Machine Assistance for Programming Language Research

Stephanie Weirich



Changes in PL Research

- Increasing complexity of language features being considered
 - E.g., module systems, dependently typed programming, types ensuring resource bounds or deadlock freedom, ...
- Increasing concern for scale and integration
- Decreasing cycle time for tech transfer
 - Extensions to production languages
 - e.g., generics in Java/C#
 - Adoption of research languages
 - e.g. (OCaml, F#, Haskell, Scheme)

Big changes in the way *proofs* are communicated...



The State of the Art

Chen and Tarditi,

A Simple Typed Intermediate Language for Object-Oriented Languages,

Principles of Programming Languages (POPL), 2005

We have proved the soundness of LIL_C , in the style of [34], and the decidability of type checking. Full proofs are in the technical report.

THEOREM 1 (PRESERVATION). If $\Sigma \vdash P : \tau$ and $P \mapsto P'$, then $\exists \Sigma'$ such that $\Sigma' \vdash P' : \tau$.

THEOREM 2 (PROGRESS). If $\Sigma \vdash P : \tau$, then either the main expression in P is a value, or $\exists P'$ such that $P \mapsto P'$.

Proof sketch: by standard induction over the typing rules.



 If Θ; •; Σ; Γ ⊢ v : ∀tvs(τ₁,, τ_n) → τ (tvs = α₁ ≪ u₁,, α_m ≪ u_m) and Θ ⊢ H : Σ, then v is a label and H(v) = fix g(tvs')(x₁ : τ'₁,, x_n : τ'_n) : τ' = e_m and tvs' = α₁ ≪ u'₁,, α_m ≪ u'_m. 						
 If Θ; •; Σ; Γ ⊢ v : ∃α ≪ τ_u. τ, then v = pack τ₀ as α ≪ τ'_u in (v' : τ'). 		Г				
 If Θ; •; Σ; Γ ⊢ v : Tag(τ), then v = tag(C) for some C. 	a $[\delta] \rightarrow (\Gamma (a))$	$\mathcal{V}[\delta] \text{ and } \Theta \mathcal{N}[\delta] \mathcal{V}[\delta] \mathcal{V}[\delta] \vdash \phi[\delta] \vdash \phi[\delta]$	By induc : τ , $\forall 0 \le i$	tion hypothesis, Θ ; $\leq n - 1$. By array	$\Delta; \Sigma; \Gamma \vdash e_i :$ $\iota : \Theta; \Delta; \Sigma; \Gamma \vdash$	τ_{mi} , $\forall 0 \le i \le n - 1$. By sub and Θ ; $\Delta \vdash \tau_{mi} \le \tau$ $\vdash E : T$.
 If Θ; •; Σ; Γ ⊢ v : Tag(C), then v = tag(C). 	$E[\delta]; \Gamma[\delta] \vdash E[$	$[\delta] : T[\delta].$ $[\delta] : T[\delta].$	Case a_s	ubscript $E = e_1[e$	2], $T = \tau$ wit	h subderivations $\Theta; \Delta; \Sigma; \Gamma \vDash e_1 : array(\tau)$ and Θ ;
7. If Θ : •: Σ : $\Gamma \vdash v$: C , then $v = C(v')$ for some value v' .	[u] with subder $u \ll u_m.(\tau_1,, \tau_n)$	rivation $\Theta; \Delta; \Sigma; \Gamma \vdash e : \tau_f$ and $\Theta; \Delta; \Sigma; \Gamma \vdash \cdots$ $(\tau, \tau_n) \rightarrow \tau, \sigma = t_1, \dots, t_m/\alpha_1, \dots, \alpha_m, \text{ and } \sigma$		and the last of the second		τ) and Θ ; Δ ; Σ ; $\Gamma \vdash e_2$; int. By subscript
8. If Θ : •: Σ : $\Gamma \vdash v$: int. then v is an integer.	$[\delta] : \tau_f[\delta]$ and	Θ ; $\Delta'[\delta]$; $\Sigma[\delta]$; $\Gamma[\delta] \vdash e_i[\delta]$: $\tau_i[\sigma][\delta] \forall 1 \le i \le$	in γ. i the kind envi-	$x g < tvs' > (x_1 : s_1, dues, (3) H' = H, (4)$	V' = V, g = e	
Proof, by inspection on supervision tuning subscheme value value and subtuning investion Lemma A_{i}	$t_n[\delta]) \rightarrow \tau[\delta].$	By permutation of substitutions Lemma 49 Lemma 15, $\Theta: \Delta'[\delta] \models t: [\delta] \ll u: [\sigma][\delta]$, By	Θ ; • $\vdash \tau \ll s$),	mma 52, (1) let $\tau_f =$ l (2) Θ ; tvs' ; Σ ; $g : \tau'_f$,	$\forall \alpha_1 \ll u_1,, c_n$ $x_1 : s_1,, x_n$:	$s_m \ll = \tau'$ with subderivations $\Theta; \Delta; \Sigma; \Gamma \models e$ $s_n \vdash \leq \tau$ and $\Theta; \Delta; \Sigma; \Gamma \models e_4 : \tau'$.
Proof: by inspection on expression typing rules, neap value rules and subtyping inversion Lemma 41.	$[\sigma']$ and call,	$\Theta; \Delta'[\delta]; \Sigma[\delta]; \Gamma[\delta] \vdash E[\delta] : T[\delta].$	- r we have	$\alpha_1 \ll \text{Topc}, \dots, \alpha_m \ll$ $n, \Theta; \bullet \vdash t_i \ll u'_i[\sigma].$	\leq Topc $\vdash \tau_i \leq i$ By type substit	s_i , (2) $ay(\tau)$, $\Theta; \Delta; \Sigma; \Gamma \vdash e_2$: int, $\Theta; \Delta; \Sigma; \Gamma \vdash$ $ay(\tau)$, $\Theta; \Delta; \Sigma; \Gamma \vdash e_2$: $int, \Theta; \Delta; \Sigma; \Gamma \vdash$
sectorem 57 If freetvs(Σ) = \emptyset , and $\Theta \vdash H : \Sigma$, and $\Theta; \Sigma \vdash V : \Gamma$ and $\Theta; \bullet; \Sigma; \Gamma \vdash E : T$, then either E is value, or E can evaluate one step, that is, $\exists H', V'$ and E' such that $(H; V; E) \mapsto (H'; V'; E')$.	$= \exists \alpha \ll \tau_u. \tau$	with subderivation $\Theta; \Delta; \Sigma; \Gamma \vdash e : \tau' [\tau / \alpha]$	- 1, we have	Σ has no free type v that type substitution	ariables, therefo preserves subt	$\tau_{re}, \tau'_{f} = \Delta; \Sigma; 1 \vdash e_{3} : \tau$. By assign $A, \Theta; \Delta; \Sigma; 1 \vdash \tau_{pr}$ upping ibderivations $\Theta; \Delta; \Sigma; \Gamma \vdash e_{1} : \tau_{m1}, \Theta; \Delta \vdash$
Proof: by induction on expression tuning rules	$[\delta] : \tau'[\tau/\alpha][\delta]$ = $\tau'[\delta][\tau[\delta]/\alpha]$	 T[δ] = ∃α ≪ τ_u[δ]. τ'[δ] and by permuta- By Lemma 15, Θ; Δ'[δ] ⊢ τ[δ] ≪ τ_u[δ]. by 	C) contains no	0000000000	on a sithe	wise an luce on each combuste one step. If
Case int, Case label, Case tag: all the expressions are values already.	the subd e ^t with subd e ^t with subd e ^t e ^t e ^t with subd e ^t	erivations $\Theta : \Lambda : \Sigma : \Gamma \vdash e_1 : \exists \beta \ll \tau, \tau \text{ and}$	(C). By c2r_c	expressi inte_one	on e_i eithe step Ot	π is a value of can evaluate one step. If therwise $\exists e_i$ such that e_i can evaluate
Case var: $E = x$ $P_{U} \oplus \Sigma = U$, Γ demain $(\Gamma) = \operatorname{demain}(U)$. From $u \in \operatorname{demain}(\Gamma)$, we know $u \in \operatorname{demain}(U)$. Let $U' = U$.	$(\Delta), \alpha \notin fro$	$e(\tau')).$	$\vdash \alpha \ll C$).	ander one	a rule. $E c$	can evaluate one step
By $\Theta; \Sigma \vdash V : 1$, domain $(1) = \text{domain}(V)$. From $x \in \text{domain}(1)$, we know $x \in \text{domain}(V)$. Let $H' = H$, = V and $E' = V(x)$. By every var $(H; V; E) \mapsto (H'; V'; E')$	$[\delta] : (\exists \beta \ll \tau,$	$_{u}$, τ) $[\delta]$ and Θ ; $\Delta'[\delta]$, $\alpha \ll \tau_{u}[\delta]$; $\Sigma[\delta]$; $\Gamma[\delta]$, $x :=$	= T since the		entions Θ	$\bullet: \Sigma: \Gamma \vdash c_1 : \operatorname{array}(\pi) \text{ and } \Theta: \bullet: \Sigma: \Gamma \vdash$
Case error: $E = \operatorname{error}[\tau]$: by ev_error, E steps to itself.	$[o] = \exists \phi \ll 0$	$\tau_{u}[o]$. $\tau[o]$ and $\tau[\alpha/\beta][o] = \tau[o][\alpha[o]/\beta] =$	δ]; Γ [δ] $\vdash e : s.$ [s/α]. By sub,	$\Theta; \Delta; \Sigma; \Gamma \vDash \ell : \Sigma$	had on co	$(\bullet, \Sigma, 1 \vdash e_1)$ analy (τ) and $(\bullet, \bullet, \Sigma, 1 \vdash e_2)$
Case object: $E = C(e)$ with subderivation $\Theta; \bullet; \Sigma; \Gamma \vdash e : R(C)$.	in e_1 else e_2	, $T = \tau'$ with subderivations $\Theta; \Delta; \Sigma; \Gamma \vdash e$:	subderivations	$\Theta; \Delta; \Sigma; \Gamma \vDash e : C$ $\Delta; \Sigma; \Gamma \vDash e : C$ a_c	fue or e_1	can evaluate one step. So does e_2 .
By induction hypothesis, either e is a value or $\exists H', V'$ and e' such that $(H; V; e) \mapsto (H'; V'; e')$. If e is	$\Theta; \Delta; \Sigma; \Gamma \vdash e$ $\delta \mid : \operatorname{Trag}(\pi)[\delta]$	$_2: \tau'$. $\Omega: \Lambda'(\delta) \to \infty = [\delta], \Sigma[\delta], \Gamma(\delta) = - \operatorname{Test}(\alpha)[\delta] \models -$	$[\delta]; \Sigma[\delta]; \Gamma[\delta] \vdash$	$(\Delta; \Delta; 1) \vdash c2r(e) : R(C)$	I form Ler	nma 56, e_1 is a label and $H(v) = [v_0,$
alue, then E is a value. Otherwise, let $E' = C(e')$. By the congruence rule, $(H; V; E) \mapsto (H'; V'; E')$.	$g(\tau)[\delta] = Tag(\tau)[\delta],$	$\mathfrak{g}(\tau[\delta])$ and $\operatorname{Tag}(\alpha)[\delta] = \operatorname{Tag}(\alpha)$ because α is	$\{l_n^{\phi_n} : \tau_n\}$ and		s check gu	iarantees that the index e_2 is within
Case c2r_c: $E = c2r(e)$ with subderivation $\Theta; \bullet; \Sigma; \Gamma \vdash e : C$. By induction hypothesis, either e is a value or $\exists H' \mid V'$ and e' such that $(H : V : e) \mapsto (H' : V' : e')$. If e	$[\delta] \vdash E[\delta] : T$	[δ].	$J^{\phi_n} := \{\delta\}$)	ne step.	
by induction hypothesis, either e is a value of $\Im H$, V and e such that $(H, V, e) \mapsto (H', V, e)$. If e a value, then by Lemma 56 $e = C(v)$. Let $E' = v$. By $ev_e c^2r$ $(H; V; E) \mapsto (H'; V'; E')$. Otherwise let	else e_2 , $T =$	= τ with subderivations $\Theta; \Delta; \Sigma; \Gamma \vdash e_{t1}$:	$\dots, n_n \dots n_n[o]f$	} or $s = \{l_1 : :$	ongruenc	e rule E can evaluate one step.
$= c2r(e')$. By the congruence rule $(H; V; E) \mapsto (H'; V'; E')$.	$\vdash e_1 : \tau (C_1 = v_1[\delta] : Tag(C_1)$	= C_2). $\lambda[\delta]$, Θ : $\Delta'[\delta]$: $\Sigma[\delta]$: $\Gamma[\delta] \vdash e_{i2}[\delta]$: Tag(C_2) $[\delta]$.	there $S = \{l_1^{+}:$		th subder	ivations Θ : •: Σ : $\Gamma \vdash e_1$: array(τ). Θ : •
Case c2r_tv: not applicable because by Lemma 17, the subderivation $\Theta; \bullet; \Sigma; \Gamma \vdash e : \alpha$ is invalid.	δ] = Tag(C_1). $\operatorname{Tag}(C_2)[\delta] = \operatorname{Tag}(C_2)$. By ifTag_eq	nd $\Theta; \Delta'[\delta]; \Sigma[\delta];$ - $E[\delta] : T[\delta].$	L		
Case record: $E = \text{new}[\tau] \{ l_1 = e_1, \dots, l_n = e_n \}$ with subderivations $\Theta; \bullet; \Sigma; \Gamma \vdash e_i : \tau_i \forall 1 \le i \le n$.			$: \tau \forall 1 \le i \le n.$ $[\delta], \dots, e_{n-1}[\delta]]^{\tau}$		lue or er i	con evoluate one sten. So do eo and e
By induction hypothesis, each subexpression e_i either is a value or can evaluate one more step. If all a second provide the lat $W' = W$ and $E' = (1 - 1) + (1 -$	$\pm e_1 \in e_2, T = \pm e_1 \pm \tau (C_1 =$	$= \tau$ with subderivations $\Theta; \Delta; \Sigma; \Gamma \vdash e_{t1} :$ (ϵC_2)	$(; \Delta; \Sigma; \Gamma \vdash e_2 :$		fue of er a	Lamma E_{i}^{c} as is a label and $U(u)$
If all e_i are values, then let $H = H, \ell \rightsquigarrow \{\ell_1 = e_1, \dots, \ell_n = e_n\}$ (ℓ is a fresh label), $V' = V$ and $E = \ell$.	01.7 (017			1 & 31 ¹ , 11 &	cal iorm	Lemma 56, e_1 is a label and $H(v) =$
If $\exists e_i$ such that e_1, \ldots, e_{i-1} are values and $\exists H', V', e'_i$ such that $(H; V; e_i) \mapsto (H'; V'; e'_i)$ If e_{i1} or	e_{t2} can evalua	te one step, then by the congruence rule	E can eval	uate one step.		guarantees that the index e_2 is within
$w[\tau]{l_1 = e_1, \dots, l_i = e'_i, \dots, l_n = e_n}$. By the congruence rule, $(H; V; E) \mapsto (H'; V'; E')$. Case if	$fag_tv: E = i$	$fEqTag^{\tau}(e_{t1}, e_{t2})$ then e_1 else e_2 with su	bderivation	$\Theta; \bullet; \Sigma; \Gamma \vdash e_{t1} : Ta$	$g(\gamma)$.	ep.
Case field: $E = e.l_i$ with subderivation $\Theta; \bullet; \Sigma; \Gamma \vdash e: \{l_1^{\phi_1} : \tau_1, \dots, l_n^{\phi_n} : \tau_n\}$ and $1 \le i \le n$	icable because	the subderivation is invalid by Lemma	17.			uence rule E can evaluate one step.
By induction hypothesis, either e is a value or e can evaluate one step. By induction hypothesis, either e is a value or e can evaluate one step.	D: $E = e$ with stion hypothes	is subderivation $\Theta; \bullet; \Sigma; \Gamma \vdash e : \tau_1$.	ne sten. Th	at is aither E is a	value or E	$\Theta; \bullet; \Sigma; \Gamma \vdash e_1 : \tau.$
If e is a value, by canonical form Lemma 56, e is a label and $H(e) = \{l_1 = v_1, \dots, l_n = i $ can evaluate	one step.	is, either e is a value of e can evaluate of	me step. In	at is, either 15 is a	value or E	an evaluate one step. If e_1 is a value,
$=H, V'=V$ and $E'=v_i$. By ev-field $(H;V;E) \mapsto (H';V';E')$. $K \supseteq H', V', e'$ such that $(H, V, e) \mapsto (H', V', e')$ then let $E'=e'$, and by the congruption rule is	one step.					rule E can evaluate one sten
$\Pi \exists H, V, e$ such that $(H; V; e) \mapsto (H; V; e)$, then let $E = e \exists_i$ and by the congruence rule Corollary $\{V, V', E'\}$	58 (progress) If Σ ⊢ P : τ, then either the main exp	pression in I	r is a value, or ∃P	such that	$\Omega(\alpha) \Sigma \Sigma \Sigma = \alpha + \Sigma(\alpha)$
In the rest cases, we state only which evaluation rule to apply, but omit the new H' , V' or $P \mapsto P'$.						$\Theta; \bullet; \Sigma; 1 \vdash e_1 : 1 (x).$
Case assign $E = e_1 l_i := e_2$ in e_3 with subderivations $\Theta; \bullet; \Sigma; \Gamma \vdash e_1 : \{l_1^{\phi_1} : \tau_1, \dots, l_i^M : \text{Theorem 5}\}$	9 LILe is Se	und Well-tuped LIL _C programs do not	aet stuck.			an evaluate one step. By $\Theta; \Sigma \vdash V : I$
and Θ ; •; Σ ; $\Gamma \vdash e_2 : \tau_i$.	5 mil 10 00	and men system and programs to not	Ace nearrai			e1 is a value, by ev_assign E can evaluate a value.
By induction hypothesis, either e_1 is a value or e_1 can evaluate one step. Similarly, either error by	y progress and	preservation.				step.
e_2 can evaluate one step. If both e_2 and e_3 are values, then by canonical form Lemma 56 e_2 is a label and $H(e_2) = I_1 = \frac{1}{16}$		C. 21 112-241 21 2 222-				lerivations $\Theta; \bullet; \Sigma; \Gamma \vdash e : \forall tvs(\tau_1,$
In both e_1 and e_2 are values, then by canonical form Lemma 50 e_1 is a label and $H(e_1) = \{i_1 = i_1, \dots, i_n = \dots\}$, and by ev_assignR E can evaluate one step.	$[\delta] : \tau_1[\delta]$. Be	cause type substitution preserves subtyping			$1,, t_m/$	tvs.
If either e_1 or e_2 can evaluate one step, by the congruence rule, E can evaluate one step.	$[\delta]; \Sigma[\delta]; \Gamma[\delta]$	$\vdash E[\delta] : T[\delta].$		1	alue or e	can evaluate one step. So does each
Case array $E = \text{new}[e_0, \dots, e_{n-1}]^{\tau}$ with subderivations $\Theta; \bullet; \Sigma; \Gamma \vdash e_i : \tau \forall 0 \le i \le n-1$	7 = 71	$\langle \alpha_1, \ldots, \alpha_m, and \forall 1 \le i \le m, \Theta : \bullet \vdash t_i \ll$				con conduct one step: too does does
	$-e[\sigma] : \tau[\sigma].$	further thanks and the 2 to 2 million to at a				The second
					anonical I	orm Lemma 56, e is a label and $H(e) =$
Case record: $E = new[\tau]\{t_1 = e_1,, t_n = e_n\}, I = \tau = \{[t_1^{-1} : \tau_1,, t_n^{r_n} : \tau_n]\}$ with subderivations $\Theta : \bullet \Sigma : \Gamma \vdash e_i : \tau_i \forall 1 \le i \le n$		A DE ALLO DA MINI DO ALLO INCOM	d aubtroing m	les nomestively	evaluate o	one step.
If the congruence rule applies, then $\exists H', V'$ and e'_i such that e_1, \ldots, e_{i-1} are all values, $(H; V; e_i) \mapsto (H; H; V; e_i) \mapsto (H; V;$		F) with subderivations $\Theta; \Delta; \Sigma; 1 \vdash e_i : \neq \forall 0 \leq i \leq 1$ $\Theta; \Delta' \vdash \Theta$	$\vdash \alpha \ll \alpha \uparrow_{\Delta}$. B	y the second part of	the cong	ruence rule E can evaluate one step.
$(H'; V'; e'_i)$ and $E' = \{e_1, \dots, e_{i-1}, e'_i, e_{i+1}, \dots, e_n\}$. By induction hypothesis, $\exists \Sigma'$ and Γ' such that expression in subavariase in the subavarias	the conclusion.	$\Delta \ll u \mid \Delta$.			$: \tau')$ with	subderivation Θ ; •; Σ ; $\Gamma \vdash e : \tau'[\tau/\alpha]$.
Lemma 27 premises. Also the side conditions are decidable.		g of Environments.			ue or e can	n evaluate one step. If e is a value, the
Proof: by in Theorem 48 Type checking of LIL_G is decidable.		Γ , $\Theta \vdash \Delta' \ll \Delta$ and Θ ; $\Delta' \vdash \Gamma' \leq \Gamma$, then Θ ; Δ' ; Σ	$; \Gamma' \vDash E : T' an$	$d \Theta; \Delta' \vdash T' \leq T.$	uence rule	E can evaluate one step.
Case as Proof: To decide whether $\Theta \land \Sigma : \Gamma \vdash e : \pi$ holds, we can first get the minimal type π .	of e such that	on algorithmic typing rules.			with sub-	derivation $\Theta : \bullet : \Sigma : \Gamma \vdash e_1 : \exists \beta \ll \pi$
Θ : $\Delta \models \tau_i \leq \Theta$ By indue Θ : Δ : Σ : $\Gamma \models e : \tau_m$, then test whether Θ : $\Delta \vdash \tau_m \leq \tau$. Because both minimal typing and	subtyping are	$\mathfrak{r}, T = \Gamma(x).$ $\mathbf{y} = \mathbf{var}, \Theta; \Delta'; \Sigma; \Gamma' \models E : T'.$ And by the defi	inition of $\Theta: \Delta'$	$\vdash \Gamma' \leq \Gamma$, we have	10.05.000	n ambusto one stop. If c. is a mbus th
followed by decidable, type checking is decidable.					ue or ei ca	in evaluate one step. If e1 is a value, th
⁵ It is synta		br, a_label and a_tag: trivial. = $C(e)$, $T = C$ with subderivation Θ : $\Lambda : \Sigma : \Gamma \models e$: R(C)		$v:\tau$). By	ev-open E can evaluate one step. If
E Soundness		hesis, $\Theta; \Delta'; \Sigma; \Gamma' \models e : \tau' \text{ and } \Theta; \Delta' \vdash \tau' \le R(C). 1$	By inversion of	subtyping Lemma 41	evaluate o	one step.
E 1 Sech-titertion		. By reflexivity of subtyping Θ ; $\Delta' \vdash T' \leq T$. By	a_object $\Theta; \Delta$	$'; \Sigma; \Gamma' \models E : T'.$	$\operatorname{id}(\alpha, x)$ is	n e_1 else e_2 with subderivation Θ : •: Σ :

We've Got a Problem Here

 As a practical matter, a large fraction of the proofs being done about programming languages today are not possible for humans to check.





Are Proof Checkers the Answer?

- Proof checking (or "proof assistant") technology has made amazing strides in recent years
 - Several mature systems
 - E.g., Coq, HOL, Isabelle, Twelf, MetaPRL, etc., etc.
 - Some very impressive achievements in several domains



A Marriage Made in Heaven

So we've got...

- A. A community with a burning need
- B. A community with a great technology

Can we put them together?



Lots of Work Already Underway

- Leroy's verified C compiler
- Nipkow et al's formalization of a large part of Java
- Appel et al's Foundational Proof-Carrying Code project
- Crary et al's machine-checked development of a typed assembly language
- Harper et al's formalization of Standard ML
- Sewell et al's formalization of TCP/IP
- Etc., etc.



But We're Not There Yet!

Challenges:

- Current achievements are mostly heroic efforts by heroic individuals and small teams
- Many different proof assistants and diverse technical machinery
- Lots of black magic; high cost of entry; little sharing of knowledge across projects
- Some significant unresolved technical issues
 - Lightweight methods for reasoning about variable binding
 - Scalability and modularity of proofs



Vision

A world where *every* PL paper is accompanied by a machine-checked appendix

- Plan:
 - Identify current best practices
 - Gather community consensus around them
 - Build additional tools and other infrastructure as needed
 - Address technical challenges specific to programming language research

First step...



The PoplMark Challenge

- A set of benchmark problems to help evaluate progress in the area
 - Based around metatheory of F<:, a typed lambdacalculus with polymorphism and subtyping
 - Presented at TPHOLs 2005
- Has generated tremendous interest in both PL and theorem proving communities
 - many solutions submitted
 - 6 different proof assistants
 - 7 different treatments of binding
- Much has been learned
 - JAR special issue CFP now open



Community Development

- Wiki & Mailing list for POPLmark challenge
 - Gathering place for news/solutions/discussion/ advice
- Workshop on Mechanizing Metatheory
 4th Wksp, 4 Sep 2009, Edinburgh
- Using Proof Assistants for Programming Language Research
 or How to write your post POPL paper in
 - or, How to write your next POPL paper in Coq
 - Tutorial for novices
 - POPL tutorial, January, 2008
 - Oregon Summer School, June 2008



Engineering Formal Metatheory



Resolving technical issues

- Engineering Formal Metatheory
 - Brian Aydemir, Arthur Charguéraud, Benjamin Pierce, Randy Pollack, and Stephanie Weirich
 - POPL 2008
- Describes a lightweight first-order methodology for representing binding and specifying induction principles
- Two essential components:
 - Locally nameless representation
 - Cofinite name quantification



What is so hard about binding?

- Alpha-equivalence
 - Identify " $\lambda x.x$ " and " $\lambda y.y$ "
- Barendregt convention
 - Assumption that bound variables are "sufficiently fresh"



Alpha-equivalence

- Important when we need to compare terms with binding structure:
 - Type system of polymorphic language
 - Confluence for pure lambda calculus
- Formalism simpler if alpha-equals is "="
 Lemma preservation: forall G e t,
 typing G e t -> red e e' ->
 exists t', alpha_eq t t' /\
 typing G e' t'.

Lemma preservation: forall G e t, typing G e t -> red e e' -> typing G e' t.



Barendregt Convention

- What if z == x?
- What if z in free variables of u?



Existing approaches

- No completely satisfactory mechanism for binding
 - Name representation: must explicitly rename terms, define alpha-equivalence
 - de Bruijn indices: difficult to work with as must shift indices
 - Nominal logic: only available in Isabelle/ HOL
 - HOAS: exotic terms, specialized logic



Locally nameless representation

Names for free variables de Bruijn indices for bound variables

Example: Untyped lambda calculus terms

 $t ::= bvar i | fvar x | app t_1 t_2 | abs t$

 $\lambda x. \lambda y. (x y) z$ and $\lambda w. \lambda v. (w v) z$ represented as abs (abs (app (app (bvar 1) (bvar 0)) (fvar "z")

ductiv	/e	exp : Set :=
bvar	•	nat -> exp
fvar	•	atom -> exp
abs	•	exp -> exp
app	•	exp -> exp -> exp.
	luctiv bvar fvar abs app	<pre>luctive bvar : fvar : abs : app :</pre>



Basic operations

- Variable opening
 - t^u replace bound index 0 with exp u
- Free variable calculation
 - FV *t* finite set of free atoms in *t*
- Substitution
 - $[x \mapsto u]t$ replace free variable x with u
- All operations have *simple* definitions.



Variable opening

Fixpoint open_rec (k : nat) (f : exp) (e : exp) {struct e} : exp := match e with | bvar i => if k === i then f else e | fvar x => fvar xl abs e1 => abs (open_rec (1 + k) f e1) | app e1 e2 => app (open_rec k f e1) (open_rec k f e2)

end.

Notation "e ^ u" := (open_rec 0 u e).



Free variable substitution



Free variable calculation

Fixpoint fv (e : exp) {struct e} : atoms := match e with | bvar i => {} | fvar x => singleton x | abs e1 => (fv e1)l e1 e2 => (fv e1) `union` (fv e2)



Local closure

- Not all members of type term are lambda calculus terms
 - abs (bvar 3)?
- Predicate *lc* picks out members datatype that are *locally-closed*
- Definitions respect local closure
 - If lc u and lc t, then $lc ([x \mapsto u]t)$

- If $t \rightarrow t'$ then lc t and lc t'



Managing local closure

• Many theorems need not refer to local closure

- If $t \rightarrow t'$ and $t \rightarrow t''$, then t' = t''

• Some theorems require it, tactics discharge assumptions

- If
$$x \neq y$$
 and $lc u$, then
 $[x \mapsto u](t^y) = ([x \mapsto u]t)^y$



Induction principles

Definition of typing rules generates an induction principle for typing derivations

$$ok E \quad (x:T) \in E$$
$$E \vdash \text{ fvar } x:T$$
$$E \vdash t_1: S \rightarrow T \quad E \vdash t_2:S$$
$$E \vdash \text{ app } t_1 t_2:T$$

$$x \notin FV \ t \cup dom(E) \quad E, x: S \vdash t^x : T$$
$$E \vdash abs \ t : S \rightarrow T$$



Exists vs. Cofinite Quantification

Induction hypothesis holds for some particular, unknown x

$$x \notin FV \ t \cup dom(E) \qquad E, x: S \vdash t^x : T$$
$$E \vdash abs \ t : S \rightarrow T$$

Induction hypothesis holds for all but some finite set of variables.

$$\begin{array}{c|c} \forall x \notin L & E, x : S \vdash t^x : T \\ \hline E \vdash abs \ t : S \twoheadrightarrow T \end{array}$$



Weakening Lemma

```
If E, G \vdash t : T and ok(E, F, G)
   then E, F, G \vdash t : T
Proof (?):
  by induction on E, G \vdash t : T
  Abstraction case:
                 x \notin FV \ t \cup dom(E,G) \quad E,G,x:S \vdash t^x : T
                             E.G \vdash abs \ t : S \rightarrow T
  WTP E, F, G \vdash abs \ t : S \rightarrow T
  By typing rule, holds if E,F,G,x:S \vdash t^x:T
     and x \notin FV \ t \cup dom(E,F,G)
```



Weakening Lemma

If E, $G \vdash t$: T and ok(E, F, G)then E, F, $G \vdash t : T$ **Proof:** by induction on $E, G \vdash t : T$ abstraction case: $\forall x \notin L \quad E, G, x: S \vdash t^x : T$ $E, G \vdash abs \ t : S \rightarrow T$ WTP: $E, F, G \vdash abs \ t : S \rightarrow T$ By rule, holds if exists a set L', such that $\forall x \notin L'. E.F.G.x: S \vdash t^x: T$ **IH:** $\forall x \notin L$, $ok(E,F,G, x:S) => E,F,G,x:S \vdash t^x : T$ Choose $L' = dom(E,F,G) \cup L$ By definition, $\forall x \notin L'$, ok (E, F, G, x:S)



Renaming lemma

- Similar reasoning proves substitution lemma If E, x:S, $F \vdash t : T$ and $E \vdash u : S$ then $E,F \vdash [z \mapsto u]t : T$
- Important corollary of substitution and weakening

Renaming: If $x \notin FV \ t \cup dom \ E$ and $E, \ x:S \vdash t^x : T$ then for all $y \notin FV \ t \cup dom \ E, \ E, \ y:S \vdash t^y:T$



Corollaries of renaming

 Renaming lemma gives strongest intro form:

If exists x, s.t. $x \notin FV \ t \cup dom \ E$ and $E, x: S \vdash t^x : T$ then $E \vdash abs \ t : S \rightarrow T$

• And strongest inversion principle: If $E \vdash abs \ t: S \rightarrow T$ then for all $x \notin FV \ t \cup dom \ E, \ E, x : S \vdash t^x : T$



Equivalence of systems

- Renaming lemma also shows equivalence of two systems:
 - $E \vdash t : T$ with exists-fresh rule for abs if and only if $E \vdash t : T$ with cofinite rule for abs

• We are proving properties about the language we actually care about!



General Form of Development

- Def. of language syntax
- Def. of variable opening and local closure
- Def. of free variable substitution
- Def. of free variable function
- Interaction lemmas
- Def. of semantic relations using cofinite quantification for binders
- Show semantic relations respect local closure
- Substitution and weakening lemmas for each relation w/ binding
- Preservation and Progress
- Derive renaming lemmas



Code Distribution

- Reference *examples* & supporting experience
 - multiple calculi (STLC, F<:, CoC)
 - multiple theorems (type soundness, confluence)
- A *library* that supports this methodology
 - atoms, finite sets
 - reasoning about freshness
 - representing environments

Let's look closer at POPLmark...





Fsub Definitions.v



| exp_trac | exp_trac | exp_abu | exp_tabu | exp_tabu | exp_tabu

spoint spen, match T with | typ_tap | typ_tame 3 | typ_fine 3 | typ_arrow | typ_all Ti

Hartold & Griff | exp_losur i | exp_losur i | exp_alsi T | exp_alsi V | exp_Calsi V | exp_Calsi V

spont open. natch e with I exp_bear i I exp_fmar x I exp_abe H I exp_tabe H I exp_tabe V I ex

Definition open. Definition open.

type_all : F type T1 -> (Forall E true (true a)

Inductive expr : $exp \rightarrow Prop :=$

automation hints

silwary.	l engr.ver : ferall s.	mb X 51 72 ···
	expr Geop_fvar s)	sub E (typ_arrow 51 52) (typ_arrow Ti T2)
1 (P)	empr.she : ferall L T ed.	<pre> sub.all : ferall 1 E 51 52 T1 T2.</pre>
	Type T in	rab E T1 51 ->
1 -9 typ	(farall x : atom, x 'notin' 1 + expr (open.ee.war el x)) =>	(forall X : ston, N 'notin' L =>
on ris top	ergr (erg.abs T el)	<pre>mub (E # X T4: T1) (open.tt.mar 52 X) (spen.tt.mar T2 X1) =</pre>
s of two of two	Lange are - forall at #2.	rah E (typ all \$1 52) [typ all Ti 72]
s of two of two.	aver at the	and a relificant at any relificant in and
Proceeding of the states	4707 42 13	
	ever (eee an at all)	Inductive typing : end -> end -> typ -> From :*
TR AND	i aver take : foreil [T al.	i traine var i furall E a T.
the set	topo T de	ski I ->
the same till same	Chronic X - atom X (actin) 1 th game domas to use al X1 th	binds a (bind two T) E ->
the same of same	source from the P all	Summing E (and from a) T
the set of set	i anno i Sonali al V.	Limburg also - Journall L E V al TL
the state of state.	entry al ->	(formild a r mine, a faction L ->
on the sector	type I =>	towing (F E a "; V) (seen as you at a) T15 -5
and Will match (11 - mark) 17 - mark). Contract 71 - mark -	store from Lang at No.	tenter E (are also E at) (ten avenue 2 71)
is not must on a difficient of the manufactory of the ta-	and confidents and the	Limiter and a family 71 E at al 72.
all time term	Tedesting hinding - Tel -s	Summer F at (inc access 71 77) ab
at 15 K and 1 then K alon (ton hour 1)	I had to be a first of the second sec	tipling a st tripperter to table?
ab the face 2	i hand the class of hinding	append a ferra and at any at
the second forces in and W.W.W.W. (come in case & H. Will	i none_typ : typ -> noneng.	allered a Auch while at any its
a se all deserve to see a lit all frame is see (0.00 if and	Reserves and the descent structure in	i speak to store a factor i an
so the per color course a cro-other control a co-	monators and to give the interest of	provent A r when a short to see at a from at an at all of
	Socarios septy := (Sail (ston = booting)).	shing on a y of all objected on any (shearer on all objected on an all objected on a real of all objected on an all objected on a real objected on
and the second state of th		typing E (exp_takes V el) (typ_all 0 Ti)
et (E : max) (U : 526) (e : exp) (attract e) : exp :=	Rotation "Z "C T" (= (Z " blod_sub T)	T typing_tapp : Dorall Ti E 41 T T2,
	(at level 31, left annulativity) : say, suge-	typing i at tryplant ti tizz ->
-o sap_bras 1	Botation "s ": T" := (s " bind_typ T)	100 L T 11 19
o exp_frar x	(at level 31, left associativity) : eav_scope.	typing E Geop.tepp ed TD Cepem.tt T2 TD
<pre>+O exp_whs (upen_tt_res X T N) (open_te_res E U e1)</pre>		T typing_out : ferall 5 E e T,
<pre>+> exp_app (spes_ts_res X T et) (open_ts_res E U e2)</pre>	Inductive wft : env -= typ -= Prop :=	tjping E e 5 ===
<pre>+> exp_tabs (spen_tt_res X T N) (spen_te_res (T R) T e1)</pre>	wft_top : forall E.	BAD I S T 10
<pre>+> exp_tapp (spen_te_res X T et) (spen_tt_res E U V)</pre>	wit E typ_top	tpping E e T.
	<pre>! wft_war : forall U E CK : stoml,</pre>	
	binds X Coint_sub ND E ->	Inductive value : exp => Prop :=
ec Dk : mat) Cf : emp0 De : exp1 - Entrect e3 : exp :=	wit E (typ_fvar E)	I walme.abs : forall T ed,
	I wft_arrow : forall E T1 T2.	espr Excpuebe T ei) 🛞
<pre>if k === i then f else [exp_bvar i]</pre>	vdt E T3 ->	value (explate T ell
++ exp_fver x	vdt E T2 ->	I valme_tabs : ferall T et,
<pre>www.abu V (open.ee.rec 15 k) d ed)</pre>	wft E DypLarrow Ti T21	empr Excp_tabs T ell 🔅
explapp (open_ee_rec k f ei) (spen_ee_rec k f e2)	I sft.all : forall L E T1 T2.	value (exploate T et).
🕫 expltaba V Copenleelrec k f ei)	vdt E TJ ->	
expltapp [spemiew.rec k f el] W	Cfurall X : stom, R 'motin' L 🗇	Induitive red : exp -> exp -> Prup :=
	wft (E & X "4: T1) (open.tt.mar T2 30) ->	<pre>/ red_app_1 : forall el el' el;</pre>
	wft E DypLall Ti T21.	c- De topie
T U := open.tt.rec O U T.		red al al/ ->
e U := open.te.rec O U e.	Induitive ski : eav -> Prop :*	red (exp_app e1 e2) (exp_app e1' e2)
el el := spesiesire: 0 el el:	ski_mapty :	red_app_2 : forall of e2 e2',
	oht sapty	value el ->
.var T R := (spen.tt T (typ_fvar R)).) sht_auk : Forall (E : ear) (X : stam) (T : typ),	ind a2 a2' ->
.var e R := (spen.te e (typ_fvar R)).	she E -> with E -7 -2 familian' she E -> she (E = 2 -7 -7 -7 -7 -7)	red (exp_app e1 e2) (exp_app e1 e2/)
_var = s := (spen_ee = (esp_Evar s)).	$ skt_typ + forall (E + exp) (x + airs) (T + typ),$	red_tapp : forall el el' %,
	shi E -> shi E T -> x 'antin' dom E -> shi (E E x "; T).	Spper V ->
p -> Peop i*		and al al/ ->
	Industive sub : may -> typ -> typ -> Prop :*	and (exp_tapp of V) (exp_tapp of V)
	sub_top : Donall E Z,	red_abs : forall T el 02,
all 3.	alt E ->	espo (esp_abs T el) ->
ar X)	uti E 2 ->	value v2 😳
seall Ti T2,	ank E Z typ_tap	red Complage Complate T all v21 Copenies al v21
	sub_refl_trar : forall E 2,	I red.tahu : forall Ti el T2,
	shi E ->	empr (axp.tabs Ti wi) >>
res T1 T2)	with E (hyp_from E) ->	type T2 +>
A11 L T1 T2,	sub E (typ_fvar E) (typ_fvar E)	red Geop_tapp (exp_tabu Ti ei) T21 Copen_te ei T21.
	I sub_trans_tvar : forall U E T X,	
atom, X 'notin' L -> type (spen_tt_tar T2 X)) ->	binds X (bind, sub E) E =0	High Constructors type amor oft sit value red.
	FIGURE IN DEPENDENCE IT &	the second second second second second second second
1 T1 T2)	anh E U T 🗇	Hint Resolve sub_tep sub_refl_tvar sub_arrow.
1 T1 T2)	sch E U T -> sch E ityp_fvar RI T	Hint Beeclee sub_tep sub_refl_tvar sub_acros. Hint Beeclee typing_var typing_app typing_tapp typing_sub.
1 11 12)	mch E O T -> anh E Oxp.stvar KI T I mch.errav : formill E Si 52 Ti T2.	Hint Repolve sub_tap sub_refl_tvar sub_acros. Hint Resolve typing_var typing_app typing_tapp typing_sub.



Fsub_Infrastructure.v

Basic operations (fv and subst)

Tactic to choose fresh atoms

Properties of open, fv and subst

Properties of subst and local closure

in toport that platest and		Test Stap, Soll-
and display (2) - 1980 (present 2) - second in	Toric territe "add, bed" user (1 hat dentid in an table clean be placed as	destruct to - C structure fits decodifies.
nter : : : : : : : : : : : : : : : : : : :	Sama aparentary para construction of a parameters of the second secon	Land California (Lando) - Parcell K. M. G. C. Tandari, "doctor 'The co- spanses of the re-random set of computer course for 20. Parcella and a state."
AND A TE OF STORE TO SAME AND THE	Mark and sectors (2.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1	and the spectral provides and the spectral spectra spectral spectral spectra spectral spectral spectra
A fight to the charge (proved a) is below it to the charge (proved a) is below it proved in the constraints of the constraint	Second Contraction of the second	teres aparteres approach i marié a jair P.
puter en la secto de la succession de la secto de la s	T + genovalence k P.C.	Berlin, M., Miller, J. & M. & Aller, A. Phys. J. 7 (1996) 201000 (1997) 111 (1997) 11100 (1997).
point P is to door to failed include and	Next and another and the second secon	table at a series y to the surger of the surger of the surger of the surger of the
na mana ta magin General di Lanana an-	State Tripp, path. resided specific and specific in P. provide provide a set.	Lands oper (a proc.) (process) in the set of (1) is the (1) of (1) of sets (b) one (1) is a 1-spectra process of the process of (1) of (1).
plant to the sequence of the second s	ingelie ingenue sures, especies 20 in Organies 2011 -	 a applications (1.2.4). France. Landonization, Antonia (1.2.1). This frag. stand in the internation frag.
plage at at the Group and Transmit Group and plage if eff the Group and plage at the Group and	Tanna and Andrews Constraints and Anna. Dimeter Constraints and Index Constraints.	toget here and the statistics
a second a second s	Band and anter index on Proceeding of the Science Science Science of Science (Science).	Lands open (A) and (Apple) Faculty in Fig. (2011) 2015 10 (1990) 10 (10) 10
N F Land. 9.749 IN TEL (MP Charge 7 IN TEL (MP	dependent in er D statusting og at statusterter - fert	Reset and a first second as balance a first do conset as balances We defense to a single channels, only using second-based-based.
prove to the first the first time to be the second state of the se	Dense ministry approximate a ferrill To TO 8.9 ft.	The Augusta Staty
A ANALY A DISTANCE OF CASE AND AND AND AND AND AND AND AND	allerium 2 P. Japan, A. Lee A. 2012 Applications in Control on P. Net. Instances 3, P. Fel., Terrard. Article and A.	Tappin Space, Stranger, and A. S. Sang, Phys. Rev. Lett. 2010, 100 (1997); 2011. [INTERCONFIDENCE. INFINIT Approx. (INCOME.), 2010. [INTERCONFIDENCE.].
A D UPA. (Jane 1 Transplane 1 (Jane 2 Transplane 2	Barras Di Li	<pre>printing to space to previous and the property to last</pre>
(c) and the strength of the	- Andread (a) (2010) and a start - Const Prop. Sect	Lawsen entering and another in the staff of
one of the solution interface of the solution of the	Det.	Scheduler I. F. et al. Processing and a scheduler of a scheduler and a scheduler. In Joseph Landscheduler of scheduler in Scheduler and a scheduler. In Joseph Landscheduler and Scheduler and Scheduler and Scheduler and Scheduler Scheduler and Scheduler and Schedule
an particular for the second data amplifies the sample for each of a support to a second sec second second sec	100 P. C. Mind, C. & D. Sagar, N. M. D. Lagar, C. Sadar, N. & B. W. Sadar, N. & T. W. Reed, and M. Satar.	ten. Leen minute gen, incent colored, e 7 (7).
A real of the second state	and an entroid open.ex.	Table 1 of the second state of the second stat
Com Fiel + moving Fighters (1000)	94.	Intel Control Party Interests
the education of a start of a local de chinesing a binding of	The First Control of C	an. Tana atau a ana ing tanà 1711.
na and a first management of the first	training of the second se	Tape E of manual E Figure a Dimagnetic Intervie C Fail Intervie C F
	mention reduction approximately and a second s	and parts.
offeet private into	10 C C C C C C C C C C C C C C C C C C C	and the characteristic and the second s
offen, states at the second se		terms when on one or one control \$15.00
$ \begin{array}{l} (A + a) = A + a \\ (A +$	Communication of a second s	Lanes salars en apas no car i mesti 6.9.9 s. 1930 - Lai 1939 - P. O
$\label{eq:constraints} \begin{array}{l} \mathrm{det}_{\mathbf{x}_1,\mathbf{y}_2,\mathbf{x}_3,\mathbf{x}_1,\mathbf{y}_2} \\ + & \mathrm{det}_{\mathbf{x}_1,\mathbf{y}_2,\mathbf{y}_3,\mathbf{x}_$	The same relations is made a lot in a second relation of the same rel	Lands and the comparison of the state of
$\label{eq:starting} \begin{split} & \text{def}(\mathbf{x}_1, \text{def}(\mathbf{x}_1)) & = 1 & \text{def}(\mathbf{x}_1) & = 1 & \text{def}(\mathbf$	$\label{eq:constraints} \begin{array}{l} \mbox{trans} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	Lines and the paper have a strength of the term of the term of the strength of the term of the strength of
$\label{eq:second} \begin{array}{l} \mathrm{second} & sec$	$ \begin{array}{l} \mbox{transmission} transmission$	Let us the constant of the second set of the se
<pre>Add</pre>	where the set of the	Land Lange AL, and AL, and AL
	 Margin M., Margin M., Ballar D. M., Land M., M. M., Margin M., Margin M. M., Land M., M. M. M., Margin M., M. M. M., Margin M., M. M. M. M., Margin M. M. M., Margin M., M. M.	
или дола () -) () () () () () () () () () () () () (<pre>improvement : Name (First) </pre>	Idea contract parts from a "contract the first of the fir
ningen of the second se	Hand South State (Section 1997) Hand State (Here server a server
And young () And young () A	"""""""""""""""""""""""""""""""""""""	Here care approximate "installation". The set of the s
And Angele () Provide a second secon	()	Idea contra cano have " result of the ". The " is a " is
<pre>Main_prove ()</pre>	<pre>import = indication. for a contrast = indication. for a con</pre>	Here realizes a sector and the secto
And And 1 And And 2 And And And 2 An		Here represent a free hard a free free free free free free free fr
And A grade ()	<pre>""""""""""""""""""""""""""""""""""""</pre>	Here results appendies in werd affende in the second appendies in the second a
And your (And yo	<pre>///</pre>	<pre>Description to the section of t</pre>
And A	<pre>************************************</pre>	Here is a sector of the sector. Here is a sector of the s
Add (Add (Add (Add (Add (Add (Add (<pre>""""""""""""""""""""""""""""""""""""</pre>	Han contract approx data of the set of the s
And A general (<pre>image is a start it is it</pre>	Hand the second of the second
And your (<pre>image: set in the set is a set is</pre>	
Answerster 1		Here representations and and a second and a
And young () And young () An	<pre>""""""""""""""""""""""""""""""""""""</pre>	Identify approved interface. The second approved interface.
	<pre>""""""""""""""""""""""""""""""""""""</pre>	Handback and a set of the set of
<pre>charged i</pre>	<pre>""""""""""""""""""""""""""""""""""""</pre>	<pre>classical case case case case case case case case</pre>
And your () And y	<pre>""""""""""""""""""""""""""""""""""""</pre>	
<pre>statusti i</pre>	<pre>""""""""""""""""""""""""""""""""""""</pre>	<pre>descriptions.com : .com 24.00.0 implications in the second s</pre>
<pre>designed i</pre>	<pre>image:</pre>	<pre>classical case case case case case case case case</pre>
	<pre> """ ""</pre>	<pre>classical case was a result of the state of the stat</pre>
And a part of the second of th	<pre> """ ""</pre>	
	<pre>image: set is a set is a</pre>	
	<pre> interface interface</pre>	
	<pre> """ ""</pre>	



Fsub_Lemmas.v

type well-formedness

environment well-formedness

substitution in environments

regularity of relations

hints about local closure







Fsub_Soundness.v

Proofs of lemmas from the Appendix of



Comparison POPLmark 1A

Author	Binding	Lemmas	Proof steps
Vouillon	de Bruijn	30	402
Leroy	Locally nameless	49	495
Stump	Levels/names	56	938
Hirschowitz & Maggesi	de Bruijn (nested datatype)	49	1574
Chlipala	Locally nameless	23	75
This work	Locally nameless	22	101



McKinna & Pollack Rule

Exists-fresh: IH holds for some fresh x

$$x \notin FV \ t \cup dom(E) \quad E, x: S \vdash t^x : T$$
$$E \vdash abs \ t : S \rightarrow T$$

Cofinite: IH holds for all but some unknown set

$$\frac{\forall x \notin L \quad E, x: S \vdash t^x : T}{E \vdash \text{abs } t: S \to T}$$

Forall-fresh: IH holds for all fresh variables

$$\forall x \notin FV \ t \cup dom(E) \qquad E, x: S \vdash t^x : T \\ E \vdash abs \ t : S \rightarrow T$$



Forall vs. Cofinite

Define system with forall-fresh rules

Define swapping and show relations are stable under swapping

Show exists-fresh intro

Prove weakening and substitution

Define system with cofinite rules

Prove weakening and substitution

Show exists-fresh intro

Prove type soundness

Prove type soundness

Conclusions

- Can use Coq's standard mechanisms for reasoning (inductive defs, tactics, etc.)
- Swapping does not appear to be essential.
- Seldom need to rename during proofs. IH applies to an infinite # of suitably fresh variables.
- Specialized tactics help
 - local closure obligations
 - fresh variable introduction



Thanks to

Brian Aydemir Arthur Charguéraud (INRIA) Randy Pollack (Edinburgh) Peter Sewell (Cambridge) Aaron Bohannon Matthew Fairbairn (Cambridge) J. Nathan Foster **Benjamin Pierce** Jeffrey Vaughan **Dimitrios Vytiniotis Geoffrey Washburn Steve Zdancewic**



Vouillon / de Bruijn indices

```
Fixpoint subst (t : term) (x : nat) (t' : term)
  {struct t} : term :=
 match t with
  l var y =>
     match lt_eq_lt_dec y x with
      l inleft (left _) => var y
      l inleft (right _) => t'
      | inright _ => var (y - 1)
     end
  | abs T1 t2 =>
    abs T1 (subst t2 (1 + x) (shift 0 t'))
  | app t1 t2 => app (subst t1 x t')
    (subst t2 x t')
  | tabs T1 t2 => tabs T1
    (subst t2 x (shift_typ 0 t'))
  | tapp t1 T2 => tapp (subst t1 x t') T2
 end.
```

