Functors are Type Refinement Systems

Paul-André Melliès[1]   Noam Zeilberger[2]

[1]CNRS, Université Paris Diderot, Sorbonne Paris Cité

[2]MSR-Inria Joint Centre

POPL 2015, Mumbai, India
15 January 2015

Complete the sequence

- number theory is about numbers
- group theory is about groups
- category theory is about categories
- type theory is about ____

Complete the sequence

- number theory is about numbers
- group theory is about groups
- category theory is about categories
- type theory is about types!

Complete the sequence

- number theory is about numbers
- group theory is about groups
- category theory is about categories
- type theory is about types!

But what on earth are types?

The intrinsic (“à la Church”) view

Sometimes, a type is like a linguist’s *part of speech*, in the sense that parts of speech (NP, VP, etc.) can be used to distinguish well-formed sentences like

the quick brown fox jumped over the lazy dog

from non-sentences such as

* *lazy over dog fox quick the brown jumped the*

Under this usage, every valid program expression carries a type, and “untyped” expressions are considered meaningless.

The extrinsic (“à la Curry”) view

On the other hand, in natural language sometimes one wants to consider sentences such as

colourless green ideas sleep furiously

or

the king of India in 2014 was bald

which although syntactically well-formed, fail to satisfy other more semantic criteria of validity. Similarly, in programming sometimes one wants to start with a relatively liberal programming language, but then use types to ensure that programs are in some sense well-behaved.

Two views of typing

John Reynolds referred to these as the *intrinsic* and the *extrinsic* views of typing in his book, *Theories of Programming Languages*.

(Also known as “types à la Church” and “types à la Curry”.)

TT through the lens of CT: the standard dogma

A type system induces a category of well-typed terms, e.g., any well-typed term

$$x_1 : A_1, \ldots, x_n : A_n \vdash e : B$$

of the simply-typed lambda calculus may be interpreted as a morphism

$$A_1 \times \cdots \times A_n \xrightarrow{e} B$$

in a cartesian-closed category [see Lambek and Scott 1986].

TT through the lens of CT: the standard dogma

The standard dogma favors the **intrinsic** view: any morphism

$$A \xrightarrow{f} B$$

of a category is intrinsically associated with a unique pair of types, namely $\mathrm{dom}(f) = A$ and $\mathrm{cod}(f) = B$.

A hiccup in the standard dogma

But what about, say, intersection types or subtyping?

$$\frac{\Gamma \vdash e : B \quad \Gamma \vdash e : C}{\Gamma \vdash e : B \cap C} \qquad \frac{\Gamma \vdash e : B \quad B \leq C}{\Gamma \vdash e : C}$$

In a category, it is *ungrammatical* for one morphism to lie between different pairs of objects.

$$* \quad \begin{array}{l} \Gamma \xrightarrow{e} B \\ \Gamma \xrightarrow{e} C \end{array}$$

What Reynolds originally observed is that an intrinsic semantics for such a type system must really interpret **typing derivations** rather than terms. But this leads to questions of coherence[1]...

[1] Do two derivations of the same typing judgment have the same meaning?

“The Meaning of Types” (Reynolds 2000)

In later work, Reynolds gave a very elegant proof of coherence for a language with subtyping. The proof begins by defining both an intrinsic semantics and an extrinsic semantics, and then connects them via a logical relations theorem and a “bracketing” theorem (with coherence as a corollary).

Although JCR did not work in the language of category theory (in the 2000 paper), these semantics can be seen as functors

$$[\![-]\!]_D : \text{Derivations} \to \text{Meanings}$$
$$[\![-]\!]_T : \text{Terms} \to \text{Meanings}$$

“The Meaning of Types” (Reynolds 2000)

But any “reasonable” type system also induces a functor

$$\begin{array}{c} \text{Derivations} \\ \big\downarrow \mathbf{U} \\ \text{Terms} \end{array}$$

and this is also implicit in Reynolds’ analysis (e.g., in the statement of the logical relations theorem and coherence).

Functors are type refinement systems

Our starting point is the idea that this can actually serve as a working *definition* of “type system” (or *refinement system*).

Definition

A **(type) refinement system** is a functor $\mathbf{U} : \mathcal{D} \to \mathcal{T}$.

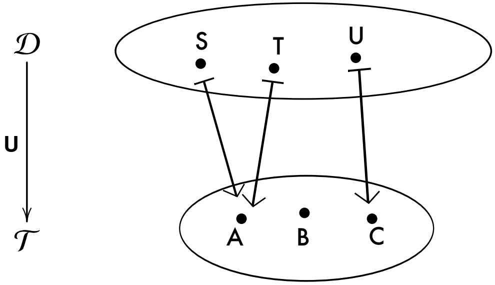
This is a “working” definition in the sense that

1. We have been working with it for a while.
2. It allows one to prove some interesting things about broad classes of type systems.
3. It is open to revision.

Reading a functor as a refinement system

Definition

We say that an object $S \in \mathcal{D}$ **refines**[2] an object $A \in \mathcal{T}$ if $\mathbf{U}(S) = A$.



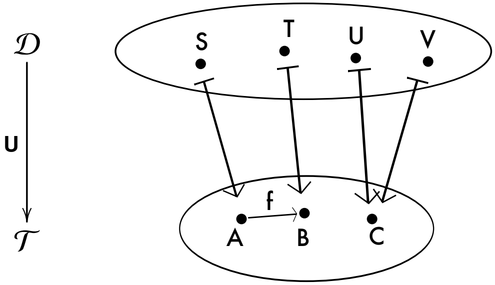
$S, T \sqsubset A$

$U \sqsubset C$

[2]With a tip of the hat to Tim Freeman & Frank Pfenning.

Definition

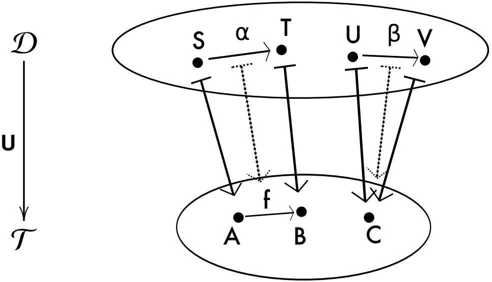
A **typing judgment** is a triple (S, f, T) such that S and T refine the domain and codomain of f, respectively, i.e., such that $f : A \to B$, $\mathbf{U}(S) = A$ and $\mathbf{U}(T) = B$, for some arbitrary A and B. In the special case of a triple with component $f = \mathrm{id}$, we also call this a **subtyping judgment**.



$$S \underset{f}{\Longrightarrow} T \qquad U \leq V$$

Definition

A **derivation** of a typing judgment (S, f, T) is a morphism $\alpha : S \to T$ in $\mathcal{D}$ such that $\mathbf{U}(\alpha) = f$.



$$S \overset{\alpha}{\underset{f}{\Longrightarrow}} T \qquad U \overset{\beta}{\leq} V$$

A convention for typing rules

We say that a *typing rule*

$$\frac{S_1 \underset{f_1}{\Longrightarrow} T_1 \quad \cdots \quad S_n \underset{f_n}{\Longrightarrow} T_n}{S \underset{f}{\Longrightarrow} T}$$

is valid (for $\mathbf{U} : \mathcal{D} \to \mathcal{T}$) if given derivations of the premises, we can construct a derivation of the conclusion**...**

Proposition

The following typing rules are valid for any refinement system:

$$\frac{S \underset{f}{\Longrightarrow} T \quad T \underset{g}{\Longrightarrow} U}{S \underset{f;g}{\Longrightarrow} U}\ ; \qquad \frac{}{S \underset{\mathrm{id}}{\Longrightarrow} S}\ \mathrm{id}$$

Proposition

Subtyping is reflexive and transitive, and admits rules of covariant and contravariant subsumption:

$$\frac{}{S \leq S} \qquad \frac{S \leq T \quad T \leq U}{S \leq U} \qquad \frac{S \underset{f}{\Longrightarrow} T \quad T \leq U}{S \underset{f}{\Longrightarrow} U} \qquad \frac{S \leq T \quad T \underset{g}{\Longrightarrow} U}{S \underset{g}{\Longrightarrow} U}$$

Reading Grothendieck in translation

Definition

A **pullback of** T **along** f is a refinement type $f^* T$

$$\frac{f : A \to B \quad T \sqsubset B}{f^* T \sqsubset A}$$

equipped with a pair of typing rules

$$\frac{}{f^* T \underset{f}{\Longrightarrow} T}\ Lf^* \qquad \frac{S \underset{g;f}{\Longrightarrow} T}{S \underset{g}{\Longrightarrow} f^* T}\ Rf^*$$

satisfying equations

$$\frac{\dfrac{\overset{\beta}{S \underset{g;f}{\Longrightarrow} T}}{S \underset{g}{\Longrightarrow} f^* T}\ Rf^* \quad \dfrac{}{f^* T \underset{f}{\Longrightarrow} T}\ Lf^*}{S \underset{g;f}{\Longrightarrow} T}\ ; \quad = \quad \overset{\beta}{S \underset{g;f}{\Longrightarrow} T} \qquad \overset{\eta}{S \underset{g}{\Longrightarrow} f^* T} \quad = \quad \dfrac{\dfrac{\overset{\eta}{S \underset{g}{\Longrightarrow} f^* T} \quad \dfrac{}{f^* T \underset{f}{\Longrightarrow} T}\ Lf^*}{S \underset{g;f}{\Longrightarrow} T}\ ;}{S \underset{g}{\Longrightarrow} f^* T}\ Rf^*$$

Definition

A **pushforward of** S **along** f is a refinement type $f\,S$

$$\frac{S \sqsubset A \quad f : A \to B}{f\,S \sqsubset B}$$

equipped with a pair of typing rules

$$\frac{S \underset{f;g}{\Longrightarrow} T}{f\,S \underset{g}{\Longrightarrow} T}\ Lf \qquad \frac{}{S \underset{f}{\Longrightarrow} f\,S}\ Rf$$

satisfying equations

$$\frac{\dfrac{}{S \underset{f}{\Longrightarrow} f\,S}\ Rf \quad \dfrac{\overset{\beta}{S \underset{f;g}{\Longrightarrow} T}}{f\,S \underset{g}{\Longrightarrow} T}\ Lf}{S \underset{f;g}{\Longrightarrow} T}\ ; \quad = \quad \overset{\beta}{S \underset{f;g}{\Longrightarrow} T} \qquad \overset{\eta}{f\,S \underset{g}{\Longrightarrow} T} \quad = \quad \dfrac{\dfrac{\dfrac{}{S \underset{f}{\Longrightarrow} f\,S}\ Rf \quad \overset{\eta}{f\,S \underset{g}{\Longrightarrow} T}}{S \underset{f;g}{\Longrightarrow} T}\ ;}{f\,S \underset{g}{\Longrightarrow} T}\ Lf$$

Fact: a functor $\mathbf{U} : \mathcal{D} \to \mathcal{T}$ is a **fibration** iff a pullback $(f^* T, Lf^*, Rf^*)$ exists for all $f : A \to B$ and $T \sqsubset B$. It is a **bifibration** iff all pullbacks and pushforwards exist.

Proof: essentially immediate by unwinding definitions.

Proposition

Whenever the corresponding pullbacks exist:

1. *the following subtyping rule is valid:*

$$\frac{T_1 \leq T_2}{f^* \, T_1 \leq f^* \, T_2}$$

2. *we have*

$$(g; f)^* \, T \equiv g^* \, f^* \, T \qquad \mathrm{id}^* \, T \equiv T$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f^* T_1 \underset{f}{\Longrightarrow} T_1}\; Lf^* \qquad T_1 \le T_2}{f^* T_1 \underset{f;\mathrm{id}}{\Longrightarrow} T_2}\; ;}{f^* T_1 \underset{\mathrm{id};f}{\Longrightarrow} T_2}\; \sim}{f^* T_1 \le f^* T_2}\; Rf^*$$

A few examples of refinement systems

Example: subsets over sets

Let $\mathcal{T} = \textbf{Set}$ be the category of sets and functions, and let $\mathcal{D} = \textbf{SubSet}$ be the category whose objects are pairs

$$(A, S \subseteq A)$$

and whose morphisms

$$(A, S) \to (B, T)$$

are functions $f : A \to B$ such that

$$\forall a. a \in S \Rightarrow f(a) \in T$$

Then consider the projection functor $\mathbf{U} : \textbf{SubSet} \to \textbf{Set}$.

Observe: pullback = inverse image, pushforward = image

Example: Hoare Logic

Take $\mathcal{T}$ as a category with one object W corresponding to the state space, and with morphisms $c : W \to W$ corresponding to commands-as-state-transformers.

Define $\mathcal{D}$ and $\mathbf{U} : \mathcal{D} \to \mathcal{T}$ so that refinements $\phi \sqsubset W$ are state predicates, and a derivation of a typing judgment

$$\phi \underset{c}{\Longrightarrow} \psi$$

corresponds exactly to a verification of a Hoare triple $\{\phi\}c\{\psi\}$.

Observations:

- usual rules of sequential composition, pre-strengthening and post-weakening are valid (see slide 17)
- pullback = weakest pre, pushforward = strongest post

Example: STLC à la Curry (after Scott)

Take $\mathcal{T}$ as a ccc including an object U and a pair of operations

$$U \underset{lam}{\overset{app}{\rightleftarrows}} U^U$$

Take $\mathcal{D}$ as a ccc including a collection of simple types

$$\sigma, \tau, fn[\sigma, \tau], \ldots$$

together with morphisms

$$fn[\sigma, \tau] \underset{Lam_{\sigma,\tau}}{\overset{App_{\sigma,\tau}}{\rightleftarrows}} \tau^\sigma$$

Define $\mathbf{U} : \mathcal{D} \to \mathcal{T}$ (as a cartesian closed functor) by

$$\sigma, \tau, \ldots \mapsto U \qquad App_{\sigma,\tau} \mapsto app \qquad Lam_{\sigma,\tau} \mapsto lam$$

Example: refining a point

Any category determines a trivial refinement system:

$$\begin{array}{c} \mathcal{C} \\ \big\downarrow {\scriptstyle !_{\mathcal{C}}} \\ 1 \end{array}$$

What next?

In the paper, we describe various ways of taking these basic ideas further, including a bit of discussion of Separation Logic, and culminating in a rational reconstruction of Reynolds' (2000) proof of coherence for a language with subtyping.

In ongoing work, we are exploring how this framework can be applied towards a better understanding of dependent types and effects, as well as the proof theory of linear logic.

We like the slogan that

functors are type refinement systems

in part because it works both ways: on the one hand it brings some basic mathematical tools to bear on type theory, but it also suggests a broader scope for type-theoretic reasoning.

Making it more than a slogan will require a lot of work, and we would love some help!

Bonus slides

Separation Logic

Separating conjunction and magic wand:

$$\phi * \psi \stackrel{\text{def}}{=} \circledast(\phi \bullet \psi) \qquad \phi -\!\!* \, \tau \stackrel{\text{def}}{=} \lambda[\circledast]^* (\phi \setminus \tau)$$

(but also Day convolution)

Condition equivalent to the Frame Rule:

$$\phi * c^* \psi \leq c^* (\phi * \psi)$$

Reynolds 2000

The logical relations theorem:

$$\begin{array}{ccc} \mathcal{D} & \xrightarrow{\rho} & \mathbf{DRel} \\ \| & (0) & \downarrow{\scriptstyle \partial_0} \\ \mathcal{D} & \xrightarrow[\llbracket - \rrbracket_D]{} & \mathbf{Dom} \end{array} \qquad \begin{array}{ccc} \mathcal{D} & \xrightarrow{\rho} & \mathbf{DRel} \\ {\scriptstyle \mathbf{U}}\downarrow & (1) & \downarrow{\scriptstyle \partial_1} \\ \mathcal{T} & \xrightarrow[\llbracket - \rrbracket_T]{} & \mathbf{Dom} \end{array}$$

If $\theta_1 \overset{\alpha}{\underset{p}{\Longrightarrow}} \theta_2$ then $\vdash \rho[\theta_1] \underset{(\llbracket\alpha\rrbracket,\llbracket p\rrbracket)}{\Longrightarrow} \rho[\theta_2]$.

(case $\alpha = \mathit{NI}$)

$$\dfrac{\dfrac{\dfrac{}{\Delta[\mathbb{N}_\perp] \underset{(i,i)}{\Longrightarrow} \Delta[\mathbb{Z}_\perp]}\ \Delta[i]}{(\mathrm{id}, i)\,\Delta[\mathbb{N}_\perp] \underset{(i,\mathrm{id})}{\Longrightarrow} \Delta[\mathbb{Z}_\perp]}\ L(\mathrm{id}, i)}{(\mathrm{id}, \Psi_\mathrm{p})^*\,(\mathrm{id}, i)\,\Delta[\mathbb{N}_\perp] \underset{(i,\mathrm{id})}{\Longrightarrow} (\mathrm{id}, \Psi_\mathrm{p})^*\,\Delta[\mathbb{Z}_\perp]}\ (\mathit{fun})$$

$$\overline{\overline{\rho[\mathit{nat}] \underset{([\![\mathit{NI}]\!],[\![\mathrm{id}]\!])}{\Longrightarrow} \rho[\mathit{int}]}}$$

(case $\alpha = \mathit{Lam}$)

$$\dfrac{\dfrac{\dfrac{}{\rho[\theta_2]^{\rho[\theta_1]} \underset{(\mathrm{id},\mathrm{id})}{\Longrightarrow} \rho[\theta_2]^{\rho[\theta_1]}}\ \mathrm{id}}{\rho[\theta_2]^{\rho[\theta_1]} \underset{(\mathrm{id},(\Phi_\mathrm{f};\Psi_\mathrm{f}))}{\Longrightarrow} \rho[\theta_2]^{\rho[\theta_1]}}\ \sim}{\rho[\theta_2]^{\rho[\theta_1]} \underset{(\mathrm{id},\Phi_\mathrm{f})}{\Longrightarrow} (\mathrm{id}, \Psi_\mathrm{f})^*\,\rho[\theta_2]^{\rho[\theta_1]}}\ R(\mathrm{id}, \Psi_\mathrm{f})^*$$

$$\overline{\overline{\rho[{\theta_2}^{\theta_1}] \underset{([\![\mathit{Lam}]\!],[\![\mathit{lam}]\!])}{\Longrightarrow} \rho[\mathit{fn}[\theta_1, \theta_2]]}}$$