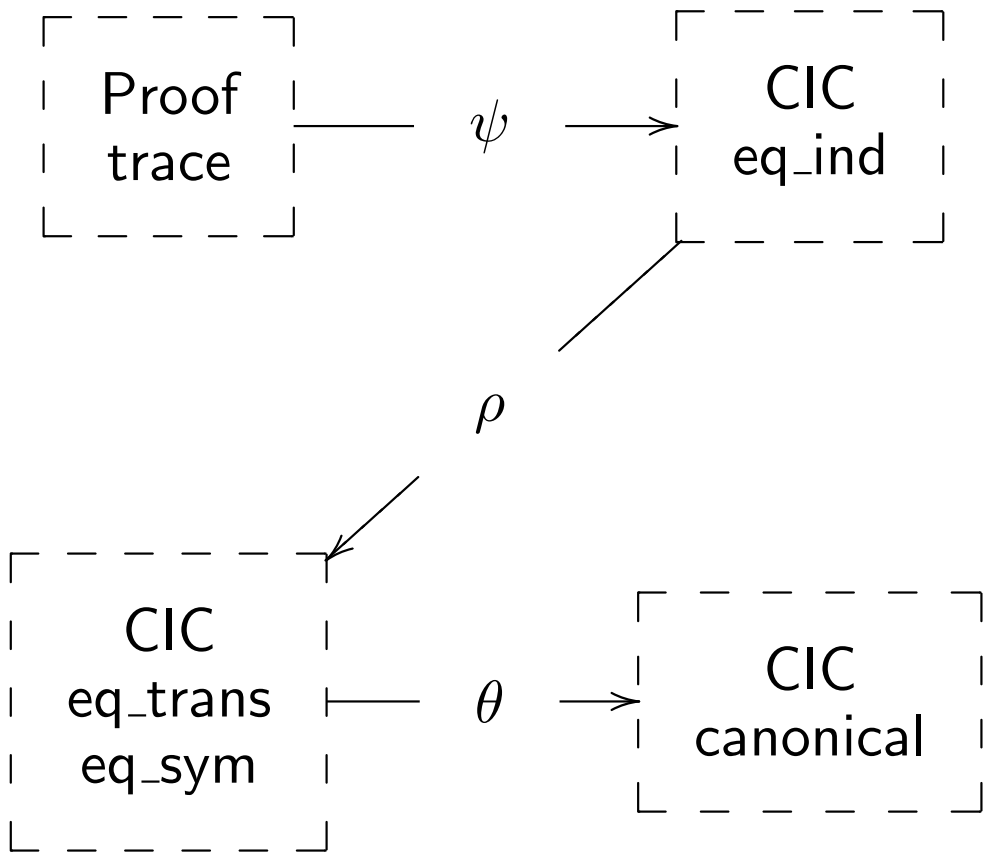
- Equality resolution

$$\frac{\alpha : t =_A s \vdash \qquad C}{\vdash \qquad C\sigma}$$

Data flow (2/3)


Proof
trace
ψ
CIC
eq_ind
ρ
CIC
eq_trans
eq_sym
θ
CIC
canonical

Data flow (2/3)


Proof
trace
ψ
CIC
eq_ind
ρ
CIC
eq_trans
eq_sym
θ
CIC
canonical

Data flow (2/3)

Problem	Search	Steps	Typing		Proof size	
			raw	opt	raw	opt
BOO069-1	2.15	27	79.50		3.1M	
BOO071-1	2.23	27	203.03		5.4M	
GRP118-1	0.11	17	7.66		546K	
GRP485-1	0.17	47	323.35		5.1M	
LAT008-1	0.48	40	22.56		933K	
LCL115-2	0.81	52	24.42		1.1M	

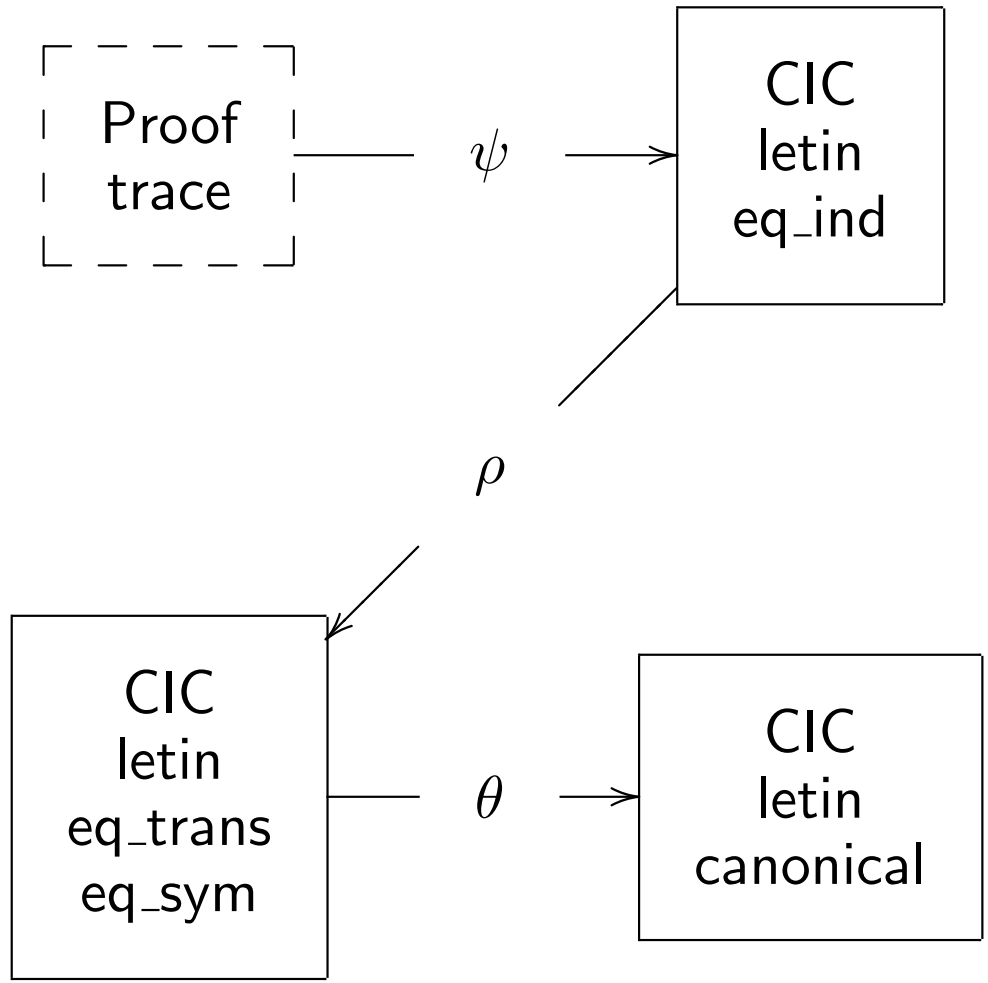
Tab. 1. Timing (in seconds) and proof size

Data flow (2/3)

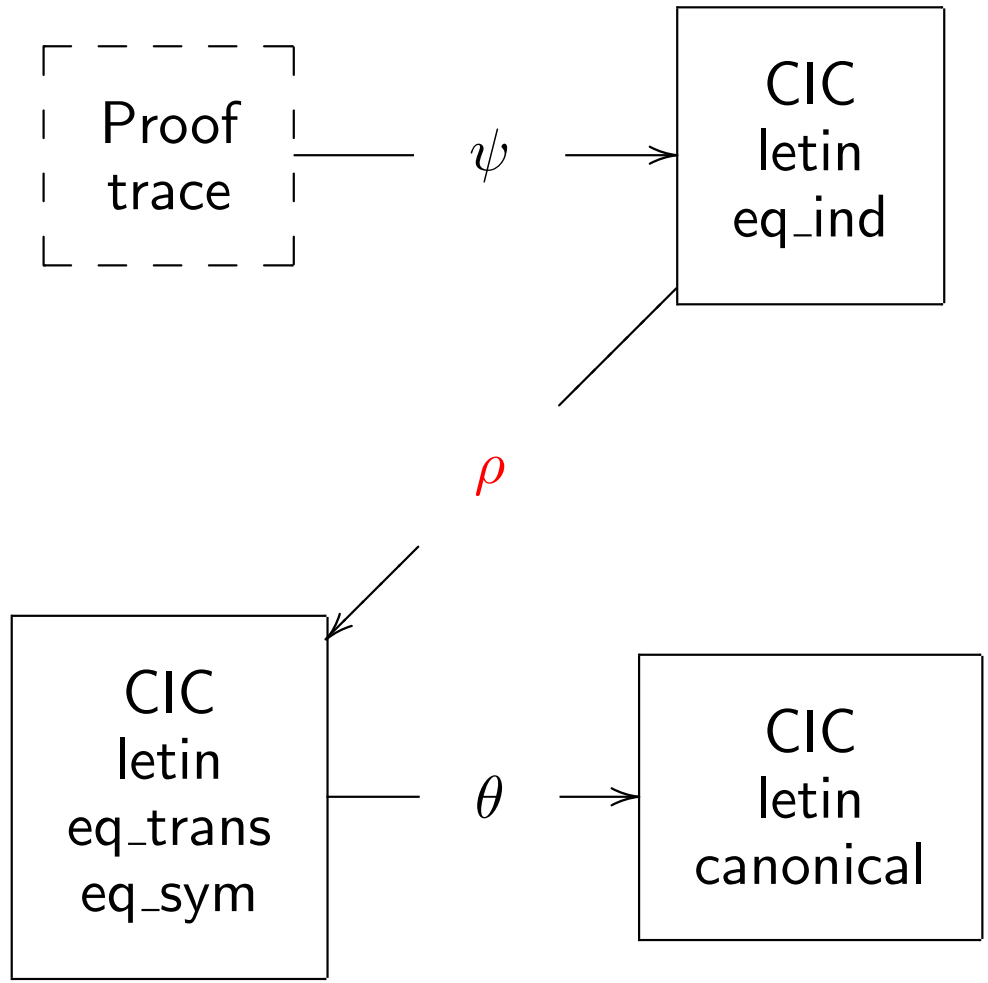
Problem	Search	Steps	Typing		Proof size	
			raw	opt	raw	opt
BOO069-1	2.15	27	79.50	0.23	3.1M	29K
BOO071-1	2.23	27	203.03	0.22	5.4M	28K
GRP118-1	0.11	17	7.66	0.13	546K	21K
GRP485-1	0.17	47	323.35	0.23	5.1M	33K
LAT008-1	0.48	40	22.56	0.12	933K	19K
LCL115-2	0.81	52	24.42	0.29	1.1M	37K

Tab. 1. Timing (in seconds) and proof size

Data flow (2/3)


Proof
trace
ψ
CIC
letin
eq_ind
ρ
CIC
letin
eq_trans
eq_sym
θ
CIC
letin
canonical

Data flow (2/3)


Proof
trace
ψ
CIC
letin
eq_ind
ρ
CIC
letin
eq_trans
eq_sym
θ
CIC
letin
canonical

Normal form

Given the following standard lemmas:

$$\begin{aligned}
&\mathrm{trans} : \forall A : \mathit{Type}.\forall x, y, z : A.x =_A y \to y =_A z \to x =_A z \\
&\mathrm{sym} : \forall A : \mathit{Type}.\forall x, y : A.x =_A y \to y =_A x \\
&\mathrm{eq_f} : \forall A, B : \mathit{Type}.\forall f : A \to B.\forall x, y : A.x =_A y \to (f\ x) =_B (f\ y)
\end{aligned}$$

Definition (Proof normal form)

$$\begin{array}{rcl}
\pi & = & \mathrm{eq_f}\ B\ C\ \Delta\ a\ b\ \mathit{axiom} \\
& | & \mathrm{eq_f}\ B\ C\ \Delta\ a\ b\ (\mathrm{sym}\ B\ b\ a\ \mathit{axiom}) \\
& | & \mathrm{trans}\ A\ a\ b\ c\ \pi\ \pi
\end{array}$$

ρ, CIC eq_ind$\to$ CIC trans sym

$$\rho(\pi) \rightsquigarrow \rho'(\lambda x{:}B.x,\ \pi) \qquad \text{when } \pi : a =_B b$$

$$\rho'(\Delta,\ \text{eq\_ind}\ A\ a\ (\lambda x.\Gamma[x] =_B m)\ \pi_1\ b\ \pi_2) \rightsquigarrow$$
$$\text{trans}\ C\ (\Delta \circ \Gamma)[b]\ (\Delta \circ \Gamma)[a]\ \Delta[m]$$
$$(\text{sym}\ C\ (\Delta \circ \Gamma)[a]\ (\Delta \circ \Gamma)[b]\ \rho'(\Delta \circ \Gamma,\ \pi_2))\ \rho'(\Delta,\ \pi_1)$$

$$\rho'(\Delta,\ \text{eq\_ind\_r}\ A\ a\ (\lambda x.\Gamma[x] =_B m)\ \pi_1\ b\ \pi_2) \rightsquigarrow$$
$$\text{trans}\ C\ (\Delta \circ \Gamma)[b]\ (\Delta \circ \Gamma)[a]\ \Delta[m]\ \rho'(\Delta \circ \Gamma,\ \pi_2)\ \rho'(\Delta,\ \pi_1)$$

$$\rho'(\Delta,\ \text{eq\_ind}\ A\ a\ (\lambda x.m =_B \Gamma[x])\ \pi_2\ b\ \pi_1) \rightsquigarrow$$
$$\text{trans}\ C\ \Delta[m]\ (\Delta \circ \Gamma)[a]\ (\Delta \circ \Gamma)[b]\ \rho'(\Delta,\ \pi_2)\ \rho'(\Delta \circ \Gamma,\ \pi_1)$$

$$\rho'(\Delta,\ \text{eq\_ind\_r}\ A\ a\ (\lambda x.m =_B \Gamma[x])\ \pi_1\ b\ \pi_2) \rightsquigarrow$$
$$\text{trans}\ C\ \Delta[m]\ (\Delta \circ \Gamma)[a]\ (\Delta \circ \Gamma)[b]$$
$$\rho'(\Delta,\ \pi_1)\ (\text{sym}\ C\ (\Delta \circ \Gamma)[b]\ (\Delta \circ \Gamma)[a]\ \rho'(\Delta \circ \Gamma,\ \pi_2))$$

$$\rho'(\Delta,\ \pi) \rightsquigarrow \text{eq\_f}\ B\ C\ \Delta\ a\ b\ \pi \qquad \text{when } \pi : a =_B b \text{ and } \Delta : B \to C$$

Theorem 1: ρ is type preserving

if $\Delta : B \to C$ and $\pi : x =_B y$, then $\rho'(\Delta,\ \pi) : \Delta[x] =_C \Delta[y]$
By induction on the size of π

$$\frac{\Gamma : A \to B \qquad \pi_1 : \Gamma[a] =_B m \qquad \pi_2 : a =_A b}{\rho'(\Delta,\ \mathrm{eq\_ind}\ A\ a\ (\lambda x.\Gamma[x] =_B m)\ \pi_1\ b\ \pi_2 : \Gamma[b] =_B m)}$$

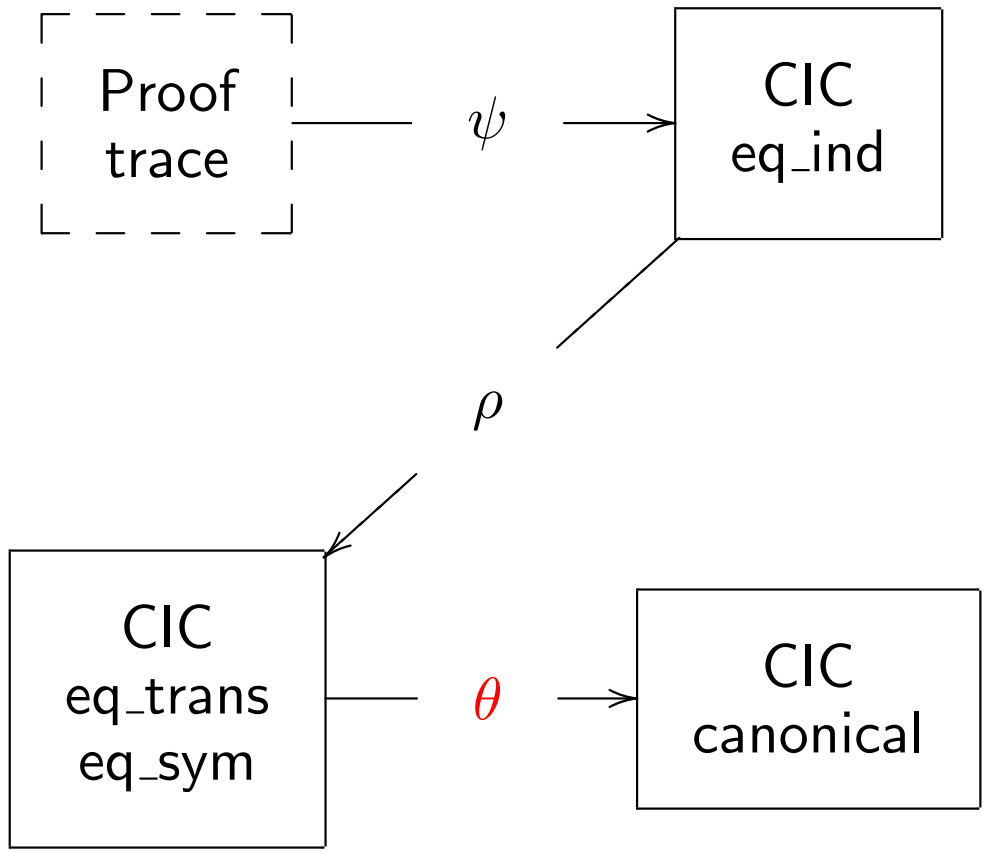
$$\Rightarrow$$

$$\frac{\rho'(\Delta \circ \Gamma,\ \pi_2) : \Delta[\Gamma[a]] =_C \Delta[\Gamma[b]]}{P \equiv (\mathrm{sym}\ \ C\ \ \Delta[\Gamma[a]]\ \ \Delta[\Gamma[b]]\ \ \rho'(\Delta \circ \Gamma,\ \pi_2)) : \Delta[\Gamma[b]] =_C \Delta[\Gamma[a]]}$$

$$\frac{\pi_1 : \Gamma[a] =_B m}{Q \equiv \rho'(\Delta,\ \pi_1) : \Delta[\Gamma[a]] =_C \Delta[m]}$$

$$\frac{P : \Delta[\Gamma[b]] =_C \Delta[\Gamma[a]] \qquad Q : \Delta[\Gamma[a]] =_C \Delta[m]}{\mathrm{trans}\ \ C\ \ \Delta[\Gamma[b]]\ \ \Delta[\Gamma[a]]\ \ \Delta[m]\ \ P\ \ Q : \Delta[\Gamma[b]] =_C \Delta[m]}$$

Data flow (3/3)


Proof
trace
ψ
CIC
eq_ind
ρ
CIC
eq_trans
eq_sym
θ
CIC
canonical

θ, CIC trans sym$\longrightarrow$ CIC canonical

Pushing sym up.

$$\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2)) \rightsquigarrow$$
$$\quad \mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1)$$
$$\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{sym}\ A\ a\ b\ \pi)) \rightsquigarrow \theta(\pi)$$
$$\theta(\mathrm{trans}\ A\ a\ c\ b\ \pi_1\ \pi_2) \rightsquigarrow$$
$$\quad \mathrm{trans}\ A\ a\ c\ b\ \theta(\pi_1)\ \theta(\pi_2)$$
$$\theta(\mathrm{sym}\ B\ \Delta[a]\ \Delta[b]\ (\mathrm{eq\_f}\ A\ B\ \Delta\ a\ b\ \pi)) \rightsquigarrow$$
$$\quad \mathrm{eq\_f}\ A\ B\ \Delta\ b\ a\ (\mathrm{sym}\ A\ a\ b\ \pi)$$
$$\theta(\pi) \rightsquigarrow \pi$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2)\qquad\qquad)$$

$$\Rightarrow$$

$$\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1)$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2) : a =_A b)$$

$$\Rightarrow$$

$$\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1)$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2) : a =_A b)$$

$$\Rightarrow$$

$$\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1) : a =_A b$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\frac{\pi_1 : b =_A c \qquad \pi_2 : c =_A a}{\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2) : a =_A b)}$$

$$\Rightarrow$$

$$\frac{}{\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1) : a =_A b}$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\frac{\pi_1 : b =_A c \qquad \pi_2 : c =_A a}{\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2) : a =_A b)}$$

$$\Rightarrow$$

$$\frac{\mathrm{sym}\ A\ c\ a\ \pi_2\ \ : a =_A c}{\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1) : a =_A b}$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\frac{\pi_1 : b =_A c \qquad \pi_2 : c =_A a}{\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2) : a =_A b)}$$

$$\Rightarrow$$

$$\frac{\mathrm{sym}\ A\ c\ a\ \pi_2\ : a =_A c \qquad \mathrm{sym}\ A\ b\ c\ \pi_1\ : c =_A b}{\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1) : a =_A b}$$

Theorem 2: θ is type preserving

By induction on the size of the proof.

$$\frac{\pi_1 : b =_A c \qquad \pi_2 : c =_A a}{\theta(\mathrm{sym}\ A\ b\ a\ (\mathrm{trans}\ A\ b\ c\ a\ \pi_1\ \pi_2) : a =_A b)}$$

$$\Rightarrow$$

$$\frac{\theta(\mathrm{sym}\ A\ c\ a\ \pi_2) : a =_A c \qquad \theta(\mathrm{sym}\ A\ b\ c\ \pi_1) : c =_A b}{\mathrm{trans}\ A\ a\ c\ b\ \theta(\mathrm{sym}\ A\ c\ a\ \pi_2)\ \theta(\mathrm{sym}\ A\ b\ c\ \pi_1) : a =_A b}$$

Examples

- Demo!

Future work

- We developed a prolog-style proof search procedure, we want nice proof objects also in this case.
- Declarative and procedural language rendering of such proof objects (work in progress).
- Make tactics more “proof reconstruction” friendly.