Notes on normalisation by evaluation and Kripke models

Pierre-Louis Curien (CNRS & Université Paris VII)

June 11, 2012

Abstract

I recollect here in (mild) categorical language the ideas which I learned from papers by Catarina Coquand, and by Herbelin and Lee, about the strong relation between normalisation by evaluation and Kripke models for intuitionnistic logic. I intend to improve and extend these notes during the first week of the Summer School.

1 Normalisation by evaluation

Let $\mathbf{C}$ be a category. Then the category $\widehat{\mathbf{C}} = \mathbf{Set}^{\mathbf{C}}$, called the category of presheaves over $\mathbf{C}$, being cartesian closed (CCC), is a model of simply typed λ-calculus. More precisely, the data of a presheaf $[\![\kappa]\!]$ associated with each basic type is required.

Normalisation by evaluation (NBE) naturally takes place in a presheaf catgory over a partially ordered set W of worlds (viewed as a category). Then a presheaf is just a collection of sets $X(w)$ indexed by worlds and related by functions $X_{w,w'} = X(w) \to X(w')$ for every pair of worlds $w \leq w'$, such that $X_{w,w} = id$ for all w and $X_{w',w''} \circ X_{w,w'} = X_{w,w''}$ for all $w \leq w' \leq w''$. We shall use the following notation:

- X_w for $X(w)$,
- $ext(S, w')$ for $X_{w,w'}(S)$ $(S \in X_w)$.

The general definition of function space in $\widehat{\mathbf{C}}$ instantiates as follows when $\mathbf{C} = W$ is a partial order: $(X \to Y)_w$ consists of the set of natural families of functions $f_{w'} : X_{w'} \to Y_{w'}$, ranging over all worlds $w' \geq w$.

More precisely, NBE takes place

- in $\widehat{\mathcal{U}}$, where $\mathcal{U}$ is the set of contexts, ordered by inclusion,
- and in the model determined by taking

$$[\![\kappa]\!] = \Lambda_{nf}(\kappa)$$

where, for all Γ, $\Lambda_{nf}(\kappa)_\Gamma$ is the set of all terms in normal form of type κ in context Γ.

We call this the universal model.

Of course, one gets more generally a presheaf $\Lambda_{nf}(\sigma)$ for every type σ, just replacing κ by σ. Also, replacing "normal form" by "neutral normal form" (where a neutral normal form is a normal form with no head λ's), we get presheaves $\Lambda_{ne}(\sigma)$. These presheaves will soon play a rôle. While we are here, here is the (mutually recursive) syntax for normal forms t and neutral normal forms e:

$$e ::= x \mid et$$
$$t ::= e \mid \lambda x.t$$

The ingredients of NBE are three-fold:

- a semantic function from terms to their interpretation in the (universal) model,
- a family of reification functions from the universal model to terms in normal form,
- an auxiliary family of reflection functions from neutral normal forms to the universal model. (The point is that reification needs reflection for its definition, by mutual induction.)

We do not need to specify the first ingredient, as it is just the usual interpretation in any CCC. The reification functions are organised as a family of morphisms $\downarrow^\sigma : [\![\sigma]\!] \to \Lambda_{nf}(\sigma)$ indexed by the types, and hence as a family of functions $\downarrow^\tau_\Gamma$ indexed by types and contexts. The reflection morphisms $\uparrow^\sigma$ go "the other way around", more precisely, from $\Lambda_{ne}(\sigma)$ to $[\![\sigma]\!]$. Schematically, we have thus:

$$\downarrow : [\![\ ]\!] \to \Lambda_{nf} \qquad \uparrow : \Lambda_{ne} \to [\![\ ]\!]$$

The reification and reflection functions are defined as follows:

$\downarrow^\kappa$ is the identity morphism

$\downarrow^{\tau\to\sigma}_\Gamma(S) = \lambda x.\downarrow^\sigma_{\Gamma,x:\tau}(S_{\Gamma,x:\tau}(\uparrow^\tau_{\Gamma,x:\tau}(x)))$ (x not in Γ)

$\uparrow^\kappa$ is the inclusion morphism

$\uparrow^{\tau\to\sigma}_\Gamma(e)_\Delta(S) = \uparrow^\sigma_\Delta(e\downarrow^\tau_\Delta(S))$

Slowly:

- In the second line on the left, S is an element of $[\![\tau \to \sigma]\!]_\Gamma$, hence $S_{\Gamma,x:\tau}$ is a function from $[\![\tau]\!]_{\Gamma,x:\tau}$ to $[\![\sigma]\!]_{\Gamma,x:\tau}$, and x is a neutral normal form of type τ in context $\Gamma, x : \tau$.
- In the second line on the right, e is a neutral normal form of type $\tau \to \sigma$ in context Γ, S is an element of $[\![\tau]\!]_\Delta$, hence $\downarrow^\tau_\Delta(S)$ is normal form of type τ in context Δ, from which it follows that $e\downarrow^\tau_\Delta(S)$ is a neutral normal form of type σ in context Δ, since a fortiori e is typed in context Δ by weakening.

We observe also that the "inclusion" in the first line on the right is actually just the identity function since $\Lambda_{ne}(\kappa) = \Lambda_{nf}(\kappa)$ at base types, for obvious typing reasons.

The NBE algorithm is then obtained as follows: take a term v of type τ in context $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$, and compute

$$\downarrow^\sigma_\Gamma([\![v]\!]_\Gamma(\uparrow^{\sigma_1}_\Gamma(x_1), \ldots, \uparrow^{\sigma_n}_\Gamma(x_n)))$$

which is a normal form.

It remains to be seen that this normal form is $\beta\eta$ convertible with M. For this, some invariants have to be shown (to be developped).

2 Axiomatic NBE

One can abstract from the particular model determined by $\Lambda_{nf}(\kappa)$, while still remaining with $\mathbf{C} = W = \mathcal{U}$. It turns out that only the following data are needed in a model over $\mathcal{U}$ in order to define reification and reflection morphisms. One needs two families of presheaves $Nf(\sigma), Ne(\sigma)$ over $\mathcal{U}$, indexed by the types, that satisfy the following axioms:

1. there is an inclusion morphism from $Ne(\kappa)$ to $Nf(\kappa)$,
2. there are natural functions and elements

$$\begin{array}{l} App : Ne(\tau \to \sigma)_\Gamma \times Nf(\tau)_\Gamma \to Ne(\sigma)_\Gamma \\ \Lambda : Nf(\sigma)_{\Gamma, x:\tau} \to Nf(\tau \to \sigma)_\Gamma \\ \pi \in Ne(\tau)_{\Gamma, x:\tau} \end{array}$$

One then fixes $[\![\kappa]\!]$ to be $Nf(\kappa)$, and one can define

$$\downarrow : [\![\ ]\!] \to Nf \qquad \uparrow : Ne \to [\![\ ]\!]$$

as follows:

$\downarrow^\kappa$ is the identity morphism

$\uparrow^\kappa$ is the inclusion morphism

$\downarrow^{\tau\to\sigma}_\Gamma(S) = \Lambda(\downarrow^\sigma_{\Gamma,x:\tau}(S_{\Gamma,x:\tau}(\uparrow^\tau_{\Gamma,x:\tau}(\pi))))$ (x not in Γ)

$\uparrow^{\tau\to\sigma}_\Gamma(e)_\Delta(S) = \uparrow^\sigma_\Delta(App(ext(e,\Delta), \downarrow^\tau_\Delta(S)))$

We can instantiate these axioms in at least two ways:

- $Nf(\sigma) = \Lambda_{nf}(\sigma)$ and $Ne(\sigma) = \Lambda_{ne}(\sigma)$ for all σ, and we are back to the standard NBE of the previous section. The source here is [3].
- $Nf(\sigma) = Test_\emptyset(\Lambda_{nf}(\sigma))$ and $Ne(\sigma) = Test_\emptyset(\Lambda_{ne}(\sigma))$ for all σ, where $Test_\emptyset : \mathbf{Set} \to \mathbf{Set}$ takes $\emptyset$ to $\emptyset$ and X to 1 (the terminal object $\{*\}$ of $\mathbf{Set}$) if X is not empty. We have that
  - the interpretation in this model amounts to the proof of soundness of Kripke models: the fact that the interpretation of a proof v that $\Gamma \vdash \sigma$ is defined amounts to the existence of a morphism from $[\![\Gamma]\!]$ to $[\![\sigma]\!]$ which unrolls as the usual Kripke semantics formula;
  - the reification from this model exhibits a constructive proof of completeness of Kripke models. Here, constructive means that from the soundness of the sequent $\Gamma \vdash \sigma$ one deduces not only the provability of $\Gamma \vdash \sigma$, but an actual proof tree (in normal form) for this judgement. (Combining with soundness, this gives a cut-elimination result for free as a bonus).

Note, to ease the understanding of what is going on here, that in this second model, we have interpreted the basic type by a "provability only" presheaf, by which we mean that $[\![\kappa]\!] = Nf(\kappa)$ is a collection of sets that are either empty or singletons. It is easy to see that this property extends to all types σ. Then the information contained in $[\![\sigma]\!]$ at world Γ is just a "yes" or a "no", and the "yes" is traditionally formulated as "Γ realises σ". (Therefore Kripke was working with very special presheaves!)

In summary, the axiomatic treatment given here provides evidence that, as observed informally in [5], the proof of completeness of Kripke semantics and normalisation by evaluation are (instances of) the same thing.

3 Extension to products

This raises no difficulty. The syntax of normal forms and neutral normal forms extends as follows.

$$e ::= \ldots \mid \mathtt{fst}(e) \mid \mathtt{snd}(e)$$
$$t ::= \ldots \mid (t, t)$$

(As an exercise, check that these are indeed the normal forms, when adding the rules $\mathtt{fst}(t_1, t_2) \to t_1$ and $\mathtt{fst}(t_1, t_2) \to t_1$ to β-reduction; note that certain configurations are forbidden because they would form redexes, some others because they would be ill-typed.)

The reification and reflection for the product case are:

$$\downarrow_\Gamma^{\sigma\times\tau}(S_1, S_2) = (\downarrow_\Gamma^\sigma(S_1), \downarrow_\Gamma^\tau(S_2)) \qquad \uparrow_\Gamma^{\sigma\times\tau}(e) = (\uparrow_\Gamma^\sigma(\mathtt{fst}(e)), \uparrow_\Gamma^\tau(\mathtt{snd}(e)))$$

What about sums? Well, try... . The sum of two presheaves is nicely defined componentwise like the product, and the reification is easy, but the reflection is not: we should guess whether e is an $\mathtt{inl}(e_1)$ or and $\mathtt{inr}(e_2)$ but this depends on the context. So we need a continuation semantics here, or, in direct style, to make use of (delimited) control. This will be further discussed in the next section, when it will be fully written. Let me just mention that this has to do with polarities: the sum is positive, unlike the implication and conjunction (the reversible rule for introducing sums would need two formulas on the right of a sequent, which is not permitted in intuitionistic logic).

4 The approach via partial evaluation and 2-level λ-calculus

This section, yet to be written, will be based on [1, 4]. This approach is purely syntactic, and relies on the 2-level λ-calculus, which was, I believe, introduced by Neil Jones. The 2-level λ-calculus has two abstractions and two applications marked with overlining and underlining, respectively.

In comparison with the previous sections, the overlining corresponds to the meta-level applications and (implicit) abstractions in the model, while the underlining corresponds to the λ-calculus syntax.

In the perspective of implementation, the overlining and underlining correspond to ML syntax and to an encoding of λ-terms as an inductive type, respectively.

Normalisation by evaluation takes a term M that has only overlinings, and returns (after evaluation in ML) a datum in the inductive type which can be viewed as a "decompiled" form of a normal form of M.

Type-directed partial evaluation extends the picture to terms M that have mixed overlinings and underlinings. Such terms are obtained by a binding time analysis of a program which we want to specialise in certain arguments. The typical example is the power function $\mathtt{power}(n, x)$ which is normally defined by recursion, but when n is fixed and know, say $n = 3$, then partial evaluation yields the more efficient program $\lambda x.x \times x \times x$.

Sums are treated using delimited control.

If you cannot wait, you may want to consult [4] up to, say, Section 2.6 included, and look closely at Figure 2, and [1] up to Section 1.2 included, and look closely at Figure 1.

The TDPE work postdates the initial paper on normalisation by evaluation of Berger and Schwichtenberg [2].

References

[1] V. Balat, Keeping sums under control (available from `www.pps.univ-paris-diderot.fr/~balat`.

[2] U. Berger and H. Schwichtenberg, An inverse of the evaluation functional for typed λ-calculus, Proceedings 6th Symposium on Logic in Computer Science (LICS'91), 203-211(1991).

[3] C. Coquand, A formalised proof of the soundness and completeness of a simply typed lambda-calculus with explicit substitutions. Higher-Order and Symbolic Computation 15(1): 57-90 (2002).

[4] O. Danvy, Type-directed partial evaluation, available from `http://www.cs.au.dk/~danvy/index-previous.html`.

[5] H. Herbelin and G. Lee, Forcing-based cut-elimination for Gentzen-style intuitionistic sequent calculus, in Proc. Logic, Language, Information and Computation, 16th International Workshop, WoLLIC 2009, Tokyo, Japan, June 21-25, 2009, 209–217, Lecture Notes in Computer Science 5514 (2009).