The Constructive Engine

Randy Pollack

LFCS, University of Edinburgh

Version of December 30, 2006

Outline

Outline

Checking Type Judgements

- Type Judgements have shape $\Gamma \vdash M : A$
  - M is a term claimed to have type A in context of assumptions, Γ.
- Correct judgements are specified inductively by a set of rules.
- Like other relations defined by inductive rule sets, a type judgement can be verified by checking a derivation tree.
  - Side conditions, decidability, ...
  - This is how NuPrl is checked.
- In some type theories, the information in the judgement is enough to build a derivation.
  - Intensional theories with Church-style terms ($\lambda x{:}A.M$):
  - Calculus of Constructions, ECC, Coq, λP (LF), ...

Type Checking and Type Synthesis

- The Type Checking (TC) problem, given Γ, M and A, is to decide $\Gamma \vdash M : A$.
- The Type Synthesis (TS) problem, given Γ and M is to find A s.t. $\Gamma \vdash M : A$.
  - ...Or to find a principle description of all such A.

We will use TS to solve TC for a class of PTS.

History of Checking PTS

- [Mar71] Martin-Löf's theory with Type:Type (λ^*); feasible abstract algorithm for checking.
- [Hue89] Coined name *constructive engine*, gave concrete ML program for checking CC.
- [HP91] Proved correctness of engine, extended to universes.
- [Pol92] Extended constructive engine to λP (LF).
- [Pol] Abstract algorithm for whole λ-cube.
- [vBJMP94] Deeper study of checking for all PTS; machine checked.
- [Pol94] Machine checked proof of correctness.
- [Pol95] Machine checked proof of correctness.
- [Bar99] Machine checked for CC with inductive types (including normalization); working typechecker extracted from proof!
- [Sev98] Simpler proof for functional PTS.

Outline

The Language

PTS is a language and a typing relation parameterised by $(\mathcal{V}, \mathcal{S}, \mathrm{ax}, \mathrm{rl})$ where:

- $\mathcal{V}$ is an infinite set of *variables*, ranged over by x, y;
- $\mathcal{S}$ is a set of *sorts*, ranged over by s, t;
- ax $\subseteq \mathcal{S} \times \mathcal{S}$ called *axioms*;
- rl $\subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ called *rules*.

The Language

Terms (M, N, A, B) are given by the grammar:

$$\begin{array}{llll} \text{atoms} & \alpha & ::= & x \mid s \\ \text{terms} & M & ::= & \alpha \mid \lambda x{:}M.M \mid \Pi x{:}M.M \mid M\,M \end{array}$$

- Write $A \overset{wh}{\twoheadrightarrow} B$ for β-weak-head-reduction to whnf.
- Write $A \simeq B$ for β-conversion.

Contexts (Γ) are lists of variable-term pairs, written as

$$\begin{array}{llllr} \text{context} & \Gamma & ::= & \bullet \mid \Gamma, x{:}A & \textit{empty, non-empty} \end{array}$$

Typing judgement of PTS has shape $\Gamma \vdash M : A$.

- (Γ, M) is the **subject** of judgement $\Gamma \vdash M : A$.
- A is the **predicate** of the judgement.

Typing Rules

$$\textsc{Ax}\quad \frac{ax(s_1 : s_2)}{\bullet \vdash s_1 : s_2} \qquad \textsc{Start}\quad \frac{\Gamma \vdash A : s \quad x \notin \Gamma}{\Gamma, x{:}A \vdash x : A}$$

$$\textsc{Weak}\quad \frac{\Gamma \vdash \alpha : B \quad \Gamma \vdash A : s \quad x \notin \Gamma}{\Gamma, x{:}A \vdash \alpha : B}$$

$$\textsc{Pi}\quad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2 \quad rl(s_1, s_2, s_3) \quad x \notin \Gamma}{\Gamma \vdash \Pi x{:}A.B : s_3}$$

$$\textsc{Lda}\quad \frac{\Gamma, x{:}A \vdash M : B \quad \Gamma \vdash \Pi x{:}A.B : s \quad x \notin \Gamma}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}$$

$$\textsc{App}\quad \frac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash M\,N : [N/x]B}$$

$$\textsc{Conv}\quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s \quad A \simeq B}{\Gamma \vdash M : B}$$

Outline

Strategy for Programming

- To solve TS we want a set of rules that is syntax directed on the subject of the judgement, Γ, M.
- Rules are read as recursion equations. For example

$$\text{PI} \quad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad x \notin \Gamma}{\Gamma \vdash \Pi x{:}A.B : s_3}$$

is read as

$$\begin{aligned} \mathrm{TS}(\Gamma, \Pi x{:}A.B) = \;& \text{let } s_1 = \mathrm{TS}(\Gamma, A) \text{ in} \\ & \text{let } s_2 = \mathrm{TS}((\Gamma, x{:}A), B) \text{ in} \\ & \text{if } \mathrm{rl}(s_1, s_2, s_3) \text{ then } s_3 \text{ else fail} \end{aligned}$$

- Reason about the relation rather than the program.
- Termination must still be proved (because of side conditions).

Problems with the Strategy

- **Relation may be too inefficient in practice:**
  - Some rules have two premises . . .
  - . . . the PTS context is constructed by weakening on every branch.

  Solution: Incrementally maintain context correctness instead of repeatedly checking it.
- **Rule CONV is not syntax directed:**
  - Structure doesn't tell when to use the rule.

$$\textsc{Conv}\ \frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s \qquad A \simeq B}{\Gamma \vdash M : B}$$

  Solution: permute the rule to end of derivations

These are the two main ideas in the Constructive Engine.

Basic Properties for Type Synthesis

- The (raw) language of PTS is Church–Rosser.
  - Conversion testing by reduction is complete, but possibly non-terminating.
  - For normalizing PTS, conversion testing is decidable on well-typed terms.
  - A TS program should only try reduction on terms already known to be well-typed.
- **Subject Reduction:** If $\Gamma \vdash M : A$ and $M \to M'$ then $\Gamma \vdash M' : A$.
- **Type Correctness:** If $\Gamma \vdash M : A$ then $\exists s \,.\, \Gamma \vdash A : s \vee A = s$.

Functional PTS

- **Definition:** A PTS is functional iff

$$\begin{array}{rcl} \mathrm{ax}(s_1 : s_2) \wedge \mathrm{ax}(s_1 : s_2') & \Longrightarrow & s_2 = s_2' \\ \mathrm{rl}(s_1, s_2, s_3) \wedge \mathrm{rl}(s_1, s_2, s_3') & \Longrightarrow & s_3 = s_3'. \end{array}$$

  - In non-functional PTS, different instances of a rule may be used:

$$\textsc{Ax}\ \frac{\mathrm{ax}(s_1 : s_2)}{\bullet \vdash s_1 : s_2}$$

  - Non-functional PTS may sometimes be handled using schematic sort variables and constraints.
    - E.g. universes in Coq and Lego.
  - But non-functional PTS lack other good properties too, so we ignore them in the rest of this talk.
  - In functional PTS, rules Ax and Pi are deterministic for TS.

Decidability Issues

- **Example:** Consider the PTS:

$$\begin{aligned} \mathcal{S} &= \text{natural numbers} \\ \mathsf{ax} &= \{(i:n) \mid \text{Turing machine } i\text{, with } i \text{ on its tape,} \\ &\qquad\qquad \text{halts in exactly } n \text{ steps}\} \\ \mathsf{rl} &= \emptyset \end{aligned}$$

- ax and rl are decidable.
- The PTS is functional and strongly normalizing (there are no well-typed redices).
- TC and TS are undecidable:

$$x{:}i \vdash x : i \iff \exists n \,.\, \mathsf{ax}(i:n) \iff \text{Turing machine } i \text{ halts on input } i.$$

- **Definition:** Sort s is a topsort iff $\neg\exists s' \,.\, \mathsf{ax}(s:s')$.

Outline

Step 1: System with Valid Contexts

Typing

$$\textsc{Ax}\ \frac{\vdash_{vc} \Gamma \quad \mathrm{ax}(s_1 : s_2)}{\Gamma \vdash_{vt} s_1 : s_2} \qquad \textsc{Var}\ \frac{\vdash_{vc} \Gamma \quad x{:}A \in \Gamma}{\Gamma \vdash_{vt} x : A}$$

$$\textsc{Pi}\ \frac{\Gamma \vdash_{vt} A : s_1 \quad \Gamma, x{:}A \vdash_{vt} B : s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad x \notin \Gamma}{\Gamma \vdash_{vt} \Pi x{:}A.B : s_3}$$

$$\textsc{Lda}\ \frac{\Gamma, x{:}A \vdash_{vt} M : B \quad \Gamma \vdash_{vt} \Pi x{:}A.B : s \quad x \notin \Gamma}{\Gamma \vdash_{vt} \lambda x{:}A.M : \Pi x{:}A.B}$$

$$\textsc{App}\ \frac{\Gamma \vdash_{vt} M : \Pi x{:}A.B \quad \Gamma \vdash_{vt} N : A}{\Gamma \vdash_{vt} M\,N : [N/x]B}$$

$$\textsc{Conv}\ \frac{\Gamma \vdash_{vt} M : A \quad \Gamma \vdash_{vt} B : s \quad A \simeq B}{\Gamma \vdash_{vt} M : B}$$

Contexts

$$\textsc{valNil}\ \frac{}{\vdash_{vc} \bullet} \qquad \textsc{valCons}\ \frac{\Gamma \vdash_{vt} A : s \quad x \notin \Gamma}{\vdash_{vc} \Gamma, x{:}A}$$

System with Valid Contexts

- $\vdash_{vc}$ and $\vdash_{vt}$ are mutually inductive.
- Correctness (for all PTS)
  - $\Gamma \vdash_{vt} M : A \Longrightarrow \Gamma \vdash M : A$
    Proof by direct induction using simple properties of $\vdash$.
  - $\Gamma \vdash M : A \Longrightarrow \Gamma \vdash_{vt} M : A$
    Proof by induction, requiring weakening for $\vdash_{vt}$ to treat the WEAK rule of $\vdash$.
  - Derivations essentially isomorphic with $\vdash$.
- Efficiency
  - Infeasible: checks $\vdash_{vc} \Gamma$ on every branch.
- Still not syntax directed: CONV rule.

Step 2: System with Locally Valid Contexts

Typing

$$\textsc{Ax}\ \frac{\mathrm{ax}(s_1 : s_2)}{\Gamma \vdash_{lvt} s_1 : s_2} \qquad \textsc{Var}\ \frac{x{:}A \in \Gamma}{\Gamma \vdash_{lvt} x : A}$$

$$\textsc{Pi}\ \frac{\Gamma \vdash_{lvt} A : s_1 \quad \Gamma, x{:}A \vdash_{lvt} B : s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad x \notin \Gamma}{\Gamma \vdash_{lvt} \Pi x{:}A.B : s_3}$$

$$\textsc{Lda}\ \frac{\Gamma, x{:}A \vdash_{lvt} M : B \quad \Gamma \vdash_{lvt} \Pi x{:}A.B : s \quad x \notin \Gamma}{\Gamma \vdash_{lvt} \lambda x{:}A.M : \Pi x{:}A.B}$$

$$\textsc{App}\ \frac{\Gamma \vdash_{lvt} M : \Pi x{:}A.B \quad \Gamma \vdash_{lvt} N : A}{\Gamma \vdash_{lvt} M\,N : [N/x]B}$$

$$\textsc{Conv}\ \frac{\Gamma \vdash_{lvt} M : A \quad \Gamma \vdash_{lvt} B : s \quad A \simeq B}{\Gamma \vdash_{lvt} M : B}$$

Contexts

$$\textsc{Nil}\ \frac{}{\vdash_{lvc} \bullet} \qquad \textsc{Cons}\ \frac{\Gamma \vdash_{lvt} A : s \quad x \notin \Gamma}{\vdash_{lvc} \Gamma, x{:}A}$$

System with Locally Valid Contexts

- $\vdash_{lvt}$ does not depend on $\vdash_{lvc}$.
- Correctness (for all PTS)
  - $(\Gamma \vdash_{lvt} M : A \wedge \vdash_{lvc} \Gamma) \Longrightarrow \Gamma \vdash_{vt} M : A$
    Proof by tricky induction on the sum of the heights of the derivations of $\Gamma \vdash_{lvt} M : A$ and $\vdash_{lvc} \Gamma$.
    - This proof says $\vdash_{lvt}$ derivations can be expanded to $\vdash_{vt}$ derivations.
  - Thus: $\Gamma \vdash M : A \iff (\vdash_{lvc} \Gamma \wedge \Gamma \vdash_{lvt} M : A)$
- Efficiency
  - Contexts are not checked on every branch ...
  - ... so local extensions (rules PI and LDA) must be checked locally. (Note rule LDA.)
  - “The derivation tree of $\vdash_{vt}$ has become a graph by identifying the duplicate derivations of $\vdash_{vc} \Gamma$.”
  - Derivations much smaller than $\vdash$.

Outline

Permuting Conv Out of Derivations

- $\vdash_{lvt}$ is not syntax directed: the shape of the subject doesn't tell where to use rule Conv.
- The idea is to permute uses of Conv to the root of derivations.
- When Conv passes through a premise of another rule, we may need to do some computation on that premise.
- E.g. rule App becomes

$$\textsc{App}\ \frac{\Gamma \vdash M \Rightarrow X \quad X \xrightarrow{wh} \Pi x{:}A.B \quad \Gamma \vdash N \Rightarrow A' \quad A \simeq A'}{\Gamma \vdash M\,N \Rightarrow [N/x]B}$$

  - Note, $\Pi x{:}A.B$ is a whnf.
- Soundness depends on subject reduction and type correctness.

Syntax Directed Relation

Notation: Write $\Gamma \vdash M \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A$ for $(\Gamma \vdash M \Rightarrow X \wedge X \xrightarrow{wh}\!\!\!\!\rightarrow A)$.

Typing

$$\textsc{Ax}\ \frac{\mathrm{ax}(s_1 : s_2)}{\Gamma \vdash s_1 \Rightarrow s_2} \qquad \textsc{Var}\ \frac{x{:}A \in \Gamma}{\Gamma \vdash x \Rightarrow A}$$

$$\textsc{Pi}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow s_1 \quad \Gamma, x{:}A \vdash B \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad x \notin \Gamma}{\Gamma \vdash \Pi x{:}A.B \Rightarrow s_3}$$

$$\textsc{Lda}\ \frac{\Gamma, x{:}A \vdash M \Rightarrow B \quad \Gamma \vdash \Pi x{:}A.B \Rightarrow s \quad x \notin \Gamma}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

$$\textsc{App}\ \frac{\Gamma \vdash M \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow \Pi x{:}A.B \quad \Gamma \vdash N \Rightarrow A' \quad A \simeq A'}{\Gamma \vdash M\, N \Rightarrow [N/x]B}$$

Contexts

$$\textsc{Nil}\ \frac{}{\vdash \bullet} \qquad \textsc{Cons}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow s \quad x \notin \Gamma}{\vdash \Gamma, x{:}A}$$

Correctness of Syntax Directed Relation

- **Soundness:** For any PTS,

$$(\vdash \Gamma \wedge \Gamma \vdash M \Rightarrow A) \Longrightarrow \Gamma \vdash M : A.$$

- **Completeness?** $\Gamma \vdash M : A \Longrightarrow (\Gamma \vdash M \Rightarrow A' \wedge A \simeq A')$
  - Counterexample for non-functional PTS [Pol92].
  - Source of incompleteness is rule LDA.
  - Open problem for arbitrary functional PTS.
- This system is syntax directed, but is not a satisfactory TS program!
  - It may not terminate when it should.

Termination

Assume a functional, normalizing PTS.

- The rules are syntax directed, so building a putative derivation tree does terminate.
- reduction terminates for any well-typed term.
- In rules PI and APP, we only apply reduction (conversion) to well-typed terms.
- There is a problem with rule LDA:

$$\textsc{Lda}\quad \frac{\Gamma, x{:}A \vdash M \Rightarrow B \qquad \Gamma \vdash \Pi x{:}A.B \Rightarrow s}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

  - The left premise must be synthesised first, to get B
  - A in the extended context is not yet checked, so some reduction may diverge.

Improve Rule LDA

- Expand the right premise:

$$\frac{\Gamma, x{:}A \vdash M \Rightarrow B \qquad \dfrac{\Gamma \vdash A \Rightarrow \overset{wh}{\twoheadrightarrow} s_1 \quad \Gamma, x{:}A \vdash B \Rightarrow \overset{wh}{\twoheadrightarrow} s_2 \quad \mathrm{rl}(s_1, s_2, s_3)}{\Gamma \vdash \Pi x{:}A.B \Rightarrow s}}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

- Move the premises around:

$$\textsc{Lda}\ \frac{\Gamma \vdash A \Rightarrow \overset{wh}{\twoheadrightarrow} s_1 \quad \Gamma, x{:}A \vdash M \Rightarrow B \quad \Gamma, x{:}A \vdash B \Rightarrow \overset{wh}{\twoheadrightarrow} s_2 \quad \mathrm{rl}(s_1, s_2, s_3)}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

Regain Completeness for Full PTS

$$\text{LDA}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\!\!\rightarrow s_1 \quad \Gamma, x{:}A \vdash M \Rightarrow B \quad \Gamma, x{:}A \vdash B \Rightarrow \xrightarrow{wh}\!\!\!\!\!\!\rightarrow s_2 \quad \mathrm{rl}(s_1, s_2, s_3)}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

- **Definition:** A PTS is full iff $\forall s_1, s_2 \,.\, \exists s_3 \,.\, \mathrm{rl}(s_1, s_2, s_3)$.
  - λ^*, CC, ECC and CIC are full.
- For full PTS, we can omit the side condition $\mathrm{rl}(s_1, s_2, s_3)$.
- Thus there is no need to actually know s_2; any sort will do.
- If B is not a topsort, the third premise is derivable from type correctness on the second premise:

$$\exists s \,.\, (\Gamma, x{:}A \vdash B \Rightarrow \xrightarrow{wh}\!\!\!\!\!\!\rightarrow s_B) \vee B = s_B.$$

  - In the second case, if $\mathrm{ax}(s_B : s)$, then $\Gamma, x{:}A \vdash B \Rightarrow s$.

$$\text{LDA}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\!\!\rightarrow s_1 \quad \Gamma, x{:}A \vdash M \Rightarrow B \quad B \text{ not a topsort}}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

Abstract Constructive Engine

Sound and complete for full PTS.

$$\text{AX}\ \frac{\mathrm{ax}(s_1 : s_2)}{\Gamma \vdash s_1 \Rightarrow s_2} \qquad \text{VAR}\ \frac{x{:}A \in \Gamma}{\Gamma \vdash x \Rightarrow A}$$

$$\text{PI}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\rightarrow s_1 \quad \Gamma, x{:}A \vdash B \Rightarrow \xrightarrow{wh}\!\!\!\!\rightarrow s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad x \notin \Gamma}{\Gamma \vdash \Pi x{:}A.B \Rightarrow s_3}$$

$$\text{LDA}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\rightarrow s_1 \quad \Gamma, x{:}A \vdash M \Rightarrow B \quad B \text{ not a topsort} \quad x \notin \Gamma}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \Pi x{:}A.B}$$

$$\text{APP}\ \frac{\Gamma \vdash M \Rightarrow \xrightarrow{wh}\!\!\!\!\rightarrow \Pi x{:}A.B \quad \Gamma \vdash N \Rightarrow A' \quad A \simeq A'}{\Gamma \vdash M\,N \Rightarrow [N/x]B}$$

$$\text{NIL}\ \frac{}{\vdash \bullet} \qquad \text{CONS}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\rightarrow s \quad x \notin \Gamma}{\vdash \Gamma, x{:}A}$$

Outline

Binding

- User input is ascii, so after parsing, raw terms have strings for variable names, x.
- Introduce a new species of checked terms, (M, A), with equality up-to α-equivalence.
  - Using de Bruijn representation, FreshOcaml, Cαml, FreshLib, ...
  - ...with a new species of bound variable, v, w.
  - Continue to use strings, x, for global variables.

$$\begin{array}{llll} \text{terms} & M & ::= & x \mid s \mid v \mid \lambda v{:}M.M \mid \Pi v{:}M.M \mid M\ M \\ \text{context} & \Gamma & ::= & \bullet \mid \Gamma, x{:}A \end{array}$$

- A judgement form (the kernel) that does type synthesis and translates to checked terms at the same time:

$$\Gamma \vdash M \Rightarrow M : A$$

  - M and Γ are abstract datatypes, only constructed by the kernel.
  - Only operation needed on black terms is structural decomposition.

Huet's Concrete Constructive Engine: Kernel

$$\textsc{Ax}\ \frac{\mathrm{ax}(s_1 : s_2)}{\Gamma \vdash s_1 \Rightarrow s_1 : s_2} \qquad \textsc{Var}\ \frac{x{:}A \in \Gamma}{\Gamma \vdash x \Rightarrow x : A}$$

$$\textsc{Pi}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\to A : s_1 \quad \Gamma, x{:}A \vdash B \Rightarrow \xrightarrow{wh}\!\!\!\!\to B : s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad \begin{matrix} x \notin \Gamma \\ v \text{ fresh} \end{matrix}}{\Gamma \vdash \Pi x{:}A.B \Rightarrow \Pi v{:}A.[v/x]B : s_3}$$

$$\textsc{Lda}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\to A : s_1 \quad \Gamma, x{:}A \vdash M \Rightarrow M : B \quad B \text{ not a topsort} \quad \begin{matrix} x \notin \Gamma \\ v \text{ fresh} \end{matrix}}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \lambda v{:}A.[v/x]M : \Pi v{:}A.[v/x]B}$$

$$\textsc{App}\ \frac{\Gamma \vdash M \Rightarrow \xrightarrow{wh}\!\!\!\!\to M : \Pi v{:}A.B \quad \Gamma \vdash N \Rightarrow N : A' \quad A \simeq A'}{\Gamma \vdash M\, N \Rightarrow M\, N : [N/v]B}$$

$$\textsc{Nil}\ \frac{}{\vdash \bullet} \qquad \textsc{Cons}\ \frac{\Gamma \vdash A \Rightarrow \xrightarrow{wh}\!\!\!\!\to A : s \quad x \notin \Gamma}{\vdash \Gamma, x{:}A}$$

Outline

A Problem: Repeated Concrete Binding Names

- Some correct judgements are not accepted because of side condition $x \not\in \Gamma$ on rules Pi and Lda:

$$\bullet \vdash \lambda x{:}\star.\lambda x{:}x.x : \Pi x{:}\star.\Pi y{:}x.x. \tag{1}$$

- One might think these side conditions can be omitted, making lookup in Γ block structured.
- For dependent types this is unsound:
  - Running (1) in the concrete engine without these side conditions yields the unsound judgement:

$$\bullet \vdash \lambda x{:}\star.\lambda x{:}x.x \Rightarrow \lambda w{:}\star.\lambda v{:}w.v : \Pi w{:}\star.\Pi v{:}w.v.$$

- Context lookup can safely be block structured ...
  - because type synthesis is directed by the syntax of the black term;
- but resolving bound variable names cannot be ...
  - because the synthesised type may have different binding dependency than the term: e.g. equation (1).

A Pretty Fix for the Problem

- Ugly fix: α-convert black terms, ...
  - we don't want to define binding opertions (e.g. α-conversion) on black terms.
- Pretty fix: to safely allow black names (x) to be duplicated in Γ, use the fresh variable names (v) to disambiguate:

$$
\begin{array}{llll}
\text{terms} & M & ::= & s \mid v \mid \lambda v{:}M.M \mid \Pi v{:}M.M \mid M\ M \\
\text{context} & \Gamma & ::= & \bullet \mid \Gamma,(x,v){:}A
\end{array}
$$

  - This only works if red terms use names for binding (e.g. FreshOcaml, Cαml or FreshLib).
    - See below a fix for de Bruijn representation.
  - For good taste, we keep black names unique in the global context.
- Concrete names (x) don't appear in red terms, and no dummy substitution $[v/x]M$ is required to fix up bound names.
- In the rules (next slide), rule VAR is split for sequential lookup.

A Pretty Fix Using Named Binding

$$\textsc{Ax}\ \frac{\mathrm{ax}(s_1 : s_2)}{\Gamma \vdash s_1 \Rightarrow s_1 : s_2}$$

$$\textsc{VarHd}\ \frac{}{\Gamma,(x,v){:}A \vdash x \Rightarrow v : A} \qquad \textsc{VarTl}\ \frac{\Gamma \vdash x \Rightarrow v : A \quad x \neq y}{\Gamma,(y,w){:}B \vdash x \Rightarrow v : A}$$

$$\textsc{Pi}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A : s_1 \quad \Gamma,(x,v){:}A \vdash B \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow B : s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad v \text{ fresh}}{\Gamma \vdash \Pi x{:}A.B \Rightarrow \Pi v{:}A.B : s_3}$$

$$\textsc{Lda}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A : s_1 \quad \Gamma,(x,v){:}A \vdash M \Rightarrow M : B \quad B \text{ not a topsort} \quad v \text{ fresh}}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \lambda v{:}A.M : \Pi v{:}A.B}$$

$$\textsc{App}\ \frac{\Gamma \vdash M \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow M : \Pi v{:}A.B \quad \Gamma \vdash N \Rightarrow N : A' \quad A \simeq A'}{\Gamma \vdash M\,N \Rightarrow M\,N : [N/v]B}$$

$$\textsc{Nil}\ \frac{}{\vdash \bullet} \qquad \textsc{Cons}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A : s \quad x \notin \Gamma \quad v \text{ fresh}}{\vdash \Gamma,(x,v){:}A}$$

A Fix With de Bruijn Representation

- When the red terms use de Bruijn representation, we can't use the "fresh" names (v) to disambiguate Γ.
- Introduce a new concept, timestamps (notation: i, j), to annotate freshness:

$$
\begin{array}{llll}
\text{terms} & M & ::= & i \mid s \mid v \mid \lambda v{:}M.M \mid \Pi v{:}M.M \mid M\,M \\
\text{context} & \Gamma & ::= & \bullet \mid \Gamma,(x,i){:}A
\end{array}
$$

- In the rules (next slide) side condition "i fresh" can be satisfied by taking $i =$ length Γ.
- This is the solution LEGO uses.

A Fix With de Bruijn Representation

$$\textsc{Ax}\ \frac{\mathrm{ax}(s_1 : s_2)}{\Gamma \vdash s_1 \Rightarrow s_1 : s_2}$$

$$\textsc{VarHd}\ \frac{}{\Gamma,(x,i){:}A \vdash x \Rightarrow i : A} \qquad \textsc{VarTl}\ \frac{\Gamma \vdash x \Rightarrow i : A \quad x \neq y}{\Gamma,(y,j){:}B \vdash x \Rightarrow i : A}$$

$$\textsc{Pi}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A : s_1 \quad \Gamma,(x,i){:}A \vdash B \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow B : s_2 \quad \mathrm{rl}(s_1, s_2, s_3) \quad v, i \text{ fresh}}{\Gamma \vdash \Pi x{:}A.B \Rightarrow \Pi v{:}A.[v/i]B : s_3}$$

$$\textsc{Lda}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A : s_1 \quad \Gamma,(x,i){:}A \vdash M \Rightarrow M : B \quad B \text{ not a topsort} \quad v, i \text{ fresh}}{\Gamma \vdash \lambda x{:}A.M \Rightarrow \lambda v{:}A.[v/i]M : \Pi v{:}A.[v/i]B}$$

$$\textsc{App}\ \frac{\Gamma \vdash M \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow M : \Pi v{:}A.B \quad \Gamma \vdash N \Rightarrow N : A' \quad A \simeq A'}{\Gamma \vdash M\,N \Rightarrow M\,N : [N/v]B}$$

$$\textsc{Nil}\ \frac{}{\vdash \bullet} \qquad \textsc{Cons}\ \frac{\Gamma \vdash A \Rightarrow\xrightarrow{wh}\!\!\!\!\rightarrow A : s \quad x \notin \Gamma \quad i \text{ fresh}}{\vdash \Gamma,(x,i){:}A}$$

Outline

References

[Bar99] Bruno Barras. *Auto-validation d'un système de preuves avec familles inductives*.
Thèse de doctorat, Université Paris 7, November 1999.

[HP91] Robert Harper and Robert Pollack. Type checking with universes.
Theoretical Computer Science, 89:107–136, 1991.

[Hue89] Gérard Huet. The constructive engine.
In R. Narasimhan, editor, *A Perspective in Theoretical Computer Science*. World Scientific Publishing, 1989.

[Mar71] Per Martin-Löf. A theory of types.
Technical Report 71-3, Univ. of Stockholm, 1971.

[Pol] Erik Poll. A typechecker for bijective pure type systems.
Computing Science Note (93/22), Eindhoven University of Technology.

[Pol92] R. Pollack. Typechecking in Pure Type Systems.
In *Informal Proceedings of the 1992 Workshop on Types for Proofs and Programs, Båstad, Sweden*.

[Pol94] Robert Pollack. *The Theory of LEGO: A Proof Checker for the Extended Calculus of Constructions*.
PhD thesis, Univ. of Edinburgh, 1994.

[Pol95] Robert Pollack. A verified typechecker.
In M.Dezani-Ciancaglini and G.Plotkin, editors, *Proceedings of the Second International Conference on Typed Lambda Calculi and Applications, TLCA'95, Edinburgh*, volume 902 of *LNCS*. Springer-Verlag, April 1995.

[Sev98] Paula Severi. Type inference for pure type systems.
INFCTRL: Information and Computation (formerly Information and Control), 143, 1998.

[vBJMP94] L.S. van Benthem Jutting, J. McKinna, and R. Pollack. Checking algorithms for Pure Type Systems.
In Barendregt and Nipkow, editors, *TYPES'93: Workshop on Types for Proofs and Programs, Selected Papers*, volume 806 of *LNCS*, pages 19–61. Springer-Verlag, 1994.