Internal Proofs of Strong Normalization

Cody Roux

AWS

Strong normalization (henceforth SN) is a useful property for a type theory to have. Among other things, it usually provides

1. consistency of the system under scrutiny
2. the existence of canonical forms for every term
3. decidability of type checking

of course, this is not the only way to provide such guarantees. For consistency, any model will suffice, and decidability of type checking, while quite nice in theory, is generally accompanied by potential runtimes that strain the meaning of the word. Canonicity can also be achieved nowadays via more subtle methods, like normalization-by-evaluation (NbE), which may be available in cases where SN may not hold, e.g. in the presence of certain η-laws.

However, SN does have the advantage of being straightforward to state, and very effective, leaving to the implementer the choice of normalization strategy, which is mostly delegated to the meta-theory in the case of NbE. We also regard the proof of normalization to have a certain combinatorial charm, with certain questions around it, notably the Barendregt-Geuvers-Klop conjecture (see, e.g. Barthe & al [2]), being still open.

Most proofs of normalization involve a certain kind of model construction, the so-called *logical relations model*. This is unsurprising based on point 1 above, for Gödelian reasons, as the proof SN $\Rightarrow$ consistency can often be carried out in a very weak system (and therefore the "tough bit" must be in the proof of normalization). This construction feels very similar to the classic realizability interpretation $t \Vdash P$ of Kleene which relates certain kinds of program encodings t to propositions P, and indeed may be seen as a generalization of it.

It is a classical fact that, in Heyting arithmetic HA, any theorem P has a realizer t, and that HA "knows" that it is a realizer:

$$HA \vdash P \quad \Rightarrow \quad HA \vdash \ulcorner t \Vdash P \urcorner$$

This is nice, but the richness of HA allows for a lot of mess in this proof, e.g. there may be some instances of induction required to carry out the second proof that are not obviously related to the first, to deal with encoding issues.

We show that this is not necessarily the case, by working in a setting potentially much poorer than HA, at least in terms of expressiveness, and adding only the requisite tools to express and prove the strong normalization theorem (for a given well-typed term) without any additional meta-theoretic tools.

It's worth noting that such a proof translation will not really guarantee normalization, of course: we need an additional fact, namely *existence of normal forms*. This may seem redundant, but it is worth noting that existence of normal forms is actually weaker than strong normalization! In essence our strategy is to bootstrap this stronger property from the weaker one.

Our proof is quite similar to that of Berger & al [3] with the main difference being that we aim to identify exactly which type theoretic mechanisms are required in the target logic,

with the aim of identifying a natural map from theories to theories which can encode their own normalization proof.

To translate the proof, move from a source type theory $\mathcal{T}$ to a richer theory $\hat{\mathcal{T}}$ which can express facts about syntax.

We will need a new sort which we call Syn. This sort will contain a number of types (Term, Var, ...), which themselves contain symbols for terms, variables, substitutions etc. as well as constructors for the normalization predicates, SN and neutral terms NE essential to the proof. We do *not* need eliminators for any of these types, since induction shall (and must!) remain at the meta-level.

We further extend the target type systems with a dependent product $\Pi x : T.U$ in the case that $T : \text{Syn}$ is a syntax form and U is a well typed type in the original system, e.g. has type Type or Prop. The resulting product is in the sort of T, allowing us to chain such products. We note here that the fact that SN and NE eliminate into that sort as well requires us to have at least a non-dependent function type of that sort, that is, if $T,\ U : \text{Type}$ then $T \to U : \text{Type}$.

Finally we extend the normal conversion rule of the theory, by adding rewrite rules over the inhabitants of Syn to encode *substitution*, which can be taken, e.g. to be the σ-rules of Abadi & al [1]. However, we do *not* add a β-rule for syntactic terms. Intuitively, these rules allow us to reason modulo substitutions, as they are confluent and terminating in the absence of β.

Now we can translate term derivations in the source theory into derivations of computability in the target theory. The idea is to translate each type $T : \text{Type}$ (or any other sort) into a predicate $[\![T]\!] : \text{Term} \to \text{Type}$, and prove, by (meta-)induction on the type derivation,

$$\Gamma \vdash_{\mathcal{T}} t : T \quad \Rightarrow \quad [\![\Gamma]\!] \vdash_{\hat{\mathcal{T}}} (\!|t|\!) : [\![T]\!]\ \ulcorner t^o \urcorner$$

where $\ulcorner t^o \urcorner$ is the *open* syntactic form of t, containing variables of type Term appearing in $[\![\Gamma]\!]$ which correspond to the original t variables. Predictably, the translation of non-dependent arrow types will be the "realizability arrow"

$$[\![A \to B]\!] := \lambda t : \text{Term}.\Pi u : \text{Term}.[\![A]\!]\ u \to [\![B]\!]\ (t\ u)$$

but type variables X will need their own variable predicates $\mathcal{X}$, along with additional constructors

$$\text{ne} : \Pi t : \text{Term}.\text{NE}\ t \to \mathcal{X}\ t$$

$$\text{sn} : \Pi t : \text{Term}.\mathcal{X}\ t \to \text{SN}\ t$$

$$\text{wh} : \Pi t\ u\ t' : \text{Term}.\text{SN}\ u \to \text{WHExp}\ t\ t'\ u \to \mathcal{X}\ t' \to \mathcal{X}\ t$$

The predicate WHExp expresses that t is the weak head expansion of t' by u, that is, $t = (\lambda x.t_1)\ u\ t_2 \dots t_n$ and $t' = t_1[u/x]\ t_2 \dots t_n$.

These conditions are exactly the *Girard conditions* (at least, the so-called "saturated sets" form), and sadly, we must add with them some common sense reductions. Thankfully these conversions do not interact with β-reductions, and so do not threaten weak normalization.

Finally, from there we build a proof $\mathcal{G} \vdash \text{sn}_t : \text{SN}\ \ulcorner t \urcorner$ for a well chosen $\mathcal{G}$ (closely related to Γ), a surprisingly non-trivial task in the absence of induction over syntax! This time, $\ulcorner t \urcorner$ is the actual syntax of t, variables and all. From there we may conclude, if the original type theory admits normal forms, that t is indeed strongly normalizing. This relies on a simple induction on normal forms of terms of type SN t, along with the following meta-theorem:

Theorem 1. *If $\mathcal{T}$ admits normal forms, so does $\hat{\mathcal{T}}$.*

The feasibility of our translation, and of course, the correctness of our final conclusion (that t is strongly normalizing) will depend on exactly which constructors we give for SN, and whether they are true in the meta-theory. Here, again, we proceed similarly to Berger & al [3] and add the (folklore) inductive characterizations of strong normalization using weak-head expansions, and an additional fact: SN t holds if SN $(t\ x)$ holds for any variable x.

We give, as illustration, a representation of the normal form of the proof of SN for $(\lambda x.x)\ y$, which, while not eminently surprising, may illustrate the strategy.

$$\mathcal{A} : \mathrm{Term} \to *,$$

$$\mathrm{ne}_{\mathcal{A}} : \Pi t : \mathrm{Term}.\mathrm{NE}\ t\ \to \mathcal{A}\ t,\ \mathrm{sn}_{\mathcal{A}} : \Pi t : \mathrm{Term}.\mathcal{A}\ t \to \mathrm{SN}\ t,$$

$$\mathrm{wh}_{\mathcal{A}} : \Pi t\ t'\ u.\mathrm{WHExp}\ t\ t'\ u \to \mathrm{SN}\ u \to \mathcal{A}\ t' \to \mathcal{A}\ t,$$

$$\vdash$$

$$\mathrm{wh}_{\mathrm{SN}}\ \ulcorner(\lambda x.x)\ y\urcorner\ \ulcorner y\urcorner\ \ulcorner\ulcorner y\urcorner\urcorner\ (\mathrm{ne}_{\mathrm{SN}}\ (\mathrm{ne}_{\mathrm{Var}}\ulcorner y\urcorner))\ (\mathrm{ne}_{\mathrm{SN}}\ (\mathrm{ne}_{\mathrm{Var}}\ulcorner y\urcorner)) : \mathrm{SN}\ \ulcorner(\lambda x.x)\ y\urcorner$$

here $\mathrm{wh}_{\mathrm{SN}}$, $\mathrm{ne}_{\mathrm{SN}}$, $\mathrm{ne}_{\mathrm{Var}}$ etc. are part of the constructors for the type families SN and NE, and as a result the normalization of the top-level term can simply be read off of the proof, and the occurrences of $\mathrm{wh}_{\mathcal{A}}$ were reduced away as the semantic variable y was replaced by its syntactic counterpart $\ulcorner y\urcorner$.

As a conclusion, we note that the idea of conducting a logical relations argument within its own type theory is not new; our work is inspired from Bernardy & Lasson [4], where they introduce, for a given type theory, an enriched theory capable of expressing logical relations between the terms of the original theory.

There are many similar such categorical investigations (see e.g. Bocquet & al [5, 6]), though the author is at the moment incapable of determining the exact relationship with this more syntactic argument. The main novelty here is to carry out the combinatorial argument for normalization internally, including the last step of "extracting" the normal form.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[2] Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. Weak normalization implies strong normalization in a class of non-dependent pure type systems. *Theoretical Computer Science*, 269(1):317–361, 2001.

[3] Ulrich Berger, Stefan Berghofer, Pierre Letouzey, and Helmut Schwichtenberg. Program extraction from normalization proofs. *Studia Logica: An International Journal for Symbolic Logic*, 82(1):25–49, 2006.

[4] Jean-Philippe Bernardy and Marc Lasson. Realizability and parametricity in pure type systems. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures*, pages 108–122, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[5] Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. For the metatheory of type theory, internal sconing is enough, 2023.

[6] Jonathan Sterling. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory*. PhD thesis, Carnegie Mellon University, 2022.