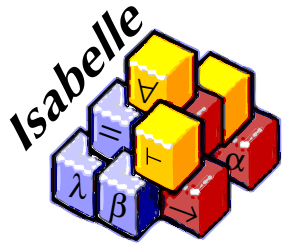
A solution to the PoplMark challenge in Isabelle/HOL

Stefan Berghofer
Technische Universität München



Motivation

Many proofs about programming languages are ...

- ... long and tedious, with few interesting cases ("write-only")
- ... straightforward with pencil and paper, but ...
- ... still "rocket science" when it comes to mechanization [Greg Morrisett]

Main problems when formalizing programming languages

- Management of many details
- Danger of small mistakes or overlooked cases
- Poor scalability: hard to keep definitions and proofs consistent
- Reuse of work
- Ensuring tight relationships between theory and implementations

The PoplMark Challenge [Benjamin Pierce et al., TPHOLs 2005]

Task: Formalize basic properties of polymorphic λ-calculus (System $F_{<:}$)

Idea behind challenge

- Assess "state-of-the-art" in the area of proof assistants
- Suitability of theorem provers for proofs about programming languages
- Vision: Papers submitted to confereces like
  - POPL (Principles of Programming Languages) or
  - ICFP (International Conference on Functional Programming)

  should be accompanied by machine-checkable proof scripts

Requirements for proof assistants

- Reasonably small technological overhead
- Notation close to "usual" conventions
- Easy to learn
- Proofs and specifications should be reusable
- Possibility to generate executable prototypes

Parts of the challenge

- **1A:** Transitivity of Subtyping
- **1B:** Transitivity of Subtyping with Records
- **2A:** Type Safety for Pure $F_{<:}$
- **2B:** Type Safety with Records and Pattern Matching
- **3:** Testing and Animating with Respect to the Semantics

Complete solutions to the challenge

Author	**System**	**Encoding**	**LOCs**
Jerome Vouillon	Coq	De Bruijn	2700
Karl Crary & Robert Harper	Twelf	HOAS	5000
S. B.	Isabelle/HOL	De Bruijn	2500

Other (partial) solutions by Xavier Leroy, Aaron Stump, Christian Urban, ...

Syntax of System $F_{<:}$

Types

datatype *type* =
  TVar nat
| *Top*
| *Fun type type* (**infixr** $\rightarrow$ *200*)
| *TyAll type type* ((*3*$\forall$<:-./ -) [*0*, *10*] *10*)

Terms

datatype *trm* =
  Var nat
| *Abs type trm* ((*3*λ:-./ -) [*0*, *10*] *10*)
| *TAbs type trm* ((*3*λ<:-./ -) [*0*, *10*] *10*)
| *App trm trm* (**infixl** $\cdot$ *200*)
| *TApp trm type* (**infixl** $\cdot_\tau$ *200*)

Example

"Pen and paper" version (with names)

$$\lambda A{<:}\mathit{Top}.\ \lambda B{<:}\mathit{Top}.\ \lambda C{<:}\mathit{Top}.\ \lambda f{:}A \to B \to C.$$
$$\lambda g{:}(\forall\, D{<:}\mathit{Top}.\ D \to D).\ \lambda x{:}A.\ \lambda y{:}B.$$
$$f \cdot (g \cdot_\tau A \cdot x) \cdot (g \cdot_\tau B \cdot y)$$

De Bruijn version

$$\lambda{<:}\mathit{Top}.\ \lambda{<:}\mathit{Top}.\ \lambda{<:}\mathit{Top}.\ \lambda{:}\mathit{TVar}\ 2 \to \mathit{TVar}\ 1 \to \mathit{TVar}\ 0.$$
$$\lambda{:}(\forall {<:}\mathit{Top}.\ \mathit{TVar}\ 0 \to \mathit{TVar}\ 0).\ \lambda{:}\mathit{TVar}\ 4.\ \lambda{:}\mathit{TVar}\ 4.$$
$$\mathit{Var}\ 3 \cdot (\mathit{Var}\ 2 \cdot_\tau \mathit{TVar}\ 6 \cdot \mathit{Var}\ 1) \cdot (\mathit{Var}\ 2 \cdot_\tau \mathit{TVar}\ 5 \cdot \mathit{Var}\ 0)$$

Notation

$\Gamma \vdash S <: T$	Type S is subtye of T in context Γ
$\Gamma \vdash t : T$	Term t has type T in context Γ
$\Gamma \vdash_{wf}$	Context Γ is well-formed
$\Gamma \vdash_{wf} T$	Type T is well-formed in context Γ

Contexts

List of bindings for term and type variables

datatype $binding = VarB\ type \mid TVarB\ type$

types $env = binding\ list$

- Variable with index i corresponds i-th element of list (denoted by $\Gamma\langle i \rangle$)
- Types in Γ may refer to type variables "further to the right"
- New elements are appended to the left using $b\ \#\ \Gamma$
- Concatenation of contexts using $\Delta\ @\ \Gamma$

Lifting and Substitution

$\uparrow_\tau\ n\ k\ T$	Increment free variables $\geq k$ in type T by n
$\uparrow\ n\ k\ t$	Increment free variables $\geq k$ in term t by n
$\uparrow_e\ n\ k\ \Gamma$	Increment free variables $\geq k$ in environment Γ by n
$T[k \mapsto_\tau S]_\tau$	Substitute type S for type variable with index k in type T
$t[k \mapsto_\tau S]$	Substitute type S for type variable with index k in term t
$t[k \mapsto s]$	Substitute term s for term variable with index k in term t
$\Gamma[k \mapsto_\tau T]_e$	Substitute type T for type variable with index k in environment Γ

Some equations

$$\uparrow\ n\ k\ (Var\ i) = (if\ i < k\ then\ Var\ i\ else\ Var\ (i + n))$$
$$\uparrow\ n\ k\ (\lambda{:}T.\ t) = (\lambda{:}\uparrow_\tau\ n\ k\ T.\ \uparrow\ n\ (k + 1)\ t)$$

$$(Var\ i)[k \mapsto s] = (if\ k < i\ then\ Var\ (i - 1)\ else\ if\ i = k\ then \uparrow k\ 0\ s\ else\ Var\ i)$$
$$(\lambda{:}T.\ t)[k \mapsto s] = (\lambda{:}T[k \mapsto_\tau Top]_\tau.\ t[k{+}1 \mapsto s])$$

$$[][k \mapsto_\tau T]_e = []$$
$$(B\ \#\ \Gamma)[k \mapsto_\tau T]_e = mapB\ (\lambda U.\ U[k + \|\Gamma\| \mapsto_\tau T]_\tau)\ B\ \#\ \Gamma[k \mapsto_\tau T]_e$$

Well-formedness of Types and Contexts

Intuition:

- A type is well-formed in a context, if all its free variables appear in the context.
- A context is well-formed, if all types only refer to type variables "further to the right"

$$\frac{\Gamma\langle i\rangle = \lfloor TVarB\ T\rfloor}{\Gamma \vdash_{wf} TVar\ i} \qquad \Gamma \vdash_{wf} Top$$

$$\frac{\Gamma \vdash_{wf} T \quad \Gamma \vdash_{wf} U}{\Gamma \vdash_{wf} T \to U} \qquad \frac{\Gamma \vdash_{wf} T \quad TVarB\ T\ \#\ \Gamma \vdash_{wf} U}{\Gamma \vdash_{wf} (\forall <: T.\ U)}$$

$$[] \vdash_{wf} \qquad \frac{\Gamma \vdash_{wf} type\text{-}ofB\ B \quad \Gamma \vdash_{wf}}{B\ \#\ \Gamma \vdash_{wf}}$$

Important property:

All terms and contexts involved in (sub)typing judgements are well-formed, i.e.

if $\Gamma \vdash S <: T$, then $\Gamma \vdash_{wf}$, $\Gamma \vdash_{wf} S$, $\Gamma \vdash_{wf} T$

if $\Gamma \vdash t : T$, then $\Gamma \vdash_{wf}$, $\Gamma \vdash_{wf} T$

Subtyping Relation

"Pen and paper" version (with names)

$$\frac{X{<:}U \in \Gamma \quad \Gamma \vdash U <: T}{\Gamma \vdash X <: T}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma, X{<:}T_1 \vdash S_2 <: T_2}{\Gamma \vdash (\forall X{<:}S_1.\ S_2) <: (\forall X{<:}T_1.\ T_2)}$$

De Bruijn version

$$\frac{\Gamma\langle i\rangle = \lfloor TVarB\ U \rfloor \quad \Gamma \vdash \uparrow_\tau (Suc\ i)\ 0\ U <: T}{\Gamma \vdash TVar\ i <: T}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \quad TVarB\ T_1 \# \Gamma \vdash S_2 <: T_2}{\Gamma \vdash (\forall <: S_1.\ S_2) <: (\forall <: T_1.\ T_2)}$$

Subtyping Relation

$$\frac{\Gamma \vdash_{wf} \quad \Gamma \vdash_{wf} S}{\Gamma \vdash S <: Top}$$

$$\frac{\Gamma \vdash_{wf} \quad \Gamma \vdash_{wf} TVar\ i}{\Gamma \vdash TVar\ i <: TVar\ i}$$

$$\frac{\Gamma\langle i\rangle = \lfloor TVarB\ U \rfloor \quad \Gamma \vdash \uparrow_\tau (Suc\ i)\ 0\ U <: T}{\Gamma \vdash TVar\ i <: T}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \quad \Gamma \vdash S_2 <: T_2}{\Gamma \vdash S_1 \to S_2 <: T_1 \to T_2}$$

$$\frac{\Gamma \vdash T_1 <: S_1 \quad TVarB\ T_1 \# \Gamma \vdash S_2 <: T_2}{\Gamma \vdash (\forall <: S_1.\ S_2) <: (\forall <: T_1.\ T_2)}$$

Typing relation

"Pen and paper" version (with names)

$$\frac{\Gamma,\ X{<:}T_1 \vdash t_2 : T_2}{\Gamma \vdash (\lambda X{<:}T_1.\ t_2) : (\forall\, X{<:}T_1.\ T_2)}$$

$$\frac{\Gamma \vdash t_1 : (\forall\, X{<:}T_{11}.\ T_{12}) \quad \Gamma \vdash T_2 <: T_{11}}{\Gamma \vdash t_1 \cdot_\tau T_2 : T_{12}[X \mapsto_\tau T_2]_\tau}$$

De Bruijn version

$$\frac{TVarB\ T_1\ \#\ \Gamma \vdash t_2 : T_2}{\Gamma \vdash (\lambda{<:}T_1.\ t_2) : (\forall{<:}T_1.\ T_2)}$$

$$\frac{\Gamma \vdash t_1 : (\forall{<:}T_{11}.\ T_{12}) \quad \Gamma \vdash T_2 <: T_{11}}{\Gamma \vdash t_1 \cdot_\tau T_2 : T_{12}[0 \mapsto_\tau T_2]_\tau}$$

Typing relation

$$\frac{\Gamma \vdash_{wf} \quad \Gamma\langle i\rangle = \lfloor VarB\ U \rfloor \quad T = \uparrow_\tau (Suc\ i)\ 0\ U}{\Gamma \vdash Var\ i : T}$$

$$\frac{VarB\ T_1\ \#\ \Gamma \vdash t_2 : T_2}{\Gamma \vdash (\lambda{:}T_1.\ t_2) : T_1 \to T_2[0 \mapsto_\tau Top]_\tau}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1 \cdot t_2 : T_{12}}$$

$$\frac{TVarB\ T_1\ \#\ \Gamma \vdash t_2 : T_2}{\Gamma \vdash (\lambda{<:}T_1.\ t_2) : (\forall{<:}T_1.\ T_2)}$$

$$\frac{\Gamma \vdash t_1 : (\forall{<:}T_{11}.\ T_{12}) \quad \Gamma \vdash T_2 <: T_{11}}{\Gamma \vdash t_1 \cdot_\tau T_2 : T_{12}[0 \mapsto_\tau T_2]_\tau}$$

$$\frac{\Gamma \vdash t : S \quad \Gamma \vdash S <: T}{\Gamma \vdash t : T}$$

Evaluation relation

Evaluation can take place . . .

- . . . at the root of a term
- . . . inside subterms (e.g. in operator or operand of an application)

Two ways of modelling evaluation in subterms

- Add extra congruence rules to definition of evaluation relation
- Introduce evaluation contexts as separate concept
  - Context $\approx$ term with "hole", i.e. function $term \Rightarrow term$
  - Expected to lead to more scalable formalizations
  - Not directly executable (needs computational content of "decomposition theorem")

Evaluation relation – using congruence rules

Values

$$(\lambda{:}T.\ t) \in value$$
$$(\lambda{<:}T.\ t) \in value$$

Evaluation rules

$$\frac{v_2 \in value}{(\lambda{:}T_{11}.\ t_{12}) \cdot v_2 \longmapsto t_{12}[0 \mapsto v_2]}$$

$$(\lambda{<:}T_{11}.\ t_{12}) \cdot_\tau T_2 \longmapsto t_{12}[0 \mapsto_\tau T_2]$$

Congruence rules

$$\frac{t \longmapsto t'}{t \cdot u \longmapsto t' \cdot u} \qquad \frac{v \in value \quad t \longmapsto t'}{v \cdot t \longmapsto v \cdot t'}$$

$$\frac{t \longmapsto t'}{t \cdot_\tau T \longmapsto t' \cdot_\tau T}$$

Evaluation relation – using contexts

Evaluation contexts

$$(\lambda t.\ t) \in ctxt$$

$$\frac{E \in ctxt}{(\lambda t.\ E\ t \cdot u) \in ctxt} \qquad \frac{v \in value \quad E \in ctxt}{(\lambda t.\ v \cdot E\ t) \in ctxt}$$

$$\frac{E \in ctxt}{(\lambda t.\ E\ t \cdot_\tau T) \in ctxt}$$

Evaluation rules

$$\frac{t \longmapsto t' \quad E \in ctxt}{E\ t \longmapsto E\ t'}$$

$$\frac{v_2 \in value}{(\lambda{:}T_{11}.\ t_{12}) \cdot v_2 \longmapsto t_{12}[0 \mapsto v_2]}$$

$$(\lambda{<:}T_{11}.\ t_{12}) \cdot_\tau T_2 \longmapsto t_{12}[0 \mapsto_\tau T_2]$$

Important properties

Weakening

$$\Gamma \vdash t : T \Longrightarrow \Delta \ @ \ \Gamma \vdash_{wf} \Longrightarrow \Delta \ @ \ \Gamma \vdash \uparrow \|\Delta\| \ 0 \ t : \uparrow_\tau \|\Delta\| \ 0 \ T$$

Substitution lemma

$$\Delta \ @ \ VarB \ U \ \# \ \Gamma \vdash t : T \Longrightarrow \Gamma \vdash u : U \Longrightarrow$$
$$\Delta[0 \mapsto_\tau Top]_e \ @ \ \Gamma \vdash t[\|\Delta\| \mapsto u] : T[\|\Delta\| \mapsto_\tau Top]_\tau$$
$$\Delta \ @ \ TVarB \ Q \ \# \ \Gamma \vdash S <: T \Longrightarrow \Gamma \vdash P <: Q \Longrightarrow$$
$$\Delta[0 \mapsto_\tau P]_e \ @ \ \Gamma \vdash S[\|\Delta\| \mapsto_\tau P]_\tau <: T[\|\Delta\| \mapsto_\tau P]_\tau$$

Type safety

$t \longmapsto t' \Longrightarrow \Gamma \vdash t : T \Longrightarrow \Gamma \vdash t' : T$ — Preservation / Subject Reduction

$[] \vdash t : T \Longrightarrow t \in value \vee (\exists t'.\ t \longmapsto t')$ — Progress

Properties of evaluation contexts

Decomposition

$$[] \vdash t : T \Longrightarrow t \in value \vee (\exists\, E\ t_0\ t_0'.\ E \in ctxt \wedge t = E\ t_0 \wedge t_0 \longmapsto t_0')$$

Typing

$$\Gamma \vdash E\ t : T \Longrightarrow E \in ctxt \Longrightarrow (\bigwedge T_0.\ \Gamma \vdash t : T_0 \Longrightarrow \Gamma \vdash t' : T_0) \Longrightarrow \Gamma \vdash E\ t' : T$$

Tricky proofs by induction

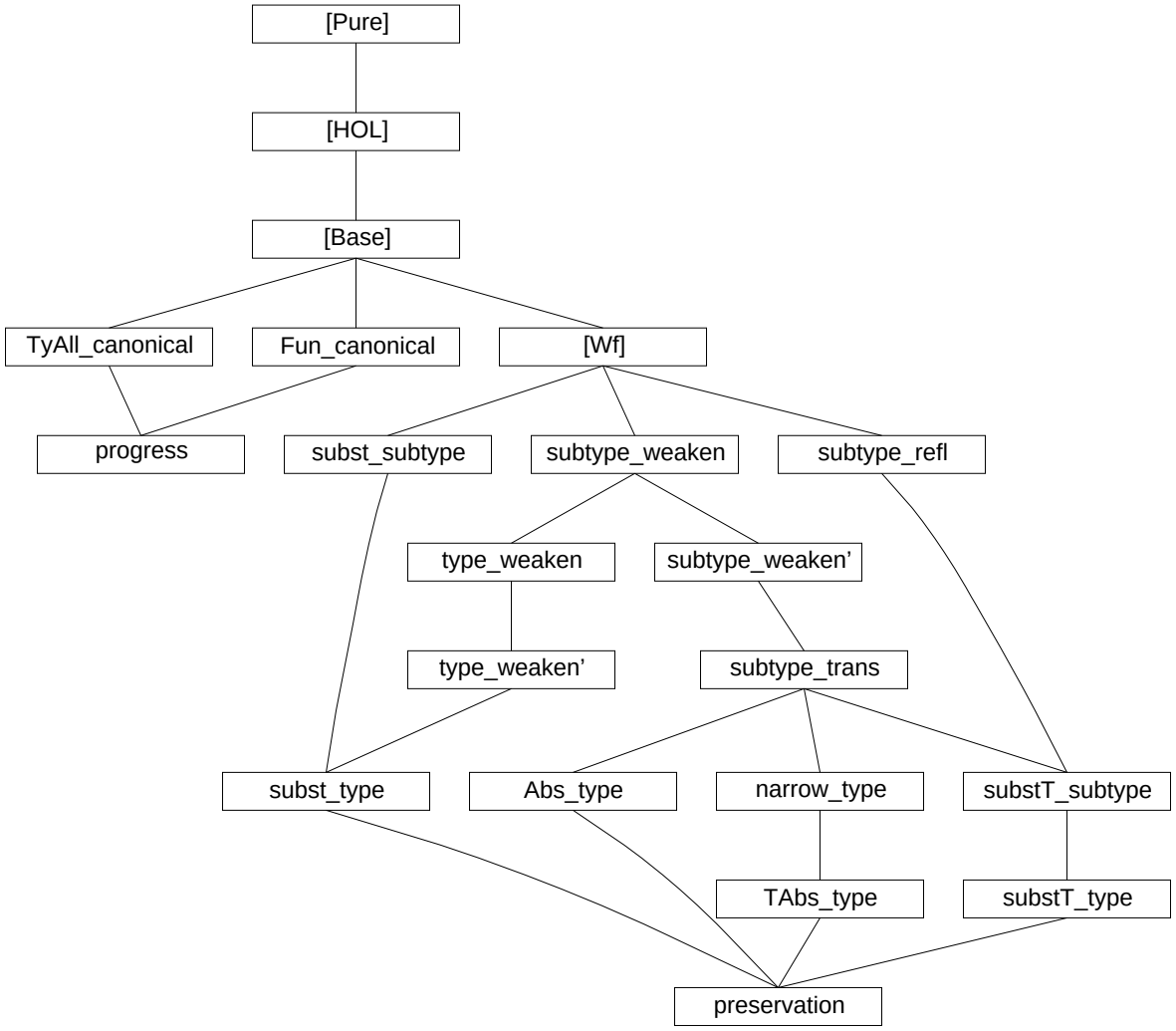
Simultaneous goals

lemma *subtype-trans*:
  $\Gamma \vdash S <: Q \Longrightarrow \Gamma \vdash Q <: T \Longrightarrow \Gamma \vdash S <: T$
  $\Delta \mathbin{@} TVarB\ Q \mathbin{\#} \Gamma \vdash M <: N \Longrightarrow \Gamma \vdash P <: Q \Longrightarrow$
    $\Delta \mathbin{@} TVarB\ P \mathbin{\#} \Gamma \vdash M <: N$
  using *wf-measure-size*
proof (*induct Q fixing*: $\Gamma\ S\ T\ \Delta\ P\ M\ N$ *rule*: *wf-induct-rule*)
  ...

Expressions of a certain form (local definition)

lemma *substT-type*:
  assumes *H*: $\Delta \mathbin{@} TVarB\ Q \mathbin{\#} \Gamma \vdash t : T$
  shows $\Gamma \vdash P <: Q \Longrightarrow$
    $\Delta[0 \mapsto_\tau P]_e \mathbin{@} \Gamma \vdash t[\|\Delta\| \mapsto_\tau P] : T[\|\Delta\| \mapsto_\tau P]_\tau$ **using** *H*
proof (*induct* $\Gamma' \equiv \Delta \mathbin{@} TVarB\ Q \mathbin{\#} \Gamma\ t\ T$ *fixing*: Δ)
  ...

Theorem dependencies



Executability

Idea

- Translate PROLOG-style inductive definitions to functional program (e.g. in ML) yielding sequence of possible outputs for a given input [B., Nipkow, TYPES 2000]
- Implementation using list monad

```
:-> : 'a Seq.seq -> ('a -> 'b Seq.seq) -> 'b Seq.seq
fun s :-> f = Seq.flat (Seq.map f s);
```

```
fun eval inp =
  Seq.single inp :->
    (fn (App (Abs (T_1_1, t_1_2), v_2)) =>
      value (v_2) :-> (fn () => Seq.single (subst t_1_2 0 v_2))
      | _ => Seq.empty) ++
  Seq.single inp :->
    (fn (TApp (TAbs (T_1_1, t_1_2), T_2)) => Seq.single (substT t_1_2 0 T_2)
      | _ => Seq.empty) ++
  Seq.single inp :->
    (fn (App (t, u)) => eval (t) :-> (fn (t') => Seq.single (App (t', u)))
      | _ => Seq.empty) ++ ...
```

Records

- Records are modelled as association lists mapping field names to terms
  types $rcd = (name \times trm)\ list$
- Record types are modelled as association lists mapping field names to types
  types $rcdT = (name \times type)\ list$
- LET expressions can be treated like nested abstractions

$$LET\ \{l_1 = x_1 : T_1, \ldots, l_n = x_n : T_n\} = \{l_1 = v_1, \ldots, l_n = v_n\}\ IN\ t$$
$$\approx$$
$$(\lambda x_1 : T_1, \ldots, x_n : T_n.\ t) \cdot v_1 \cdot \cdots \cdot v_n$$

- Pattern typing judgement $\vdash p : T \Rightarrow \Delta$
  yields context assigning types to variables bound in pattern
- Pattern matching judgement $\vdash p \rhd t \Rightarrow ts$
  yields list of terms bound to variables in pattern

New constructors for records

Types

datatype $type =$

$\quad \ldots$

$\quad \mid RcdT\ (name \times type)\ list$

Patterns

datatype $pat = PVar\ type \mid PRcd\ (name \times pat)\ list$

Terms

datatype $trm =$

$\quad \ldots$

$\quad \mid Rcd\ (name \times trm)\ list$

$\quad \mid Proj\ trm\ name\ \ ((\text{-}..\text{-})\ [90,\ 91]\ 90)$

$\quad \mid LET\ pat\ trm\ trm\ ((LET\ (\text{-} =/\ \text{-})/\ IN\ (\text{-}))\ 10)$

Well-formedness and subtyping of record types

Well-formedness

$$\frac{unique\ fs \quad \forall\,(l,\ T){\in}set\ fs.\ \Gamma \vdash_{wf} T}{\Gamma \vdash_{wf} RcdT\ fs}$$

Subtyping

$$\frac{\Gamma \vdash_{wf} \quad \Gamma \vdash_{wf} RcdT\ fs \quad unique\ fs' \quad \forall\,(l,\ T){\in}set\ fs'.\ \exists\,(k,\ S){\in}set\ fs.\ k = l \wedge \Gamma \vdash S <: T}{\Gamma \vdash RcdT\ fs <: RcdT\ fs'}$$

Additional typing rules for records

$$\frac{\Gamma \vdash t_1 : T_1 \quad \vdash p : T_1 \Rightarrow \Delta \quad \Delta \mathbin{@} \Gamma \vdash t_2 : T_2}{\Gamma \vdash (LET\ p = t_1\ IN\ t_2) : \downarrow_\tau \|\Delta\|\ 0\ T_2}$$

$$\frac{\Gamma \vdash fs\ [:]\ fTs}{\Gamma \vdash Rcd\ fs : RcdT\ fTs} \qquad \frac{\Gamma \vdash t : RcdT\ fTs \quad fTs\langle l\rangle_? = \lfloor T \rfloor}{\Gamma \vdash t..l : T}$$

$$\frac{\Gamma \vdash_{wf}}{\Gamma \vdash [\,]\ [:]\ [\,]} \qquad \frac{\Gamma \vdash t : T \quad \Gamma \vdash fs\ [:]\ fTs \quad fs\langle l\rangle_? = \bot}{\Gamma \vdash (l,\ t)\ \#\ fs\ [:]\ (l,\ T)\ \#\ fTs}$$

Pattern typing

$$\vdash PVar\ T : T \Rightarrow [VarB\ T] \qquad \frac{\vdash fps\ [:]\ fTs \Rightarrow \Delta}{\vdash PRcd\ fps : RcdT\ fTs \Rightarrow \Delta}$$

$$\vdash [\,]\ [:]\ [\,] \Rightarrow [\,] \qquad \frac{\vdash p : T \Rightarrow \Delta_1 \quad \vdash fps\ [:]\ fTs \Rightarrow \Delta_2 \quad fps\langle l\rangle_? = \bot}{\vdash (l,\ p)\ \#\ fps\ [:]\ (l,\ T)\ \#\ fTs \Rightarrow \uparrow_e \|\Delta_1\|\ 0\ \Delta_2 \mathbin{@} \Delta_1}$$

Additional evaluation rules for records

$$\frac{v \in value \quad \vdash p \rhd v \Rightarrow ts}{(LET\ p = v\ IN\ t) \longmapsto t[0 \mapsto_s ts]} \qquad \frac{fs\langle l\rangle_? = \lfloor v \rfloor \quad v \in value}{Rcd\ fs..l \longmapsto v}$$

Contexts

$$\frac{E \in ctxt}{(\lambda t.\ E\ t..l) \in ctxt} \qquad \frac{E \in rctxt}{(\lambda t.\ Rcd\ (E\ t)) \in ctxt} \qquad \frac{E \in ctxt}{(\lambda t.\ LET\ p = E\ t\ IN\ u) \in ctxt}$$

$$\frac{E \in ctxt}{(\lambda t.\ (l,\ E\ t)\ \#\ fs) \in rctxt} \qquad \frac{v \in value \quad E \in rctxt}{(\lambda t.\ (l,\ v)\ \#\ E\ t) \in rctxt}$$

Matching

$$\vdash PVar\ T \rhd t \Rightarrow [t] \qquad \frac{\vdash fps\ [\rhd]\ fs \Rightarrow ts}{\vdash PRcd\ fps \rhd Rcd\ fs \Rightarrow ts}$$

$$\vdash []\ [\rhd]\ fs \Rightarrow [] \qquad \frac{fs\langle l\rangle_? = \lfloor t \rfloor \quad \vdash p \rhd t \Rightarrow ts \quad \vdash fps\ [\rhd]\ fs \Rightarrow us}{\vdash (l,\ p)\ \#\ fps\ [\rhd]\ fs \Rightarrow ts\ @\ us}$$

Additional theorems for records

Matched patterns preserve types

$$\vdash p : T_1 \Rightarrow \Delta \Longrightarrow \Gamma_2 \vdash t_1 : T_1 \Longrightarrow \Gamma_1 \mathbin{@} \Delta \mathbin{@} \Gamma_2 \vdash t_2 : T_2 \Longrightarrow \vdash p \rhd t_1 \Rightarrow ts \Longrightarrow \downarrow_e \|\Delta\|\ 0\ \Gamma_1 \mathbin{@} \Gamma_2 \vdash t_2[\|\Gamma_1\| \mapsto_s ts] : \downarrow_\tau \|\Delta\|\ \|\Gamma_1\|\ T_2$$

$$\vdash fps\ [:]\ fTs \Rightarrow \Delta \Longrightarrow \Gamma_2 \vdash fs\ [:]\ fTs \Longrightarrow \Gamma_1 \mathbin{@} \Delta \mathbin{@} \Gamma_2 \vdash t_2 : T_2 \Longrightarrow \vdash fps\ [\rhd]\ fs \Rightarrow ts \Longrightarrow \downarrow_e \|\Delta\|\ 0\ \Gamma_1 \mathbin{@} \Gamma_2 \vdash t_2[\|\Gamma_1\| \mapsto_s ts] : \downarrow_\tau \|\Delta\|\ \|\Gamma_1\|\ T_2$$

Well-typed pattern matching is defined

$$\vdash p : T \Rightarrow \Delta \Longrightarrow [] \vdash t : T \Longrightarrow t \in value \Longrightarrow \exists\, ts.\ \vdash p \rhd t \Rightarrow ts$$

$$\vdash fps\ [:]\ fTs \Rightarrow \Delta \Longrightarrow [] \vdash fs\ [:]\ fTs \Longrightarrow \forall\,(l,\ t) \in set\ fs.\ t \in value \Longrightarrow \exists\, us.\ \vdash fps\ [\rhd]\ fs \Rightarrow us$$

Conclusion

- Complete and executable formalization of System $F_{<:}$
- Formalized evaluation relation using both evaluation contexts and congruence rules (including proof of equivalence)
- Use of evaluation contexts did not shorten the formalization (but rendered the specification non-executable)
- Use of de Bruijn indices not too much of a heavy burden (technical lemmas from simply-typed λ-calculus could be reused)
- Extension with records increased size of formalization by about factor 2

Future work

- Formalize more complex properties of System $F_{<:}$, e.g. strong normalization (see Thorsten Altenkirch's Ph.D.)
- Use nominal syntax to formalize bound variables (work in progress, together with Christian Urban)

Further information

http://www.in.tum.de/~berghofe/papers/Poplmark/