NbE for LNL via Adjoint Meta-modalities

James Wood[1]

Huawei Research Centre, Edinburgh, UK
lamudri@gmail.com

Abstract

Linear/non-Linear Logic (LNL) was introduced by Benton [1994], and is a calculus comprising both linear and intuitionistic terms. In this work, we give an intrinsically typed Agda-mechanised (see Wood [2025] for the code) normalisation by evaluation (NbE) [Berger and Schwichtenberg, 1991] procedure for LNL, using metalanguage constructs F and G to mediate between linear and intuitionistic parts throughout the metatheory.

Overview. We give an NbE procedure based on that for Simply Typed λ-Calculus given by Allais et al. [2017], in particular reusing the context-implicit notational style of that work. Our model construction and *reify*, *reflect*, and *evaluation* functions appear to be very similar, just with refinements to deal with linearity and modifications for the choice of object language connectives. However, the underlying definitions and lemmas, particularly pertaining to *environments*, are different and new, and are our main focus in this extended abstract.

Contexts. We take the LNL types of Benton [1994], split into *linear* and *intuitionistic* types. We let intuitionistic contexts (Θ and Λ) be lists of intuitionistic types, and let linear contexts (Γ and Δ) be lists of arbitrary types tagged **lin** or **int** as appropriate. The two kinds of context can be related by $\sim$, with $\Gamma \sim \Theta$ whenever all types in Γ are intuitionistic and Γ and Θ are pointwise equal. Note that $\sim$ is functional in both directions and total from right to left.

We often consider families of sets indexed over contexts, with $\mathcal{L}$Fam and $\mathcal{I}$Fam being the sets indexed over linear and intuitionistic contexts respectively. We have operators on $\mathcal{L}$Fam and $\mathcal{I}$Fam called *meta-connectives*, coloured blue. These include the *proof relevant separation logic* connectives of Rouvoet et al. [2020] — namely *separating conjunction* $*$, *separating unit* I, and *separating implication* $-\!\!*$ — on $\mathcal{L}$Fam. Acting on both $\mathcal{L}$Fam and $\mathcal{I}$Fam are the pointwise operators $\dot{\times}$, $\dot{\to}$, and $\dot{\cup}$. We also have the *meta-modalities* F and G of definition 1. Binders Σ and Π stand for the dependent pair/function type formers of the ambient theory (Agda).

Definition 1 (Meta-modalities)**.** *Let* $\mathrm{F} : \mathcal{I}\mathrm{Fam} \to \mathcal{L}\mathrm{Fam}$ *be defined by* $(\mathrm{F}\,T)\ \Gamma \coloneqq \Sigma\Theta.\ \Gamma \sim \Theta \times T\,\Theta$*, and let* $\mathrm{G} : \mathcal{L}\mathrm{Fam} \to \mathcal{I}\mathrm{Fam}$ *be defined by* $(\mathrm{G}\,T)\ \Theta \coloneqq \Sigma\Gamma.\ \Gamma \sim \Theta \times T\,\Theta$*.*

Our first use of the meta-modalities is in defining the object-language (LNL) modalities: F and G. The introduction forms use the corresponding meta-modalities directly, similarly to how $\otimes$-introduction uses $*$ directly to split the context between the two subterms. G-elimination also uses meta-F directly, to restrict the rule's use to purely intuitionistic contexts. However, F-elimination does not use any meta-modalities, instead using binding of an intuitionistic variable to cross modes. F-elimination does, nevertheless, use meta-connective $*$ to split the free linear variables between the two premises. Each of these rules is understood to take an arbitrary ambient linear/intuitionistic (as appropriate) context, which is the context of the conclusion. The contexts of the premises are derived from the ambient context via the meta-connectives.

$$\frac{\mathrm{F}\,(\vdash_{\mathcal{I}} X)}{\vdash_{\mathcal{L}} \mathrm{F}X}\ \mathrm{FI} \qquad \frac{\mathrm{G}\,(\vdash_{\mathcal{L}} A)}{\vdash_{\mathcal{I}} \mathrm{G}A}\ \mathrm{GI} \qquad \frac{\vdash_{\mathcal{L}} \mathrm{F}X \ \ *\ \ \mathbf{int}\,X \vdash_{\mathcal{L}} A}{\vdash_{\mathcal{L}} A}\ \mathrm{FE} \qquad \frac{\mathrm{F}\,(\vdash_{\mathcal{I}} \mathrm{G}A)}{\vdash_{\mathcal{L}} A}\ \mathrm{GE}$$

Aside from modalities F and G, we have linear and intuitionistic function types, negative intuitionistic products, and positive linear tensor products, all with rules equivalent to those of Benton [1994, Fig. 3]. We also have linear and intuitionistic base types, with no rules.

Variables and environments. The variable judgements are defined as follows. The variable judgements embed into their respective term judgements via the variable rules (not pictured).

Definition 2 (Variables)**.** *Let $\Theta \ni_{\mathcal{I}} X$ be the set of entries of Θ equal to X. Let $\Gamma \ni_{\mathcal{L}} A$ be a singleton if there is exactly one linear entry in Γ and that entry has type A, and empty otherwise.*

We choose very intensionally different representations of environments in the intuitionistic and linear cases. This is partially to avoid any coincidences and sleights of hand when passing between environment types, but also reflects the path of least resistance in each mode considered separately. The functional definition given in definition 3 is standard, but unavailable in the linear case without some overarching linearity condition [Wood, 2024, sec. 5.1]. These definitions are parametrised on an intuitionistic judgement form $\triangleright$ and a linear judgement form $\blacktriangleright$. Both of these judgement forms are indexed on a context and a type, like the variable and term judgement forms, where the context and type are intuitionistic for $\triangleright$ and linear for $\blacktriangleright$.

Definition 3 (Intuitionistic environment)**.** *Let $\Theta \overset{\triangleright}{\Longrightarrow}_{\mathcal{I}} \Lambda \coloneqq \Pi X.\ \Lambda \ni_{\mathcal{I}} X \to \Theta \triangleright X$.*

Definition 4 (Linear environment)**.** *Let $(-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta$ be defined inductively by the following inclusions, where $\forall[\ T\ ] \coloneqq \Pi\Gamma.\ T\,\Gamma$ and $(S \ \dot{\to}\ T)\ \Gamma \coloneqq S\,\Gamma \to T\,\Gamma$:*

$$\forall\Big[\ \mathrm{I}\ \dot{\to}\ (-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \cdot\ \Big] \qquad \forall\Big[\ (-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta * \quad (-) \blacktriangleright A \ \ \dot{\to}\ (-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta, \mathbf{lin}\, A\ \Big]$$

$$\forall\Big[\ (-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta * \mathrm{F}\,((-) \triangleright X)\ \ \dot{\to}\ (-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta, \mathbf{int}\, X\ \Big]$$

We convert between linear and intuitionistic contexts using the following lemma. We only convert at purely intuitionistic contexts, as enforced by the meta-modalities, so the linear value judgement $\blacktriangleright$ does not matter. Note that it is somewhat unusual to apply meta-connectives to a family with its open place on the *right* of the environment judgements.

Lemma 5. *Given linear contexts Γ, Δ and intuitionistic contexts Θ, Λ, we have a functions from $\mathrm{G}\left(\Gamma \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} (-)\right) \Lambda$ to $\mathrm{F}\left((-) \overset{\triangleright}{\Longrightarrow}_{\mathcal{I}} \Lambda\right) \Gamma$ and from $\mathrm{F}\left(\Theta \overset{\triangleright}{\Longrightarrow}_{\mathcal{I}} (-)\right) \Delta$ to $\mathrm{G}\left((-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta\right) \Theta$.*

On the intuitionistic side, distribution of an environment between subterms is trivial (by copying the whole environment). On the linear side, however, we have that an environment into a context which splits yields a splitting of the source context and two smaller environments.

Lemma 6. *If we have an environment of type $\Gamma \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta$ and Δ splits into Δ_l and Δ_r, then we have $\left((-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta_l * (-) \overset{\blacktriangleright,\triangleright}{\Longrightarrow}_{\mathcal{L}} \Delta_r\right) \Gamma$. If, instead of splitting, Δ contains no linear assumptions, then Γ also contains no linear assumptions, i.e., $\mathrm{I}\,\Gamma$.*

We use environments to define both renamings (environments in which the values are variables) and evaluation environments (environments in which the values are elements from the NbE model). With renamings come the meta-modalities $\square_{\mathcal{L}}$ and $\square_{\mathcal{I}}$, which take a family and produce the largest family stable under renaming contained in it, as per Allais et al. [2017].

NbE. Our NbE model is given by families $\vDash_{\mathcal{L}}$ and $\vDash_{\mathcal{I}}$, defined below with the contexts left implicit and manipulated via meta-connectives. Once again, meta-F and meta-G help in interpreting object-F and object-G, and similar relationships hold for other meta-connectives and types. As is standard, both function types have their interpretation coerced into being renameable, and positive types may be interpreted as neutral terms as well as semantic values.

$$
\begin{array}{llll}
\vDash_{\mathcal{L}} \iota_{\mathcal{L}} & := \vdash^{\mathrm{ne}}_{\mathcal{L}} \iota_{\mathcal{L}} & \vDash_{\mathcal{I}} \iota_{\mathcal{I}} & := \vdash^{\mathrm{ne}}_{\mathcal{I}} \iota_{\mathcal{I}} \\
\vDash_{\mathcal{L}} A \otimes B & := (\vDash_{\mathcal{L}} A \mathbin{\dot{*}} \vDash_{\mathcal{L}} B) \ \dot{\cup}\ \vdash^{\mathrm{ne}}_{\mathcal{L}} A \otimes B & \vDash_{\mathcal{I}} X \times Y & := \vDash_{\mathcal{I}} X \ \dot{\times}\ \vDash_{\mathcal{I}} Y \\
\vDash_{\mathcal{L}} A \multimap B & := \Box_{\mathcal{L}} (\vDash_{\mathcal{L}} A \mathbin{\dot{-\!*}} \vDash_{\mathcal{L}} B) & \vDash_{\mathcal{I}} X \to Y & := \Box_{\mathcal{I}} (\vDash_{\mathcal{I}} X \ \dot{\to}\ \vDash_{\mathcal{I}} Y) \\
\vDash_{\mathcal{L}} \mathrm{F}X & := \mathrm{F} (\vDash_{\mathcal{I}} X) \ \dot{\cup}\ \vdash^{\mathrm{ne}}_{\mathcal{L}} \mathrm{F}X & \vDash_{\mathcal{I}} \mathrm{G}A & := \mathrm{G} (\vDash_{\mathcal{L}} A)
\end{array}
$$

With these definitions, the *reify* and *reflect* functions follow largely as may be expected. The evaluator $eval_{\mathcal{L}} : \Gamma \overset{\vDash_{\mathcal{L}},\vDash_{\mathcal{I}}}{\Longrightarrow}_{\mathcal{L}} \Delta \to \Delta \vdash_{\mathcal{L}} A \to \Gamma \vDash_{\mathcal{L}} A$ (and similar for the intuitionistic mode) is somewhat more involved, and uses most of the properties and operations established about meta-connectives (including their functoriality), environments, and renaming. Details can be found in the associated artefact [Wood, 2025].

References

Guillaume Allais, James Chapman, Conor McBride, and James McKinna. Type-and-scope safe programs and their proofs. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2017, page 195–207, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450347051. doi: 10.1145/3018610.3018613. URL https://doi.org/10.1145/3018610.3018613.

P.N. Benton. A mixed linear and non-linear logic: Proofs, terms and models. pages 121–135. Springer-Verlag, 1994.

Ulrich Berger and Helmut Schwichtenberg. An inverse of the evaluation functional for typed lambda-calculus. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991*, pages 203–211. IEEE Computer Society, 1991. doi: 10.1109/LICS.1991.151645. URL https://doi.org/10.1109/LICS.1991.151645.

Arjen Rouvoet, Casper Bach Poulsen, Robbert Krebbers, and Eelco Visser. Intrinsically-typed definitional interpreters for linear, session-typed languages. In *CPP 2020*, pages 284–298, 2020. ISBN 9781450370974. doi: 10.1145/3372885.3373818.

James Wood. *A framework for semiring-annotated type systems*. PhD thesis, 2024. URL https://doi.org/10.48730/pa0y-ct71.

James Wood. NbE for LNL via adjoint meta-modalities (artefact), 2025. URL https://github.com/laMudri/lin-env/blob/types25-submission/src/Modal/LnL.agda.