A Linear Algebra Approach to Linear Metatheory

James Wood*
University of Strathclyde
Glasgow, United Kingdom
james.wood.100@strath.ac.uk

Robert Atkey
University of Strathclyde
Glasgow, United Kingdom
robert.atkey@strath.ac.uk

Linear typed λ-calculi are more delicate than their simply typed siblings when it comes to metatheoretic results like preservation of typing under renaming and substitution. Tracking the usage of variables in contexts places more constraints on how variables may be renamed or substituted. We present a methodology based on linear algebra over semirings, extending McBride's *kits and traversals* approach for the metatheory of syntax with binding to linear usage-annotated terms. Our approach is readily formalisable, and we have done so in Agda.

1 Introduction

The basic metatheoretic results for typed λ-calculi, such as preservation of typing under renaming, weakening, substitution and so on, are crucial but quite boring to prove. In calculi with substructural typing disciplines and modalities, it can also be quite easy to break these properties [Wad92, BBPH93]. It is desirable therefore to use a proof assistant to prove these properties. This has the double benefit of both confidence in the results and in focusing on the essential properties required to obtain them.

Mechanisation of the metatheory of substructural λ-calculi has not received the same level of attention as intuitionistic typing. "Straightforward" translations from paper presentations to formal presentations make metatheory difficult, due to incompatibilies between the standard de Bruijn representation of binding and the splitting of contexts. For formalisations of linear sequent calculi, sticking to the paper presentation using lists and permutations is common [PW99, XORN17, Lau18], but explicit permutations make the resulting encodings difficult to use. Multisets for contexts are more convenient [CLR19], but do not work well for Curry-Howard uses, as noted by Laurent. For natural deduction, Allais [All18] uses an I/O model to track usage of variables, Rouvoet et al. [RBPKV20] use a co-de Bruijn representation to distribute variables between subterms, and Crary uses mutually defined typing and linearity judgements with HOAS [Cra10].

In this paper, we adapt the generic *kits and traversals* technique for proving admissibility of renaming and substitution due to McBride [McB05] to a linear typed λ-calculus where variables are annotated with values from a skew semiring denoting those variables' *usage* by terms. Our calculus, $\lambda\mathcal{R}$, is a prototypical example of a linear "quantitative" or "coeffect" calculus in the style of [RP10, BGMZ14, GS14, POM14, OLE19]. The key advantages of $\lambda\mathcal{R}$ over the formalisations listed above are that the shape of typing contexts is maintained, so de Bruijn indices behave the same as in non-substructural calculi, and by selecting different semirings, we obtain from $\lambda\mathcal{R}$ well known systems, including Barber's Dual Intuitionistic Linear Logic [Bar96] and Pfenning and Davis' S4 modal type theory [PD99].

McBride's kits and traversals technique isolates properties required to form binding-respecting traversals of simply typed λ-terms, so that renaming and substitution arise as specific instantiations. Benton, Hur, Kennedy, and McBride [BHKM12] implement the technique in Coq and extend it to polymorphic

*James Wood is supported by an EPSRC Studentship.

Submitted to:
Linearity & TLLA 2020

terms. Allais *et al.* [AAC+20] generalise to a wider class of syntax with binding and show that more general notions of *semantics* can be handled. Using methods like these can reduce the effort required to develop a new calculus and its metatheory.

To adapt kits and traversals to linear usage-annotated terms requires us to not only respect the binding structure, but to also respect the usage annotations. For instance, the usages associated with a term being substituted in must be correctly distributed across all the occurrences of that term in the result. To aid us in tracking usages, we employ the linear algebra of vectors and matrices induced by the skew semiring we are using. Usage annotations on contexts are vectors, usage-preserving maps of contexts are matrices, and the linearity properties of the maps induced by matrices are exactly the lemmas we need for showing that traversals (and hence renaming, subusaging, and substitution) preserve typing and usages.

The paper proceeds as follows. In section 2, we specify our requirements on the set of annotations that will track usage of variables. A consequence of our formalisation is that we learn that we only need *skew* semirings, a weaker structure than the partially ordered semirings usually used. In section 3, we use these annotations to define the system $\lambda\mathcal{R}$ in an intrinsically typed style. Then, in section 4, we prove that $\lambda\mathcal{R}$ admits renaming, subusaging, and substitution by our extension of McBride's kits and traversals technique. We conclude in section 5 with some directions for future work.

The Agda formalisation of this work can be found at `https://github.com/laMudri/generic-lr/tree/master/src/Specific`. It contains our formalisation of vectors and matrices (approx. 790 lines) and the definition of $\lambda\mathcal{R}$ and proofs of renaming and substitution (approx. 530 lines).

2 Skew semirings

We shall use skew semirings where other authors have previously used partially ordered semirings (see, for example, the Granule definition of a *resource algebra* [OLE19]). Elements of a skew semiring are used as *usage annotations*, and describe *how* values are used in a program. In the syntax for $\lambda\mathcal{R}$, each assumption will have a usage annotation, describing how that assumption can be used in the derivation. Addition describes how to combine multiple usages of an assumption, and multiplication describes the action our graded !-modality can have. The ordering describes the specificness of annotations. If $\pi \trianglelefteq \rho$, π can be the annotation for a variable wherever ρ can be. We can read this relation as "supply $\trianglelefteq$ demand" — given a variable annotated π, we can coerce to treat it as if it has the desired annotation ρ.

Skew semirings are a generalisation of partially ordered semirings, which are in turn a generalisation of commutative semirings. As such, readers unfamiliar with the more general structures may wish to think in terms of the more specific structures. Our formalisation was essential for noticing and sticking to this level of generality.

Definition 2.1. *A* (left) skew monoid *is a structure* $(\mathbf{R}, \trianglelefteq, 1, *)$ *such that* $(\mathbf{R}, \trianglelefteq)$ *forms a partial order,* $*$ *is monotonic with respect to* $\trianglelefteq$*, and the following laws hold (with* $x * y$ *henceforth being written as* xy*).*

$$1x \trianglelefteq x \qquad\qquad x \trianglelefteq x1 \qquad\qquad (xy)z \trianglelefteq x(yz)$$

Remark 2.2. *A commutative skew monoid is just a commutative monoid.*

Skew-monoidal categories are due to Szlachányi [Szl12], and the notion introduced here of a skew monoid is a decategorification of the notion of skew-monoidal category.

Definition 2.3. *A* (left) skew semiring *is a structure* $(\mathbf{R}, \trianglelefteq, 0, +, 1, *)$ *such that* $(\mathbf{R}, \trianglelefteq)$ *forms a partial order,* $+$ *and* $*$ *are monotonic with respect to* $\trianglelefteq$*,* $(\mathbf{R}, 0, +)$ *forms a commutative monoid,* $(\mathbf{R}, \trianglelefteq, 1, *)$ *forms a skew monoid, and we have the following distributivity laws.*

$$0z \trianglelefteq 0 \qquad\qquad (x+y)z \trianglelefteq xz + yz \qquad\qquad 0 \trianglelefteq x0 \qquad\qquad xy + xz \trianglelefteq x(y+z)$$

Example 2.4. *In light of the above remark, most "skew" semirings are actually just partially ordered semirings. An example that yields a system equivalent to Barber's DILL is the* $0 \triangleright \omega \triangleleft 1$ *semiring of "unused", "unrestricted", and "linear", respectively. See [OLE19] for more examples.*

We will only speak of *left* skew semirings, and thus generally omit the word "left". A mnemonic for (left) skew semirings is "multiplication respects operators on the left from left to right, and respects operators on the right from right to left". One may also describe multiplication as "respecting" and "corespecting" operators on the left and right, respectively.

From a skew semiring $\mathbf{R}$, we form finite vectors, which we notate as $\mathbf{R}^n$, and matrices, which we notate as $\mathbf{R}^{m\times n}$. In Agda, we represent vectors in $\mathbf{R}^n$ as functions $\mathrm{Idx}\ n \to \mathbf{R}$, where $\mathrm{Idx}\ n$ is the type of valid indexes in an n-tuple, and matrices in $\mathbf{R}^{m\times n}$ as functions $\mathrm{Idx}\ m \to \mathrm{Idx}\ n \to \mathbf{R}$. Whereas elements of $\mathbf{R}$ describe how individual *variables* are used, elements of $\mathbf{R}^n$ describe how all of the variables in an n-length *context* are used. We call such vectors *usage contexts*, and take them to be row vectors. Matrices in $\mathbf{R}^{m\times n}$ will be used to describe how usage contexts are transformed by renaming and substitution in section 4. We define $\trianglelefteq$, 0 and $+$ on vectors and matrices pointwise. Basis vectors $\langle i|$ (used to represent usage contexts for individual variables), identity matrices $\mathbf{I}$, matrix multiplication $*$, and matrix reindexing $-_{-\times -}$ are defined as follows:

$$\begin{aligned}&\langle - | : \mathrm{Idx}\ n \to \mathbf{R}^n \\ &\langle i|_j := \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise}\end{cases}\end{aligned} \qquad \begin{aligned}&\mathbf{I} : \mathbf{R}^{m\times m} \\ &\mathbf{I}_{ij} := \langle i|_j\end{aligned} \qquad \begin{aligned}&* : \mathbf{R}^{m\times n} \times \mathbf{R}^{n\times o} \to \mathbf{R}^{m\times o} \\ &(MN)_{ik} := \textstyle\sum_j M_{ij} N_{jk}\end{aligned}$$

$$\begin{aligned}&-_{-\times -} : \mathbf{R}^{m'\times n'} \times (\mathrm{Idx}\ m \to \mathrm{Idx}\ m') \times (\mathrm{Idx}\ n \to \mathrm{Idx}\ n') \to \mathbf{R}^{m\times n} \\ &(M_{f\times g})_{i,j} := M_{f\,i, g\,j}\end{aligned}$$

We define vector-matrix multiplication by treating vectors as 1-height matrices.

3 Syntax

We present the syntax of $\lambda\mathcal{R}$ as an *intrinsically* typed syntax, as it is in our Agda formalisation. Intrinisic typing means that we define well-typed terms as inhabitants of an inductive family $\mathcal{R}\Gamma \vdash A$ indexed by typing contexts Γ, usage contexts $\mathcal{R}$, and types A. Typing contexts are lists of types. Usage contexts $\mathcal{R}$ are vectors of elements of some fixed skew semiring $\mathbf{R}$, with the same number of elements as the typing context they are paired with. To highlight how usage annotations are used in the syntax, we write all elements of $\mathbf{R}$, and vectors and matrices thereof, in green.

The types of $\lambda\mathcal{R}$ are given by the grammar: $A,B,C ::= \iota \mid A \multimap B \mid 1 \mid A \otimes B \mid 0 \mid A \oplus B \mid \top \mid A \,\&\, B \mid !_\rho A$. We have a base type ι, function types $A \multimap B$, tensor product types $A \otimes B$ with unit 1, sum types $A \oplus B$ with unit 0, "with" product types $A \,\&\, B$ with unit $\top$, and an exponential $!_\rho A$ indexed by a usage ρ.

We distinguish between *plain* variables, values of type $\Gamma \ni A$, which are indices into a context with a specified type, and *usage-checked* variables, values of type $\mathcal{R}\Gamma \sqsupseteq A$ which are pairs of a plain variable $i : \Gamma \ni A$ and proof that $\mathcal{R} \trianglelefteq \langle i|$. The force of the latter condition is that the selected variable i must have a usage annotation $\trianglelefteq 1$ in $\mathcal{R}$, and all other variables must have a usage annotation $\trianglelefteq 0$. We will sometimes silently cast between the types $\mathrm{Idx}\ m$ and $\Gamma \ni A$, particularly when using the reindexing operation $-_{-\times -}$.

The constructors for our intrinsically typed terms are presented in Figure 1. In keeping with our intrinsic typing methodology, terms of $\lambda\mathcal{R}$ are presented as constructors of the inductive family $\mathcal{R}\Gamma \vdash A$, hence the notation $M : \mathcal{R}\Gamma \vdash A$ instead of the more usual $\mathcal{R}\Gamma \vdash M : A$. Our Agda formalisation uses

Figure 1: Typing rules of $\lambda\mathcal{R}$

$$\frac{x : \mathcal{R}\Gamma \ni A}{x : \mathcal{R}\Gamma \vdash A}\ \textsc{var} \qquad \frac{M : \mathcal{P}\Gamma \vdash A \multimap B \qquad N : \mathcal{Q}\Gamma \vdash A \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{M\,N : \mathcal{R}\Gamma \vdash B}\ {\multimap}\text{-E}$$

$$\frac{M : \mathcal{R}\Gamma, 1A \vdash B}{\lambda M : \mathcal{R}\Gamma \vdash A \multimap B}\ {\multimap}\text{-I} \qquad \frac{M : \mathcal{P}\Gamma \vdash 1 \qquad N : \mathcal{Q}\Gamma \vdash C \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{\text{let } ({}_{\otimes}) = M \text{ in } N : \mathcal{R}\Gamma \vdash C}\ 1\text{-E}$$

$$\frac{\mathcal{R} \trianglelefteq \mathbf{0}}{({}_{\otimes}) : \mathcal{R}\Gamma \vdash 1}\ 1\text{-I} \qquad \frac{M : \mathcal{P}\Gamma \vdash A \otimes B \qquad N : \mathcal{Q}\Gamma, 1A, 1B \vdash C \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{\text{let } (- {}_{\otimes} -) = M \text{ in } N : \mathcal{R}\Gamma \vdash C}\ {\otimes}\text{-E}$$

$$\frac{M : \mathcal{P}\Gamma \vdash A \qquad N : \mathcal{Q}\Gamma \vdash B \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{(M {}_{\otimes} N) : \mathcal{R}\Gamma \vdash A \otimes B}\ {\otimes}\text{-I} \qquad \frac{M : \mathcal{P}\Gamma \vdash 0 \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{\text{ex-falso}\, M : \mathcal{R}\Gamma \vdash C}\ 0\text{-E}$$

$$\frac{M : \mathcal{P}\Gamma \vdash A \oplus B \qquad N : \mathcal{Q}\Gamma, 1A \vdash C \qquad O : \mathcal{Q}\Gamma, 1B \vdash C \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{\text{case } M \text{ of inj}_L - \mapsto N \text{ ; inj}_R - \mapsto O : \mathcal{R}\Gamma \vdash C}\ {\oplus}\text{-E}$$

$$\frac{M : \mathcal{R}\Gamma \vdash A}{\text{inj}_L M : \mathcal{R}\Gamma \vdash A \oplus B}\ {\oplus}\text{-I}\textsc{l} \qquad \frac{M : \mathcal{R}\Gamma \vdash B}{\text{inj}_R M : \mathcal{R}\Gamma \vdash A \oplus B}\ {\oplus}\text{-I}\textsc{r} \qquad \frac{}{({}_{\&}) : \mathcal{R}\Gamma \vdash \top}\ \top\text{-I}$$

$$\frac{M : \mathcal{R}\Gamma \vdash A \,\&\, B}{\text{proj}_L M : \mathcal{R}\Gamma \vdash A}\ \&\text{-E}\textsc{l} \qquad \frac{M : \mathcal{R}\Gamma \vdash A \,\&\, B}{\text{proj}_R M : \mathcal{R}\Gamma \vdash B}\ \&\text{-E}\textsc{r} \qquad \frac{M : \mathcal{R}\Gamma \vdash A \qquad N : \mathcal{R}\Gamma \vdash B}{(M {}_{\&} N) : \mathcal{R}\Gamma \vdash A \,\&\, B}\ \&\text{-I}$$

$$\frac{M : \mathcal{P}\Gamma \vdash !_{\rho} A \qquad N : \mathcal{Q}\Gamma, \rho A \vdash C \qquad \mathcal{R} \trianglelefteq \mathcal{P} + \mathcal{Q}}{\text{let } [-] = M \text{ in } N : \mathcal{R}\Gamma \vdash C}\ !_{\rho}\text{-E} \qquad \frac{M : \mathcal{P}\Gamma \vdash A \qquad \mathcal{R} \trianglelefteq \rho\mathcal{P}}{[M] : \mathcal{R}\Gamma \vdash !_{\rho} A}\ !_{\rho}\text{-I}$$

de Bruijn indices to represent variables, but we have annotated the rules with variable names for ease of reading. Ignoring the usages, the typing rules all look like their simply typed counterparts; the only difference between the $\otimes$ and $\&$ products being their presentation in terms of pattern matching and projections, respectively. Thus the addition of usage contexts and constraints on them refines the usual simple typing to be usage constrained. For instance, in the $\otimes$-I rule, the usage context $\mathcal{R}$ on the conclusion is constrained to be able to supply the sum $\mathcal{P} + \mathcal{Q}$ of the usage contexts of the premises. If we instantiate $\mathbf{R}$ to be the $0 \triangleright \omega \triangleleft 1$ semiring, then we obtain a system that is equivalent to Barber's DILL [Bar96].

4 Metatheory

McBride defines *kits* [McB05, BHKM12], which provide a general method for giving admissible rules that are usually proven by induction on the derivation. To produce a kit, we give an indexed family $\blacklozenge : \text{Ctx} \times \text{Ty} \to \text{Set}$ and explain how to inject variables, extract terms, and weaken by new variables coming from binders. In return, given a type-preserving map from variables in one context to $\blacklozenge$-stuff in another (an *environment*), we get a type-preserving function between terms in these contexts. Such a function is the intrinsic typing equivalent of an admissible rule.

To make the kit-based approach work in our usage-constrained setting, we make modifications to both kits and environments. Kits need not support arbitrary weakening, but only weakening by the introduction of 0-use variables. The family $\blacklozenge$ must also respect $\trianglelefteq$ of usage contexts. Environments are equipped with a matrix mapping input usages to output usages.

We prove simultaneous substitution using renaming. We take both renaming and substitution as corollaries of the *traversal* principle (Theorem 4.3) yielded from kits and environments.

Definition of *kit*s, *environment*s and traversal are in the module Specific.Syntax.Traversal.

4.1 Kits, Environments, and Traversals

A kit is a structure on $\vdash$-like relations $\blacklozenge$, intuitively giving a way in which $\blacklozenge$ lives between the usage-checked variable judgement $\sqsupseteq$ and the typing judgement $\vdash$. The components *vr* and *tm* are basically unchanged from McBride's original kits. The component *wk* only differs in that new variables are given annotation 0, which intuitively marks them as weakenable. The requirement *psh* is new, and allows us to fix up usage contexts via skew algebraic reasoning.

Definition 4.1 (Kit)**.** *For any* $\blacklozenge : \mathrm{Ctx} \times \mathrm{Ty} \to \mathrm{Set}$*, let* $\mathrm{Kit}(\blacklozenge)$ *denote the type of* kits. *A kit comprises the following functions for all* $\mathcal{P}$*,* $\mathcal{Q}$*,* Γ*,* Δ*, and* A*.*

$$psh : \mathcal{P} \trianglelefteq \mathcal{Q} \to \mathcal{Q}\Gamma \blacklozenge A \to \mathcal{P}\Gamma \blacklozenge A \qquad vr : \mathcal{P}\Gamma \sqsupseteq A \to \mathcal{P}\Gamma \blacklozenge A$$

$$tm : \mathcal{P}\Gamma \blacklozenge A \to \mathcal{P}\Gamma \vdash A \qquad wk : \mathcal{P}\Gamma \blacklozenge A \to \mathcal{P}\Gamma, \mathbf{0}\Delta \blacklozenge A$$

An inhabitant of $\mathcal{P}\Gamma \blacklozenge A$ is described as *stuff in* $\mathcal{P}\Gamma$ *of type* A.

Environments In simple intuitionistic type theory, an environment is a type-preserving function from variables in the old context Δ to stuff in the new context Γ: an inhabitant of $\Delta \ni A \to \Gamma \blacklozenge A$. The traversal function turns such an environment into a map between terms, $\Delta \vdash A \to \Gamma \vdash A$.

For $\lambda\mathcal{R}$, we want maps of usaged terms $\mathcal{Q}\Delta \vdash A \to \mathcal{P}\Gamma \vdash A$. We can see that an environment of type $\mathcal{Q}\Delta \sqsupseteq A \to \mathcal{P}\Gamma \blacklozenge A$ would be insufficient — $\mathcal{Q}\Delta \sqsupseteq A$ can only be inhabited when $\mathcal{Q}$ is compatible with a basis vector, so our environment would be trivial in more general cases. Instead, we care about non-usage-checked variables $\Delta \ni A$.

Our understanding of an environment is that it should simultaneously map all of the usage-checked variables in $\mathcal{Q}\Delta$ to stuff in $\mathcal{P}\Gamma$ in a way that preserves usage. As such, we want to map each variable $j : \Delta \ni A$ not to A-stuff in $\mathcal{P}\Gamma$, but rather A-stuff in $\mathcal{P}_j\Gamma$, where $\mathcal{P}_j$ is some fragment of $\mathcal{P}$. Precisely, when weighted by $\mathcal{Q}|j\rangle$, we want these $\mathcal{P}_j$ to sum to $\mathcal{P}$, so as to provide "enough" usage to cover all of the variables j. When we collect all of the $\mathcal{P}_j$ into a matrix Ψ, we notice that the condition just described is stated succinctly via a vector-matrix multiplication $\mathcal{Q}\Psi$. This culminates to give us the following:

Definition 4.2 (Env)**.** *For any* $\blacklozenge$*,* $\mathcal{P}$*,* $\mathcal{Q}$*,* Γ*, and* Δ*, where* Γ *and* Δ *have lengths* m *and* n *respectively, let* $\mathcal{P}\Gamma \overset{\blacklozenge}{\Rightarrow} \mathcal{Q}\Delta$ *denote the type of* environments. *An environment comprises a pair of a matrix* $\Psi : \mathbf{R}^{n\times m}$ *and a mapping of variables* $act : (j : (\Delta \ni A)) \to (\langle j|\Psi)\Gamma \blacklozenge A$*, such that* $\mathcal{P} \trianglelefteq \mathcal{Q}\Psi$*.*

Our main result is the following, which we will instantiate to prove admissibility of renaming (Corollary 4.6), subusaging (Corollary 4.7), and substitution (Corollary 4.9). The proof is in subsection 4.4.

Theorem 4.3 (traversal, trav)**.** *Given a kit on* $\blacklozenge$ *and an environment* $\mathcal{P}\Gamma \overset{\blacklozenge}{\Rightarrow} \mathcal{Q}\Delta$*, we get a function* $\mathcal{Q}\Delta \vdash A \to \mathcal{P}\Gamma \vdash A$*.*

4.2 Renaming

We now show how to use traversals to prove that renaming (including weakening) and subusaging are admissible. This subsection corresponds to the Agda module Specific.Syntax.Renaming.

Definition 4.4 (LVar-kit)**.** *Let $\ni$ -kit : $\mathrm{Kit}(\ni)$ be defined with the following fields.*

psh $(PQ : \mathcal{P} \trianglelefteq \mathcal{Q}) : \mathcal{Q}\Gamma \ni A \to \mathcal{P}\Gamma \ni A$**:** *The only occurrence of the usage context $\mathcal{Q}$ in the definition of $\ni$ is to the left of a $\trianglelefteq$. Applying transitivity in this place gets us the required term.*

$vr : \mathcal{P}\Gamma \ni A \to \mathcal{P}\Gamma \ni A := \mathrm{id}$.

$tm : \mathcal{P}\Gamma \ni A \to \mathcal{P}\Gamma \vdash A := \textbf{VAR}$.

$wk : \mathcal{P}\Gamma \ni A \to \mathcal{P}\Gamma, \mathbf{0}\Delta \ni A$**:** *A basis vector extended by 0s is still a basis vector: if that we have $\mathcal{P} \trianglelefteq \langle i|$ for some i, we also have $\mathcal{P}, \mathbf{0} \trianglelefteq \langle \mathrm{inl}\, i|$.*

Environments for renamings are special in that the matrix Ψ can be calculated from the action of the renaming on non-usage-checked variables.

Lemma 4.5 (ren-env)**.** *Given a type-preserving mapping of plain variables $f : \Delta \ni A \to \Gamma \ni A$ such that $\mathcal{P} \trianglelefteq \mathcal{Q}I_{f\times\mathrm{id}}$, we can produce a $\ni$-environment of type $\mathcal{P}\Gamma \overset{\ni}{\Rightarrow} \mathcal{Q}\Delta$.*

Proof. The environment has $\Psi := I_{f\times\mathrm{id}}$, so the usage condition holds by assumption. Now, *act* is required to have type $(j : \Delta \ni A) \to (\langle j|\Psi)\Gamma \ni A$. Take arbitrary $j : \Delta \ni A$. Then, we have $f\ j : \Gamma \ni A$, so all that is left is to show that $f\ j$ forms a usage-checked variable of type $(\langle j|\Psi)\Gamma \ni A$. This amounts to proving $\langle j|\Psi \trianglelefteq \langle f\ j|$. Let $i : \Gamma \ni A$, then we have $(\langle j|\Psi)_i \trianglelefteq \Psi_{j,i} = I_{f\,j,i} = \langle f\ j|_i$. □

Corollary 4.6 (renaming, ren)**.** *Given a type-preserving mapping of plain variables $f : \Delta \ni A \to \Gamma \ni A$ such that $\mathcal{P} \trianglelefteq \mathcal{Q}I_{f\times\mathrm{id}}$, we can produce a function of type $\mathcal{Q}\Delta \vdash A \to \mathcal{P}\Gamma \vdash A$.*

Corollary 4.7 (subusaging, subuse)**.** *Given $\mathcal{P} \trianglelefteq \mathcal{Q}$, then we have a function $\mathcal{Q}\Gamma \vdash A \to \mathcal{P}\Gamma \vdash A$.*

4.3 Substitution

Now that we have renaming, we can use it with traversals to prove that simultaneous well-usaged substitution is admissible. This subsection corresponds to the Agda module Specific.Syntax.Substitution.

Definition 4.8 (Tm-kit)**.** *Let $\vdash$ -kit : $\mathrm{Kit}(\vdash)$ be defined with the following fields.*

psh $(PQ : \mathcal{P} \trianglelefteq \mathcal{Q}) : \mathcal{Q}\Gamma \vdash A \to \mathcal{P}\Gamma \vdash A$**:** *This is Corollary 4.7 (subusaging).*

$vr : \mathcal{P}\Gamma \ni A \to \mathcal{P}\Gamma \vdash A := \textbf{VAR}$.

$tm : \mathcal{P}\Gamma \vdash A \to \mathcal{P}\Gamma \vdash A := \mathrm{id}$.

$wk : \mathcal{P}\Gamma \vdash A \to \mathcal{P}\Gamma, \mathbf{0}\Delta \vdash A$**:** *We use Corollary 4.6 (renaming), with $f : \Gamma \ni A \to \Gamma, \Delta \ni A$ being the embedding* inl*. It remains to check that $(\mathcal{P}, \mathbf{0}) \trianglelefteq \mathcal{P}I_{\mathrm{inl}\times\mathrm{id}}$. We prove this pointwise. Let $i : \Gamma, \Delta \ni A$, and take cases on whether i is from Γ or from Δ. If $i = \mathrm{inl}\, i'$ for an $i' : \Gamma \ni A$, we must show that $\mathcal{P}_{i'} \trianglelefteq (\mathcal{P}I_{\mathrm{inl}\times\mathrm{id}})_{\mathrm{inl}\, i'}$. But we have the following.*

$$\mathcal{P}_{i'} \trianglelefteq (\mathcal{P}I)_{i'} = \sum_{j:\Gamma\ni A} \mathcal{P}_j I_{j,i'} = \sum_{j:\Gamma\ni A} \mathcal{P}_j I_{\mathrm{inl}\,j,\mathrm{inl}\,i'} = (\mathcal{P}I_{\mathrm{inl}\times\mathrm{id}})_{\mathrm{inl}\,i'}.$$

If $i = \mathrm{inr}\, i'$ for an $i' : \Delta \ni A$, we must show that $0 \trianglelefteq (\mathcal{P}I_{\mathrm{inl}\times\mathrm{id}})_{\mathrm{inr}\,i'}$. But we have the following.

$$0 \trianglelefteq (\mathcal{P}\mathbf{0})_{i'} = \sum_{j:\Gamma\ni A} \mathcal{P}_j \mathbf{0}_{j,i'} = \sum_{j:\Gamma\ni A} \mathcal{P}_j I_{\mathrm{inl}\,j,\mathrm{inr}\,i'} = (\mathcal{P}I_{\mathrm{inl}\times\mathrm{id}})_{\mathrm{inr}\,i'}.$$

Corollary 4.9 (substitution, sub)**.** *Given an environment of type $\mathcal{P}\Gamma \overset{\vdash}{\Rightarrow} \mathcal{Q}\Delta$ (i.e., a well-usaged simultaneous substitution), we get a function of type $\mathcal{Q}\Delta \vdash A \to \mathcal{P}\Gamma \vdash A$.*

4.4 Proof of traversal

The proof of the traversal theorem follows the same structure as in McBride's article, extended with proof obligations to show that we are correctly respecting the usage annotations. We must first prove a lemma that shows that environments can be pushed under binders.

Lemma 4.10 (bind, bind)**.** *Given a kit on* $\blacklozenge$*, we can extend an environment of type* $\mathcal{P}\Gamma \xRightarrow{\blacklozenge} \mathcal{Q}\Delta$*, to an environment of type* $\mathcal{P}\Gamma, \mathcal{R}\Theta \xRightarrow{\blacklozenge} \mathcal{Q}\Delta, \mathcal{R}\Theta$.

Proof. Let the environment we are given be $(\Psi : \mathbf{R}^{n\times m}, act : (j : \Delta \ni A) \to (\langle j|\Psi)\Gamma \blacklozenge A)$, with $\mathcal{P} \trianglelefteq \mathcal{Q}\Psi$. We are trying to construct $(\Psi' : \mathbf{R}^{(n+o)\times(m+o)}, act' : (j : \Delta, \Theta \ni A) \to (\langle j|\Psi')(\Gamma, \Theta) \blacklozenge A)$, with $\mathcal{P}, \mathcal{R} \trianglelefteq (\mathcal{Q}, \mathcal{R})\Psi'$. Let $\Psi' := \left(\begin{array}{c|c} \Psi & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right)$. With this definition, our required condition splits into the easily checked conditions $\mathcal{P} \trianglelefteq \mathcal{Q}\Psi + \mathcal{R}\mathbf{0}$ and $\mathcal{R} \trianglelefteq \mathcal{Q}\mathbf{0} + \mathcal{R}\mathbf{I}$. For act', we take cases on whether j is from Δ or from Θ. In the Δ case, act gets us an inhabitant of $(\langle j|\Psi)\Gamma \blacklozenge A$. Notice that $\langle j|\Psi' = \langle j|\Psi, \mathbf{0}$, so we want to get from $(\langle j|\Psi)\Gamma \blacklozenge A$ to $(\langle j|\Psi)\Gamma, \mathbf{0}\Theta \blacklozenge A$. We can get this using *wk* from our kit. In the Θ case, notice that $\langle j|\Psi' = \mathbf{0}, \langle j|$. In other words, $\langle j|\Psi'$ is a basis vector, so we actually have usage-checked $(\langle j|\Psi')(\Gamma, \Theta) \ni A$. Thus, we can use *vr* from our kit to get $(\langle j|\Psi')(\Gamma, \Theta) \blacklozenge A$, as required. □

Theorem 4.3 (traversal, trav)**.** *Given a kit on* $\blacklozenge$ *and an environment* $\mathcal{P}\Gamma \xRightarrow{\blacklozenge} \mathcal{Q}\Delta$*, we get a function* $\mathcal{Q}\Delta \vdash A \to \mathcal{P}\Gamma \vdash A$.

Proof. By induction on the syntax of M. In the VAR x case, where $x : \mathcal{Q}\Delta \ni A$: By definition of $\ni$, we have that $\mathcal{Q} \trianglelefteq \langle j|$ for some j. Applying the action of the environment, we have $(\langle j|\Psi)\Gamma \blacklozenge A$. We then have $\mathcal{P} \trianglelefteq \mathcal{Q}\Psi \trianglelefteq \langle j|\Psi$, so using the fact that stuff appropriately respects subusaging (*psh*), we have $\mathcal{P}\Gamma \blacklozenge A$. Finally, using *tm*, we get a term $\mathcal{P}\Gamma \vdash A$, as required.

Non-VAR cases are generally handled in the following way. If the input usage context $\mathcal{Q}$ is split up into a linear combination of zero or more usage contexts $\mathcal{Q}_i$, obtain a similar splitting of $\mathcal{P}$ by setting $\mathcal{P}_i := \mathcal{Q}_i\Psi$. This works out because of the linearity of matrix multiplication (in particular, multiplication respects operations on the left). This yields environments of type $\mathcal{P}_i\Gamma \xRightarrow{\blacklozenge} \mathcal{Q}_i\Delta$ for the subterms to use with the inductive hypothesis. If any subterms bind variables, apply Lemma 4.10 as appropriate. □

5 Conclusion

We have extended McBride's method of kits and traversals to proving admissibility of renaming, subusaging and substitution to the usage-annotated calculus $\lambda\mathcal{R}$. In doing so, we have discovered that only skew semirings are required, and the importance of linear algebra for stating and proving these results. Though we have not had space to elaborate here, $\lambda\mathcal{R}$ is capable of representing several well known linear and modal type theories by choice of semiring.

Our work is similar in spirit to the work of Licata, Shulman, and Riley [LSR17], which gives a proof of cut elimination for a large class of substructural single-conclusion sequent calculi. The class of natural deduction systems we consider here is likely less general, but not directly comparable. We leave a complete comparison to future work. They have not formalised their work.

We plan to build on our work to generalise the framework of Allais *et al.* [AAC+20] to include usage annotations, allowing generic metatheory and semantics for an even wider class of substructural calculi.

Acknowledgements We are thankful for comments from Guillaume Allais.

References

[AAC+20] Guillaume Allais, Robert Atkey, James Chapman, Conor McBride & James McKinna (2020): *A Type and Scope Safe Universe of Syntaxes with Binding: Their Semantics and Proofs*. Accepted for JFP.

[All18] Guillaume Allais (2018): *Typing with Leftovers - A mechanization of Intuitionistic Multiplicative-Additive Linear Logic*. In: *TYPES 2017*, pp. 1:1–1:22, doi:10.4230/LIPIcs.TYPES.2017.1.

[Bar96] Andrew Barber (1996): *Dual Intuitionistic Linear Logic*. Technical Report, University of Edinburgh.

[BBPH93] P. N. Benton, Gavin M. Bierman, Valeria de Paiva & Martin Hyland (1993): *A Term Calculus for Intuitionistic Linear Logic*. In: *Typed Lambda Calculi and Applications*, *LNCS* 664, Springer, pp. 75–90, doi:10.1007/BFb0037099.

[BGMZ14] A. Brunel, M. Gaboardi, D. Mazza & S. Zdancewic (2014): *A Core Quantitative Coeffect Calculus*. In: *ESOP 2014*, pp. 351–370.

[BHKM12] Nick Benton, Chung-Kil Hur, Andrew Kennedy & Conor McBride (2012): *Strongly typed term representations in Coq*. *J. of Autom Reasoning* 49(2), doi:10.1007/s10817-011-9219-0.

[CLR19] Kaustuv Chaudhuri, Leonardo Lima & Giselle Reis (2019): *Formalized meta-theory of sequent calculi for linear logics*. *Theor. Comput. Sci.* 781, pp. 24–38, doi:10.1016/j.tcs.2019.02.023.

[Cra10] Karl Crary (2010): *Higher-Order Representation of Substructural Logics*. *SIGPLAN Not.* 45(9), p. 131–142, doi:10.1145/1932681.1863565.

[GS14] Dan R. Ghica & Alex I. Smith (2014): *Bounded Linear Types in a Resource Semiring*. In: *ESOP 2014*, pp. 331–350.

[Lau18] Olivier Laurent (2018): *Preliminary Report on the Yalla Library*. Available at `https://perso.ens-lyon.fr/olivier.laurent/yalla/`. Coq Workshop.

[LSR17] Daniel R. Licata, Michael Shulman & Mitchell Riley (2017): *A Fibrational Framework for Substructural and Modal Logics*. In: *FSCD 2017*, pp. 25:1–25:22, doi:10.4230/LIPIcs.FSCD.2017.25.

[McB05] Conor McBride (2005): *Type-preserving renaming and substitution*. Available at `http://www.strictlypositive.org/ren-sub.pdf`.

[OLE19] Dominic A. Orchard, Vilem Liepelt & Harley Eades (2019): *Quantitative program reasoning with graded modal types*. *Proceedings of the ACM on Programming Languages* 3.

[PD99] Frank Pfenning & Rowan Davies (1999): *A Judgmental Reconstruction of Modal Logic*. In: *Mathematical Structures in Computer Science*, p. 2001.

[POM14] Tomas Petricek, Dominic A. Orchard & Alan Mycroft (2014): *Coeffects: a calculus of context-dependent computation*. In: *ICFP 2014*, pp. 123–135.

[PW99] James Power & Caroline Webster (1999): *Working with Linear Logic in Coq*. 12th International Conference on Theorem Proving in Higher Order Logics (Work-in-progress paper).

[RBPKV20] Arjen Rouvoet, Casper Bach Poulsen, Robbert Krebbers & Eelco Visser (2020): *Intrinsically-Typed Definitional Interpreters for Linear, Session-Typed Languages*. In: *CPP 2020*, pp. 284–298, doi:10.1145/3372885.3373818.

[RP10] J. Reed & B. C. Pierce (2010): *Distance Makes the Types Grow Stronger*. In P. Hudak & S. Weirich, editors: *ICFP 2010*, pp. 157–168.

[Szl12] Kornél Szlachányi (2012): *Skew-monoidal categories and bialgebroids*. *Advances in Mathematics* 231(3-4), pp. 1694–1730, doi:10.1016/j.aim.2012.06.027.

[Wad92] Philip Wadler (1992): *There's no substitute for linear logic*. In: *8th International Workshop on the Mathematical Foundations of Programming Semantics*.

[XORN17] Bruno Xavier, Carlos Olarte, Giselle Reis & Vivek Nigam (2017): *Mechanizing Focused Linear Logic in Coq*. In: *12th Workshop on Logical and Semantic Frameworks, with Applications*, *ENTCS* 338, Elsevier, pp. 219–236, doi:10.1016/j.entcs.2018.10.014.