Reasoning on a semantics of higher-order states using guarded recursion and forcing

Guilhem Jaber and Nicolas Tabareau

Ecole des Mines de Nantes
LINA - Ascola

Réalisabilité à Chambéry
16th of June 2011

Introduction

- How to define a syntactic model of λ-calculus with :
  - higher-order states with pointers,
  - impredicative polymorphism,
  - and recursive types ?

  $\leadsto$ Works from the last 10 years (Pitts, Appel, Ahmed, Dreyer, Birkedal, ...)

- How to define a logic to reason about this semantics ?
  $\leadsto$ Extension of LSLR and LADR.

- How to prove its coherence ?
  $\leadsto$ Using forcing !

The Language

$$
\begin{aligned}
\tau, \sigma &\stackrel{def}{=} \mathrm{Nat} \mid \tau \to \sigma \mid \forall \alpha.\tau \mid \mu\alpha.\tau \mid \mathrm{ref}\,\tau \\
v &\stackrel{def}{=} \hat{n} \mid l \mid \lambda x.M \mid \ldots \\
M, N &\stackrel{def}{=} v \mid x \mid \tau \mid MN \mid \mathrm{ref}\,M \mid\, !M \mid M := N \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
((\lambda x.M)v, h) &\mapsto (M\{v/x\}, h) \\
(!l, h) &\mapsto (v, h) \text{ when } h(l) = v \\
(\mathrm{ref}\, v, h) &\mapsto (l, h \bullet [l \mapsto v]) \text{ with } l \notin dom(h) \\
(l := v, h) &\mapsto ((), h[l \mapsto v]) \text{ when } l \in dom(h)
\end{aligned}
$$

$$
\frac{(M_1, h_1) \mapsto (M_1, h_2)}{(K[M_1], h_1) \mapsto (K[M_2], h_2)}
$$

A first glance of our realizability semantics

- Associate to a type τ a set of values $[\![\tau]\!]$:

$$[\![\mathrm{Nat}]\!] \stackrel{def}{=} \{\hat{n} \mid n \in \mathbb{N}\}$$

$$[\![\sigma \to \tau]\!] \stackrel{def}{=} \{f \mid \forall u \in [\![\sigma]\!], fu \in \mathcal{E}\,[\![\tau]\!]\}$$

- Lifting the semantics from values to terms using *biorthogonality* :

$$\mathcal{K}\,[\![\tau]\!] = \{K \mid \forall v \in [\![\tau]\!], K[v] \Uparrow\}$$

$$\mathcal{E}\,[\![\tau]\!] = \{M \mid \forall K \in \mathcal{K}\,[\![\tau]\!], K[M] \Uparrow\}$$

- In this talk : predicate semantics
  $\rightsquigarrow$ But extension to relational semantics is straightforward using logical relations.

Higher order states

- What should $[\![\mathrm{ref}\,\tau]\!]$ be ?
  $\rightsquigarrow$ Locations which contain values in $[\![\tau]\!]$
  $\rightsquigarrow$ But this depends on the heap !
  $\rightsquigarrow$ Need to abstract the heap : notion of worlds.
- Worlds should map location to set of values :
  $\rightsquigarrow$ $\mathrm{World} \stackrel{\mathit{def}}{=} \mathrm{Loc} \rightarrow_{\mathit{fin}} \mathcal{P}(\mathrm{Val})$.
- What about locations which can contains themselves locations ?
  $\rightsquigarrow$ It depends on worlds too !

Mutual recursive equations

$$\text{World} = \text{Loc} \rightarrow_{fin} \text{SemType}$$
$$\text{SemType} = \text{World} \rightarrow \mathcal{P}(\text{Val})$$

- How to solve it ?
  $\rightsquigarrow$ Impose a stratification :

$$\text{World}_n = \text{Loc} \rightarrow_{fin} \text{SemType}_{n-1}$$
$$\text{SemType}_n = \text{World}_n \rightarrow \mathcal{P}(\text{Val})$$

- So need to define $[\![\tau]\!]_n$.
- Can this stratification be completely artificial ?
  $\rightsquigarrow$ No, because of impredicative polymorphism.

Step-indexing in a nutshell

- Need to give a meaning of n in $[\![\tau]\!]_n$.
- An idea : $[\![\tau]\!]_n$ is an approximation of $[\![\tau]\!]$.
- $t \in \mathcal{E}\,[\![\tau]\!]_n$ is true for n-th steps of reductions of t.
- Once t has been reduce n-th times, $t \in \mathcal{E}\,[\![\tau]\!]_n$ gives no information anymore.

Changing the notion of observation

- $[\![\sigma \to \tau]\!]_n = \{f \mid \forall k \leq n.\forall u \in [\![\sigma]\!]_k .fu \in \mathcal{E}[\![\tau]\!]_k\}$
$\rightsquigarrow$ So we get monotonicity !

- Changing the notion of observation in $\mathcal{K}[\![\tau]\!]$ and $\mathcal{E}[\![\tau]\!]$
$\rightsquigarrow$ From $M \Uparrow$ (divergence) to $M \Uparrow_n$: M can be reduced at least n steps.
$\rightsquigarrow$ $\mathcal{K}[\![\tau]\!]_n = \{K \mid \forall i \leq n.\forall v \in [\![\tau]\!], K[v] \Uparrow_n\}$.
$\rightsquigarrow$ $\mathcal{E}[\![\tau]\!]_n = \{M \mid \forall i \leq n.\forall K \in \mathcal{K}[\![\tau]\!], K[M] \Uparrow_n\}$.

Semantics for references

- Dereferencing a location takes at least one step.
  $\leadsto$ This explains $\mathrm{World}_n = \mathrm{Loc} \rightarrow_{fin} \mathrm{SemType}_{n-1}$
- $[\![\mathrm{ref}\,\tau]\!]_n\, w \stackrel{def}{=} \{l \mid l \in \mathrm{dom}(w)$ and $w(l)(n) = [\![\tau]\!]_n\}$
- $h :_k w \stackrel{def}{=} \mathrm{dom}(h) = \mathrm{dom}(w)$ and $\forall i < k. \forall l \in \mathrm{dom}(w) h(l) \in w(l)(i)(w)$

Some results

Theorem (Operational correctness)

If $t \in \mathcal{E}\,[\![\mathrm{Nat}]\!]$ *then* $t \Uparrow$ *or there exists* $n \in \mathbb{N}$ *such that* $t \mapsto^{*} \hat{n}$

Theorem (Adequacy)

if $\vdash t : \tau$ *then for all* $k \in \mathbb{N}, t \in [\![\tau]\!]_k$.

Reasoning about the semantics

- Build a logic like LCF or Plotkin-Abadi for parametric polymorphism.
- Want to reason on properties like $t \in \mathcal{E}\,[\![\tau]\!]$
- ground elements of our logic : λ-terms.
- Need higher-order logic to define our semantics.
  - $\rightsquigarrow$ Kinds : $\mathrm{Term}, \mathrm{Nat}, \mathrm{Prop}, T \to U$.
  - $\rightsquigarrow$ Constructions like $(x : T).P$ and $a \in P$.
  - $\rightsquigarrow$ Typing judgments :

$$\Gamma \vdash_{\mathcal{P}} P : T$$

  - $\rightsquigarrow$ Logical judgments :

$$\Gamma ; \mathcal{C} \vdash_{\mathcal{P}} \varphi$$

Abstracting step-indexes

- Step-indexing is ugly !
- Introduce a modal operator "later" : $\triangleright$
- Useful to state that a property has to be true *in the future*.

$$\triangleright\text{-Mono}\ \frac{\Gamma; \mathcal{C}_1, \mathcal{C}_2 \vdash_{\mathcal{P}} P}{\Gamma; \triangleright\mathcal{C}_1, \mathcal{C}_2 \vdash_{\mathcal{P}} \triangleright P} \qquad \text{Löb}\ \frac{\Gamma; \mathcal{C}, \triangleright P \vdash_{\mathcal{P}} P}{\Gamma; \mathcal{C} \vdash_{\mathcal{P}} P}$$

$$\frac{\Gamma; \mathcal{C} \vdash (M, h) \to (M', h) \qquad \Gamma; \mathcal{C} \vdash_{\mathcal{P}} \triangleright(M' \in \mathcal{E}\,[\![\tau]\!])}{\Gamma; \mathcal{C} \vdash_{\mathcal{P}} M \in \mathcal{E}\,[\![\tau]\!]}$$

Recursive kinds, what for ?

- Want to define kinds $\mu T.U$.
- So that step-indexing is also abstract in the definition of worlds :

$$\mathrm{World} \quad \stackrel{def}{=} \quad \mu W.(\mathrm{Loc} \to_{fin} W \to \mathrm{Prop}),$$

- Need to introduce a modality on kinds $\blacktriangleright T$ to guard recursion.

$$\frac{\Gamma \vdash_{\mathcal{P}} P : \blacktriangleright U\{\mu T.U/T\}}{\Gamma \vdash_{\mathcal{P}} P : \mu T.U}$$

$$\frac{\Gamma \vdash_{\mathcal{P}} P : \blacktriangleright \mathrm{Prop}}{\Gamma \vdash_{\mathcal{P}} \triangleright P : \mathrm{Prop}}$$

Forcing, what for ?

- A syntactic transformation of formulas : from P to $p \Vdash P$.
- Links between the truth of P and of $p \Vdash P$.
- Use to give meaning of new connectives
  $\rightsquigarrow$ $\triangleright$ and $\blacktriangleright$here.
  $\rightsquigarrow$ In Cohen's Forcing, the generic ultrafilter G.
- Forcing for higher-order logic :
  $\rightsquigarrow$ Need to define a translation of kinds too.

Forcing for higher order logic

- $\vdash_{\mathcal{P}}$ is the judgment in the forcing layer.
  $\rightsquigarrow$ We can in fact compose different forcing layers.
- Need to transform terms and kinds.
- A new kind for forcing conditions : $\mathcal{P}$.
- Translation of kinds needs to be stratified by forcing conditions :
  $\rightsquigarrow$ From T to $[T]_p$.

Translating formulas

Definition

$$
\begin{array}{lcl}
[t] & \stackrel{def}{=} & (p : \mathcal{P}).t \text{ for ground terms} \\
[x] & \stackrel{def}{=} & x \\
[P \Rightarrow Q] & \stackrel{def}{=} & (p : \mathcal{P}).\forall q \leq p.(q \Vdash P) \Rightarrow (q \Vdash Q) \\
[\forall x : T.P] & \stackrel{def}{=} & (p : \mathcal{P}).\forall q \leq p.\forall x : [T]_q.q \Vdash P \\
[(x).P] & \stackrel{def}{=} & (x).[P] \\
[Q \in P] & \stackrel{def}{=} & [Q] \in [P] \\
[\triangleright P] & \stackrel{def}{=} & (p : \mathcal{P}).\forall q < p.q \Vdash P
\end{array}
$$

Then $p \Vdash \varphi$ is defined as $p \in [\varphi]$

Translating typing judgements

$$\Gamma \vdash_{\mathcal{P};\bar{F}} P : T$$

will be translated in the underlying layer as

$$p : \mathcal{P}, [\Gamma]_p \vdash_{\bar{F}} [P] : [T]_p.$$

Translating kinds

- Want monotonicity for $[T]_p$
- Need implicit dependent kinds to get it.

Definition

$$
\begin{array}{lcl}
[\mathrm{Prop}]_p & \stackrel{def}{=} & \mathcal{P}_p \to \mathrm{Prop} \\
[\mathrm{Term}]_p & \stackrel{def}{=} & \mathrm{Term} \\
[T \to U]_p & \stackrel{def}{=} & \forall p' : \mathcal{P}_p . [T]_{p'} \to [U]_{p'} \\
[\blacktriangleright T]_0 & \stackrel{def}{=} & \top \\
[\blacktriangleright T]_{p+1} & \stackrel{def}{=} & [T]_p \\
[\mu T . U]_0 & \stackrel{def}{=} & \top \\
[\mu T . U]_{p+1} & \stackrel{def}{=} & [U[\mu . T / T]]_p
\end{array}
$$

Translating logical judgments

Theorem

$$\frac{\Gamma_1; \mathcal{C}_1 \vdash_{\mathcal{P}} \varphi_1 \qquad \dots \qquad \Gamma_n; \mathcal{C}_n \vdash_{\mathcal{P}} \varphi_n}{\Gamma; \mathcal{C} \vdash_{\mathcal{P}} \varphi}$$

will be valid if for all forcing condition p

$$\frac{[\Gamma_1]_p; p \Vdash \mathcal{C}_1 \vdash p \Vdash \varphi_1 \qquad \dots \qquad [\Gamma_n]_p; p \Vdash \mathcal{C}_n \vdash p \Vdash \varphi_n}{[\Gamma]_p; p \Vdash \mathcal{C} \vdash p \Vdash \varphi}$$

What about a direct model of our logic ?

- Use the restatement of forcing in terms of sheaves.
  $\rightsquigarrow$ Work in the category of sheaves over forcing conditions.
  $\rightsquigarrow$ Kripke-Joyal semantics gives a forcing relation.
- $\{p \mid p \Vdash P\}$ is a downward closed subset of forcing conditions.
  $\rightsquigarrow$ $\widetilde{P} = \max\{p \mid p \Vdash P\}$
- Sheaves are needed to represents how forcing modify terms (and not only propositions).
- Use recent work of Birkedal et al. about topos of trees for synthetic guarded domain theory.

The Topos of presheaves of tree

- A presheaf X over ω :
  $\rightsquigarrow$ A collection of sets $X(1), \ldots, X(n), \ldots$
  $\rightsquigarrow$ Restrictions maps $r_n : X(n+1) \to X(n)$
  $\rightsquigarrow$ Morphisms between two presheaves X and Y : family of maps $f_n : X(n) \to Y(n)$ commuting with restriction maps.
- It's a Topos :
  $\rightsquigarrow$ Subobject classifier : $\Omega(n) = \{0, \ldots, n\}$.
- A morphism $\triangleright : \Omega \to \Omega$
  $\rightsquigarrow$ $k \in \Omega(n) \mapsto min(k, n+1)$.
- $\blacktriangleright X(1) \stackrel{def}{=} \{*\}$ and $\blacktriangleright X(n+1) \stackrel{def}{=} X(n)$.

Using presheaves to build a model

- Interpret a kind T by a presheaf X
  $\rightsquigarrow$ Prop is interpreted by Ω.
  $\rightsquigarrow$ Ground kinds by constant presheafs.
  $\rightsquigarrow$ $[T]_p$ corresponds to $X(p)$.

- $[\mathrm{Prop}]_p$ do not require downward closure contrary to $\Omega(p)$
  $\rightsquigarrow$ Because our forcing translation of propositions impose monotonicity.

- The monotonicity condition required by $[T \rightarrow U]_p$ corresponds to the commutation of morphisms with restrictions maps.

Future Works

- Extend the logic with proof terms :
  $\rightsquigarrow$ Using Miquel's interpretation of forcing.
- Formalize smarter worlds like STS.
- Define other layers to reason more abstractly about the heap.
- Formalization in Coq :
  $\rightsquigarrow$ Using Parametric Higher Order Abstract Syntax.
- Proof of compiler correctness : realizers are assembly codes.