# Equivalence of System F and $\lambda 2$ in Abella

Jonas Kaiser    Gert Smolka

TTT 2017, Paris

January 15, 2017

SAARLAND UNIVERSITY
COMPUTER SCIENCE

# System F [Girard '72] / PTLC [Reynolds '74]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

## The Problem

- Everybody defines their own version.
- *Equivalence* is tacitly assumed when external results are applied.

## We consider two particular versions

- F: two-sorted, explicit, separate type variable context, e.g. [Harper '13]
- $\lambda 2$: single-sorted pure type system (PTS) [Barendregt '91]

## Are the two presentations equivalent? In what sense?

$$F \quad \overset{?}{\approx} \quad \lambda 2$$

# Overview

SAARLAND UNIVERSITY
COMPUTER SCIENCE

# Historical Context

*"To show that the two representations of these systems are in fact the same requires some technical but not difficult work."*

**Herman Geuvers**, in *Logics and Type Systems*, '93

- Mostly discusses PTSs.
- "Traditional" systems not defined precisely.
- Desired correspondence only stated, not proven.
- We want a formal/mechanised proof of:

$$\vdash_{\mathsf{F}} s : A \iff \vdash_{2} s' : A'$$

$$\vdash_{2} a : b \iff \vdash_{\mathsf{F}} a' : b'$$

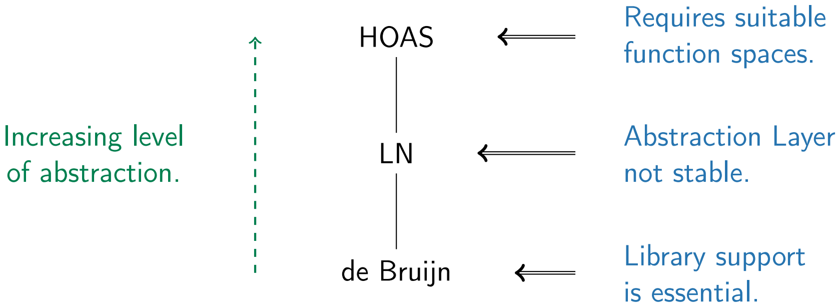
The *technical* part involves dealing with *syntax* and *variable binding*.

# Dealing with Syntax



- Binding of variables (of different kinds).
- Free variables vs. bound identifiers; freshness.
- Capture-avoiding substitution(s).

Various approaches:



# Reduction of the Typing Problem – 2 Proofs

SAARLAND UNIVERSITY
COMPUTER SCIENCE

HOAS proof using syntax relations in Abella:

$$\{s :_F A\} \iff \exists ab.\ \{s \approx a\} \wedge \{A \sim b\} \wedge \{a :_2 b\}$$

$$\{a :_2 b\} \iff \exists sA.\ \{s \approx a\} \wedge \{A \sim b\} \wedge \{s :_F A\}$$

De Bruijn proof using translation functions in Coq [K/Tebbi/Smolka CPP'17]:

$$\vdash_F s : A \iff \vdash_2 \lfloor s \rfloor : \lfloor A \rfloor$$

$$\vdash_2 a : b \iff \vdash_F \lceil a \rceil : \lceil b \rceil$$

Note: Utilises the Autosubst de Bruijn library [Schäfer/Tebbi/Smolka '15].

# Brief Overview of Abella [Miller, Chaudhuri et al. '14]

- Interactive theorem prover with two layers (*two-level logic approach*).
- *Meta level:* $\mathcal{G}$
  - Intuitionistic, predicative fragment of Church's STT,
  - + (co-)inductive predicates,
  - + built-in natural numbers and natural induction,
  - + nominal quantification ($\nabla x.s$): $x$ in $s$ is guaranteed *fresh*
  - **Note:** no induction on types, no functions.
- *Specification level:* Hereditary Harrop Formulas / $\lambda$Prolog
  - Horn clauses (cf. Prolog): $A$ :- $C, D$
  - + hypothetical reasoning: $A$ :- $C, E \Rightarrow D$
  - + quantification: $A$ :- $C, \Pi x.\ D\,x$
- *Logical Embedding:*
  - HHOP-derivations are inductive
  - $\{J\}$ holds in $\mathcal{G} \iff J$ has a $\lambda$Prolog derivation
  - $\{L \vdash J\}$ holds in $\mathcal{G} \iff J$ has a derivation, given hypotheses $L$

# HOAS Signatures of System F and $\lambda 2$

SAARLAND UNIVERSITY
COMPUTER SCIENCE

| $\mathsf{Ty}_F, \mathsf{Tm}_F$ | **type** | $\mathsf{Tm}_2$ | **type** |
|---|---|---|---|
| | | $*, \Box$ | $\mathsf{Tm}_2$ |
| $\_ \to \_$ | $\mathsf{Ty}_F \to \mathsf{Ty}_F \to \mathsf{Ty}_F$ | $\Pi \_ . \_$ | $\mathsf{Tm}_2 \to (\mathsf{Tm}_2 \to \mathsf{Tm}_2) \to \mathsf{Tm}_2$ |
| $\forall . \_$ | $(\mathsf{Ty}_F \to \mathsf{Ty}_F) \to \mathsf{Ty}_F$ | | |
| $\_ \mathbin{@} \_$ | $\mathsf{Tm}_F \to \mathsf{Tm}_F \to \mathsf{Tm}_F$ | $\_ \mathbin{@} \_$ | $\mathsf{Tm}_2 \to \mathsf{Tm}_2 \to \mathsf{Tm}_2$ |
| $\_ \mathbin{\overline{@}} \_$ | $\mathsf{Tm}_F \to \mathsf{Ty}_F \to \mathsf{Tm}_F$ | | |
| $\lambda \_ . \_$ | $\mathsf{Ty}_F \to (\mathsf{Tm}_F \to \mathsf{Tm}_F) \to \mathsf{Tm}_F$ | $\lambda \_ . \_$ | $\mathsf{Tm}_2 \to (\mathsf{Tm}_2 \to \mathsf{Tm}_2) \to \mathsf{Tm}_2$ |
| $\Lambda . \_$ | $(\mathsf{Ty}_F \to \mathsf{Tm}_F) \to \mathsf{Tm}_F$ | | |
| | | $\mathcal{U}\ \_$ | $\mathsf{Tm}_2 \to o$ |
| $\_\ \mathbf{ty}$ | $\mathsf{Ty}_F \to o$ | $\_ :_2 \_$ | $\mathsf{Tm}_2 \to \mathsf{Tm}_2 \to o$ |
| $\_ :_F \_$ | $\mathsf{Tm}_F \to \mathsf{Ty}_F \to o$ | | |

# Proof in Abella – Syntax Relations



$$\_ \sim \_ \quad \mathsf{Ty}_F \to \mathsf{Tm}_2 \to o$$

$$\_ \approx \_ \quad \mathsf{Tm}_F \to \mathsf{Tm}_2 \to o$$

$$\frac{A \sim a \qquad \Pi x.\ B \sim b\,x}{A \to B \sim \Pi a.b} \qquad \frac{\Pi x\,y.\ x \sim y \ \Rightarrow\ A\,x \sim a\,y}{\forall.A \sim \Pi *.a}$$

$$\frac{s \approx a \qquad t \approx b}{s \mathbin{@} t \approx a \mathbin{@} b} \qquad \frac{s \approx a \qquad B \sim b}{s \mathbin{\overline{@}} B \approx a \mathbin{@} b}$$

$$\frac{A \sim a \qquad \Pi x\,y.\ x \approx y \ \Rightarrow\ s\,x \approx b\,y}{\lambda A.s \approx \lambda a.b} \qquad \frac{\Pi x\,y.\ x \sim y \ \Rightarrow\ s\,x \approx b\,y}{\Lambda.s \approx \lambda *.b}$$

# Proof in Abella, ctnd.

- Show that $\sim, \approx$ are both *injective* and *functional*.
- Show that $\sim, \approx$ are conditionally *left- & right-total* and *preserve judgements*, e.g. for $\sim$:

$$\{A\ \mathbf{ty}\} \Rightarrow \exists a.\ \ \{A \sim a\} \wedge \{a :_2 *\}$$
$$\{a :_2 *\} \Rightarrow \exists A.\ \ \{A \sim a\} \wedge \{A\ \mathbf{ty}\}$$

- Due to injectivity and functionality, witnesses are *unique*.
- Thus both inverse implications ($\Leftarrow$) also hold.
- Similar for $\approx$ but more verbose.

# Contexts



- We have to generalise to open terms and hypothetical contexts.
- Consider functionality for $\sim$:

$$C_R(L) \Rightarrow \{L \vdash A \sim a\} \Rightarrow \{L \vdash A \sim b\} \Rightarrow a = b \qquad L : o\ \text{list}$$

- Inductive definition of $C_R(L)$ using nominals:

$$\frac{}{C_R(\bullet)} \qquad \frac{C_R(L) \qquad x, y \text{ fresh for } L}{C_R(L, x \sim y)} \qquad \frac{C_R(L) \qquad x, y \text{ fresh for } L}{C_R(L, x \approx y)}$$

Define $C_R : o\ \text{list} \to \text{prop}$ by

$$\begin{aligned} &C_R(\bullet); \\ &\nabla x\, y,\ \ C_R(L, x \sim y) \ := \ C_R(L); \\ &\nabla x\, y,\ \ C_R(L, x \approx y) \ := \ C_R(L). \end{aligned}$$

# Contexts, ctnd.



- For *totality/preservation* the situation is similar, e.g.

$$\{L_F \vdash A\ \mathbf{ty}\} \Rightarrow \forall L_R\, L_2.\quad C(L_F \mid L_R \mid L_2) \Rightarrow \exists a.\quad \{L_R \vdash A \sim a\} \wedge \{L_2 \vdash a :_2 *\}$$

- $C(L_F \mid L_R \mid L_2)$ is interesting:
  - Entails $C_F(L_F)$, $C_R(L_R)$ and $C_2(L_2)$ by construction.
  - Recall that $L_R$ is effectively a *relation on type- and term-variables*.
  - Ensures that $L_R$ precisely relates the typing contexts $L_F$ and $L_2$.
- Inductive definition:

$$\frac{}{C(\bullet \mid \bullet \mid \bullet)} \qquad \frac{C(L_F \mid L_R \mid L_2) \qquad x, y \text{ fresh for } L_F, L_R, L_2}{C(L_F, x\ \mathbf{ty} \mid L_R, x \sim y \mid L_2, y :_2 *)}$$

$$\frac{\{L_F \vdash A\ \mathbf{ty}\} \qquad \{L_R \vdash A \sim a\} \qquad \{L_2 \vdash a :_2 *\} \qquad C(L_F \mid L_R \mid L_2) \qquad x, y \text{ fresh for } L_F, L_R, L_2, A, a}{C(L_F, x :_F A \mid L_R, x \approx y \mid L_2, y :_2 a)}$$

# Proof – Technical Remarks



- Proof relies heavily on useful *inversion lemmas*. Reason:
  - Our contexts only contain variable information.
  - But every simple case analysis on $\{L \vdash J\}$ considers the case $J \in L$, even if $J$ is a non-variable judgement.
- Applications of $\lambda 2$ are particularly involved:

$$\{L_R \vdash s \approx a \mathbin{@} b\} \quad \xrightarrow{??} \quad \{L_R \vdash s' \mathbin{\overline{@}} B \approx a \mathbin{@} b\}$$

$$\{L_R \vdash s \approx a \mathbin{@} b\} \quad \xrightarrow{??} \quad \{L_R \vdash s' \mathbin{@} t \approx a \mathbin{@} b\}$$

Solving this solely from typing information for $b$ under some $L_2$ appears to rely on the predicate $C$ to connect $L_R$ and $L_2$. Only having $C_R(L_R)$ and $C_2(L_2)$ is not enough.

# Comparison with Coq Proof – de Bruijn



## Similarities of both proofs

- Overall proof structure.
- *Propagation/Type Correctness* plays a major role.
- The ($\Leftarrow$)-directions are obtained from the respective other ($\Rightarrow$)-result.
- Hardest case: disambiguation of PTS applications.

## Differences

- Relations avoid cancellation laws (about a third of the Coq proof).
- The de Bruijn proof clearly separates type formation from typing, in the HOAS proof they are connected much closer.

## Main Observation

- The predicate $C(L_F \mid L_R \mid L_2)$ appears to be the *relational combination* of all four de Bruijn morphism conditions. The latter express that certain renaming functions map variable typings from one context to another.

# Remarks on Abella Usability



- Overall experience of working in Abella was quite pleasant.
- The combination of the two-level approach and nominals was particularly useful.
- Proof scripts are extremely fragile when it comes to refactoring.
  - Automatically named, but explicitly referenced hypotheses.
  - No means to enforce separation of proof tree branches (cf. Coq bullets).
  - So `case H3.` might still work, while `H3` now denotes sth. different.
  - Thus hard to track down where changes are required.
- Currently only a single specification may be imported into $\mathcal{G}$.
- Extending $\mathcal{G}$ with actual functions would also be nice.
- Why does Abella admit (with a warning) potentially consistency breaking inductive predicates with negative occurrences?
- Merging the Abella Proof General fork back into trunk would be desirably to avoid duplicate environments.

# Summary

- Contributions:
  - Reduction of type formation and typing problems, formalised in Abella.
  - Comparison of de Bruijn and HOAS techniques for this proof.
  - Comparison of syntax translation via functions vs. relations.
  - Small usability study of the Abella theorem prover.
- Current & Future Work:
  - Rework Coq proof using relations instead of functions.
  - Improve HOAS support in Coq, see [Capretta & Felty '06].

*Thank you for your attention.*

http://www.ps.uni-saarland.de/extras/ttt17-sysf/

Note: Presentation of the de Bruijn proof @ CPP:
Tuesday, January 17, 2017 – 17:00