Effects and Coeffects in Call-By-Push-Value (Extended Version)

ANONYMOUS AUTHOR(S)

Effect and coeffect tracking are a useful way to integrate many types of compile-time analysis, such as cost, liveness or dataflow, into a language's type system. However, their interactions with call-by-push-value (CBPV), a computational model useful in compilation for its isolation of effects and its ability to encompass both call-by-name and call-by-value computations, are still poorly understood. We present fundamental results about those interactions, in particular *effect* and *coeffect soundness*. The former asserts that our CBPV-with-effects system accurately predicts the effects that the program may trigger during execution, and the latter asserts that our CBPV-with-coeffects system accurately tracks the demands a program makes on its environment. We prove our results for a core CBPV calculus and maintain generality across a broad range of effects and coeffects, laying the groundwork for future investigation.

CCS Concepts: • **Theory of computation → Type theory**.

Additional Key Words and Phrases: Types,CBPV,Effects,Coeffects

ACM Reference Format:
Anonymous Author(s). 2023. Effects and Coeffects in Call-By-Push-Value (Extended Version) . *Proc. ACM Program. Lang.* 1, 1 (October 2023), 50 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Computations interact with the world in which they run. Sometimes the computation does something the world can observe, known as *effects* [Lucassen and Gifford 1988], and sometimes computations demands something the world must provide, known as *coeffects* [Brunel et al. 2014; Orchard and et al. 2022; Petricek et al. 2014]. For example, running a computation might take time (running time is an effect) and might require resources (using input parameters is a coeffect).

To statically track these behaviors, some programming languages abstractly incorporate effects and coeffects into the type system. Languages that track effects include OCaml 5 [Leroy et al. 2023], Koka [Leijen 2023] and the Verse functional logic language [Verse development team 2023]. Languages that track coeffects include Linear Haskell [Bernardy et al. 2017], which uses linear types for resource management, and Agda and Idris 2 [Brady 2021], which track whether arguments are relevant to computation or may be erased. The Granule language [Orchard et al. 2019] provides a general structure that tracks both effects and coeffects in the same system.

While effects and coeffects have been well explored as extensions to the simply-typed λ-calculus, there is little work that adapts these analyses to other computational frameworks. This is unfortunate, because while the λ-calculus captures the essence of pure functional programming, it falls short as a model of modern languages with their rich computational structure.

In this paper, we integrate effect and coeffect tracking with call-by-push-value (CBPV), a computational model designed to describe the semantics of effectful programs [Levy 1999, 2022]. More specifically, CBPV separates pure "values" from "computations," isolating effects in the latter and

2475-1421/2023/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

leading to a rich equational theory. Because of its precision, CBPV works well as a low-level language and is closely related to compiler intermediate representations [Garbuzov et al. 2018; Maurer et al. 2017; Rizkallah et al. 2018].

The distinction between values and computations allows CBPV to treat strict and nonstrict language features explicitly, enabling it to model both call-by-value and call-by-name languages with the same facility. With the addition of effects and coeffects, this distinction allows us to observe how evaluation order changes the way a program alters and makes demands on the world. Levy characterizes the difference between values and computations with the slogan: "a value *is*, a computation *does*." Our interpretation of this slogan is that only computations may contain effectful subcomponents—values must be pure throughout. Conversely, coeffects describe demands a program makes on its inputs, which are always values.

In CBPV, values and computations have separate types. Values are described by positive types A computations by negative types B. These two forms are connected via an adjunction: the thunk type $\mathbf{U}\,B$ embeds a suspended computation inside the language of values and the returner type $\mathbf{F}\,A$ threads values through computations. Due to the structure of the adjunction, the combination $\mathbf{U}\,(\mathbf{F}\,A)$ forms a monad and the combination $\mathbf{F}\,(\mathbf{U}\,B)$ forms a comonad.

The duality between values and computations is reflected in a duality between the structures that we employ to statically track effects and coeffects. For effects, we add effect information ϕ to the thunk type $\mathbf{U}_{\phi}\ B$, recording the latent effect of suspended computations. Similarly, to track coeffects, we add coeffect information q in the returner type $\mathbf{F}_{q}\ A$ recording demands placed on the returned value by subsequent computation. With this augmentation, the types $\mathbf{U}_{\phi}\ (\mathbf{F}\,A)$ and $\mathbf{F}_{q}\ (\mathbf{U}\,B)$ correspond to the graded monads and comonads associated with effect and coeffect tracking. By working in the context of CBPV we have access to the fine building blocks of computation.

Following this duality, this paper comes in two mirrored halves. The first part (Section 2) extends CBPV with effect tracking and shows how we can recover the graded monad by grading the thunk type with latent effects. The second part (Section 3) extends CBPV with coeffect tracking and recovers a graded comonad by grading the returner type with latent coeffects. For clarity, we consider the two extensions separately, but, although we do not do so here, they may be combined together into the same system. However, by presenting these extensions side-by-side, we better understand each of them through comparison.

More specifically, we present the following results:

- We prove *effect soundness* for our effect-annotated extension of CBPV: that the type-and-effect system accurately bounds what happens at runtime. To do so, we define an environment-based big-step operational semantics for CBPV instrumented to precisely tracks effects during evaluation and use a logical relation to prove our soundness theorem. (Section 2.3)
- We prove that the standard translations from call-by-value (CBV) and call-by-name (CBN) lambda calculi to CBPV are *type-and-effect preserving*. Starting with a well-typed program, we can produce a well-typed CBPV program. Our effectful CBN language uses a graded monad to encapsulate effects, which we translate to the monadic type found in CBPV. (Section 2.4)
- We prove *coeffect soundness* for our coeffect-annotated extension of CBPV: that the statically tracked coeffects accurately bound the demands that the program makes on its inputs. We do so using an environment-based big-step operational semantics for CBPV, where the environment has been instrumented track coeffects during evaluation. (Section 3.1)
- While our our coeffect-tracking operational semantics is generalized over any coeffect structure, we observe that its behavior is not useful for reasoning about resource usage. Therefore,

we modify the rules of our semantics so that it does not use resources for discarded values, providing a more accurate model of the computation we would like to reason about. (Section 3.2)

- We prove that the standard translations from both CBN and CBV to CBPV are *type-and-coeffect* preserving. Starting with a well-typed CBN or CBV program, we can produce a well-typed CBPV program. We also consider variations of both systems in which comonads isolate coeffects and show how those variants can use the comonadic type in CBPV. (Section 3.3)
- CBPV augmented with coeffects allows us to compare the duality between values and computations with the duality between shared and disjoint resources. We observe that these two notions do not need to align, and explore two new forms of products that are available in this context. (Section 3.4)

Our effect type system for CBPV is similar to type systems found in prior work [Forster et al. 2017]. However, all other definitions and results of this paper are novel. In particular, we have found little work that explores the interaction between CBPV and coeffects. Furthermore, while we deliberately use the standard translations to interpret CBV and CBN in CBPV, designing the effect and coeffect systems so that these translations work naturally is part of the contribution of this paper. Our approach to reasoning about type, effect, and coeffect soundness is also new—we employ a novel environment-based big-step semantics for CBPV that leads to simple proofs of these results.

This paper is the product of multiple collaborators. The results in the effect section of this paper have been completely formalized in Agda and are available as supplementary material. We include footnotes with each result specifying the relevant theorems in the Agda code. The results of the coeffect portion have been typeset and appear inline.

2 CALL-BY-PUSH-VALUE (CBPV) AND EFFECT TRACKING

In this section, we extend the type system of CBPV with effect tracking. Our modifications to the base system, which are limited to reasoning about effect annotations ϕ, are marked in red.

CBPV syntactically separates terms into *values* V, inhabiting positive types A, and *computations* M, inhabiting negative types B, as shown by the following grammar.

$$\begin{array}{llll}
\textit{value types} & A & ::= & \texttt{unit} \mid \mathbf{U}_{\phi}\ B \mid A_1 \times A_2 \mid A_1 + A_2 \\
\textit{computation types} & B & ::= & A \rightarrow B \mid \mathbf{F}\, A \mid B_1 \,\&\, B_2 \\
 & & & \\
\textit{values} & V & ::= & x \mid () \mid \{M\} \mid (V_1, V_2) \mid \mathbf{inl}\ V \mid \mathbf{inr}\ V \\
\textit{computations} & M & ::= & \lambda x.M \mid M\ V \mid V! \mid \mathbf{case}\ V\ \mathbf{of}\ (x_1, x_2)\ \rightarrow\ N \mid \langle M_1, M_2 \rangle \mid M.1 \mid M.2 \\
 & & \mid & \mathbf{return}\ V \mid x \leftarrow M\ \mathbf{in}\ N \mid \mathbf{tick} \\
 & & \mid & V; M \mid \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2
\end{array}$$

Values in CBPV generally correspond to the values found in a call-by-value typed functional language, and include unit, positive products and sum values. Variables always represent values, so they are always declared with value types in the context. CBPV values also include the unit value $()$, pairs (V_1, V_2) and suspended computations, called *thunks*, and written $\{M\}$.

Computations in CBPV include abstractions ($\lambda x.M$) and applications ($M\ V$ as well as the forcing of thunks ($V!$) and pattern matching for value pairs ($\mathbf{case}\ V\ \mathbf{of}\ (x_1, x_2)\ \rightarrow\ N$). In addition to positive products, CBPV also includes negative products, of type $B_1 \,\&\, B_2$. These are introduced by a pair of computations $\langle M_1, M_2 \rangle$ and eliminated by projecting either the first or second component (*i.e.* $M.1$ and $M.2$).

Computations manipulate values through **return** V, and through sequencing with the "letin" computation. The latter, written $x \leftarrow M\ \mathbf{in}\ N$, orders computations, where the second may depend on the value returned by the first.

One of the advantages of CBPV is that it is readily extensible with effectful language features. Levy [Levy 2001, 2006, 2022] demonstrates how to add nontermination, nondeterminism, errors, I/O, state and control effects to CBPV. In each case, he extends the language with new computations and modifies the operational semantics to account for the new features.

For concreteness, we describe a single effect in this paper, the **tick** computation. This effect advances a virtual clock in the operational semantics.

2.1 CBPV: Type-and-effect system

Our type-and-effect system for CBPV is shown in in Figure 1. Under some typing context Γ, this system assigns a value type to values, $\Gamma \vdash_{\mathit{eff}} V : A$, and both a computation type and effect to computations, $\Gamma \vdash_{\mathit{eff}} M :^{\phi} B$. The effect annotation ϕ is an upper bound on the effects that could occur during evaluation of M. The judgement for values does not need an effect annotation because values are pure. The latent effect of suspended computations are recorded in the thunk type $\mathbf{U}_{\phi}\ B$ using rule EFF-THUNK.

To statically bound the number of **tick**s that will be evaluated, we extend the type system with a primitive effect *tock*, produced by the effectful **tick** computation, and structure the rest of the type system to count *tock*s. For simplicity, **tick** is the *only* effectful computation that this system includes. However, following Katsumata [2014], the only part of the system that is specific to time tracking are the rules for **tick** itself. All other rules are presented generically and are adaptable to other effects.

By tracking the *tock* effect, this type system performs cost analysis. For example, the type system tells us that each of the computations $x \leftarrow \mathbf{tick}\ \mathbf{in}\ \mathbf{tick}$ and $\langle \mathbf{tick}, y \leftarrow \mathbf{tick}\ \mathbf{in}\ \mathbf{tick} \rangle$ advances the clock at most twice. In the second case, if the first component of the pair is projected, the type system overapproximates the effect produced during execution.

Our system models effects using an arbitrary *preordered monoid* to preserve generality. This gives us an identity element $\perp$, an associative combining operation $\phi_1 \cdot \phi_2$, and a preorder relation $\leq_{\mathit{eff}}$ that respects the operation. For cost analysis, this monoid is the natural numbers with their usual ordering; the identity element is 0; the combining operation is addition, and the *tock* effect is the number 1. To modify the type system to track other effects, we need only update our monoid and preorder accordingly.

The general rules require a monoid structure because we need an associative notion of effect composition. In the $x \leftarrow M\ \mathbf{in}\ N$ computation, both M and N have effects. Therefore, we need a way to describe the effect of the entire computation and a notion of identity for that composition (in case one of the computations has no effect).

We require the preorder $\phi_1 \leq_{\mathit{eff}} \phi_2$ to allow for imprecision. In a program with branching, different branches may have different effects. That is, an effect annotation ϕ on the type of a program indicates that the program will have *at most* ϕ as its effect; it may have less. Choosing the discrete ordering (equality) forces the type system to track effects precisely.

This type system is syntax directed but admits a subeffecting property. If the type system determines that the computation will complete within 5 ticks, it is also sound, but less precise, for it to say that it will complete within 7 ticks.

LEMMA 2.1 (SUBEFFECTING FOR CBPV). *If* $\Gamma \vdash_{\mathit{eff}} M :^{\phi_1} B$ *and* $\phi_1 \leq_{\mathit{eff}} \phi_2$ *then* $\Gamma \vdash_{\mathit{eff}} M :^{\phi_2} B$.[1]

[1] CBPV/Effects/SyntacticTyping.agda: type-subeff

$\boxed{\Gamma \vdash_{\mathit{eff}} V : A}$ *(value effect typing)*

$$\frac{x : A \in \Gamma}{\Gamma \vdash_{\mathit{eff}} x : A}\ \textsc{eff-var} \qquad \frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} \{M\} : \mathbf{U}_{\phi}\, B}\ \textsc{eff-thunk} \qquad \frac{}{\Gamma \vdash_{\mathit{eff}} () : \mathbf{unit}}\ \textsc{eff-unit} \qquad \frac{\Gamma \vdash_{\mathit{eff}} V_1 : A_1 \quad \Gamma \vdash_{\mathit{eff}} V_2 : A_2}{\Gamma \vdash_{\mathit{eff}} (V_1, V_2) : A_1 \times A_2}\ \textsc{eff-pair}$$

$$\frac{\Gamma \vdash_{\mathit{eff}} V : A_1}{\Gamma \vdash_{\mathit{eff}} \mathbf{inl}\, V : A_1 + A_2}\ \textsc{eff-inl} \qquad \frac{\Gamma \vdash_{\mathit{eff}} V : A_2}{\Gamma \vdash_{\mathit{eff}} \mathbf{inr}\, V : A_1 + A_2}\ \textsc{eff-inr}$$

$\boxed{\Gamma \vdash_{\mathit{eff}} M :^{\phi} B}$ *(computation effect typing)*

$$\frac{\mathit{tock} \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} \mathbf{tick} :^{\phi} \mathbf{F}\,\mathbf{unit}}\ \textsc{eff-tick} \qquad \frac{\Gamma, x : A \vdash_{\mathit{eff}} M :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} \lambda x.M :^{\phi} A \to B}\ \textsc{eff-abs} \qquad \frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi} A \to B \quad \Gamma \vdash_{\mathit{eff}} V : A}{\Gamma \vdash_{\mathit{eff}} M\, V :^{\phi} B}\ \textsc{eff-app}$$

$$\frac{\Gamma \vdash_{\mathit{eff}} V : \mathbf{U}_{\phi_1}\, B \quad \phi_1 \leq_{\mathit{eff}} \phi_2}{\Gamma \vdash_{\mathit{eff}} V! :^{\phi_2} B}\ \textsc{eff-force} \qquad \frac{\Gamma \vdash_{\mathit{eff}} V : A \quad \bot \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} \mathbf{return}\, V :^{\phi} \mathbf{F}\, A}\ \textsc{eff-ret} \qquad \frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi_1} \mathbf{F}\, A \quad \Gamma, x : A \vdash_{\mathit{eff}} N :^{\phi_2} B \quad \phi_1 \cdot \phi_2 \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} x \leftarrow M\, \mathbf{in}\, N :^{\phi} B}\ \textsc{eff-letin}$$

$$\frac{\Gamma \vdash_{\mathit{eff}} V : A_1 \times A_2 \quad \Gamma, x_1 : A_1, x_2 : A_2 \vdash_{\mathit{eff}} N :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} \mathbf{case}\ V\ \mathbf{of}\ (x_1, x_2) \to N :^{\phi} B}\ \textsc{eff-split} \qquad \frac{\Gamma \vdash_{\mathit{eff}} M_1 :^{\phi} B_1 \quad \Gamma \vdash_{\mathit{eff}} M_2 :^{\phi} B_2}{\Gamma \vdash_{\mathit{eff}} \langle M_1, M_2 \rangle :^{\phi} B_1 \,\&\, B_2}\ \textsc{eff-cpair} \qquad \frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi} B_1 \,\&\, B_2}{\Gamma \vdash_{\mathit{eff}} M.1 :^{\phi} B_1}\ \textsc{eff-fst}$$

$$\frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi} B_1 \,\&\, B_2}{\Gamma \vdash_{\mathit{eff}} M.2 :^{\phi} B_2}\ \textsc{eff-snd}$$

$$\frac{\Gamma \vdash_{\mathit{eff}} V : \mathbf{unit} \quad \Gamma \vdash_{\mathit{eff}} N :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} V; N :^{\phi} B}\ \textsc{eff-sequence} \qquad \frac{\Gamma \vdash_{\mathit{eff}} V : A_1 + A_2 \quad \Gamma, x_1 : A_1 \vdash_{\mathit{eff}} M_1 :^{\phi} B \quad \Gamma, x_2 : A_2 \vdash_{\mathit{eff}} M_2 :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\, x_1 \to M_1; \mathbf{inr}\, x_2 \to M_2 :^{\phi} B}\ \textsc{eff-case}$$

Fig. 1. CBPV typing and effect tracking

To make sure that subeffecting is admissible, several rules in the type system include a constraint that allows for weakening of the effect on the judgement. For example, rule EFF-TICK asserts that the **tick** computation advances the clock at least once, *i.e.* has any effect greater than or equal to a single *tock*. Rule EFF-FORCE allows any effect that is greater than or equal to the latent effect of the suspended computation. Similarly, in rule EFF-RET, returning a value has no effect, so ϕ may be any effect greater than or equal to $\bot$.

Unlike in effect systems for the λ-calculus, the latent effects of function bodies are not recorded in function types. Instead, they are propagated to the conclusion of rule EFF-ABS. This makes sense because abstractions are not values in CBPV. From an operational sense, they are computations that pop the argument off the stack before continuing.

Rule EFF-LETIN type checks sequences of computations. In this rule, both premises are always evaluated, in the order specified by the syntax. As a result, the typing rule for this computation calculates the resulting effect using the combining operation, $\phi_1 \cdot \phi_2$, which then may be coarsened.

2.2 Instrumented operational semantics and effect soundness

We next define a big-step, *environment-based* semantics for CBPV. This semantics is new but straightforward. Past presentations of CBPV define its operational behavior using small-step, big-step or stack-based semantics, but always use substitution [Levy 2022]. We choose this form of semantics for two reasons. First, the big-step structure corresponds closely to the structure of the type system. Used with the environment semantics, which eliminates the need for substitution lemmas, our soundness proofs is remarkably straightforward (Section 2.3). Second, the environment lets us track the demands that computations make on their inputs in our coeffect soundness proof (Section 3.1). For example, with a resource usage, we can include annotations that count how many times the program accesses each variable during computation. A substitution-based semantics does not support this instrumentation.

Figure 2 shows the definition of the operational semantics. This semantics consists of two mutually defined relations. The first, $\rho \vdash V \Downarrow W$, shows how a value uses the provided environment ρ to "evaluate" to a closed value W. This operation resembles a substitution operation in that it replaces each variable found in the value with its definition in the environment.

The second relation, $\rho \vdash_{\mathit{eff}} M \Downarrow T \,\#\, \phi$, shows how computations evaluate to *closed terminal computations*, T, or computations that have no free variables and cannot step further. The effect annotation ϕ on this relation counts the number of ticks that occur during evaluation of M.

Environments ρ are mappings from variables to *closed values*, W, and can be thought of as delayed substitutions. Closed values include closures, *i.e.* suspended computations paired with closing environments, as well as unit, positive products and sums of closed values. Terminals include returned (closed) values and closures for suspended abstractions and pairs.

$$
\begin{array}{llcl}
\textit{closed values} & W & ::= & \mathbf{clo}(\rho, \{M\}) \mid () \mid (W_1, W_2) \mid \mathbf{inl}\ W \mid \mathbf{inr}\ W \\
\textit{environments} & \rho & ::= & \emptyset \mid \rho\,, x \mapsto W \\
\textit{closed terminals} & T & ::= & \mathbf{return}\ W \mid \mathbf{clo}(\rho, \lambda x.M) \mid \mathbf{clo}(\rho, \langle M_1, M_2 \rangle)
\end{array}
$$

The operational semantics of the **tick** computation is trivial (it merely produces a unit value and a single *tock* effect). Other rules either assert that they produce no effect (such as rule EVAL-EFF-COMP-ABS) or combine the effects of their subcomponents (such as rule EVAL-EFF-COMP-APP-ABS). As in the type-and-effect system, the only rule that is specific to the *tock* effect is the rule for **tick**. All other effects in these rules are specified using the generic monoid structure.

While the type system allows for imprecision, the operational semantics precisely tracks the effects of computation. Each evaluation results in exactly one terminal and one computed effect.

THEOREM 2.2 (DETERMINISM). *If $\rho \vdash_{\mathit{eff}} M \Downarrow T \,\#\, \phi$ and $\rho \vdash_{\mathit{eff}} M \Downarrow T' \,\#\, \phi'$, then $T = T'$ and $\phi = \phi'$.*[2]

2.3 Type and effect soundness

We state our type and effect soundness theorem as follows: closed, well-typed computations of type $\mathbf{F}\,A$ return closed values and produce effects that are bounded by the type system. Combined

[2] CBPV/Effects/Determinism.agda: determinism-comp

$\boxed{\rho \vdash V \Downarrow W}$ *(Value closing)*

$$\textsc{eval-val-var}\ \frac{x \mapsto W \in \rho}{\rho \vdash x \Downarrow W} \qquad \textsc{eval-val-unit}\ \frac{}{\rho \vdash () \Downarrow ()} \qquad \textsc{eval-val-thunk}\ \frac{}{\rho \vdash \{M\} \Downarrow \mathbf{clo}(\rho, \{M\})}$$

$$\textsc{eval-val-vpair}\ \frac{\rho \vdash V_1 \Downarrow W_1 \quad \rho \vdash V_2 \Downarrow W_2}{\rho \vdash (V_1, V_2) \Downarrow (W_1, W_2)}$$

$$\textsc{eval-val-inl}\ \frac{\rho \vdash V \Downarrow W}{\rho \vdash \mathbf{inl}\, V \Downarrow \mathbf{inl}\, W} \qquad \textsc{eval-val-inr}\ \frac{\rho \vdash V \Downarrow W}{\rho \vdash \mathbf{inr}\, V \Downarrow \mathbf{inr}\, W}$$

$\boxed{\rho \vdash_{\mathit{eff}} M \Downarrow T \# \phi}$ *(Computation rules)*

$$\textsc{eval-eff-comp-tick}\ \frac{}{\rho \vdash_{\mathit{eff}} \mathbf{tick} \Downarrow \mathbf{return}\, () \# \mathit{tock}} \qquad \textsc{eval-eff-comp-force-thunk}\ \frac{\rho \vdash V \Downarrow \mathbf{clo}(\rho', \{M\}) \qquad \rho' \vdash_{\mathit{eff}} M \Downarrow T \# \phi}{\rho \vdash_{\mathit{eff}} V! \Downarrow T \# \phi}$$

$$\textsc{eval-eff-comp-return}\ \frac{\rho \vdash V \Downarrow W}{\rho \vdash_{\mathit{eff}} \mathbf{return}\, V \Downarrow \mathbf{return}\, W \# \bot} \qquad \textsc{eval-eff-comp-letin-ret}\ \frac{\rho \vdash_{\mathit{eff}} M \Downarrow \mathbf{return}\, W \# \phi_1 \qquad \rho, x \mapsto W \vdash_{\mathit{eff}} N \Downarrow T \# \phi_2}{\rho \vdash_{\mathit{eff}} x \leftarrow M\, \mathbf{in}\, N \Downarrow T \# \phi_1 \cdot \phi_2}$$

$$\textsc{eval-eff-comp-split}\ \frac{\rho \vdash V \Downarrow (W_1, W_2) \quad \rho, x_1 \mapsto W_1, x_2 \mapsto W_2 \vdash_{\mathit{eff}} N \Downarrow T \# \phi}{\rho \vdash_{\mathit{eff}} \mathbf{case}\ V\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N \Downarrow T \# \phi}$$

$$\textsc{eval-eff-comp-app-abs}\ \frac{\rho \vdash_{\mathit{eff}} M \Downarrow \mathbf{clo}(\rho', \lambda x.M') \# \phi_1 \quad \rho \vdash V \Downarrow W \qquad \rho', x \mapsto W \vdash_{\mathit{eff}} M' \Downarrow T \# \phi_2}{\rho \vdash_{\mathit{eff}} M\, V \Downarrow T \# \phi_1 \cdot \phi_2}$$

$$\textsc{eval-eff-comp-abs}\ \frac{}{\rho \vdash_{\mathit{eff}} \lambda x.M \Downarrow \mathbf{clo}(\rho, \lambda x.M) \# \bot} \qquad \textsc{eval-eff-comp-cpair}\ \frac{}{\rho \vdash_{\mathit{eff}} \langle M_1, M_2 \rangle \Downarrow \mathbf{clo}(\rho, \langle M_1, M_2 \rangle) \# \bot}$$

$$\textsc{eval-eff-comp-fst}\ \frac{\rho \vdash_{\mathit{eff}} M \Downarrow \mathbf{clo}(\rho', \langle M_1, M_2 \rangle) \# \phi_1 \quad \rho' \vdash_{\mathit{eff}} M_1 \Downarrow T \# \phi_2}{\rho \vdash_{\mathit{eff}} M.1 \Downarrow T \# \phi_1 \cdot \phi_2} \qquad \textsc{eval-eff-comp-snd}\ \frac{\rho \vdash_{\mathit{eff}} M \Downarrow \mathbf{clo}(\rho', \langle M_1, M_2 \rangle) \# \phi_1 \quad \rho' \vdash_{\mathit{eff}} M_2 \Downarrow T \# \phi_2}{\rho \vdash_{\mathit{eff}} M.2 \Downarrow T \# \phi_1 \cdot \phi_2}$$

$$\textsc{eval-eff-comp-sequence}\ \frac{\rho \vdash V \Downarrow () \quad \rho \vdash_{\mathit{eff}} N \Downarrow T \# \phi}{\rho \vdash_{\mathit{eff}} V; N \Downarrow T \# \phi}$$

$$\textsc{eval-eff-comp-case-inl}\ \frac{\rho \vdash V \Downarrow \mathbf{inl}\, W \quad \rho, x_1 \mapsto W \vdash_{\mathit{eff}} M_1 \Downarrow T \# \phi}{\rho \vdash_{\mathit{eff}} \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\, x_1 \rightarrow M_1; \mathbf{inr}\, x_2 \rightarrow M_2 \Downarrow T \# \phi}$$

$$\textsc{eval-eff-comp-case-inr}\ \frac{\rho \vdash V \Downarrow \mathbf{inr}\, W \quad \rho, x_2 \mapsto W \vdash_{\mathit{eff}} M_2 \Downarrow T \# \phi}{\rho \vdash_{\mathit{eff}} \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\, x_1 \rightarrow M_1; \mathbf{inr}\, x_2 \rightarrow M_2 \Downarrow T \# \phi}$$

Fig. 2. Operational semantics of CBPV with effect tracking

with determinism, this result tells us that computations cannot go wrong (*i.e.* crash) or diverge. (It also implies that all computations terminate in this simple system.)

Theorem 2.3 (Effect soundness). *If* $\varnothing \vdash_{\mathit{eff}} M \;:^{\phi}\; \mathbf{F}\,A$ *then* $\emptyset \vdash_{\mathit{eff}} M \Downarrow \mathbf{return}\; W \,\#\, \phi'$ *where* $\phi' \leq_{\mathit{eff}} \phi$.[3]

The reason that this theorem is limited to type $\mathbf{F}\,A$ is because we do not assume $\bot \leq_{\mathit{eff}} \phi$ in the preordered monoid. As a result, at other types, the soundness theorem is more complex. For a general type B, we know that M will evaluate to some terminal T with effect ϕ' such that there is some ϕ'' where $\phi' \cdot \phi'' \leq_{\mathit{eff}} \phi$. This extra ϕ'' is the latent effect from the case where T is a closure.

The theorem above is a corollary of the fundamental theorem of the following logical relation. This relation consists of four mutual definitions: closed values $\mathcal{W}[\![A]\!]$, closed terminal computations $\mathcal{T}[\![B]\!]$, values paired with environments $\mathcal{V}[\![A]\!]$, and computations tupled with environments and effects $\mathcal{M}[\![B]\!]$.

Definition 2.4 (CBPV with Effects: Logical Relation).

$$
\begin{array}{lcll}
\mathcal{W}[\![\mathbf{U}_{\phi}\; B]\!] & = & \{\, \mathbf{clo}(\rho, \{M\}) & \mid (\rho, M, \phi) \in \mathcal{M}[\![B]\!] \} \\
\mathcal{W}[\![\mathbf{unit}]\!] & = & \{\, () \,\} & \\
\mathcal{W}[\![A_1 \times A_2]\!] & = & \{\, (W_1, W_2) & \mid W_1 \in \mathcal{W}[\![A_1]\!] \textit{ and } W_2 \in \mathcal{W}[\![A_2]\!] \} \\
\mathcal{W}[\![A_1 + A_2]\!] & = & \{\, \mathbf{inl}\; W & \mid W \in \mathcal{W}[\![A_1]\!] \} \cup \{\, \mathbf{inr}\; W \mid W \in \mathcal{W}[\![A_2]\!] \} \\
 & & & \\
\mathcal{T}[\![\mathbf{F}\,A]\!] & = & \{\, (\mathbf{return}\; W, \phi) & \mid W \in \mathcal{W}[\![A]\!] \textit{ and } \bot \leq_{\mathit{eff}} \phi \,\} \\
\mathcal{T}[\![A \to B]\!] & = & \{\, (\mathbf{clo}(\rho, \lambda x.M), \phi) & \mid \textit{for all } W \in \mathcal{W}[\![A]\!],\; ((\rho\,, x \mapsto W), M, \phi) \in \mathcal{M}[\![B]\!] \,\} \\
\mathcal{T}[\![B_1 \,\&\, B_2]\!] & = & \{\, (\mathbf{clo}(\rho, \langle M_1, M_2 \rangle), \phi) & \mid (\rho, M_1, \phi) \in \mathcal{M}[\![B_1]\!] \textit{ and } (\rho, M_2, \phi) \in \mathcal{M}[\![B_2]\!] \,\} \\
 & & & \\
\mathcal{V}[\![A]\!] & = & \{\, (\rho, V) & \mid \rho \vdash V \Downarrow W \textit{ and } W \in \mathcal{W}[\![A]\!] \,\} \\
\mathcal{M}[\![B]\!] & = & \{\, (\rho, M, \phi) & \mid \rho \vdash_{\mathit{eff}} M \Downarrow T \,\#\, \phi_1 \textit{ and } (T, \phi_2) \in \mathcal{T}[\![B]\!] \textit{ and } \phi_1 \cdot \phi_2 \leq_{\mathit{eff}} \phi \}
\end{array}
$$

We use this relation to define semantic typing for environments, values and computations.

Definition 2.5 (CBPV with Effects: Semantic Typing).

$$
\begin{array}{lcl}
\Gamma \vDash \rho & = & x : A \in \Gamma \textit{ implies } x \mapsto W \in \rho \textit{ and } W \in \mathcal{W}[\![A]\!] \\
\Gamma \vDash_{\mathit{eff}} V : A & = & \Gamma \vDash \rho \textit{ implies } (\rho, V) \in \mathcal{V}[\![A]\!] \\
\Gamma \vDash_{\mathit{eff}} M \;:^{\phi}\; B & = & \Gamma \vDash \rho \textit{ implies } (\rho, M, \phi) \in \mathcal{M}[\![B]\!]
\end{array}
$$

Using these definitions, we can prove semantic typing lemmas corresponding to each of the syntactic typing rules shown in Figure 1. With these lemmas, we can then prove the fundamental lemma for the logical relation with a straightforward induction, and the effect soundness theorem follows as a corollary.

Lemma 2.6 (Fundamental Lemma). (1) *If* $\Gamma \vdash_{\mathit{eff}} V : A$ *then* $\Gamma \vDash_{\mathit{eff}} V : A$.[4]
(2) *If* $\Gamma \vdash_{\mathit{eff}} M \;:^{\phi}\; B$ *then* $\Gamma \vDash_{\mathit{eff}} M \;:^{\phi}\; B$.[5]

2.4 Type-and-effect preserving translations

Levy [2006] provides translations from call-by-value (CBV) and call-by-name (CBN) λ-calculi to CBPV and shows that those translations preserve types, denotational semantics, and (substitution-based) big-step operational semantics. We show here that those translations also preserve effects.

[3] CBPV/Effects/EffectSoundness.agda: effect-soundness
[4] CBPV/Effects/EffectSoundness.agda: fundamental-lemma-val
[5] CBPV/Effects/EffectSoundness.agda: fundamental-lemma-comp

For the CBV translation, we start with a λ-calculus that has a simple type-and-effect system, loosely based on Lucassen and Gifford [1988]. However, as few CBN languages directly include effects, for the CBN translation we use as the source language a simply-typed λ-calculus that encapsulates effects using a *graded monad*. Furthermore, we show that we can also use this monad with the CBV translation because effects are encapsulated.

2.4.1 CBV type-and-effect system. The simple CBV language with effect tracking in this subsection features the same **tick** term and *tock* effect as before along with the usual forms of the λ-calculus.

$$\begin{array}{llll}
\textit{types} & \tau & ::= & \texttt{unit} \mid \tau_1 \xrightarrow{\phi} \tau_2 \mid \tau_1 \otimes \tau_2 \mid \tau_1 + \tau_2 \\
\textit{terms} & e & ::= & \textbf{tick} \mid x \mid \lambda x.e \mid e_1\ e_2 \mid ()\ e_1; e_2 \\
 & & \mid & (e_1, e_2) \mid (x_1, x_2) = e_1\ \textbf{in}\ e_2 \\
 & & \mid & \textbf{inl}\ e \mid \textbf{inr}\ e \mid \textbf{case}\ e\ \textbf{of}\ x_1 \longrightarrow e_1; x_2 \longrightarrow e_2
\end{array}$$

$\boxed{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau}$ *(STLC + effect typing)*

$$\textsc{lam-eff-var}\ \frac{x : \tau \in \Gamma \qquad \bot \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} x :^{\phi} \tau}
\qquad
\textsc{lam-eff-abs}\ \frac{\Gamma, x : \tau_1 \vdash_{\mathit{eff}} e :^{\phi'} \tau_2 \qquad \bot \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} \lambda x.e :^{\phi} \tau_1 \xrightarrow{\phi'} \tau_2}$$

$$\textsc{lam-eff-app}\ \frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \tau_1 \xrightarrow{\phi_3} \tau_2 \qquad \Gamma \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau_1 \qquad \phi_1 \cdot \phi_2 \cdot \phi_3 \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} e_1\ e_2 :^{\phi} \tau_2}$$

$$\textsc{lam-eff-unit}\ \frac{\bot \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} () :^{\phi} \textbf{unit}}
\qquad
\textsc{lam-eff-sequence}\ \frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \textbf{unit} \qquad \Gamma \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau \qquad \phi_1 \cdot \phi_2 \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} e_1; e_2 :^{\phi} \tau}$$

$$\textsc{lam-eff-pair}\ \frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \tau_1 \qquad \Gamma \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau_2 \qquad \phi_1 \cdot \phi_2 \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} (e_1, e_2) :^{\phi} \tau_1 \otimes \tau_2}$$

$$\textsc{lam-eff-split}\ \frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \tau_1 \otimes \tau_2 \qquad \Gamma, x_1 : \tau_1, x_2 : \tau_2 \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau \qquad \phi_1 \cdot \phi_2 \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} (x_1, x_2) = e_1\ \textbf{in}\ e_2 :^{\phi} \tau}$$

$$\textsc{lam-eff-inl}\ \frac{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau_1}{\Gamma \vdash_{\mathit{eff}} \textbf{inl}\ e :^{\phi} \tau_1 + \tau_2}
\qquad
\textsc{lam-eff-inr}\ \frac{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau_2}{\Gamma \vdash_{\mathit{eff}} \textbf{inr}\ e :^{\phi} \tau_1 + \tau_2}$$

$$\textsc{lam-eff-case}\ \frac{\Gamma \vdash_{\mathit{eff}} e :^{\phi_1} \tau_1 + \tau_2 \qquad \Gamma, x : \tau_1 \vdash_{\mathit{eff}} e_1 :^{\phi_2} \tau \qquad \Gamma, x : \tau_2 \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau \qquad \phi_1 \cdot \phi_2 \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} \textbf{case}\ e\ \textbf{of}\ x_1 \longrightarrow e_1; x_2 \longrightarrow e_2 :^{\phi} \tau}
\qquad
\textsc{lam-eff-tick}\ \frac{\mathit{tock} \leq_{\mathit{eff}} \phi}{\Gamma \vdash_{\mathit{eff}} \textbf{tick} :^{\phi} \textbf{unit}}$$

Function types, written $\tau_1 \xrightarrow{\phi} \tau_2$, are annotated with *latent effects*, which occur when the function is called. In the application rule rule LAM-EFF-APP, this latent effect is combined with ϕ_1, the effects that occur when evaluating the function e_1 to a λ expression and ϕ_2, the effects that occur when evaluating the argument to a value. As in CBPV, this system supports subeffecting, *i.e.* the analogue of Lemma 2.1[6].

[6] CBV/Effects/SyntacticTyping: type-subeff

The CBV type and term translations follow directly from Levy [2022]. Besides adding a case for the **tick** expression, the only change that we make is moving the latent effect from the function type to the thunk type. All other cases are exactly as in prior work.

Type translation

$$
\begin{aligned}
[\![\tau_1 \xrightarrow{\phi_1} \tau_2]\!]_{\mathrm{v}} &= \mathbf{U}_{\phi}\ ([\![\tau_1]\!]_{\mathrm{v}} \to \mathbf{F}\ [\![\tau_2]\!]_{\mathrm{v}}) \\
[\![\mathbf{unit}]\!]_{\mathrm{v}} &= \mathtt{unit} \\
[\![\tau_1 \otimes \tau_2]\!]_{\mathrm{v}} &= [\![\tau_1]\!]_{\mathrm{v}} \times [\![\tau_2]\!]_{\mathrm{v}} \\
[\![\tau_1 + \tau_2]\!]_{\mathrm{v}} &= [\![\tau_1]\!]_{\mathrm{v}} + [\![\tau_2]\!]_{\mathrm{v}}
\end{aligned}
$$

Context translation

$$
\begin{aligned}
[\![\varnothing]\!]_{\mathrm{v}} &= \varnothing \\
[\![\Gamma, x : \tau]\!]_{\mathrm{v}} &= [\![\Gamma]\!]_{\mathrm{v}}, x : [\![\tau]\!]_{\mathrm{v}}
\end{aligned}
$$

Term translation

$$
\begin{aligned}
[\![\mathbf{tick}]\!]_{\mathrm{v}} &= \mathbf{tick} \\
[\![x]\!]_{\mathrm{v}} &= \mathbf{return}\ x \\
[\![\lambda x.e]\!]_{\mathrm{v}} &= \mathbf{return}\ \{\lambda x.[\![e]\!]_{\mathrm{v}}\} \\
[\![e_1\ e_2]\!]_{\mathrm{v}} &= x \leftarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ y \leftarrow [\![e_2]\!]_{\mathrm{v}}\ \mathbf{in}\ x!\ y \\
[\![()]\!]_{\mathrm{v}} &= \mathbf{return}\ () \\
[\![e_1; e_2]\!]_{\mathrm{v}} &= x \leftarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ x; [\![e_2]\!]_{\mathrm{v}} \\
[\![(e_1, e_2)]\!]_{\mathrm{v}} &= x \leftarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ y \leftarrow [\![e_2]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}\ (x, y) \\
[\![(x_1, x_2) = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{v}} &= x \leftarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{case}\ x\ \mathbf{of}\ (x_1, x_2) \to [\![e_2]\!]_{\mathrm{v}} \\
[\![\mathbf{inl}\ e]\!]_{\mathrm{v}} &= x \leftarrow [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}\ (\mathbf{inl}\ x) \\
[\![\mathbf{inr}\ e]\!]_{\mathrm{v}} &= x \leftarrow [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}\ (\mathbf{inr}\ x) \\
[\![\mathbf{case}\ e\ \mathbf{of}\ x_1 \to e_1; x_2 \to e_2]\!]_{\mathrm{v}} &= x \leftarrow [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{case}\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \to [\![e_1]\!]_{\mathrm{v}}; \mathbf{inr}\ x_2 \to [\![e_2]\!]_{\mathrm{v}}
\end{aligned}
$$

This translation preserves types and effects from the source language.

LEMMA 2.7. *If* $\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau$ *then* $[\![\Gamma]\!]_V \vdash_{\mathit{eff}} [\![e]\!]_V :^{\phi} \mathbf{F}\ [\![\tau]\!]_V$.[7]

This result is easy to prove, reassuring us that our effect system design is correct: we can use CBPV to encode the well-studied type-and-effect systems developed over the past 40 years.

2.4.2 Graded monads. CBPV is designed to serve as a convenient translation target for both CBV and CBN languages. However, in CBN languages, effects are usually tracked using *graded monads* [Wadler and Thiemann 2003]. Therefore, here we translate a CBN language with graded monads to CBPV. Our source language for this translation is the simply-typed λ-calculus with unit, products and sums, together with a graded monadic type $\mathbf{T}_{\phi}\ \tau$, the monadic operations **return** and **bind**, and the tick operation. The **tick** operation has a monadic type, isolating its effects from the rest of the pure λ-calculus.

[7] CBV/Effects/Translation.agda: translation-preservation-exp

$\boxed{\Gamma \vdash_{mon} e : \tau}$ *(STLC + graded monad)*

$$\textsc{lam-mon-var}\ \frac{x : \tau \in \Gamma}{\Gamma \vdash_{mon} x : \tau} \qquad \textsc{lam-mon-abs}\ \frac{\Gamma , x : \tau_1 \vdash_{mon} e : \tau_2}{\Gamma \vdash_{mon} \lambda x.e : \tau_1 \to \tau_2}$$

$$\textsc{lam-mon-app}\ \frac{\Gamma \vdash_{mon} e_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash_{mon} e_2 : \tau_1}{\Gamma \vdash_{mon} e_1\ e_2 : \tau_2} \qquad \textsc{lam-mon-unit}\ \frac{}{\Gamma \vdash_{mon} () : \mathbf{unit}}$$

$$\textsc{lam-mon-sequence}\ \frac{\Gamma \vdash_{mon} e_1 : \mathbf{unit} \quad \Gamma \vdash_{mon} e_2 : \tau}{\Gamma \vdash_{mon} e_1; e_2 : \tau} \qquad \textsc{lam-mon-pair}\ \frac{\Gamma \vdash_{mon} e_1 : \tau_1 \quad \Gamma \vdash_{mon} e_2 : \tau_2}{\Gamma \vdash_{mon} (e_1, e_2) : \tau_1 \otimes \tau_2}$$

$$\textsc{lam-mon-split}\ \frac{\Gamma \vdash_{mon} e_1 : \tau_1 \otimes \tau_2 \quad \Gamma , x_1 : \tau_1 , x_2 : \tau_2 \vdash_{mon} e_2 : \tau}{\Gamma \vdash_{mon} (x_1, x_2) = e_1 \ \mathbf{in}\ e_2 : \tau}$$

$$\textsc{lam-mon-with}\ \frac{\Gamma \vdash_{mon} e_1 : \tau_1 \quad \Gamma \vdash_{mon} e_2 : \tau_2}{\Gamma \vdash_{mon} \langle e_1, e_2 \rangle : \tau_1 \,\&\, \tau_2} \qquad \textsc{lam-mon-fst}\ \frac{\Gamma \vdash_{mon} e : \tau_1 \,\&\, \tau_2}{\Gamma \vdash_{mon} e.1 : \tau_1}$$

$$\textsc{lam-mon-snd}\ \frac{\Gamma \vdash_{mon} e : \tau_1 \,\&\, \tau_2}{\Gamma \vdash_{mon} e.1 : \tau_2} \qquad \textsc{lam-mon-inl}\ \frac{\Gamma \vdash_{mon} e : \tau_1}{\Gamma \vdash_{mon} \mathbf{inl}\ e : \tau_1 + \tau_2}$$

$$\textsc{lam-mon-inr}\ \frac{\Gamma \vdash_{mon} e : \tau_2}{\Gamma \vdash_{mon} \mathbf{inr}\ e : \tau_1 + \tau_2}$$

$$\textsc{lam-mon-case}\ \frac{\Gamma \vdash_{mon} e : \tau_1 + \tau_2 \quad \Gamma , x : \tau_1 \vdash_{mon} e_1 : \tau \quad \Gamma , x : \tau_2 \vdash_{mon} e_2 : \tau}{\Gamma \vdash_{mon} \mathbf{case}\ e\ \mathbf{of}\ x_1 \to e_1; x_2 \to e_2 : \tau}$$

$$\textsc{lam-mon-return}\ \frac{\Gamma \vdash_{mon} e : \tau \quad \bot \leq_{eff} \phi}{\Gamma \vdash_{mon} \mathbf{return}\ e : \mathbf{T}_{\phi}\ \tau}$$

$$\textsc{lam-mon-bind}\ \frac{\Gamma \vdash_{mon} e_1 : \mathbf{T}_{\phi_1}\ \tau_1 \quad \Gamma , x : \tau_1 \vdash_{mon} e_2 : \mathbf{T}_{\phi_2}\ \tau_2 \quad \phi_1 \cdot \phi_2 \leq_{eff} \phi}{\Gamma \vdash_{mon} \mathbf{bind}\ x = e_1\ \mathbf{in}\ e_2 : \mathbf{T}_{\phi}\ \tau_2} \qquad \textsc{lam-mon-tick}\ \frac{tock \leq_{eff} \phi}{\Gamma \vdash_{mon} \mathbf{tick} : \mathbf{T}_{\phi}\ \mathbf{unit}}$$

Below, we extend Levy's translation of the CBN λ-calculus to include the graded monad. The translation of the core language is as in prior work and all effects are isolated to the monadic type. The typing rules of our monadic language include both negative ($\tau_1 \,\&\, \tau_2$) and positive ($\tau_1 \otimes \tau_2$) products, but CBN languages typically only include the former and CBV languages typically only include the latter. Therefore, our translations below follow this pattern.

$$
\begin{array}{ll}
\textit{Type translation} & \\
[\![\mathbf{unit}]\!]_{\mathrm{N}} & = \mathbf{F}\,\mathbf{unit} \\
[\![\tau_1 \longrightarrow \tau_2]\!]_{\mathrm{N}} & = (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}}) \longrightarrow [\![\tau_2]\!]_{\mathrm{N}} \\
[\![\tau_1 \,\&\, \tau_2]\!]_{\mathrm{N}} & = [\![\tau_1]\!]_{\mathrm{N}} \,\&\, [\![\tau_2]\!]_{\mathrm{N}} \\
[\![\tau_1 + \tau_2]\!]_{\mathrm{N}} & = \mathbf{F}\,(\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}} + \mathbf{U}\,[\![\tau_2]\!]_{\mathrm{N}}) \\
[\![\mathbf{T}_{\phi}\ \tau]\!]_{\mathrm{N}} & = \mathbf{F}\,(\mathbf{U}_{\phi}\ \mathbf{F}\,(\mathbf{U}_{\perp}\ [\![\tau]\!]_{\mathrm{N}})) \\
& \\
\textit{Context translation} & \\
[\![\varnothing]\!]_{\mathrm{N}} & = \varnothing \\
[\![\Gamma\,, x : \tau]\!]_{\mathrm{N}} & = [\![\Gamma]\!]_{\mathrm{N}}\,, x : \mathbf{U}_{\perp}\ [\![\tau]\!]_{\mathrm{N}} \\
& \\
\textit{Term translation} & \\
[\![x]\!]_{\mathrm{N}} & = x! \\
[\![\lambda x.e]\!]_{\mathrm{N}} & = \lambda x.[\![e]\!]_{\mathrm{N}} \\
[\![e_1\ e_2]\!]_{\mathrm{N}} & = [\![e_1]\!]_{\mathrm{N}}\ \{[\![e_2]\!]_{\mathrm{N}}\} \\
[\![()]\!]_{\mathrm{N}} & = \mathbf{return}\ () \\
[\![e_1; e_2]\!]_{\mathrm{N}} & = x \leftarrow [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ x; [\![e_2]\!]_{\mathrm{N}} \\
[\![\langle e_1, e_2\rangle]\!]_{\mathrm{N}} & = \langle [\![e_1]\!]_{\mathrm{N}}, [\![e_2]\!]_{\mathrm{N}}\rangle \\
[\![e.1]\!]_{\mathrm{N}} & = [\![e]\!]_{\mathrm{N}}.1 \\
[\![e.2]\!]_{\mathrm{N}} & = [\![e]\!]_{\mathrm{N}}.2 \\
[\![\mathbf{inl}\ e]\!]_{\mathrm{N}} & = \mathbf{return}\ \mathbf{inl}\ \{[\![e]\!]_{\mathrm{N}}\} \\
[\![\mathbf{inr}\ e]\!]_{\mathrm{N}} & = \mathbf{return}\ \mathbf{inr}\ \{[\![e]\!]_{\mathrm{N}}\} \\
[\![\mathbf{case}\ e\ \mathbf{of}\ x_1 \longrightarrow e_1; x_2 \longrightarrow e_2]\!]_{\mathrm{N}} & = x \leftarrow [\![e]\!]_{\mathrm{N}}\ \mathbf{in}\ \mathbf{case}\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \longrightarrow [\![e_1]\!]_{\mathrm{N}}; \mathbf{inr}\ x_2 \longrightarrow [\![e_2]\!]_{\mathrm{N}} \\
[\![\mathbf{return}\ e]\!]_{\mathrm{N}} & = \mathbf{return}\ \{\mathbf{return}\ \{[\![e]\!]_{\mathrm{N}}\}\} \\
[\![\mathbf{bind}\ x = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{N}} & = \mathbf{return}\ \{x \leftarrow (y \leftarrow [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ y!)\ \mathbf{in}\ z \leftarrow [\![e_2]\!]_{\mathrm{N}}\ \mathbf{in}\ z!\} \\
[\![\mathbf{tick}]\!]_{\mathrm{N}} & = \mathbf{return}\ \{x \leftarrow \mathbf{tick}\ \mathbf{in}\ \mathbf{return}\ \{\mathbf{return}\ x\}\}
\end{array}
$$

This translation preserves types (with embedded effects) from the source language. Note that, because the monadic type marks effectful code, the translation produces CBPV computations that can be checked with the "pure" effect $\perp$.

LEMMA 2.8. *If* $\Gamma \vdash_{mon} e : \tau$ *then* $[\![\Gamma]\!]_N \vdash_{eff} [\![e]\!]_N :^{\perp} [\![\tau]\!]_N$.[8]

One difficulty of this translation is that the monadic type in the CBPV adjunction is **U F**. This type is a value type, and the CBN translation produces terms with computation types. Therefore to use **U F** as the monad in our CBN translation, we need to bracket it: on the outside by **F** to form a computation type and then on the inside by **U** to construct the value type that the monad expects. This bracketing produces an awkward translation of the monadic operations with doubled thunking, needed to make the types work out.

Because the graded monad isolates effects, we can also evaluate the monadic language using a call-by-value semantics, reusing the same translation we used for the CBV language with effects. For the CBV translation, the monadic type is more accessible: the type translation produces value types, so we don't need the additional bracketing in the translations for **return** e and **tick**.

[8] CBN/Monadic/Translation.agda: translation-preservation

Type translation

$[\![\mathbf{T}_\phi\ \tau]\!]_{\mathrm{v}} = \mathbf{U}_\phi\ \mathbf{F}\ [\![\tau]\!]_{\mathrm{v}}$

Term translation

$[\![\mathbf{return}\ e]\!]_{\mathrm{v}} = \mathbf{return}\ \{[\![e]\!]_{\mathrm{v}}\}$

$[\![\mathbf{tick}]\!]_{\mathrm{v}} = \mathbf{return}\ \{x \leftarrow \mathbf{tick}\ \mathbf{in}\ \mathbf{return}\ x\}$

Lemma 2.9. *If* $\Gamma \vdash_{mon} e : \tau$ *then* $[\![\Gamma]\!]_V \vdash_{\mathit{eff}} [\![e]\!]_V :^{\perp} \mathbf{F}\ [\![\tau]\!]_V$.[9]

3 CBPV AND COEFFECTS

Next, we extend CBPV's type system with *coeffect*s. Figure 3 lists the typing rules, with coeffect annotations in blue. Coeffect systems are designed for reasoning about inputs, so we annotate variables, which always represent values in CBPV, at their binding sites and in the context.

Coeffects annotations consist of *grades* q taken from a *preordered semiring*. This gives us an addition operation $q_1 + q_2$, an additive identity element 0, a multiplication operation $q_1 \cdot q_2$, a multiplicative identity 1, and a reflexive and transitive binary relation $\leq_{co}$ that respects addition and multiplication. (The preorder does not have to be the one defined by the addition operation.) The need for a more complex structure (semiring rather than monoid) arises from the fact that we have multiple inputs to a program compared to the single output.

As with effects, our type system is general across structural coeffects [10] and can be specialized via the choice of semiring and preorder. For clarity, however, we discuss it in terms of a running example, resource usage. In this example, grades come from the natural number semiring and count the *uses* of variables, as in bounded linear logic. The additive and multiplicative identity elements of this semiring mark 0 and 1 uses of a variable respectively, and the addition and multiplication semiring operations are used in the type system to calculate the total number of times each variable is used in the program.

As in many systems for bounded linear logic, $q_1 \leq_{co} q_2$ indicates that q_1 is *less precise* or *less restrictive* than q_2. When counting variable usage, this has the opposite order from the usual one—we have $3 \leq_{co} 2$ because allowing 3 uses is less restrictive than 2. With other coeffects, such as security levels, this ordering has a more intuitive interpretation: a higher grade corresponds to a higher security level, which is more restrictive than a low security level.

Analogous to the effect system, this type system supports sub-coeffecting. If a judgment holds with some annotation q_2 on a variable in the context, then it is also derivable with any $q_1 \leq_{co} q_2$. This means the environment can provide for more demands than the program actually makes. For example, when when counting variable uses, we can weaken a judgment that a computation makes zero (0) uses of a variable to observe at most one use (affine) or any other number. This corresponds to the usual weakening lemma from typed λ-calculi.

Also like effects, including a preorder with the semiring allows for imprecision, which we need when analyzing branching computations. For example, if one branch requires 1 use of a variable x, but the other branch requires 0 uses, the system will record that the program must have the resources to use x at least once, because $1 \leq_{co} 0$. This is dual to the preorder's role in the effect system—if one branch ticks once and one branch does not tick, then the system will record at most one tick because $0 \leq_{\mathit{eff}} 1$. In both cases, choosing the discrete preorder means that the type system must be precise and would reject both of these examples.

The type system uses a *grade vector* γ, a comma-separated list of grades, as the annotations for the variables in a typing context. When combined with a typing context Γ, written $\gamma \cdot \Gamma$, the grade

[9] CBV/Monadic/Translation.agda: translation-preservation-exp

[10] In this paper, we focus on structural coeffects, as opposed to flat coeffects. Structural coeffects occur at the variable level Petricek et al. [2014], whereas *flat* coeffects mirror effects in that they occur at the program level.

$\boxed{\gamma \cdot \Gamma \vdash_{coeff} V : A}$ *(value coeffect typing)*

$$\frac{\gamma_1 \leq_{co} \overline{0}_1 \qquad q \leq_{co} 1 \qquad \gamma_2 \leq_{co} \overline{0}_2}{\gamma_1 \cdot \Gamma_1, x :^{q} A, \gamma_2 \cdot \Gamma_2 \vdash_{coeff} x : A}\ \textsc{coeff-var} \qquad \frac{\gamma \cdot \Gamma \vdash_{coeff} M : B}{\gamma \cdot \Gamma \vdash_{coeff} \{M\} : \mathbf{U}\, B}\ \textsc{coeff-thunk}$$

$$\frac{\gamma \leq_{co} \overline{0}}{\gamma \cdot \Gamma \vdash_{coeff} () : \mathbf{unit}}\ \textsc{coeff-unit} \qquad \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} V_1 : A_1 \qquad \gamma_2 \cdot \Gamma \vdash_{coeff} V_2 : A_2 \qquad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} (V_1, V_2) : A_1 \times A_2}\ \textsc{coeff-pair}$$

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} V : A_1}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{inl}\, V : A_1 + A_2}\ \textsc{coeff-inl} \qquad \frac{\gamma \cdot \Gamma \vdash_{coeff} V : A_2}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{inr}\, V : A_1 + A_2}\ \textsc{coeff-inr}$$

$\boxed{\gamma \cdot \Gamma \vdash_{coeff} M : B}$ *(computation coeffect typing)*

$$\frac{\gamma \cdot \Gamma, x :^{q} A \vdash_{coeff} M : B}{\gamma \cdot \Gamma \vdash_{coeff} \lambda x^{q}.M : A^{q} \to B}\ \textsc{coeff-abs} \qquad \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} M : A^{q} \to B \qquad \gamma_2 \cdot \Gamma \vdash_{coeff} V : A \qquad \gamma \leq_{co} \gamma_1 + q \cdot \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} M\, V : B}\ \textsc{coeff-app}$$

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} V : \mathbf{U}\, B}{\gamma \cdot \Gamma \vdash_{coeff} V! : B}\ \textsc{coeff-force} \qquad \frac{\begin{array}{c}\gamma_1 \cdot \Gamma \vdash_{coeff} V : A_1 \times A_2 \\ \gamma_2 \cdot \Gamma, x_1 :^{q} A_1, x_2 :^{q} A_2 \vdash_{coeff} N : B \qquad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \to N : B}\ \textsc{coeff-split}$$

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} V : A \qquad \gamma \leq_{co} q \cdot \gamma_1}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{return}_q\ V : \mathbf{F}_q\ A}\ \textsc{coeff-ret} \qquad \frac{\begin{array}{c}\gamma_1 \cdot \Gamma \vdash_{coeff} M : \mathbf{F}_{q_1}\ A \\ \gamma_2 \cdot \Gamma, x :^{q_1 \cdot q_2} A \vdash_{coeff} N : B \qquad \gamma \leq_{co} (q_2 \cdot \gamma_1) + \gamma_2\end{array}}{\gamma \cdot \Gamma \vdash_{coeff} x \leftarrow^{q_2} M\ \mathbf{in}\ N : B}\ \textsc{coeff-letin}$$

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} M_1 : B_1 \qquad \gamma \cdot \Gamma \vdash_{coeff} M_2 : B_2}{\gamma \cdot \Gamma \vdash_{coeff} \langle M_1, M_2 \rangle : B_1 \,\&\, B_2}\ \textsc{coeff-cpair} \qquad \frac{\gamma \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2}{\gamma \cdot \Gamma \vdash_{coeff} M.1 : B_1}\ \textsc{coeff-fst} \qquad \frac{\gamma \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2}{\gamma \cdot \Gamma \vdash_{coeff} M.2 : B_2}\ \textsc{coeff-snd}$$

$$\frac{\begin{array}{c}\gamma_1 \cdot \Gamma \vdash_{coeff} V : \mathbf{unit} \\ \gamma_2 \cdot \Gamma \vdash_{coeff} N : B \\ \gamma \leq_{co} \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \Gamma \vdash_{coeff} V; N : B}\ \textsc{coeff-sequence} \qquad \frac{\begin{array}{c}\gamma_1 \cdot \Gamma \vdash_{coeff} V : A_1 + A_2 \\ \gamma_2 \cdot \Gamma, x_1 :^{q} A_1 \vdash_{coeff} M_1 : B \\ \gamma_2 \cdot \Gamma, x_2 :^{q} A_2 \vdash_{coeff} M_2 : B \\ q \leq_{co} 1 \qquad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \to M_1; \mathbf{inr}\ x_2 \to M_2 : B}\ \textsc{coeff-case}$$

Fig. 3. CBPV with coeffect tracking

vector must have the same length as Γ. We extend a combined grade vector and typing context simultaneously with the notation $\gamma \cdot \Gamma, x :^{q} A$, equivalent to $(\gamma, q) \cdot (\Gamma, x : A)$.

The grade vector written $\overline{0}$ contains only zeros and is used where its length can be inferred from context. Grade vectors of the same length can be added together pointwise, written $\gamma_1 + \gamma_2$, and compared pointwise, written $\gamma_1 \leq_{co} \gamma_2$. Grade vectors can also be pointwise scaled, written $q \cdot \gamma$.

The key rule in this system is rule COEFF-VAR, which declares that when introducing a variable, the context may grade that variable with any q, where $q \leq_{co} 1$. No other variables in the context should affect the typing judgement, so they can have a grade of 0 or anything less restrictive. (Informally, we say that those variables are "discardable" in this typing judgement.) We write this as $\gamma_1 \leq_{co} \overline{0}_1$ and $\gamma_2 \leq_{co} \overline{0}_2$, where the zero grade vectors $\overline{0}_1$ and $\overline{0}_2$ must have the same lengths as Γ_1 and Γ_2, respectively. Similarly, the () value can make no demands on the environment, so rule COEFF-UNIT requires that all variables in the typing context be discardable, *i.e.*, $\gamma \leq_{co} \overline{0}$.

In rule COEFF-THUNK ,rule COEFF-INL, rule COEFF-INR,and rule COEFF-FORCE, there is a single subterm that makes exactly the same demands on its environment as the term in the conclusion, so we use the same grade vector in the conclusion and the premise. For example, in a sum type, **inl** V makes the same demands as V. Any imprecision in the premise gets passed along to the conclusion, so we do not need to explicitly weaken the grade vector in the conclusion. (We do need to explicitly weaken when adding or scaling grade vectors in our judgement, because we can't guarantee arbitrary vectors can be written with operations on vectors in the premises.)

In other rules, the term in the conclusion has multiple subterms, so we combine the demands made by each. In rule COEFF-PAIR, the subterms are both evaluated and do not directly interact, so we combine the grade vectors via simple pointwise addition. Conversely, in negative products (*i.e.* pairs of computations), the two subterms must share resources, so we use the same grade vector in each premise and the conclusion. Intuitively, we can only ever project out one subterm from a computation pair (see rule COEFF-FST and rule COEFF-SND), so the projected term can then make all the same demands on the environment as the pair.

In the effect system, we annotate the type $\mathbf{U}_{\phi}\ B$ with the effect of the suspended computation. In the coeffect system, we dually annotate the returner type $\mathbf{F}_{q}\ A$. In our resource usage example, the q in the returner type indicates that we require enough resources from the environment to produce q copies of a value. For example, $\mathbf{return}_3\ V$ indicates that we require the resources to create 3 copies of V. Therefore, in rule COEFF-RET we must scale the demands needed to create V by q. For example, if γ records the number of copies of each variable in the context that we need to create one copy of V, and we wish to return 3 copies of V, we then require 3 times as many copies of each variable in the context, *i.e.*, $(3 \cdot \gamma)$.

In rule COEFF-ABS, we know from the premise that M will require a grade of q on x, so we store that grade as an annotation on A in the type and on x in the term. Both the premise and the conclusion make the same demands on the variables in Γ, so γ is otherwise the same in both.

In some rules, we must combine the grade vectors of subterms using both scaling and addition. For example, in rule COEFF-APP, γ_1 denotes the demands the operator M makes on the environment, and γ_2 denotes the demands the argument V makes. M has the type $A^{q} \rightarrow B$, indicating that when it is reduced to the terminal $\lambda x^{q}.M'$, then M' will require x to have a grade of q. This means we must scale γ_1 by q before adding it to γ_2 to calculate the total demands that $M\ V$ makes on its environment.

Rule COEFF-SPLIT follows a similar pattern. In this rule, we require a grade of q on x_1 and x_2 in N, so we scale γ_1, or the demands made by V, by q. (Imprecision allows us to use the same grade for x_1 and x_2 even though the exact demands N makes on each may be different.) In rule COEFF-CASE, we additionally require that $q \leq_{co} 1$. We need to evaluate V to either **inl** V_1 or **inr** V_2 for some V_1 or V_2

in order for this branching to be well-defined, so in our resource usage example we can interpret this as requiring at least 1 copy of V in order to proceed.

The need for the scaling annotation in $x \leftarrow^{q} M$ **in** N, denoting that N requires a grade of q on x, derives from the translation of a CBV λ-calculus to CBPV described in Section 3.3.2. CBV terms always translate to computations in CBPV to ensure strictness. This means that when translating an application, we must use a let binding, converting the translated argument from a computation to a value before applying the translated function to it. However, the function may require a particular grade q on its argument. In order to retain this information through the translation, we require a grade annotation on let bindings. In rule COEFF-LETIN, M also has returner type $\mathbf{F}_{q_1}\ A$, so we write $x \leftarrow^{q_2} M$ **in** N in the typing rule to distinguish the two grades. If γ_1 denotes the demands M makes on its environment, q_2 denotes the grade N requires x to have, and γ_2 denotes the demands N makes from the rest of the environment, we need $q_2 \cdot \gamma_1 + \gamma_2$ to type the entire term.

We now state what it means formally to weaken the precision of coeffect analysis. Given any judgment, we can always use a weaker grade vector to check a value or computation.

Lemma 3.1 (Sub-coeffecting). *Suppose $\gamma' \leq_{co} \gamma$. Then*

- *If $\gamma \cdot \Gamma \vdash_{coeff} V : A$ then $\gamma' \cdot \Gamma \vdash_{coeff} V : A$.*
- *If $\gamma \cdot \Gamma \vdash_{coeff} M : B$ then $\gamma' \cdot \Gamma \vdash_{coeff} M : B$.*

Proof. We assume $\gamma' \leq_{co} \gamma$ and proceed by structural induction on $\gamma \cdot \Gamma \vdash_{coeff} V : A$.

- Case $\gamma_1 \cdot \Gamma_1, x :^{q} A, \gamma_2 \cdot \Gamma_2 \vdash_{coeff} x : A$,
  (1) By inversion on the typing judgment, $\gamma_1 \leq_{co} \overline{0}_1$
  (2) By inversion on the typing judgment, $\gamma_2 \leq_{co} \overline{0}_2$
  (3) By inversion on $\gamma' \leq_{co} (\gamma_1, q, \gamma_2)$, $\gamma' = (\gamma'_1, q', \gamma'_2)$ whereby $\gamma'_1 \leq_{co} \gamma_1$, $\gamma'_2 \leq_{co} \gamma_2$, and $q' \leq_{co} q$
  (4) By transitivity of $\leq_{co}$, $\gamma'_1 \leq_{co} \overline{0}_1$, $\gamma'_2 \leq_{co} \overline{0}_2$, $q' \leq_{co} 1$
  (5) By definition of typing, $\gamma'_1 \cdot \Gamma_1, x :^{q'} A, \gamma'_2 \cdot \Gamma_2 \vdash_{coeff} x : A$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \{M\} : \mathbf{U}\, B$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} M : B$
  (2) Using induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} M : B$
  (3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \{M\} : \mathbf{U}\, B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} () : \mathbf{unit}$
  (1) By inversion on the typing judgment, $\gamma \leq_{co} \overline{0}$
  (2) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} \gamma$
  (3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} () : \mathbf{unit}$
- Case $\gamma \cdot \Gamma \vdash_{coeff} (V_1, V_2) : A_1 \times A_2$,
  (1) By inversion on the typing judgment, $\gamma \leq_{co} \gamma_1 + \gamma_2$
  (2) By inversion on the typing judgment, $\gamma_1 \cdot \Gamma \vdash_{coeff} V_1 : A_1$
  (3) By inversion on the typing judgment, $\gamma_2 \cdot \Gamma \vdash_{coeff} V_2 : A_2$
  (4) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} \gamma_1 + \gamma_2$
  (5) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} (V_1, V_2) : A_1 \times A_2$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \mathbf{inl}\, V : A_1 + A_2$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} V : A_1$
  (2) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} V : A_1$
  (3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \mathbf{inl}\, V : A_1 + A_2$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \mathbf{inr}\, V : A_1 + A_2$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} V : A_2$

(2) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} V : A_2$
(3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \mathbf{inr}\ V : A_1 + A_2$

Now we proceed by structural induction on $\gamma \cdot \Gamma \vdash_{coeff} M : B$.

- Case $\gamma \cdot \Gamma \vdash_{coeff} \lambda x^q.M : A^q \to B$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma, x :^q A \vdash_{coeff} M : B$
  (2) By definition of $\leq_{co}$ with $\gamma' \leq_{co} \gamma$, $(\gamma', q) \leq_{co} (\gamma, q)$
  (3) Using the induction hypothesis, $(\gamma', q) \cdot \Gamma, x : A \vdash_{coeff} M : B$
  (4) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \lambda x^q.M : A^q \to B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} M\ V : B$,
  (1) By inversion on the typing judgment $\gamma_1 \cdot \Gamma \vdash_{coeff} M : A^q \to B$
  (2) By inversion on the typing judgment $\gamma_2 \cdot \Gamma \vdash_{coeff} V : A$
  (3) By inversion on the typing judgment $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$
  (4) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} \gamma_1 + q \cdot \gamma_2$
  (5) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} M\ V : B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} V! : B$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} V : \mathbf{U}\,B$
  (2) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} V : \mathbf{U}\,B$
  (3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} V! : B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \mathbf{return}_q\ V : \mathbf{F}_q\ A$,
  (1) By inversion on the typing judgment, $\gamma_1 \cdot \Gamma \vdash_{coeff} V : A$
  (2) By inversion on the typing judgment, $\gamma \leq_{co} q \cdot \gamma_1$
  (3) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} q \cdot \gamma_1$
  (4) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \mathbf{return}_q\ V : \mathbf{F}_q\ A$
- Case $\gamma \cdot \Gamma \vdash_{coeff} x \leftarrow^{q_2} M\ \mathbf{in}\ N : B$,
  (1) By inversion on the typing judgment, $\gamma_1 \cdot \Gamma \vdash_{coeff} M : \mathbf{F}_{q_1}\ A$
  (2) By inversion on the typing judgment, $\gamma_2 \cdot \Gamma, x :^{(q_1 \cdot q_2)} A \vdash_{coeff} N : B$
  (3) By inversion on the typing judgment, $\gamma \leq_{co} (q_2 \cdot \gamma_1) + \gamma_2$
  (4) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} (q_2 \cdot \gamma_1) + \gamma_2$
  (5) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} x \leftarrow^{q_2} M\ \mathbf{in}\ N : B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \ \to\ N : B$,
  (1) By inversion on the typing judgment, $\gamma_1 \cdot \Gamma \vdash_{coeff} V : A_1 \times A_2$
  (2) By inversion on the typing judgment, $\gamma_2 \cdot \Gamma, x_1 :^q A_1, x_2 :^q A_2 \vdash_{coeff} N : B$
  (3) By inversion on the typing judgment, $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$
  (4) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} q \cdot \gamma_1 + \gamma_2$
  (5) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \ \to\ N : B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} V; N : B$,
  (1) By inversion on the typing judgment, $\gamma_1 \cdot \Gamma \vdash_{coeff} V : \mathbf{unit}$
  (2) By inversion on the typing judgment, $\gamma_2 \cdot \Gamma \vdash_{coeff} N : B$
  (3) By inversion on the typing judgment, $\gamma \leq_{co} \gamma_1 + \gamma_2$
  (4) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} \gamma_1 + \gamma_2$
  (5) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} V; N : B$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \langle M_1, M_2 \rangle : B_1 \,\&\, B_2$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} M_1 : B_1$
  (2) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} M_2 : B_2$
  (3) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} M_1 : B_1$
  (4) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} M_2 : B_2$
  (5) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \langle M_1, M_2 \rangle : B_1 \,\&\, B_2$

- Case $\gamma \cdot \Gamma \vdash_{coeff} M.1 : B_1$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2$
  (2) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2$
  (3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} M.1 : B_1$
- Case $\gamma \cdot \Gamma \vdash_{coeff} M.2 : B_2$,
  (1) By inversion on the typing judgment, $\gamma \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2$
  (2) Using the induction hypothesis, $\gamma' \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2$
  (3) By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} M.2 : B_2$
- Case $\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of\ inl}\ x_1 \rightarrow\ M_1; \mathbf{inr}\ x_2 \rightarrow\ M_2 : B$,
  (1) By inversion on the typing judgment, $\gamma_1 \cdot \Gamma \vdash_{coeff} V : A_1 + A_2$
  (2) By inversion on the typing judgment, $\gamma_2 \cdot \Gamma, x :^q A_1 \vdash_{coeff} M_1 : B$
  (3) By inversion on the typing judgment, $\gamma_2 \cdot \Gamma, x :^q A_2 \vdash_{coeff} M_2 : B$
  (4) By inversion on the typing judgment, $q \leq_{co} 1$ and $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$
  (5) By transitivity of $\leq_{co}$, $\gamma' \leq_{co} q \cdot \gamma_1 + \gamma_2$
  (6)By definition of typing, $\gamma' \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of\ inl}\ x_1 \rightarrow\ M_1; \mathbf{inr}\ x_2 \rightarrow\ M_2 : B$

Therefore sub-coeffecting holds for any value or computation. □

3.1 General instrumented operational semantics and coeffect soundness

Next, we develop an instrumented operational semantics (shown in Figure 4) that tracks coeffects using an environment ρ, which maps variables to closed values, and a grade vector γ of equal length, which implicitly maps variables to their coeffects. As in the typing rules, we extend both a grade vector and corresponding environment simultaneously with the notation $\gamma \cdot \rho,\ x \mapsto^q W$, equivalent to $(\gamma, q) \cdot (\rho, x \mapsto W)$.

As before, we use W as a metavariable for *closed* values, and T as a metavariable for *closed terminal* computations. However, closed terminals include coeffects here. They have the form $\mathbf{return}_q W$, $\mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M)$, or $\mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle)$, where $\mathbf{clo}(\gamma \cdot \rho, M)$ denotes the *closure* of M under $\gamma \cdot \rho$. The grade vector in the closure indicates the demands on the variables used by M.

Unlike our instrumented operational semantics for effects, which calculates the exact effect of a computation, this semantics cannot track coeffects with complete precision. For example, in $\lambda x^q.M$, evaluating M may require different exact grades on x depending how the function computes (q is just a bound), so we cannot write a precise rule for evaluating abstractions to their closures. This imprecision gives us the following semantic sub-coeffecting property:

Lemma 3.2 (Operational semantics sub-coeffecting). *Suppose $\gamma' \leq_{co} \gamma$. Then*

- *If $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$ then $\gamma' \cdot \rho \vdash_{coeff} V \Downarrow W$.*
- *If $\gamma \cdot \rho \vdash_{coeff} M \Downarrow T$ then $\gamma' \cdot \rho \vdash_{coeff} M \Downarrow T$.*

Proof. By mutual induction on the derivations of $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$ and $\gamma \cdot \rho \vdash_{coeff} M \Downarrow T$.
In each case, we apply the same rule as the hypothesis, using γ' in place of γ. The hypotheses are fulfilled directly by inversion (for evaluations) and transitivity (for comparisons). □

As in the semantics for CBPV without coeffects, we define "evaluation" of values using the given environment (see Figure 4). Rule eval-coeff-val-var requires the variable to have "at most" 1 as its corresponding grade and all other variables in the environment must be discardable. In a resource usage coeffect system, looking up the value of a variable in the environment counts as one use of the variable. In general, we can think of 1 as denoting "default" usage. Rule eval-coeff-val-unit requires that every variable in the context be discardable. Rule eval-coeff-val-thunk simply includes the grade vector in the closure along with the environment. Rule eval-coeff-val-vpair requires at most the sum of all the grades needed to evaluate subterms to their closures.

$\boxed{\gamma \cdot \rho \vdash_{coeff} V \Downarrow W}$ *(Value rules)*

$$\textsc{eval-coeff-val-var}\quad \frac{\gamma_1 \leq_{co} \bar{0}_1 \qquad q \leq_{co} 1 \qquad \gamma_2 \leq_{co} \bar{0}_2}{\gamma_1 \cdot \rho_1 ,\ x \mapsto^{q} W ,\ \gamma_2 \cdot \rho_2 \vdash_{coeff} x \Downarrow W}$$

$$\textsc{eval-coeff-val-unit}\quad \frac{\gamma \leq_{co} \bar{0}}{\gamma \cdot \rho \vdash_{coeff} () \Downarrow ()}$$

$$\textsc{eval-coeff-val-thunk}\quad \frac{\gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{coeff} \{M\} \Downarrow \mathbf{clo}(\gamma' \cdot \rho, \{M\})}$$

$$\textsc{eval-coeff-val-vpair}\quad \frac{\gamma_1 \cdot \rho \vdash_{coeff} V_1 \Downarrow W_1 \qquad \gamma_2 \cdot \rho \vdash_{coeff} V_2 \Downarrow W_2 \qquad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \rho \vdash_{coeff} (V_1, V_2) \Downarrow (W_1, W_2)}$$

$$\textsc{eval-coeff-val-inl}\quad \frac{\gamma \cdot \rho \vdash_{coeff} V \Downarrow W}{\gamma \cdot \rho \vdash_{coeff} \mathbf{inl}\, V \Downarrow \mathbf{inl}\, W}$$

$$\textsc{eval-coeff-val-inr}\quad \frac{\gamma \cdot \rho \vdash_{coeff} V \Downarrow W}{\gamma \cdot \rho \vdash_{coeff} \mathbf{inr}\, V \Downarrow \mathbf{inr}\, W}$$

$\boxed{\gamma \cdot \rho \vdash_{coeff} M \Downarrow T}$ *(Computation rules)*

$$\textsc{eval-coeff-comp-abs}\quad \frac{\gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{coeff} \lambda x^{q}.M \Downarrow \mathbf{clo}(\gamma' \cdot \rho, \lambda x^{q}.M)}$$

$$\textsc{eval-coeff-comp-cpair}\quad \frac{\gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{coeff} \langle M_1, M_2 \rangle \Downarrow \mathbf{clo}(\gamma' \cdot \rho, \langle M_1, M_2 \rangle)}$$

$$\textsc{eval-coeff-comp-app-abs}\quad \frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{coeff} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^{q}.M') \\ \gamma_2 \cdot \rho \vdash_{coeff} V \Downarrow W \\ \gamma' \cdot \rho' ,\ x \mapsto^{q} W \vdash_{coeff} M' \Downarrow T \qquad \gamma \leq_{co} \gamma_1 + q \cdot \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{coeff} M\, V \Downarrow T}$$

$$\textsc{eval-coeff-comp-split}\quad \frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow (W_1, W_2) \\ \gamma_2 \cdot \rho ,\ x_1 \mapsto^{q} W_1 ,\ x_2 \mapsto^{q} W_2 \vdash_{coeff} N \Downarrow T \\ \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_{q}\ V\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N \Downarrow T}$$

$$\textsc{eval-coeff-comp-return}\quad \frac{\begin{array}{c}\gamma' \cdot \rho \vdash_{coeff} V \Downarrow W \\ \gamma \leq_{co} q \cdot \gamma'\end{array}}{\gamma \cdot \rho \vdash_{coeff} \mathbf{return}_{q}\ V \Downarrow \mathbf{return}_{q}\, W}$$

$$\textsc{eval-coeff-comp-letin-ret}\quad \frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{coeff} M \Downarrow \mathbf{return}_{q_1} W \\ \gamma_2 \cdot \rho ,\ x \mapsto^{q_1 \cdot q_2} W \vdash_{coeff} N \Downarrow T \qquad \gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{coeff} x \leftarrow^{q_2} M\ \mathbf{in}\ N \Downarrow T}$$

$$\textsc{eval-coeff-comp-force-thunk}\quad \frac{\gamma \cdot \rho \vdash_{coeff} V \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \{M\}) \qquad \gamma' \cdot \rho' \vdash_{coeff} M \Downarrow T}{\gamma \cdot \rho \vdash_{coeff} V! \Downarrow T}$$

$$\textsc{eval-coeff-comp-fst}\quad \frac{\begin{array}{c}\gamma \cdot \rho \vdash_{coeff} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle) \\ \gamma' \cdot \rho' \vdash_{coeff} M_1 \Downarrow T\end{array}}{\gamma \cdot \rho \vdash_{coeff} M.1 \Downarrow T}$$

$$\textsc{eval-coeff-comp-snd}\quad \frac{\begin{array}{c}\gamma \cdot \rho \vdash_{coeff} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle) \\ \gamma' \cdot \rho' \vdash_{coeff} M_2 \Downarrow T\end{array}}{\gamma \cdot \rho \vdash_{coeff} M.2 \Downarrow T}$$

$$\textsc{eval-coeff-comp-sequence}\quad \frac{\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow () \qquad \gamma_2 \cdot \rho \vdash_{coeff} N \Downarrow T \qquad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \rho \vdash_{coeff} V ; N \Downarrow T}$$

$$\textsc{eval-coeff-comp-case-inl}\quad \frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow \mathbf{inl}\, W \\ \gamma_2 \cdot \rho ,\ x_1 \mapsto^{q} W \vdash_{coeff} M_1 \Downarrow T \qquad q \leq_{co} 1 \qquad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_{q}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1 ; \mathbf{inr}\ x_2 \rightarrow M_2 \Downarrow T}$$

$$\textsc{eval-coeff-comp-case-inr}\quad \frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow \mathbf{inr}\, W \\ \gamma_2 \cdot \rho ,\ x_2 \mapsto^{q} W \vdash_{coeff} M_2 \Downarrow T \qquad q \leq_{co} 1 \qquad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_{q}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1 ; \mathbf{inr}\ x_2 \rightarrow M_2 \Downarrow T}$$

Figure 4 shows the operational semantics for computations. Rules EVAL-COEFF-COMP-ABS, EVAL-COEFF-COMP-FORCE-THUNK, EVAL-COEFF-COMP-CPAIR, EVAL-COEFF-COMP-FST, and EVAL-COEFF-COMP-SND are largely the same as before, just with the inclusion of grade vectors along with environments. Rule EVAL-COEFF-COMP-SEQUENCE simply sums the vectors required to evaluate each subterm. The sum type elimination rules scale the demands made by the term being eliminated by q before adding them to the demands needed to evaluate the rest of the computation, as in the typing rules. They also require that $q \leq_{co} 1$ for the branching behavior to be well-defined, as in the typing rules. Intuitively, in a resource counting context, if we have 0 copies of a value, we should not be able to use it to determine which branch to take.

In rule EVAL-COEFF-COMP-RETURN, we scale the grade needed to evaluate the subterm to its closure by q. In the elimination rules EVAL-COEFF-COMP-APP-ABS, EVAL-COEFF-COMP-LETIN-RET, and EVAL-COEFF-COMP-SPLIT, if we are eliminating a value V and binding it to a variable x with a grade q for use in some computation M, we must scale the grade vector needed to evaluate V by q before adding it to the grade vector needed to continue with M.

We prove a coeffect soundness theorem stating that if a term is well-typed with some grade vector γ, it can evaluate to a terminal given γ and some environment ρ that provides values of the correct type for all free variables, formalized as $\Gamma \vDash \rho$ in our logical relation below. Because values and computations have distinct syntax and semantics, and because both values and computations make demands on their inputs, we state this property for both. Formally:

THEOREM 3.3 (COEFFECT SOUNDNESS THEOREM). *Let Γ be a context and ρ an environment mapping all variables in the domain of Γ to closed values of the expected type. Then:*

(1) *If $\gamma \cdot \Gamma \vdash_{coeff} V : A$ then $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$ for some closed value W.*
(2) *If $\gamma \cdot \Gamma \vdash_{coeff} M : B$ then $\gamma \cdot \rho \vdash_{coeff} M \Downarrow T$ for some closed terminal computation T.*

The proof of the coeffect soundness theorem is similar to the proof of the effect soundness theorem, and requires the following logical relation, based on sets of closed values, terminals, values paired with environments, and computations paired with environments.

Definition 3.4 (CBPV with Coeffects: General Logical Relation).

$$
\begin{array}{lcll}
\mathcal{W}[\![\mathbf{U}\, B]\!] & = & \{\, \mathbf{clo}(\gamma \cdot \rho, \{M\}) & \mid (\gamma \cdot \rho, M) \in \mathcal{M}[\![B]\!] \,\} \\
\mathcal{W}[\![\mathbf{unit}]\!] & = & \{\, ()\, \} & \\
\mathcal{W}[\![A_1 \times A_2]\!] & = & \{\, (W_1, W_2) & \mid W_1 \in \mathcal{W}[\![A_1]\!] \textit{ and } W_2 \in \mathcal{W}[\![A_2]\!]\} \\
\mathcal{W}[\![A_1 + A_2]\!] & = & \{\, \mathbf{inl}\, W & \mid W \in \mathcal{W}[\![A_1]\!]\} \cup \{\, \mathbf{inr}\, W \mid W \in \mathcal{W}[\![A_2]\!]\} \\
\\
\mathcal{T}[\![\mathbf{F}_q\, A]\!] & = & \{\, \mathbf{return}_q\, W & \mid W \in \mathcal{W}[\![A]\!] \,\} \\
\mathcal{T}[\![A^q \to B]\!] & = & \{\, \mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M) & \mid \textit{for all } W \in \mathcal{W}[\![A]\!], ((\gamma \cdot \rho,\ x \mapsto^q W), M) \in \mathcal{M}[\![B]\!] \,\} \\
\mathcal{T}[\![B_1 \,\&\, B_2]\!] & = & \{\, \mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle) & \mid (\gamma, \rho, M_1) \in \mathcal{M}[\![B_1]\!] \textit{ and } (\gamma, \rho, M_2) \in \mathcal{M}[\![B_2]\!] \,\} \\
\\
\mathcal{V}[\![A]\!] & = & \{\, (\gamma \cdot \rho, V) & \mid \gamma \cdot \rho \vdash_{coeff} V \Downarrow W \textit{ and } W \in \mathcal{W}[\![A]\!] \,\} \\
\mathcal{M}[\![B]\!] & = & \{\, (\gamma \cdot \rho, M) & \mid \gamma \cdot \rho \vdash_{coeff} M \Downarrow T \textit{ and } T \in \mathcal{T}[\![B]\!] \,\}
\end{array}
$$

Definition 3.5 (CBPV with Coeffects: General Semantic Typing).

$$
\begin{array}{lcl}
\Gamma \vDash \rho & = & x : A \in \Gamma \textit{ implies } x \mapsto W \in \rho \textit{ and } W \in \mathcal{W}[\![A]\!] \\
\gamma \cdot \Gamma \vDash_{coeff} V : A & = & \textit{for all } \rho, \Gamma \vDash \rho \textit{ implies } (\gamma \cdot \rho, V) \in \mathcal{V}[\![A]\!] \\
\gamma \cdot \Gamma \vDash_{coeff} M : B & = & \textit{for all } \rho, \Gamma \vDash \rho \textit{ implies } (\gamma \cdot \rho, M) \in \mathcal{M}[\![B]\!]
\end{array}
$$

We can now state the fundamental lemma of this relation, which derives the soundness theorem as a corollary.

Theorem 3.6 (Fundamental lemma: coeffect soundness). *For all* γ, Γ, *if* $\gamma \cdot \Gamma \vdash_{coeff} V : A$ *then* $\gamma \cdot \Gamma \vDash_{coeff} V : A$, *and for all* γ, Γ, *if* $\gamma \cdot \Gamma \vdash_{coeff} M : B$ *then* $\gamma \cdot \Gamma \vDash_{coeff} M : B$.

Proof. By mutual induction on the typing derivations, using the case lemmas below. □

Lemma 3.7 (Semantic coeff-var). *If* $\gamma_1 \leq_{co} \overline{0}_1$ *and* $q \leq_{co} 1$ *and* $\gamma_2 \leq_{co} \overline{0}_2$ *then* $((\gamma_1 , q) , \gamma_2) \cdot ((\Gamma_1 , x : A) , \Gamma_2) \vDash_{coeff} x : A$.

Proof. Given $(\Gamma_1 , x : A) , \Gamma_2 \vDash \rho$, we have by definition some $W \in \mathcal{W}[\![A]\!]$ such that $x \mapsto W \in \rho$.
So by rule EVAL-COEFF-VAL-VAR, $(\gamma_1 , q) , \gamma_2 \cdot \rho \vdash_{coeff} x \Downarrow W$, so $((\gamma_1 , q) , \gamma_2, \rho, x) \in \mathcal{V}[\![A]\!]$. □

Lemma 3.8 (Semantic coeff-thunk). *If* $\gamma \cdot \Gamma \vDash_{coeff} M : B$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \{M\} : \mathbf{U}\, B$.

Proof. Given $\Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, M) \in \mathcal{M}[\![B]\!]$, so by definition $\mathbf{clo}(\gamma \cdot \rho, \{M\}) \in \mathcal{W}[\![\mathbf{U}\, B]\!]$.
By rule EVAL-COEFF-VAL-THUNK, since $\gamma \leq_{co} \gamma$ we get that $\gamma \cdot \rho \vdash_{coeff} \{M\} \Downarrow \mathbf{clo}(\gamma \cdot \rho, \{M\})$, so $(\gamma, \rho, \{M\}) \in \mathcal{V}[\![\mathbf{U}\, B]\!]$. □

Lemma 3.9 (Semantic coeff-unit). *If* $\gamma \leq_{co} \overline{0}$ *then* $\gamma \cdot \Gamma \vDash_{coeff} () : \mathbf{unit}$.

Proof. Given $\Gamma \vDash \rho$, we can use our assumption that $\gamma \leq_{co} \overline{0}$ to derive by rule EVAL-COEFF-VAL-UNIT that $\gamma \cdot \rho \vdash_{coeff} () \Downarrow ()$, so $(\gamma, \rho, ()) \in \mathcal{V}[\![\mathbf{unit}]\!]$. □

Lemma 3.10 (Semantic coeff-pair). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V_1 : A_1$ *and* $\gamma_2 \cdot \Gamma \vDash_{coeff} V_2 : A_2$ *and* $\gamma \leq_{co} \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} (V_1, V_2) : A_1 \times A_2$.

Proof. Given $\Gamma \vDash \rho$, by assumption we know that $(\gamma_1, \rho, V_1) \in \mathcal{V}[\![A_1]\!]$ and and $(\gamma_2, \rho, V_2) \in \mathcal{V}[\![A_2]\!]$.
So there exists $W_1 \in \mathcal{W}[\![A_1]\!]$ and $W_2 \in \mathcal{W}[\![A_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} V_1 \Downarrow W_1$ and $\gamma_2 \cdot \rho \vdash_{coeff} V_2 \Downarrow W_2$.
So by rule EVAL-COEFF-VAL-VPAIR, since $\gamma \leq_{co} \gamma_1 + \gamma_2$, we have that $\gamma \cdot \rho \vdash_{coeff} (V_1, V_2) \Downarrow (W_1, W_2)$. $(W_1, W_2) \in \mathcal{W}[\![A_1 \times A_2]\!]$, so $(\gamma', \rho, (V_1, V_2)) \in \mathcal{V}[\![A_1 \times A_2]\!]$. □

Lemma 3.11 (Semantic coeff-inl). *If* $\gamma \cdot \Gamma \vDash_{coeff} V : A_1$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{inl}\, V : A_1 + A_2$.

Proof. Given $\Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, V) \in \mathcal{V}[\![A_1]\!]$, i.e., there exists $W \in \mathcal{W}[\![A_1]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$.
So by rule EVAL-COEFF-VAL-INL, $\gamma \cdot \rho \vdash_{coeff} \mathbf{inl}\, V \Downarrow \mathbf{inl}\, W$. $\mathbf{inl}\, W \in \mathcal{W}[\![A_1 + A_2]\!]$, so $(\gamma, \rho, \mathbf{inl}\, V) \in \mathcal{V}[\![A_1 + A_2]\!]$. □

Lemma 3.12 (Semantic coeff-inr). *If* $\gamma \cdot \Gamma \vDash_{coeff} V : A_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{inr}\, V : A_1 + A_2$.

Proof. Given $\Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, V) \in \mathcal{V}[\![A_2]\!]$, i.e., there exists $W \in \mathcal{W}[\![A_2]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$.
So by rule EVAL-COEFF-VAL-INR, $\gamma \cdot \rho \vdash_{coeff} \mathbf{inr}\, V \Downarrow \mathbf{inr}\, W$. $\mathbf{inr}\, W \in \mathcal{W}[\![A_1 + A_2]\!]$, so $(\gamma, \rho, \mathbf{inr}\, V) \in \mathcal{V}[\![A_1 + A_2]\!]$. □

Lemma 3.13 (Semantic coeff-abs). *If* $(\gamma , q) \cdot (\Gamma , x : A) \vDash_{coeff} M : B$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \lambda x^q.M : A^q \to B$

Proof. Given $\Gamma \vDash \rho$, we can fix arbitrary $W \in \mathcal{W}[\![A]\!]$ and get that $\Gamma , x : A \vDash \rho , x \mapsto W$. So by assumption, $(\gamma , q, \rho , x \mapsto W, M) \in \mathcal{M}[\![B]\!]$.
So we have that $\mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M) \in \mathcal{T}[\![A^q \to B]\!]$.
By rule EVAL-COEFF-COMP-ABS and because $\gamma \leq_{co} \gamma$, we have that $\gamma \cdot \rho \vdash_{coeff} \lambda x^q.M \Downarrow \mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M)$, so $(\gamma, \rho, \lambda x^q.M) \in \mathcal{M}[\![A^q \to B]\!]$. □

Lemma 3.14 (Semantic coeff-app). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} M : A^q \to B$ *and* $\gamma_2 \cdot \Gamma \vDash_{coeff} V : A$ *and* $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} M\, V : B$.

Proof. Given $\Gamma \vDash \rho$, we have by assumption that $(\gamma_1, \rho, M) \in \mathcal{M}[\![A^q \to B]\!]$ and $(\gamma_2, \rho, V) \in \mathcal{V}[\![A]\!]$.
So there exists $T' \in \mathcal{T}[\![A^q \to B]\!]$ and $W \in \mathcal{W}[\![A]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} M \Downarrow T'$ and $\gamma_2 \cdot \rho \vdash_{coeff} V \Downarrow W$.
By definition, T' must have the form $\mathbf{clo}(\gamma' \cdot \rho', \lambda x^q.M')$ such that $(\gamma', q, \rho', x \mapsto W, M') \in \mathcal{M}[\![B]\!]$.
So there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma', q \cdot \rho', x \mapsto W \vdash_{coeff} M' \Downarrow T$.
Because $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$, we have by rule eval-coeff-comp-app-abs that $\gamma \cdot \rho \vdash_{coeff} M\, V \Downarrow T$, so $(\gamma, \rho, M\, V) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.15 (Semantic coeff-force). *If* $\gamma \cdot \Gamma \vDash_{coeff} V : \mathbf{U}\, B$ *then* $\gamma \cdot \Gamma \vDash_{coeff} V! : B$

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $W \in \mathcal{W}[\![\mathbf{U}\, B]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$.
By definition, W must have the form $\mathbf{clo}(\gamma' \cdot \rho', \{M\})$ such that $(\gamma', \rho', M) \in \mathcal{M}[\![B]\!]$.
So there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma' \cdot \rho' \vdash_{coeff} M \Downarrow T$.
By rule eval-coeff-comp-force-thunk, $\gamma \cdot \rho \vdash_{coeff} V! \Downarrow T$, so $(\gamma, \rho, V!) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.16 (Semantic coeff-return). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : A$ *and* $\gamma \leq_{co} q \cdot \gamma_1$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{return}_q\ V : \mathbf{F}_q\ A$

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $W \in \mathcal{W}[\![A]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow W$.
So by definition, $\mathbf{return}_q W \in \mathcal{T}[\![\mathbf{F}_q\ A]\!]$.
Because $\gamma \leq_{co} q \cdot \gamma_1$, we have by rule eval-coeff-comp-return that $\gamma \cdot \rho \vdash_{coeff} \mathbf{return}_q\ V \Downarrow \mathbf{return}_q W$, so $(\gamma, \rho, \mathbf{return}_q\ V) \in \mathcal{M}[\![\mathbf{F}_q\ A]\!]$. □

Lemma 3.17 (Semantic coeff-letin). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} M : \mathbf{F}_{q_1}\ A$ *and* $(\gamma_2, q_1 \cdot q_2) \cdot (\Gamma, x : A) \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} x \leftarrow^{q_2} M \mathbf{\ in\ } N : B$

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $T' \in \mathcal{T}[\![\mathbf{F}_{q_1}\ A]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} M \Downarrow T'$.
By definition, T' must have the form $\mathbf{return}_{q_1} W$ for some $W \in \mathcal{W}[\![A]\!]$.
So $\Gamma, x : A \vDash \rho, x \mapsto W$, so by assumption we have that there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q_1 \cdot q_2 \cdot \rho, x \mapsto W \vdash_{coeff} N \Downarrow T$.
By assumption, $\gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2$, so by rule eval-coeff-comp-letin-ret, we get that $\gamma \cdot \rho \vdash_{coeff} x \leftarrow^{q_2} M \mathbf{\ in\ } N \Downarrow T$.
So $(\gamma, \rho, x \leftarrow^{q_2} M \mathbf{\ in\ } N) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.18 (Semantic coeff-split). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : A_1 \times A_2$ *and* $((\gamma_2, q), q) \cdot ((\Gamma, x_1 : A_1), x_2 : A_2) \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{case}_q\ V \mathbf{\ of\ } (x_1, x_2) \to N : B$.

Proof. Given $\Gamma \vDash \rho$, there exists by assumption $W \in \mathcal{W}[\![A_1 \times A_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow W$.
By definition, W must have the form (W_1, W_2) for some $W_1 \in \mathcal{W}[\![A_1]\!]$ and $W_2 \in \mathcal{W}[\![A_2]\!]$.
So $\Gamma, x_1 : A_1, x_2 : A_2 \vDash \rho, x_1 \mapsto W_1, x_2 \mapsto W_2$, so by assumption there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q, q \cdot \rho, x_1 \mapsto W_1, x_2 \mapsto W_2 \vdash_{coeff} N \Downarrow T$.
Because $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ by assumption, we get that $\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_q\ V \mathbf{\ of\ } (x_1, x_2) \to N \Downarrow T$.
So $(\gamma, \rho, \mathbf{case}_q\ V \mathbf{\ of\ } (x_1, x_2) \to N) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.19 (Semantic coeff-sequence). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : \mathbf{unit}$ *and* $\gamma_2 \cdot \Gamma \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} V; N : B$.

Proof. Given $\Gamma \vDash \rho$, there exist by assumption $W \in \mathcal{W}[\![\mathbf{unit}]\!]$ and $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow W$ and $\gamma_2 \cdot \rho \vdash_{coeff} N \Downarrow T$.

By definition, W must have the form $()$.

Because $\gamma \leq_{co} \gamma_1 + \gamma_2$, we have by rule EVAL-COEFF-COMP-SEQUENCE that $\gamma \cdot \rho \vdash_{coeff} V; N \Downarrow T$.

So $(\gamma, \rho, V; N) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.20 (Semantic coeff-case). *If $\gamma_1 \cdot \Gamma \vDash_{coeff} V : A_1 + A_2$ and $(\gamma_2, q) \cdot (\Gamma, x_1 : A_1) \vDash_{coeff} M_1 : B$ and $(\gamma_2, q) \cdot (\Gamma, x_2 : A_2) \vDash_{coeff} M_2 : B$ and $q \leq_{co} 1$ and $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ then $\gamma \cdot \Gamma \vDash_{coeff}$ $\mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \to M_1; \mathbf{inr}\ x_2 \to M_2 : B$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $W \in \mathcal{W}[\![A_1 + A_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} V \Downarrow W$.

By definition, W must have the form $\mathbf{inl}\ W_1$ or $\mathbf{inr}\ W_2$ for some $W_1 \in \mathcal{W}[\![A_1]\!]$ or $W_2 \in \mathcal{W}[\![A_2]\!]$.

If W has the form $\mathbf{inl}\ W_1$, then $\Gamma, x_1 : A_1 \vDash \rho, x_1 \mapsto W_1$, so by assumption we get that there exists $T_1 \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q \cdot \rho, x_1 \mapsto W_1 \vdash_{coeff} M_1 \Downarrow T_1$.

By assumption, $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ and $q \leq_{co} 1$, so by rule EVAL-COEFF-COMP-CASE-INL, we have that $\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \to M_1; \mathbf{inr}\ x_2 \to M_2 \Downarrow T_1$.

So $(\gamma, \rho, \mathbf{case}_q\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \to M_1; \mathbf{inr}\ x_2 \to M_2) \in \mathcal{M}[\![B]\!]$ in this case.

We use the same logic and rule EVAL-COEFF-COMP-CASE-INR to conclude that $(\gamma, \rho, \mathbf{case}_q\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \to M_1; \mathbf{inr}\ x_2 \to M_2) \in \mathcal{M}[\![B]\!]$ in the case where W has the form $\mathbf{inr}\ W_2$ also. □

Lemma 3.21 (Semantic coeff-cpair). *If $\gamma \cdot \Gamma \vDash_{coeff} M_1 : B_1$ and $\gamma \cdot \Gamma \vDash_{coeff} M_2 : B_2$ then $\gamma \cdot \Gamma \vDash_{coeff} \langle M_1, M_2 \rangle : B_1 \,\&\, B_2$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, M_1) \in \mathcal{M}[\![B_1]\!]$ and $(\gamma, \rho, M_2) \in \mathcal{M}[\![B_2]\!]$.

So by definition, $\mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle) \in \mathcal{T}[\![B_1 \,\&\, B_2]\!]$.

$\gamma \leq_{co} \gamma$, so by rule EVAL-COEFF-COMP-CPAIR, we get that $\gamma \cdot \rho \vdash_{coeff} \langle M_1, M_2 \rangle \Downarrow \mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle)$

So $(\gamma, \rho, \langle M_1, M_2 \rangle) \in \mathcal{M}[\![B_1 \,\&\, B_2]\!]$. □

Lemma 3.22 (Semantic coeff-fst). *If $\gamma \cdot \Gamma \vDash_{coeff} M : B_1 \,\&\, B_2$ then $\gamma \cdot \Gamma \vDash_{coeff} M.1 : B_1$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $T \in \mathcal{T}[\![B_1 \,\&\, B_2]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} M \Downarrow T$.

By definition, T must have the form $\mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle)$ such that $(\gamma', \rho', M_1) \in \mathcal{M}[\![B_1]\!]$, i.e., there exists $T_1 \in \mathcal{T}[\![B_1]\!]$ such that $\gamma' \cdot \rho' \vdash_{coeff} M_1 \Downarrow T_1$.

So by rule EVAL-COEFF-COMP-FST, we have that $\gamma \cdot \rho \vdash_{coeff} M.1 \Downarrow T_1$.

So $(\gamma, \rho, M.1) \in \mathcal{M}[\![B_1]\!]$. □

Lemma 3.23 (Semantic coeff-snd). *If $\gamma \cdot \Gamma \vDash_{coeff} M : B_1 \,\&\, B_2$ then $\gamma \cdot \Gamma \vDash_{coeff} M.2 : B_2$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $T \in \mathcal{T}[\![B_1 \,\&\, B_2]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} M \Downarrow T$.

By definition, T must have the form $\mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle)$ such that $(\gamma', \rho', M_2) \in \mathcal{M}[\![B_2]\!]$, i.e., there exists $T_2 \in \mathcal{T}[\![B_2]\!]$ such that $\gamma' \cdot \rho' \vdash_{coeff} M_2 \Downarrow T_2$.

So by rule EVAL-COEFF-COMP-FST, we have that $\gamma \cdot \rho \vdash_{coeff} M.2 \Downarrow T_2$.

So $(\gamma, \rho, M.2) \in \mathcal{M}[\![B_2]\!]$. □

3.2 Instrumented operational semantics and resource soundness

The operational semantics and soundness proof in the previous section are generic and work for any instantiation of the coeffect semiring.

However, this semantics has strange implications when the coeffect is resource usage. The trouble stems from rules in the operational semantics that scale resources based on some annotation in

$\boxed{\gamma \cdot \rho \vdash_{lin} M \Downarrow T}$ *(Zero computation rules)*

EVAL-LIN-COMP-APP-ABS-ZERO
$$\frac{\gamma \cdot \rho \vdash_{lin} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^0 . M') \qquad \gamma' \cdot \rho' ,\ x \mapsto^0 \lightning \vdash_{lin} M' \Downarrow T}{\gamma \cdot \rho \vdash_{lin} M\, V \Downarrow T}$$

EVAL-LIN-COMP-RETURN-ZERO
$$\frac{\gamma \leq_{co} \overline{0}}{\gamma \cdot \rho \vdash_{lin} \mathbf{return}_0\ V \Downarrow \mathbf{return}_0\ \lightning}$$

EVAL-LIN-COMP-LETIN-RET-ZERO
$$\frac{\gamma' \cdot \rho ,\ x \mapsto^0 \lightning \vdash_{lin} N \Downarrow T \qquad \gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{lin} x \leftarrow^0 M \ \mathbf{in}\ N \Downarrow T}$$

EVAL-LIN-COMP-SPLIT-ZERO
$$\frac{\gamma' \cdot \rho ,\ x_1 \mapsto^0 \lightning ,\ x_2 \mapsto^0 \lightning \vdash_{lin} N \Downarrow T \qquad \gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{lin} \mathbf{case}_0\ V\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N \Downarrow T}$$

$\boxed{\gamma \cdot \rho \vdash_{lin} M \Downarrow T}$ *(Modified computation rules)*

EVAL-LIN-COMP-APP-ABS
$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{lin} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^q . M') \\ \gamma_2 \cdot \rho \vdash_{lin} V \Downarrow W \\ (\gamma' \cdot \rho'),\ (x \mapsto^q W) \vdash_{lin} M' \Downarrow T \\ \gamma \leq_{co} \gamma_1 + q \cdot \gamma_2 \qquad q \neq 0\end{array}}{\gamma \cdot \rho \vdash_{lin} M\, V \Downarrow T}$$

EVAL-LIN-COMP-RETURN
$$\frac{\begin{array}{c}\gamma' \cdot \rho \vdash_{lin} V \Downarrow W \\ \gamma \leq_{co} q \cdot \gamma' \qquad q \neq 0\end{array}}{\gamma \cdot \rho \vdash_{lin} \mathbf{return}_q\ V \Downarrow \mathbf{return}_q\, W}$$

EVAL-LIN-COMP-LETIN-RET
$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{lin} M \Downarrow \mathbf{return}_{q_1} W \\ \gamma_2 \cdot \rho ,\ x \mapsto^{q_1 \cdot q_2} W \vdash_{lin} N \Downarrow T \\ \gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2 \qquad q_2 \neq 0\end{array}}{\gamma \cdot \rho \vdash_{lin} x \leftarrow^{q_2} M \ \mathbf{in}\ N \Downarrow T}$$

EVAL-LIN-COMP-SPLIT
$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow (W_1, W_2) \\ \gamma_2 \cdot \rho ,\ x_1 \mapsto^q W_1 ,\ x_2 \mapsto^q W_2 \vdash_{lin} N \Downarrow T \\ \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2 \qquad q \neq 0\end{array}}{\gamma \cdot \rho \vdash_{lin} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N \Downarrow T}$$

Fig. 5. Instrumented operational semantics for resource tracking

the terms. For example, in rule EVAL-COEFF-COMP-APP-ABS, the resources used by the evaluation of the argument γ_2 are scaled by q, the grade on the function argument. The total resources of the application γ must be less precise than this scaled vector added to the resources used to evaluate the function γ_1, *i.e.*, we must have $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$. What happens if q is 0? In this case, then the demands made on the environment when computing the argument are not accounted for in γ the coeffects of the expression as a whole. This suggests that we should not evaluate the argument at all in this case, so we need to adjust our operational semantics.

In this section, we discuss how, with a few additional axioms, we can modify our operational semantics and produce a better model for resource tracking.

First, we require that the semiring is nontrivial. If $1 = 0$, resource tracking via grades is meaningless, and our general semantics degenerates to standard CBPV. Second, we require that if $0 \leq_{co} q_1 + q_2$, then $q_1 = 0$ and $q_2 = 0$. If either subterm in a value pair requires nonzero resources, we should not be able to evaluate the pair with no resources. Finally, for similar reasons we require that there be no nonzero zero divisors in the semiring, *i.e.*, if $0 = q_1 \cdot q_2$, then $q_1 = 0$ or $q_2 = 0$.

In this system, the 0 grade denotes that the corresponding variable is inaccessible, so anywhere we eliminate a value and bind it to an inaccessible variable (or return a value with grade 0), we require special treatment. Rules EVAL-COEFF-COMP-APP-ABS, EVAL-COEFF-COMP-RETURN, EVAL-COEFF-COMP-LETIN-RET, and EVAL-COEFF-COMP-SPLIT all have this property, so in each of these we modify the rule to require that the annotated grade be nonzero. We also add corresponding new rules that apply when the grade *is* zero. These rules, shown in Figure 5, use the untyped, closed value $\lightning$ in place of a closed value and discard the unneeded and unevaluated value entirely. Because values are pure, discarding an unused cannot change the effect of the computation. We do not include any rules for evaluating a $\lightning$ term, so by showing in our soundness lemma that we can still evaluate terms with variables mapped to $\lightning$ in the environment as long as those variables have a grade of 0, we show that we do not use inaccessible variables.

With these modifications, we restate the soundness theorem as follows:

Theorem 3.24 (Resource soundness theorem). *Fix γ, Γ. Let ρ be an environment which maps each variable with* nonzero *grade in $\gamma \cdot \Gamma$ to a closed value of the expected type. Then:*

(1) *If $\gamma \cdot \Gamma \vdash_{coeff} V : A$, then $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$ for some closed value W, and*
(2) *If $\gamma \cdot \Gamma \vdash_{coeff} M : B$, then $\gamma \cdot \rho \vdash_{coeff} M \Downarrow T$ for some closed terminal computation T.*

We update our logical relation with a special case for zero resources below.

Definition 3.25 (CBPV with Resource Coeffects: Logical Relation).

$$
\begin{array}{lcll}
\textit{Closed graded values} & & & \\
\mathcal{W}_0[\![A]\!] & = & \{ \lightning \} & \\
\mathcal{W}_q[\![A]\!] & = & \mathcal{W}[\![A]\!] \text{ when } q \neq 0 & \\
& & & \\
\textit{Closed values} & & & \\
\mathcal{W}[\![\mathrm{U}\, B]\!] & = & \{ \mathbf{clo}(\gamma \cdot \rho, M) & \mid (\gamma \cdot \rho, M) \in \mathcal{M}[\![B]\!] \} \\
\mathcal{W}[\![\mathbf{unit}]\!] & = & \{ () \} & \\
\mathcal{W}[\![A_1 \times A_2]\!] & = & \{ (W_1, W_2) & \mid W_1 \in \mathcal{W}[\![A_1]\!] \textit{ and } W_2 \in \mathcal{W}[\![A_2]\!] \} \\
\mathcal{W}[\![A_1 + A_2]\!] & = & \{ \mathbf{inl}\, W & \mid W \in \mathcal{W}[\![A_1]\!] \} \cup \\
& & \{ \mathbf{inr}\, W & \mid W \in \mathcal{W}[\![A_2]\!] \} \\
& & & \\
\mathcal{T}[\![\mathbf{F}_q\, A]\!] & = & \{ \mathbf{return}_q\, W & \mid W \in \mathcal{W}_q[\![A]\!] \} \\
\mathcal{T}[\![A^q \to B]\!] & = & \{ \mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M) & \mid \textit{forall } W \in \mathcal{W}_q[\![A]\!],\ ((\gamma \cdot \rho,\ x \mapsto^q W), M) \in \mathcal{M}[\![B]\!] \} \\
\mathcal{T}[\![B_1 \,\&\, B_2]\!] & = & \{ \mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle) & \mid (\gamma, \rho, M_1) \in \mathcal{M}[\![B_1]\!] \textit{ and } (\gamma, \rho, M_2) \in \mathcal{M}[\![B_2]\!] \}
\end{array}
$$

Furthermore, we update our semantic typing relation for environments to also include a special case for zero.

Definition 3.26 (CBPV with Resource Coeffects: Semantic Typing).

$$
\begin{array}{lcl}
\textit{Closures} & & \\
\mathcal{V}[\![A]\!] & = & \{ (\gamma \cdot \rho, V) \mid \gamma \cdot \rho \vdash_{lin} V \Downarrow W \textit{ and } W \in \mathcal{W}[\![A]\!] \} \\
\mathcal{M}[\![B]\!] & = & \{ (\gamma \cdot \rho, M) \mid \gamma \cdot \rho \vdash_{lin} M \Downarrow T \textit{ and } T \in \mathcal{T}[\![B]\!] \} \\
& & \\
\textit{Environments} & & \\
\gamma \cdot \Gamma \vDash \rho & = & x :^q A \in \gamma \cdot \Gamma \textit{ and } q \neq 0 \textit{ implies } x \mapsto W \in \rho \textit{ and } W \in \mathcal{W}[\![A]\!] \\
& & \\
\textit{Semantic typing} & & \\
\gamma \cdot \Gamma \vDash_{lin} V : A & = & \textit{forall } \rho, \gamma \cdot \Gamma \vDash \rho \textit{ implies } (\gamma \cdot \rho, V) \in \mathcal{V}[\![A]\!] \\
\gamma \cdot \Gamma \vDash_{lin} M : B & = & \textit{forall } \rho, \gamma \cdot \Gamma \vDash \rho \textit{ implies } (\gamma \cdot \rho, M) \in \mathcal{M}[\![B]\!]
\end{array}
$$

We can now state the fundamental lemma.

Theorem 3.27 (Fundamental lemma: resource soundness). *For all γ, Γ, if $\gamma \cdot \Gamma \vdash_{coeff} V : A$, then $\gamma \cdot \Gamma \vDash_{lin} V : A$, and for all γ, Γ, if $\gamma \cdot \Gamma \vdash_{coeff} M : B$, then $\gamma \cdot \Gamma \vDash_{lin} M : B$.*

Proof. By mutual induction on the typing derivations, using the case lemmas below. □

This lemma refers to the rule:

$$\frac{\begin{array}{c}\gamma_1 \le_{co} \overline{0}_1 \\ q \le_{co} 1 \qquad \gamma_2 \le_{co} \overline{0}_2\end{array}}{\gamma_1 \cdot \rho_1\,,\ x \mapsto^{q} W\,,\ \gamma_2 \cdot \rho_2 \vdash_{lin} x \Downarrow W}\ \text{EVAL-LIN-VAL-VAR}$$

Lemma 3.28 (Semantic lin-var). *If $\gamma_1 \le_{co} \overline{0}_1$ and $q \le_{co} 1$ and $\gamma_2 \le_{co} \overline{0}_2$ then $((\gamma_1\,, q)\,, \gamma_2) \cdot ((\Gamma_1\,, x : A)\,, \Gamma_2) \vDash_{coeff} x : A$.*

Proof. Given $((\gamma_1\,, q)\,, \gamma_2) \cdot ((\Gamma_1\,, x : A)\,, \Gamma_2) \vDash \rho$, we have that there exists some $W \in \mathcal{W}[\![A]\!]$ such that $x \mapsto W \in \rho$. (This follows because we assume in this lemma that the grade corresponding to x is q and that $q \le_{co} 1$, and we assume in this resource soundness section that $0 \le_{co} 1$ never holds.)

By rule EVAL-LIN-VAL-VAR, we get that $\gamma_1\,, q\,, \gamma_2 \cdot \rho \vdash_{lin} x \Downarrow W$, so $(\gamma_1\,, q\,, \gamma_2, \rho, x) \in \mathcal{V}[\![A]\!]$. □

This lemma refers to the rule:

$$\frac{\gamma \le_{co} \gamma'}{\gamma \cdot \rho \vdash_{lin} \{M\} \Downarrow \mathbf{clo}(\gamma' \cdot \rho, \{M\})}\ \text{EVAL-LIN-VAL-THUNK}$$

Lemma 3.29 (Semantic lin-thunk). *If $\gamma \cdot \Gamma \vDash_{coeff} M : B$ then $\gamma \cdot \Gamma \vDash_{coeff} \{M\} : \mathrm{U}\,B$.*

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, M) \in \mathcal{M}[\![B]\!]$.

So by definition, $\mathbf{clo}(\gamma \cdot \rho, M) \in \mathcal{W}[\![\mathrm{U}\,B]\!]$. Because $\gamma \le_{co} \gamma$, we have by rule EVAL-LIN-VAL-THUNK that $\gamma \cdot \rho \vdash_{lin} \{M\} \Downarrow \mathbf{clo}(\gamma \cdot \rho, \{M\})$.

So $(\gamma, \rho, \{M\}) \in \mathcal{V}[\![\mathrm{U}\,B]\!]$. □

This lemma refers to the rule:

$$\frac{\gamma \le_{co} \overline{0}}{\gamma \cdot \rho \vdash_{lin} () \Downarrow ()}\ \text{EVAL-LIN-VAL-UNIT}$$

Lemma 3.30 (Semantic lin-unit). *If $\gamma \le_{co} \overline{0}$ then $\gamma \cdot \Gamma \vDash_{coeff} () : \mathbf{unit}$.*

Proof. Suppose $\gamma \cdot \Gamma \vDash \rho$.

By definition, $() \in \mathcal{W}[\![\mathbf{unit}]\!]$, and because $\gamma \le_{co} \overline{0}$, we know by rule EVAL-LIN-VAL-UNIT that $\gamma \cdot \rho \vdash_{lin} () \Downarrow ()$.

So $(\gamma, \rho, ()) \in \mathcal{V}[\![\mathbf{unit}]\!]$. □

This lemma refers to the rule:

$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{lin} V_1 \Downarrow W_1 \\ \gamma_2 \cdot \rho \vdash_{lin} V_2 \Downarrow W_2 \\ \gamma \le_{co} \gamma_1 + \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{lin} (V_1, V_2) \Downarrow (W_1, W_2)}\ \text{EVAL-LIN-VAL-VPAIR}$$

Lemma 3.31 (Semantic lin-pair). *If $\gamma_1 \cdot \Gamma \vDash_{coeff} V_1 : A_1$ and $\gamma_2 \cdot \Gamma \vDash_{coeff} V_2 : A_2$ and $\gamma \le_{co} \gamma_1 + \gamma_2$ then $\gamma \cdot \Gamma \vDash_{coeff} (V_1, V_2) : A_1 \times A_2$.*

Proof. Suppose $\gamma \cdot \Gamma \vDash \rho$.

For any q in γ, we have $q \le_{co} q_1 + q_2$ where q_1 and q_2 are the corresponding entries in γ_1 and γ_2. So if $q = 0$, we have that $q_1 = 0$ and $q_2 = 0$. So we get that $\gamma_1 \cdot \Gamma \vDash \rho$ and $\gamma_2 \cdot \Gamma \vDash \rho$.

So by assumption, there exist $W_1 \in \mathcal{W}[\![A_1]\!]$ and $W_2 \in \mathcal{W}[\![A_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} V_1 \Downarrow W_1$ and $\gamma_2 \cdot \rho \vdash_{lin} V_2 \Downarrow W_2$.

So $(W_1, W_2) \in \mathcal{W}[\![A_1 \times A_2]\!]$.

Because $\gamma \leq_{co} \gamma_1 + \gamma_2$, we get by rule EVAL-LIN-VAL-VPAIR that $\gamma \cdot \rho \vdash_{lin} (V_1, V_2) \Downarrow (W_1, W_2)$.

So $(\gamma, \rho, (V_1, V_2)) \in \mathcal{V}[\![A_1 \times A_2]\!]$. □

This lemma refers to the rule:

$$\text{EVAL-LIN-VAL-INL}\quad \frac{\gamma \cdot \rho \vdash_{lin} V \Downarrow W}{\gamma \cdot \rho \vdash_{lin} \mathbf{inl}\, V \Downarrow \mathbf{inl}\, W}$$

Lemma 3.32 (Semantic lin-inl). *If* $\gamma \cdot \Gamma \vDash_{coeff} V : A_1$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{inl}\, V : A_1 + A_2$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, there exists by assumption $W \in \mathcal{W}[\![A_1]\!]$ such that $\gamma \cdot \rho \vdash_{lin} V \Downarrow W$.
By definition, $\mathbf{inl}\, W \in \mathcal{W}[\![A_1 + A_2]\!]$, and by rule EVAL-LIN-VAL-INL, $\gamma \cdot \rho \vdash_{lin} \mathbf{inl}\, V \Downarrow \mathbf{inl}\, W$.
So $(\gamma, \rho, \mathbf{inl}\, V) \in \mathcal{V}[\![A_1 + A_2]\!]$. □

This lemma refers to the rule:

$$\text{EVAL-LIN-VAL-INR}\quad \frac{\gamma \cdot \rho \vdash_{lin} V \Downarrow W}{\gamma \cdot \rho \vdash_{lin} \mathbf{inr}\, V \Downarrow \mathbf{inr}\, W}$$

Lemma 3.33 (Semantic lin-inr). *If* $\gamma \cdot \Gamma \vDash_{coeff} V : A_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{inr}\, V : A_1 + A_2$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, there exists by assumption $W \in \mathcal{W}[\![A_2]\!]$ such that $\gamma \cdot \rho \vdash_{lin} V \Downarrow W$.
By definition, $\mathbf{inr}\, W \in \mathcal{W}[\![A_1 + A_2]\!]$, and by rule EVAL-LIN-VAL-INR, $\gamma \cdot \rho \vdash_{lin} \mathbf{inr}\, V \Downarrow \mathbf{inr}\, W$.
So $(\gamma, \rho, \mathbf{inr}\, V) \in \mathcal{V}[\![A_1 + A_2]\!]$. □

This lemma refers to the rule:

$$\text{EVAL-LIN-COMP-ABS}\quad \frac{\gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{lin} \lambda x^q.M \Downarrow \mathbf{clo}(\gamma' \cdot \rho, \lambda x^q.M)}$$

Lemma 3.34 (Semantic lin-abs). *If* $(\gamma, q) \cdot (\Gamma, x : A) \vDash_{coeff} M : B$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \lambda x^q.M : A^q \to B$

Proof. Suppose $\gamma \cdot \Gamma \vDash \rho$.
Fix arbitrary $W \in \mathcal{W}_q[\![A]\!]$. $\gamma, q \cdot \Gamma, x : A \vDash \rho, x \mapsto W$, because if $q \neq 0$ then $W \in \mathcal{W}[\![A]\!]$.
So by assumption, $(\gamma, q, \rho, x \mapsto W, M) \in \mathcal{M}[\![B]\!]$. W was arbitrary, so we have that $\mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M) \in \mathcal{T}[\![A^q \to B]\!]$.
$\gamma \leq_{co} \gamma$, so by rule EVAL-LIN-COMP-ABS, we get that $\gamma \cdot \rho \vdash_{lin} \lambda x^q.M \Downarrow \mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M)$.
So $(\gamma, \rho, \lambda x^q.M) \in \mathcal{M}[\![A^q \to B]\!]$. □

Lemma 3.35 (Semantic lin-app). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} M : A^q \to B$ *and* $\gamma_2 \cdot \Gamma \vDash_{coeff} V : A$ *and* $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} M\, V : B$.

Proof. Suppose $\gamma \cdot \Gamma \vDash \rho$.
For any q_0 in γ, let q_1 and q_2 denote the corresponding grades in γ_1 and γ_2, so $q_0 \leq_{co} q_1 + q_2$. Then if $q_0 = 0$, we know that $q_1 = 0$ and $q \cdot q_2 = 0$, so either $q = 0$ or $q_2 = 0$. So we have that $\gamma_1 \cdot \Gamma \vDash \rho$ and that if $q \neq 0$, then $\gamma_2 \cdot \Gamma \vDash \rho$.
By assumption, we know that there exists $T' \in \mathcal{T}[\![A^q \to B]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} M \Downarrow T'$. By definition, T' must have the form $\mathbf{clo}(\gamma' \cdot \rho', \lambda x^q.M')$ such that forall $W' \in \mathcal{W}_q[\![A]\!]$, $(\gamma', q, \rho', x \mapsto W', M') \in \mathcal{M}[\![B]\!]$.
If $q \neq 0$, we have by assumption that there exists $W \in \mathcal{W}[\![A]\!]$ such that $\gamma_2 \cdot \rho \vdash_{lin} V \Downarrow W$.
Let W' denote W if $q \neq 0$ and $\lightning$ if $q = 0$, so $W' \in \mathcal{W}_q[\![A]\!]$ either way.

Then $(\gamma', q, \rho', x \mapsto W', M') \in \mathcal{M}[\![B]\!]$, i.e., there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma', q \cdot \rho', x \mapsto W' \vdash_{lin} M' \Downarrow T$.

If $q \neq 0$, we can use the assumption that $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$ and conclude from rule EVAL-LIN-COMP-APP-ABS that $\gamma \cdot \rho \vdash_{lin} M\ V \Downarrow T$.

If $q = 0$, we can use rule EVAL-LIN-COMP-APP-ABS-ZERO to conclude that $\gamma \cdot \rho \vdash_{lin} M\ V \Downarrow T$.

So either way, $(\gamma, \rho, M\ V) \in \mathcal{M}[\![B]\!]$. □

This lemma refers to the rule:

$$\text{EVAL-LIN-COMP-FORCE-THUNK}\quad \frac{\gamma \cdot \rho \vdash_{lin} V \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \{M\}) \qquad \gamma' \cdot \rho' \vdash_{lin} M \Downarrow T}{\gamma \cdot \rho \vdash_{lin} V! \Downarrow T}$$

Lemma 3.36 (Semantic lin-force). *If* $\gamma \cdot \Gamma \vDash_{coeff} V : \mathbf{U}\, B$ *then* $\gamma \cdot \Gamma \vDash_{coeff} V! : B$

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have by assumption that there exists $W \in \mathcal{W}[\![\mathbf{U}\, B]\!]$ such that $\gamma \cdot \rho \vdash_{lin} V \Downarrow W$.

By definition, W must have the form $\mathbf{clo}(\gamma' \cdot \rho', M)$ such that $(\gamma', \rho', M) \in \mathcal{M}[\![B]\!]$. So there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma' \cdot \rho' \vdash_{lin} M \Downarrow T$.

By rule EVAL-LIN-COMP-FORCE-THUNK, $\gamma \cdot \rho \vdash_{lin} V! \Downarrow T$.

So $(\gamma, \rho, V!) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.37 (Semantic lin-return). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : A$ *and* $\gamma \leq_{co} q \cdot \gamma_1$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{return}_q\ V : \mathbf{F}_q\ A$

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have that either $q = 0$ or $\gamma_1 \cdot \Gamma \vDash \rho$. (For any q_0 in γ with corresponding q_1 in γ_1, $q_0 \leq_{co} q \cdot q_1$, so if $q_0 = 0$ and $q \neq 0$ then $q_1 = 0$.)

If $q = 0$, then $\gamma \leq_{co} \overline{0}$, so by rule EVAL-LIN-COMP-RETURN-ZERO, $\gamma \cdot \rho \vdash_{lin} \mathbf{return}_q\ V \Downarrow \mathbf{return}_0\ \lightning$. $\lightning \in \mathcal{W}_0[\![A]\!]$, so $\mathbf{return}_0\ \lightning \in \mathcal{T}[\![\mathbf{F}_q\ A]\!]$.

If $q \neq 0$, then by assumption there exists $W \in \mathcal{W}[\![A]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow W$. So because $\gamma \leq_{co} q \cdot \gamma_1$, we get by rule EVAL-LIN-COMP-RETURN that $\gamma \cdot \rho \vdash_{lin} \mathbf{return}_q\ V \Downarrow \mathbf{return}_q W$, and $\mathbf{return}_q W \in \mathcal{T}[\![\mathbf{F}_q\ A]\!]$.

So either way, $(\gamma, \rho, \mathbf{return}_q\ V) \in \mathcal{M}[\![\mathbf{F}_q\ A]\!]$. □

Lemma 3.38 (Semantic lin-letin). *If* $\gamma_1 \Gamma \vDash_{coeff} M : \mathbf{F}_{q_1}\ A$ *and* $(\gamma_2, q_1 \cdot q_2) \cdot (\Gamma, x : A) \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} (q_2 \cdot \gamma_1) + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} x \leftarrow^{q_2} M\ \mathbf{in}\ N : B$

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we know that $\gamma_2 \cdot \Gamma \vDash \rho$ and either $q_2 = 0$ or $\gamma_1 \cdot \Gamma \vDash \rho$.

If $q_2 \neq 0$, then by assumption there exists $T' \in \mathcal{T}[\![\mathbf{F}_{q_1}\ A]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} M \Downarrow T'$, and T' must have the form $\mathbf{return}_{q_1} W'$ for some $W' \in \mathcal{W}_{q_1}[\![A]\!]$.

Let W be W' if $q_2 \neq 0$ and $\lightning$ otherwise.

Then $\gamma_2, q_1 \cdot q_2 \cdot \Gamma, x : A \vDash \rho, x \mapsto W$, because if $q_1 \cdot q_2 \neq 0$ then $q_1 \neq 0$ and $q_2 \neq 0$, so $W \in \mathcal{W}[\![A]\!]$.

So by assumption, there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q_1 \cdot q_2 \cdot \rho, x \mapsto W \vdash_{lin} N \Downarrow T$.

If $q_2 = 0$, then because $\gamma \leq_{co} \gamma_2$, we get by rule EVAL-LIN-COMP-LETIN-RET-ZERO that $\gamma \cdot \rho \vdash_{lin} x \leftarrow^{q_2} M\ \mathbf{in}\ N \Downarrow T$.

If $q_2 \neq 0$, then by rule EVAL-LIN-COMP-LETIN-RET, $\gamma \cdot \rho \vdash_{lin} x \leftarrow^{q_2} M\ \mathbf{in}\ N \Downarrow T$.

So either way, $(\gamma, \rho, x \leftarrow^{q_2} M\ \mathbf{in}\ N) \in \mathcal{M}[\![B]\!]$. □

Lemma 3.39 (Semantic lin-split). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : A_1 \times A_2$ *and* $((\gamma_2, q), q) \cdot ((\Gamma, x_1 : A_1), x_2 : A_2) \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \to N : B$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have that $\gamma_2 \cdot \Gamma \vDash \rho$ and either $q = 0$ or $\gamma_1 \cdot \Gamma \vDash \rho$.

If $q \neq 0$, then by assumption there exists $W' \in \mathcal{W}[\![A_1 \times A_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow W'$. W' must by definition have the form (W_1', W_2') where $W_1' \in \mathcal{W}[\![A_1]\!]$ and $W_2' \in \mathcal{W}[\![A_2]\!]$.

Let W_1 denote W_1' if $q \neq 0$ and $\lightning$ otherwise. Let W_2 denote W_2' if $q \neq 0$ and $\lightning$ otherwise. So either way, $\gamma_2, q, q \cdot \Gamma, x_1 : A_1, x_2 : A_2 \vDash \rho, x_1 \mapsto W_1, x_2 \mapsto W_2$.

Then by assumption, there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q, q \cdot \rho, x_1 \mapsto W_1, x_2 \mapsto W_2 \vdash_{lin} N \Downarrow T$.

If $q = 0$, then because $\gamma \leq_{co} \gamma_2$, we have by rule EVAL-LIN-COMP-SPLIT-ZERO that $\gamma \cdot \rho \vdash_{lin}$ $\mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \rightarrow N \Downarrow T$.

If $q \neq 0$, then by rule EVAL-LIN-COMP-SPLIT, we have that $\gamma \cdot \rho \vdash_{lin} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \rightarrow N \Downarrow T$.

So either way, $(\gamma, \rho, \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2) \rightarrow N) \in \mathcal{M}[\![B]\!]$. □

This lemma refers to the rule:

EVAL-LIN-COMP-SEQUENCE

$$\frac{\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow () \quad \gamma_2 \cdot \rho \vdash_{lin} N \Downarrow T \quad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \rho \vdash_{lin} V; N \Downarrow T}$$

Lemma 3.40 (Semantic lin-sequence). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : \mathbf{unit}$ *and* $\gamma_2 \cdot \Gamma \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} V; N : B$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have that $\gamma_1 \cdot \Gamma \vDash \rho$ and $\gamma_2 \cdot \Gamma \vDash \rho$.

So by assumption, there exists $W \in \mathcal{W}[\![\mathbf{unit}]\!]$ and $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow W$ and $\gamma_2 \cdot \rho \vdash_{lin} N \Downarrow T$.

By definition, W must be $()$. So by rule EVAL-LIN-COMP-SEQUENCE, we get that $\gamma \cdot \rho \vdash_{lin} V; N \Downarrow T$.

So $(\gamma, \rho, V; N) \in \mathcal{M}[\![B]\!]$. □

This lemma refers to the rules:

EVAL-LIN-COMP-CASE-INL

$$\frac{\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow \mathbf{inl}\ W \quad \gamma_2 \cdot \rho,\ x_1 \mapsto^q W \vdash_{lin} M_1 \Downarrow T \quad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2 \quad q \leq_{co} 1}{\gamma \cdot \rho \vdash_{lin} \mathbf{case}_q\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2 \Downarrow T}$$

EVAL-LIN-COMP-CASE-INR

$$\frac{\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow \mathbf{inr}\ W \quad \gamma_2 \cdot \rho,\ x_2 \mapsto^q W \vdash_{lin} M_2 \Downarrow T \quad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2 \quad q \leq_{co} 1}{\gamma \cdot \rho \vdash_{lin} \mathbf{case}_q\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2 \Downarrow}$$

Lemma 3.41 (Semantic lin-case). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} V : A_1 + A_2$ *and* $(\gamma_2, q) \cdot (\Gamma, x_1 : A_1) \vDash_{coeff} M_1 : B$ *and* $(\gamma_2, q) \cdot (\Gamma, x_2 : A_2) \vDash_{coeff} M_2 : B$ *and* $q \leq_{co} 1$ *and* $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff}$ $\mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2 : B$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have that $\gamma_1 \cdot \Gamma \vDash \rho$ and $\gamma_2 \cdot \Gamma \vDash \rho$, because $q \leq_{co} 1$ implies that $q \neq 0$ given our other axioms.

So there exists $W \in \mathcal{W}[\![A_1 + A_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{lin} V \Downarrow W$. By definition, W must have the form $\mathbf{inl}\ W_1$ or $\mathbf{inr}\ W_2$ for some $W_1 \in \mathcal{W}[\![A_1]\!]$ or $W_2 \in \mathcal{W}[\![A_2]\!]$.

If W has the form $\mathbf{inl}\ W_1$, then we know that $\gamma_2, q \cdot \Gamma, x_1 : A_1 \vDash \rho, x_1 \mapsto W_1$, so by assumption, there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q \cdot \rho, x_1 \mapsto W_1 \vdash_{lin} M_1 \Downarrow T$. So by rule EVAL-LIN-COMP-CASE-INL, we get that $\gamma \cdot \rho \vdash_{lin} \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2 \Downarrow T$.

If W has the form $\mathbf{inr}\ W_2$, then we know that $\gamma_2, q \cdot \Gamma, x_2 : A_2 \vDash \rho, x_2 \mapsto W_2$, so by assumption, there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q \cdot \rho, x_2 \mapsto W_2 \vdash_{lin} M_2 \Downarrow T$. So by rule EVAL-LIN-COMP-CASE-INR, we get that $\gamma \cdot \rho \vdash_{lin} \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2 \Downarrow T$.

So in either case, $(\gamma, \rho, \mathbf{case}\ V\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow M_1; \mathbf{inr}\ x_2 \rightarrow M_2) \in \mathcal{M}[\![B]\!]$. □

This lemma refers to the rule:

$$\textsc{eval-lin-comp-cpair}\quad \frac{\gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{lin} \langle M_1, M_2 \rangle \Downarrow \mathbf{clo}(\gamma' \cdot \rho, \langle M_1, M_2 \rangle)}$$

Lemma 3.42 (Semantic lin-cpair). *If* $\gamma \cdot \Gamma \vDash_{coeff} M_1 : B_1$ *and* $\gamma \cdot \Gamma \vDash_{coeff} M_2 : B_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \langle M_1, M_2 \rangle : B_1 \,\&\, B_2$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, M_1) \in \mathcal{M}[\![B_1]\!]$ and $(\gamma, \rho, M_2) \in \mathcal{M}[\![B_2]\!]$. So by definition, $\mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle) \in \mathcal{T}[\![B_1 \,\&\, B_2]\!]$. $\gamma \leq_{co} \gamma$, so by rule eval-lin-comp-cpair, $\gamma \cdot \rho \vdash_{lin} \langle M_1, M_2 \rangle \Downarrow \mathbf{clo}(\gamma \cdot \rho, \langle M_1, M_2 \rangle)$.

So $(\gamma, \rho, \langle M_1, M_2 \rangle) \in \mathcal{M}[\![B_1 \,\&\, B_2]\!]$. □

This lemma refers to the rule:

$$\textsc{eval-lin-comp-fst}\quad \frac{\gamma \cdot \rho \vdash_{lin} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle) \qquad \gamma' \cdot \rho' \vdash_{lin} M_1 \Downarrow T}{\gamma \cdot \rho \vdash_{lin} M.1 \Downarrow T}$$

Lemma 3.43 (Semantic lin-fst). *If* $\gamma \cdot \Gamma \vDash_{coeff} M : B_1 \,\&\, B_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} M.1 : B_1$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have by assumption that there exists $T' \in \mathcal{T}[\![B_1 \,\&\, B_2]\!]$ such that $\gamma \cdot \rho \vdash_{lin} M \Downarrow T'$.

By definition, T' must have the form $\mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle)$ such that $(\gamma', \rho', M_1) \in \mathcal{M}[\![B_1]\!]$. So there exists $T \in \mathcal{T}[\![B_1]\!]$ such that $\gamma' \cdot \rho' \vdash_{lin} M_1 \Downarrow T$.

So by rule eval-lin-comp-fst, $\gamma \cdot \rho \vdash_{lin} M.1 \Downarrow T$.

So $(\gamma, \rho, M.1) \in \mathcal{M}[\![B_1]\!]$. □

This lemma refers to the rule:

$$\textsc{eval-lin-comp-snd}\quad \frac{\gamma \cdot \rho \vdash_{lin} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle) \qquad \gamma' \cdot \rho' \vdash_{lin} M_2 \Downarrow T}{\gamma \cdot \rho \vdash_{lin} M.2 \Downarrow T}$$

Lemma 3.44 (Semantic lin-snd). *If* $\gamma \cdot \Gamma \vDash_{coeff} M : B_1 \,\&\, B_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} M.2 : B_2$.

Proof. Given $\gamma \cdot \Gamma \vDash \rho$, we have by assumption that there exists $T' \in \mathcal{T}[\![B_1 \,\&\, B_2]\!]$ such that $\gamma \cdot \rho \vdash_{lin} M \Downarrow T'$.

By definition, T' must have the form $\mathbf{clo}(\gamma' \cdot \rho', \langle M_1, M_2 \rangle)$ such that $(\gamma', \rho', M_2) \in \mathcal{M}[\![B_2]\!]$. So there exists $T \in \mathcal{T}[\![B_2]\!]$ such that $\gamma' \cdot \rho' \vdash_{lin} M_2 \Downarrow T$.

So by rule eval-lin-comp-snd, $\gamma \cdot \rho \vdash_{lin} M.2 \Downarrow T$.

So $(\gamma, \rho, M.2) \in \mathcal{M}[\![B_2]\!]$. □

3.3 Translation soundness

As with effects, we explore the translation of coeffect-aware CBN and CBV λ-calculi to CBPV. The type-and-coeffect system that we consider as the starting point of our translation is adapted from the simple type system of Choudhury et al. [2021] and is similar to the system developed by Abel and Bernardy [2020]. As in our CBPV extension with coeffects, the source type system is parameterized by a preordered semiring structure of coeffects and combines the typing context with γ, a vector of coeffect annotations that describe the demands on each variable. The differences between this source language and the related work are minor: the most significant one is that this system is syntax directed, instead of having a separate rule for weakening the coeffect vector.

$\boxed{\gamma \cdot \Gamma \vdash_{coeff} e : \tau}$ *(coeffect typing)*

$$\textsc{lam-coeff-var}\ \frac{\gamma_1 \leq_{co} \overline{0}_1 \quad q \leq_{co} 1 \quad \gamma_2 \leq_{co} \overline{0}_2}{\gamma_1 \cdot \Gamma_1, x :^{q} \tau, \gamma_2 \cdot \Gamma_2 \vdash_{coeff} x : \tau}$$

$$\textsc{lam-coeff-abs}\ \frac{\gamma \cdot \Gamma, (x :^{q} \tau_1) \vdash_{coeff} e : \tau_2}{\gamma \cdot \Gamma \vdash_{coeff} \lambda^{q} x.e : \tau_1^{q} \to \tau_2}$$

$$\textsc{lam-coeff-app}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e_1 : \tau_1^{q} \to \tau_2 \quad \gamma_2 \cdot \Gamma \vdash_{coeff} e_2 : \tau_1 \quad \gamma \leq_{co} \gamma_1 + q \cdot \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} e_1\ e_2 : \tau_2}$$

$$\textsc{lam-coeff-unit}\ \frac{\gamma \leq_{co} \overline{0}}{\gamma \cdot \Gamma \vdash_{coeff} () : \mathbf{unit}}$$

$$\textsc{lam-coeff-sequence}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e_1 : \mathbf{unit} \quad \gamma_2 \cdot \Gamma \vdash_{coeff} e_2 : \tau \quad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} e_1; e_2 : \tau}$$

$$\textsc{lam-coeff-pair}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e_1 : \tau_1 \quad \gamma_2 \cdot \Gamma \vdash_{coeff} e_2 : \tau_2 \quad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} (e_1, e_2) : \tau_1 \otimes \tau_2}$$

$$\textsc{lam-coeff-split}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e_1 : \tau_1 \otimes \tau_2 \quad \gamma_2 \cdot \Gamma, x_1 :^{q} \tau_1, x_2 :^{q} \tau_2 \vdash_{coeff} e_2 : \tau \quad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ e_1\ \mathbf{of}\ (x_1, x_2) \ \to\ e_2 : \tau}$$

$$\textsc{lam-coeff-inl}\ \frac{\gamma \cdot \Gamma \vdash_{coeff} e : \tau_1}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{inl}\, e : \tau_1 + \tau_2}$$

$$\textsc{lam-coeff-inr}\ \frac{\gamma \cdot \Gamma \vdash_{coeff} e : \tau_2}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{inr}\, e : \tau_1 + \tau_2}$$

$$\textsc{lam-coeff-with}\ \frac{\gamma \cdot \Gamma \vdash_{coeff} e_1 : \tau_1 \quad \gamma \cdot \Gamma \vdash_{coeff} e_2 : \tau_2}{\gamma \cdot \Gamma \vdash_{coeff} \langle e_1, e_2 \rangle : \tau_1 \,\&\, \tau_2}$$

$$\textsc{lam-coeff-fst}\ \frac{\gamma \cdot \Gamma \vdash_{coeff} e : \tau_1 \,\&\, \tau_2}{\gamma \cdot \Gamma \vdash_{coeff} e.1 : \tau_1}$$

$$\textsc{lam-coeff-snd}\ \frac{\gamma \cdot \Gamma \vdash_{coeff} e : \tau_1 \,\&\, \tau_2}{\gamma \cdot \Gamma \vdash_{coeff} e.2 : \tau_2}$$

$$\textsc{lam-coeff-case}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e : \tau_1 + \tau_2 \quad \gamma_2 \cdot \Gamma, x_1 :^{q} \tau_1 \vdash_{coeff} e_1 : \tau \quad \gamma_2 \cdot \Gamma, x_2 :^{q} \tau_2 \vdash_{coeff} e_2 : \tau \quad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2 \quad q \leq_{co} 1}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ e\ \mathbf{of}\ \mathbf{inl}\ x_1 \ \to\ e_1; \mathbf{inr}\ x_2 \ \to\ e_2 : \tau}$$

$$\textsc{lam-coeff-box}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e : \tau \quad \gamma \leq_{co} q \cdot \gamma_1}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{box}_q\ e : \Box_q\ \tau}$$

$$\textsc{lam-coeff-unbox}\ \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e_1 : \Box_{q_1}\ \tau \quad \gamma_2 \cdot \Gamma, x :^{q_1 \cdot q_2} \tau \vdash_{coeff} e_2 : \tau' \quad \gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{unbox}_{q_2}\ x = e_1\ \mathbf{in}\ e_2 : \tau'}$$

Above, the rules for variables, abstractions and applications are similar to the related features in our extension of CBPV. The variable rule requires a grade of at most 1 for that variable and at most 0 for any other variable. The abstraction rule uses a coeffect annotation on λ-expressions to track the demands a function makes on its argument. (This information also appears in the function type as $\tau_1^{q} \to \tau_2$.) When we apply the function, we scale the grade vector used to check the argument of the function by that annotation and add it to the grade vector used to check the function itself.

The terms **box** and **unbox** introduce and eliminate the modal type $\Box_q\ \tau$. The introduction form requires a grade of q on its argument build a box. This box can then be passed around as a first-class value. When we unbox the argument, the continuation has access to it with grade $q_1 \cdot q_2$. The q_1 comes from when the box was created, and the q_2 comes from the unboxing term, as in let bindings in CBPV.

3.3.1 Call-by-name translation. The languages in Choudhury et al. [2021] and Abel and Bernardy [2020] employ a call-by-name operational semantics. Therefore, we first consider a call-by-name translation to CBPV. As in the prior section, our CBN translation only includes negative products.

Type translation

$$
\begin{array}{ll}
[\![\tau_1^{q} \rightarrow \tau_2]\!]_{\mathrm{N}} & = (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}})^{q} \rightarrow [\![\tau_2]\!]_{\mathrm{N}} \\
[\![\mathbf{unit}]\!]_{\mathrm{N}} & = \mathbf{F}_1\ \mathbf{unit} \\
[\![\tau_1 \,\&\, \tau_2]\!]_{\mathrm{N}} & = [\![\tau_1]\!]_{\mathrm{N}} \,\&\, [\![\tau_2]\!]_{\mathrm{N}} \\
[\![\tau_1 + \tau_2]\!]_{\mathrm{N}} & = \mathbf{F}_1\ (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}} + \mathbf{U}\,[\![\tau_2]\!]_{\mathrm{N}}) \\
[\![\Box_q\ \tau]\!]_{\mathrm{N}} & = \mathbf{F}_q\ (\mathbf{U}\,[\![\tau]\!]_{\mathrm{N}})
\end{array}
$$

Context translation

$$
\begin{array}{ll}
[\![\varnothing]\!]_{\mathrm{N}} & = \varnothing \\
[\![\Gamma\,, x : \tau]\!]_{\mathrm{N}} & = [\![\Gamma]\!]_{\mathrm{N}}\,, x : \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}
\end{array}
$$

Term translation

$$
\begin{array}{ll}
[\![x]\!]_{\mathrm{N}} & = x! \\
[\![\lambda^{q} x.e]\!]_{\mathrm{N}} & = \lambda x^{q}.[\![e]\!]_{\mathrm{N}} \\
[\![e_1\ e_2]\!]_{\mathrm{N}} & = [\![e_1]\!]_{\mathrm{N}}\ \{[\![e_2]\!]_{\mathrm{N}}\} \\
 & \\
[\![()]\!]_{\mathrm{N}} & = \mathbf{return}_1\,() \\
[\![e_1; e_2]\!]_{\mathrm{N}} & = x \leftarrow^{1} [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ x; [\![e_2]\!]_{\mathrm{N}} \\
[\![\langle e_1, e_2\rangle]\!]_{\mathrm{N}} & = \langle [\![e_1]\!]_{\mathrm{N}}, [\![e_2]\!]_{\mathrm{N}} \rangle \\
[\![e.1]\!]_{\mathrm{N}} & = [\![e]\!]_{\mathrm{N}}.1 \\
[\![e.2]\!]_{\mathrm{N}} & = [\![e]\!]_{\mathrm{N}}.2 \\
[\![\mathbf{inl}\ e]\!]_{\mathrm{N}} & = \mathbf{return}_1\ (\mathbf{inl}\,\{[\![e]\!]_{\mathrm{N}}\}) \\
[\![\mathbf{inr}\ e]\!]_{\mathrm{N}} & = \mathbf{return}_1\ (\mathbf{inr}\,\{[\![e]\!]_{\mathrm{N}}\}) \\
[\![\mathbf{case}_q\ e\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow\ e_1; \mathbf{inr}\ x_2 \rightarrow\ e_2]\!]_{\mathrm{N}} & = x \leftarrow^{q} [\![e]\!]_{\mathrm{N}}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow\ [\![e_1]\!]_{\mathrm{N}}; \mathbf{inr}\ x_2 \rightarrow\ [\![e_2]\!]_{\mathrm{N}} \\
 & \\
[\![\mathbf{box}_q\ e]\!]_{\mathrm{N}} & = \mathbf{return}_q\ \{[\![e]\!]_{\mathrm{N}}\} \\
[\![\mathbf{unbox}_q\ x = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{N}} & = x \leftarrow^{q} [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{N}}
\end{array}
$$

In this translation, the coeffect on the λ-calculus function type translates directly to the coeffect on the CBPV function type. Furthermore, the modal type $\Box_q\ \tau$ is a graded comonad, so it can be translated to the comonad in CBPV, adding the grade to the returner type.

The CBN translation of λ terms is as usual. However, the translation of the box introduction and elimination forms follows from the definition of the CBPV comonadic type. To create a box, we return the thunked translation of the expression. To eliminate a box, we use the "letin" computation to add the thunk to the environment.

The preservation property states that we can translate each part of a CBN λ-calculus typing judgment to its CBPV version.

Lemma 3.45 (CBN and coeffects). *If $\gamma \cdot \Gamma \vdash_{coeff} e : \tau$ then $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau]\!]_N$.*

Proof. By induction on the derivation of the hypothesis, using the lemmas below. □

Lemma 3.46 (CBN translation: var). *If $\gamma_1 \leq_{co} \overline{0}_1$, $q \leq_{co} 1$, and $\gamma_2 \leq_{co} \overline{0}_2$, then $(\gamma_1\,, q\,, \gamma_2) \cdot [\![\Gamma_1\,, x : \tau\,, \Gamma_2]\!]_N \vdash_{coeff} [\![x]\!]_N : [\![\tau]\!]_N$.*

Proof. $[\![\Gamma_1\,, x : \tau\,, \Gamma_2]\!]_{\mathrm{N}} = [\![\Gamma_1]\!]_{\mathrm{N}}\,, x : \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}\,, [\![\Gamma_2]\!]_{\mathrm{N}}$.

$[\![x]\!]_{\mathrm{N}} = x!$.

By rule coeff-var, $(\gamma_1\,, q\,, \gamma_2) \cdot ([\![\Gamma_1]\!]_{\mathrm{N}}\,, x : \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}\,, [\![\Gamma_2]\!]_{\mathrm{N}}) \vdash_{coeff} x : \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}$, so by rule coeff-force, $(\gamma_1\,, q\,, \gamma_2) \cdot ([\![\Gamma_1]\!]_{\mathrm{N}}\,, x : \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}\,, [\![\Gamma_2]\!]_{\mathrm{N}}) \vdash_{coeff} x! : [\![\tau]\!]_{\mathrm{N}}$. □

Lemma 3.47 (CBN translation: abs). *If* $(\gamma, q) \cdot [\![\Gamma, x : \tau_1]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_2]\!]_N$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\lambda^q x.e]\!]_N : [\![\tau_1^q \to \tau_2]\!]_N$.

Proof. $[\![\Gamma, x : \tau_1]\!]_N = [\![\Gamma]\!]_N, x : \mathbf{U}\,[\![\tau]\!]_N$. $[\![\lambda^q x.e]\!]_N = \lambda x^q.[\![e]\!]_N$. $[\![\tau_1^q \to \tau_2]\!]_N = (\mathbf{U}\,[\![\tau_1]\!]_N)^q \to [\![\tau_2]\!]_N$.
By assumption, $(\gamma, q) \cdot [\![\Gamma]\!]_N, x : \mathbf{U}\,[\![\tau]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_2]\!]_N$, so by rule coeff-abs, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \lambda x^q.[\![e]\!]_N : (\mathbf{U}\,[\![\tau]\!]_N)^q \to [\![\tau_2]\!]_N$. □

Lemma 3.48 (CBN translation: app). *If* $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\tau_1^q \to \tau_2]\!]_N$, $\gamma_2 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau_1]\!]_N$*, and* $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1\ e_2]\!]_N : [\![\tau_2]\!]_N$.

Proof. $[\![e_1\ e_2]\!]_N = [\![e_1]\!]_N\,\{[\![e_2]\!]_N\}$. $[\![\tau_1^q \to \tau_2]\!]_N = (\mathbf{U}\,[\![\tau_1]\!]_N)^q \to [\![\tau_2]\!]_N$.
By assumption, $\gamma_2 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau_1]\!]_N$, so by rule coeff-thunk, $\gamma_2 \cdot [\![\Gamma]\!]_N \vdash_{coeff} \{[\![e_2]\!]_N\} : \mathbf{U}\,[\![\tau_1]\!]_N$.
By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : (\mathbf{U}\,[\![\tau_1]\!]_N)^q \to [\![\tau_2]\!]_N$ and $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$, so by rule coeff-app, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N\,\{[\![e_2]\!]_N\} : [\![\tau_2]\!]_N$. □

Lemma 3.49 (CBN translation: unit). *If* $\gamma \leq_{co} \overline{0}$ *then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![()]\!]_N : [\![\mathbf{unit}]\!]_N$.

Proof. $[\![()]\!]_N = \mathbf{return}_1\,()$. $[\![\mathbf{unit}]\!]_N = \mathbf{F}_1\ \mathbf{unit}$.
By rule coeff-unit, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} () : \mathbf{unit}$, and $\gamma \leq_{co} 1 \cdot \gamma$, so by rule coeff-ret, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \mathbf{return}_1\ () : \mathbf{F}_1\ \mathbf{unit}$. □

Lemma 3.50 (CBN translation: sequence). *If* $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\mathbf{unit}]\!]_N$, $\gamma_2 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau]\!]_N$*, and* $\gamma \leq_{co} \gamma_1 + \gamma_2$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1; e_2]\!]_N : [\![\tau]\!]_N$.

Proof. $[\![\mathbf{unit}]\!]_N = \mathbf{F}_1\ \mathbf{unit}$. $[\![e_1; e_2]\!]_N = x \leftarrow^1 [\![e_1]\!]_N$ **in** $x; [\![e_2]\!]_N$.
$(\gamma_2, 0) \cdot ([\![\Gamma]\!]_N, x : \mathbf{unit}) \vdash_{coeff} [\![e_2]\!]_N : [\![\tau]\!]_N$ by assumption, $(\overline{0}, 1) \cdot ([\![\Gamma]\!]_N, x : \mathbf{unit}) \vdash_{coeff} x : \mathbf{unit}$ by rule coeff-var, and $(\gamma_2, 1) \leq_{co} (\overline{0}, 1) + (\gamma_2, 0)$, so by rule coeff-sequence, $(\gamma_2, 1) \cdot ([\![\Gamma]\!]_N, x : \mathbf{unit}) \vdash_{coeff} x; [\![e_2]\!]_N : [\![\tau]\!]_N$.
By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : \mathbf{F}_1\ \mathbf{unit}$ and $\gamma \leq_{co} 1 \cdot \gamma_1 + \gamma_2$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} x \leftarrow^1 [\![e_1]\!]_N$ **in** $x; [\![e_2]\!]_N : [\![\tau]\!]_N$ □

Lemma 3.51 (CBN translation: with). *If* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\tau_1]\!]_N$, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau_2]\!]_N$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\langle e_1, e_2 \rangle]\!]_N : [\![\tau_1 \,\&\, \tau_2]\!]_N$.

Proof. $[! < e1, e2 >!]n = \langle [\![e_1]\!]_N, [\![e_2]\!]_N \rangle$.
$[\![\tau_1 \,\&\, \tau_2]\!]_N = [\![\tau_1]\!]_N \,\&\, [\![\tau_2]\!]_N$.
$\gamma [\![\Gamma]\!]_N \vdash_{coeff} \langle [\![e_1]\!]_N, [\![e_2]\!]_N \rangle : [\![\tau_1]\!]_N \,\&\, [\![\tau_2]\!]_N$ follows immediately from our assumptions by rule coeff-cpair. □

Lemma 3.52 (CBN translation: fst). *If* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1 \,\&\, \tau_2]\!]_N$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e.1]\!]_N : [\![\tau_1]\!]_N$.

Proof. $[\![\tau_1 \,\&\, \tau_2]\!]_N = [\![\tau_1]\!]_N \,\&\, [\![\tau_2]\!]_N$.
$[\![e.1]\!]_N = [\![e]\!]_N.1$.
$\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1]\!]_N \,\&\, [\![\tau_2]\!]_N$ by assumption, so by rule coeff-fst, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N.1 : [\![\tau_1]\!]_N$. □

Lemma 3.53 (CBN translation: snd). *If* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1 \,\&\, \tau_2]\!]_N$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e.2]\!]_N : [\![\tau_2]\!]_N$.

Proof. $[\![\tau_1 \,\&\, \tau_2]\!]_N = [\![\tau_1]\!]_N \,\&\, [\![\tau_2]\!]_N$.
$[\![e.2]\!]_N = [\![e]\!]_N.2$.
$\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1]\!]_N \,\&\, [\![\tau_2]\!]_N$ by assumption, so by rule coeff-snd, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N.2 : [\![\tau_2]\!]_N$. □

Lemma 3.54 (CBN translation: inl). *If* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1]\!]_N$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{inl}\ e]\!]_N : [\![\tau_1 + \tau_2]\!]_N$.

Proof. $[\![\mathbf{inl}\ e]\!]_N = \mathbf{return}_1\ (\mathbf{inl}\ \{[\![e]\!]_N\})$. $[\![\tau_1 + \tau_2]\!]_N = \mathbf{F}_1\ (\mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N)$.
$\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1]\!]_N$ by assumption, so by rule coeff-thunk, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \{[\![e]\!]_N\} : \mathbf{U}\ [\![\tau_1]\!]_N$.
By rule coeff-inl, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \mathbf{inl}\ \{[\![e]\!]_N\} : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N$.
$\gamma \leq_{co} 1 \cdot \gamma$, so by rule coeff-ret, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \mathbf{return}_1\ (\mathbf{inl}\ \{[\![e]\!]_N\}) : \mathbf{F}_1\ (\mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N)$. □

Lemma 3.55 (CBN translation: inr). *If* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_2]\!]_N$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{inr}\ e]\!]_N : [\![\tau_1 + \tau_2]\!]_N$.

Proof. $[\![\mathbf{inr}\ e]\!]_N = \mathbf{return}_1\ (\mathbf{inr}\ \{[\![e]\!]_N\})$. $[\![\tau_1 + \tau_2]\!]_N = \mathbf{F}_1\ (\mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N)$.
$\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_2]\!]_N$ by assumption, so by rule coeff-thunk, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \{[\![e]\!]_N\} : \mathbf{U}\ [\![\tau_2]\!]_N$.
By rule coeff-inr, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \mathbf{inr}\ \{[\![e]\!]_N\} : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N$.
$\gamma \leq_{co} 1 \cdot \gamma$, so by rule coeff-ret, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \mathbf{return}_1\ (\mathbf{inr}\ \{[\![e]\!]_N\}) : \mathbf{F}_1\ (\mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N)$. □

Lemma 3.56 (CBN translation: case). *If* $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_1 + \tau_2]\!]_N$, $(\gamma_2, q) \cdot [\![\Gamma, x_1 : \tau_1]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\tau]\!]_N$, $(\gamma_2, q) \cdot [\![\Gamma, x_2 : \tau_2]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau]\!]_N$, $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$*, and* $q \leq_{co} 1$ *then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{case}_q\ e\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow e_1; \mathbf{inr}\ x_2 \rightarrow e_2]\!]_N : [\![\tau]\!]_N$.

Proof. $[\![\mathbf{case}\ e\ \mathbf{of}\ x_1 \rightarrow e_1; x_2 \rightarrow e_2]\!]_N = x \leftarrow^q [\![e]\!]_N\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow [\![e_1]\!]_N; \mathbf{inr}\ x_2 \rightarrow [\![e_2]\!]_N$. $[\![\tau_1 + \tau_2]\!]_N = \mathbf{F}_1\ (\mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N)$. $[\![\Gamma, x_1 : \tau_1]\!]_N = [\![\Gamma]\!]_N, x_1 : \mathbf{U}\ [\![\tau_1]\!]_N$. $[\![\Gamma, x_2 : \tau_2]\!]_N = [\![\Gamma]\!]_N, x_2 : \mathbf{U}\ [\![\tau_2]\!]_N$.
By rule coeff-var, $(\bar{0}, 1) \cdot ([\![\Gamma]\!]_N, x : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N) \vdash_{coeff} x : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N$.
By assumption, $(\gamma_2, 0, q) \cdot ([\![\Gamma]\!]_N, x : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N, x_1 : \mathbf{U}\ [\![\tau_1]\!]_N) \vdash_{coeff} [\![e_1]\!]_N : [\![\tau]\!]_N$, $(\gamma_2, 0, q) \cdot ([\![\Gamma]\!]_N, x : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N, x_2 : \mathbf{U}\ [\![\tau_2]\!]_N) \vdash_{coeff} [\![e_2]\!]_N : [\![\tau]\!]_N$, and $q \leq_{co} 1$.
$(\gamma_2, q) \leq_{co} q \cdot (\bar{0}, 1) + (\gamma_2, 0)$, so by rule coeff-case, $(\gamma_2, q) \cdot ([\![\Gamma]\!]_N, x : \mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N) \vdash_{coeff} \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow [\![e_1]\!]_N; \mathbf{inr}\ x_2 \rightarrow [\![e_2]\!]_N : [\![\tau]\!]_N$.
By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : \mathbf{F}_1\ (\mathbf{U}\ [\![\tau_1]\!]_N + \mathbf{U}\ [\![\tau_2]\!]_N)$ and $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} x \leftarrow^q [\![e]\!]_N\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \rightarrow [\![e_1]\!]_N; \mathbf{inr}\ x_2 \rightarrow [\![e_2]\!]_N : [\![\tau]\!]_N$ □

Lemma 3.57 (CBN translation: box). *If* $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau]\!]_N$ *and* $\gamma \leq_{co} q \cdot \gamma_1$ *then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{box}_q\ e]\!]_N : [\![\Box_q\ \tau]\!]_N$.

Proof. $[\![\mathbf{box}_q\ e]\!]_N = \mathbf{return}_q\ \{[\![e]\!]_N\}$. $[\![\Box_q\ \tau]\!]_N = \mathbf{F}_q\ (\mathbf{U}\ [\![\tau]\!]_N)$.
By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau]\!]_N$, so by rule coeff-thunk, $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} \{[\![e]\!]_N\} : \mathbf{U}\ [\![\tau]\!]_N$.
By assumption, $\gamma \leq_{co} q \cdot \gamma_1$, so by rule coeff-ret, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} \mathbf{return}_q\ \{[\![e]\!]_N\} : \mathbf{F}_q\ (\mathbf{U}\ [\![\tau]\!]_N)$. □

Lemma 3.58 (CBN translation: unbox). *If* $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\Box_{q_1}\ \tau]\!]_N$, $(\gamma_2, q_1 \cdot q_2) \cdot [\![\Gamma, x : \tau]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau']\!]_N$*, and* $\gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2$*, then* $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{unbox}_{q_2}\ x = e_1\ \mathbf{in}\ e_2]\!]_N : [\![\tau']\!]_N$.

Proof. $[\![\Box_{q_1}\ \tau]\!]_N = \mathbf{F}_{q_1}\ (\mathbf{U}\ [\![\tau]\!]_N)$. $[\![\Gamma, x : \tau]\!]_N = [\![\Gamma]\!]_N, x : \mathbf{U}\ [\![\tau]\!]_N$. $[\![\mathbf{unbox}_{q_2}\ x = e_1\ \mathbf{in}\ e_2]\!]_N = x \leftarrow^{q_2} [\![e_1]\!]_N\ \mathbf{in}\ [\![e_2]\!]_N$. By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : \mathbf{F}_{q_1}\ (\mathbf{U}\ [\![\tau]\!]_N)$, $(\gamma_2, q_1 \cdot q_2) \cdot ([\![\Gamma]\!]_N, x : \mathbf{U}\ [\![\tau]\!]_N) \vdash_{coeff} [\![e_2]\!]_N : [\![\tau']\!]_N$, and $\gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_N \vdash_{coeff} x \leftarrow^{q_2} [\![e_1]\!]_N\ \mathbf{in}\ [\![e_2]\!]_N : [\![\tau']\!]_N$. □

3.3.2 *Call-by-value translation.*

The type-and-coeffect system above can also be given a *call-by-value* interpretation. Therefore, in this subsection, we define the CBV translation of this system to CBPV and show that it too is type and coeffect preserving.

We observe here that changing the operational semantics does not invalidate coeffect tracking: the evaluation of a term makes the *same* demands on its context no matter whether it is evaluated using a CBN or CBV semantics. This differs from the previous section: there the type-and-effect system was only valid for a call-by-value semantics.

As we discuss below, supporting this CBV translation motivates part of our extension of CBPV with coeffects.

Type translation

$$
\begin{array}{ll}
[\![\tau_1^{q} \to \tau_2]\!]_{\mathrm{v}} & = \mathbf{U}\,([\![\tau_1]\!]_{\mathrm{v}}^{q} \to \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{v}}) \\
[\![\mathbf{unit}]\!]_{\mathrm{v}} & = \mathtt{unit} \\
[\![\tau_1 \otimes \tau_2]\!]_{\mathrm{v}} & = [\![\tau_1]\!]_{\mathrm{v}} \times [\![\tau_2]\!]_{\mathrm{v}} \\
[\![\tau_1 + \tau_2]\!]_{\mathrm{v}} & = [\![\tau_1]\!]_{\mathrm{v}} + [\![\tau_2]\!]_{\mathrm{v}} \\
[\![\Box_q\ \tau]\!]_{\mathrm{v}} & = \mathbf{U}\,(\mathbf{F}_q\ [\![\tau]\!]_{\mathrm{v}})
\end{array}
$$

Context translation

$$
\begin{array}{ll}
[\![\varnothing]\!]_{\mathrm{v}} & = \varnothing \\
[\![\Gamma, x : \tau]\!]_{\mathrm{v}} & = [\![\Gamma]\!]_{\mathrm{v}}, x : [\![\tau]\!]_{\mathrm{v}}
\end{array}
$$

Term translation

$$
\begin{array}{ll}
[\![x]\!]_{\mathrm{v}} & = \mathbf{return}_1\ x \\
[\![\lambda^{q} x.e]\!]_{\mathrm{v}} & = \mathbf{return}_1\ \{\lambda x^{q}.[\![e]\!]_{\mathrm{v}}\} \\
[\![e_1\ e_2]\!]_{\mathrm{v}} & = x \leftarrow^{1} [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ y \leftarrow^{q} [\![e_2]\!]_{\mathrm{v}}\ \mathbf{in}\ x!\ y \\
[\![()]\!]_{\mathrm{v}} & = \mathbf{return}_1\,() \\
[\![e_1; e_2]\!]_{\mathrm{v}} & = x \leftarrow^{1} [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ x; [\![e_2]\!]_{\mathrm{v}} \\
[\![(e_1, e_2)]\!]_{\mathrm{v}} & = x_1 \leftarrow^{1} [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ (x_2 \leftarrow^{1} [\![e_2]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}_1\ (x_1, x_2)) \\
[\![\mathbf{case}_q\ e_1\ \mathbf{of}\ (x_1, x_2)\ \to\ e_2]\!]_{\mathrm{v}} & = x \leftarrow^{q} [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ (x_1, x_2)\ \to\ [\![e_2]\!]_{\mathrm{v}} \\
[\![\mathbf{inl}\ e]\!]_{\mathrm{v}} & = x \leftarrow^{1} [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}_1\ (\mathbf{inl}\ x) \\
[\![\mathbf{inr}\ e]\!]_{\mathrm{v}} & = x \leftarrow^{1} [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}_1\ (\mathbf{inr}\ x) \\
[\![\mathbf{case}_q\ e\ \mathbf{of}\ \mathbf{inl}\ x_1 \to\ e_1; \mathbf{inr}\ x_2 \to\ e_2]\!]_{\mathrm{v}} & = x \leftarrow^{q} [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \to\ [\![e_1]\!]_{\mathrm{v}}; \mathbf{inr}\ x_2 \to\ [\![e_2]\!]_{\mathrm{v}} \\
 & \\
[\![\mathbf{box}_q\ e]\!]_{\mathrm{v}} & = \mathbf{return}_1\ \{x \leftarrow^{q} [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}_q\ x\} \\
[\![\mathbf{unbox}_q\ x = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{v}} & = y \leftarrow^{q} [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ x \leftarrow^{q} y!\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{v}}
\end{array}
$$

As above, we propagate the coeffect from the λ-calculus function type directly to the CBPV function type. Similarly, we propagate the grade in the modal type to the inner returner type and let binding in CBPV.

The translations for variables and abstractions are also straightforward. Like the usual CBV translation, many cases of this translation use **return** to construct values. In each case, the annotation on this return is always one, as only one copy of the constructed value is returned by the translation.

However, in the application case, the ability for the "letin" computation to duplicate the right-hand side is needed in the case of the evaluation of the argument to the function. We need to make sure that this argument has coeffect q in the application, and this is only possible with this annotation.

The translation of the **box** term follows its type definition. In CBPV, the computation $x \leftarrow^{1} M$ **in** $\mathbf{return}_1\ x$ is equivalent to M, but the computation $x \leftarrow^{q} M$ **in** $\mathbf{return}_q\ x$ corresponds to duplicating M q times in a resource usage coeffect. This propagation of the grade is exactly the feature that we need to translate the **box** term. Note also in this translation that **box** must thunk its argument, even in a call-by-value language. In the translation of the **unbox** term, we must also use the annotation capability of the "letin" computation (twice) to mirror the annotation in the source language.

As before, the preservation property states that we can translate each part of a CBV λ-calculus typing judgment to its CBPV version. As in the effects section, we include a returner type in the CBPV version, and we give it a grade of 1, representing default usage of the value it contains.

Theorem 3.59 (CBV and coeffects). *If* $\gamma \cdot \Gamma \vdash_{coeff} e : \tau$ *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$.

Proof. By induction on the derivation of the hypothesis, using the case lemmas below. □

Lemma 3.60 (CBV translation: var). *If* $\gamma_1 \leq_{co} \overline{0}_1$, $q \leq_{co} 1$, *and* $\gamma_2 \leq_{co} \overline{0}_2$, *then* $(\gamma_1, q, \gamma_2) \cdot [\![\Gamma_1, x : \tau, \Gamma_2]\!]_V \vdash_{coeff} [\![x]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$.

Proof. $[\![\Gamma_1, x : \tau, \Gamma_2]\!]_V = [\![\Gamma_1]\!]_V, x : [\![\tau]\!]_V, [\![\Gamma_2]\!]_V$.

$[\![x]\!]_V = \mathbf{return}_1\ x$.

We have $(\gamma_1, q, \gamma_2) \cdot ([\![\Gamma_1]\!]_V, x : [\![\tau]\!]_V, [\![\Gamma_2]\!]_V) \vdash_{coeff} x : [\![\tau]\!]_V$ by rule coeff-var.

We also have $(\gamma_1, q, \gamma_2) \leq_{co} 1 \cdot (\gamma_1, q, \gamma_2)$, so by rule coeff-ret, $(\gamma_1, q, \gamma_2) \cdot ([\![\Gamma_1]\!]_V, x : [\![\tau]\!]_V, [\![\Gamma_2]\!]_V) \vdash_{coeff} \mathbf{return}_1\ x : \mathbf{F}_1\ [\![\tau]\!]_V$. □

Lemma 3.61 (CBV translation: abs). *If* $(\gamma, q) \cdot [\![\Gamma, x : \tau_1]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\lambda^q x.e]\!]_V : \mathbf{F}_1\ [\![\tau_1^q \to \tau_2]\!]_V$.

Proof. $[\![\Gamma, x : \tau_1]\!]_V = [\![\Gamma]\!]_V, x : [\![\tau]\!]_V$. $[\![\lambda^q x.e]\!]_V = \mathbf{return}_1\ \{\lambda x^q.[\![e]\!]_V\}$. $[\![\tau_1^q \to \tau_2]\!]_V = \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)$. By rule coeff-abs, $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} \lambda x^q.[\![e]\!]_V : [\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V$, so by rule coeff-thunk, $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} \{\lambda x^q.[\![e]\!]_V\} : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)$, and by rule coeff-ret, since $\gamma \leq_{co} 1 \cdot \gamma$, $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} \mathbf{return}_1\ \{\lambda x^q.[\![e]\!]_V\} : \mathbf{F}_1\ (\mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V))$. □

Lemma 3.62 (CBV translation: app). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\tau_1^q \to \tau_2]\!]_V$ *and* $\gamma_2 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_1]\!]_V$ *and* $\gamma \leq_{co} \gamma_1 + q \cdot \gamma_2$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1\ e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$.

Proof. By assumption, we have

$$[\![e_1\ e_2]\!]_V = x \leftarrow^1 [\![e_1]\!]_V\ \mathbf{in}\ y \leftarrow^q [\![e_2]\!]_V\ \mathbf{in}\ x!\,y$$

and

$$[\![\tau_1^q \to \tau_2]\!]_V = \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V).$$

By rule coeff-var, $(\overline{0}, 1, 0) \cdot (([\![\Gamma]\!]_V, x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)), y : [\![\tau_1]\!]_V) \vdash_{coeff} x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)$, so by rule coeff-force, $(\overline{0}, 1, 0) \cdot (([\![\Gamma]\!]_V, x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)), y : [\![\tau_1]\!]_V) \vdash_{coeff} x! : [\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V$.

By rule coeff-var, $(\overline{0}, 0, 1) \cdot (([\![\Gamma]\!]_V, x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)), y : [\![\tau_1]\!]_V) \vdash_{coeff} y : [\![\tau_1]\!]_V$, and $(\overline{0}, 1, q) \leq_{co} (\overline{0}, 1, 0) + q \cdot (\overline{0}, 0, 1)$, so by rule coeff-app $(\overline{0}, 1, q) \cdot (([\![\Gamma]\!]_V, x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)), y : [\![\tau_1]\!]_V) \vdash_{coeff} x!\,y : \mathbf{F}_1\ [\![\tau_2]\!]_V$

By assumption, $(\gamma_2, 0) \cdot ([\![\Gamma]\!]_V, x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)) \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_1]\!]_V$ and $(q \cdot \gamma_2, 1) \leq_{co} q \cdot (\gamma_2, 0) + ($ so by rule coeff-letin $(q \cdot \gamma_2, 1) \cdot ([\![\Gamma]\!]_V, x : \mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V)) \vdash_{coeff} y \leftarrow^q [\![e_2]\!]_V\ \mathbf{in}\ x!\,y : \mathbf{F}_1\ [\![\tau_2]\!]_V$

By assumption, $\gamma \leq_{co} 1 \cdot \gamma_1 + q \cdot \gamma_2$ and $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ (\mathbf{U}\,([\![\tau_1]\!]_V^q \to \mathbf{F}_1\ [\![\tau_2]\!]_V))$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} x \leftarrow^1 [\![e_1]\!]_V\ \mathbf{in}\ y \leftarrow^q [\![e_2]\!]_V\ \mathbf{in}\ x!\,y : \mathbf{F}_1\ [\![\tau_2]\!]_V$. □

Lemma 3.63 (CBV translation: unit). *If* $\gamma \leq_{co} \overline{0}$ *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![()]\!]_V : \mathbf{F}_1\ [\![\mathbf{unit}]\!]_V$.

Proof. $[\![()]\!]_V = \mathbf{return}_1()$. $[\![\mathbf{unit}]\!]_V = \mathtt{unit}$. By rule coeff-unit, $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} () : \mathbf{unit}$, so by rule coeff-unit, since $\gamma \leq_{co} 1 \cdot \gamma$, $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} \mathbf{return}_1\ () : \mathbf{F}_1\ \mathbf{unit}$. □

Lemma 3.64 (CBV translation: sequence). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\mathbf{unit}]\!]_V$, $\gamma_2 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$, *and* $\gamma \leq_{co} \gamma_1 + \gamma_2$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1; e_2]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$.

Proof. $[\![e_1; e_2]\!]_\mathrm{V} = x \leftarrow^1 [\![e_1]\!]_\mathrm{V}\ \mathbf{in}\ x; [\![e_2]\!]_\mathrm{V}$.

$(\overline{0}, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x : \mathbf{unit}) \vdash_{coeff} x : \mathbf{unit}$ by rule coeff-var. By assumption, $(\gamma_2, 0) \cdot ([\![\Gamma]\!]_\mathrm{V}, x : \mathbf{unit}) \vdash_{coeff} [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau]\!]_\mathrm{V}$, and $(\gamma_2, 1) \leq_{co} (\overline{0}, 1) + (\gamma_2, 0)$, so by rrefcoeff-sequence, $(\gamma_2, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x : \mathbf{unit}) \vdash_{coeff} x; [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau]\!]_\mathrm{V}$.

By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} [\![e_1]\!]_\mathrm{V} : \mathbf{F}_1\ \mathbf{unit}$ and $\gamma \leq_{co} 1 \cdot \gamma_1 + \gamma_2$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} x \leftarrow^1 [\![e_1]\!]_\mathrm{V}\ \mathbf{in}\ x; [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau]\!]_\mathrm{V}$. ◻

Lemma 3.65 (CBV translation: pair). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\tau_1]\!]_V$, $\gamma_2 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$, *and* $\gamma \leq_{co} \gamma_1 + \gamma_2$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![(e_1, e_2)]\!]_V : \mathbf{F}_1\ [\![\tau_1 \otimes \tau_2]\!]_V$.

Proof. $[\![(e_1, e_2)]\!]_\mathrm{V} = x_1 \leftarrow^1 [\![e_1]\!]_\mathrm{V}\ \mathbf{in}\ x_2 \leftarrow^1 [\![e_2]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{return}_1\ (x_1, x_2)$. $[\![\tau_1 \otimes \tau_2]\!]_\mathrm{V} = [\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V}$.

By rule coeff-var, $(\overline{0}, 1, 0) \cdot ([\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}, x_2 : [\![\tau_2]\!]_\mathrm{V}) \vdash_{coeff} x_1 : [\![\tau_1]\!]_\mathrm{V}$ and $(\overline{0}, 0, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}, x_2 : [\![\tau_2]\!]_\mathrm{V}) \vdash_{coeff} x_2 : [\![\tau_2]\!]_\mathrm{V}$, and $(\overline{0}, 1, 1) \leq_{co} (\overline{0}, 1, 0) + (\overline{0}, 0, 1)$, so by rule coeff-pair, $(\overline{0}, 1, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}, x_2 : [\![\tau_2]\!]_\mathrm{V}) \vdash_{coeff} (x_1, x_2) : ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})$.

$(\overline{0}, 1, 1) \leq_{co} 1 \cdot (\overline{0}, 1, 1)$, so by rule coeff-ret $(\overline{0}, 1, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}, x_2 : [\![\tau_2]\!]_\mathrm{V}) \vdash_{coeff} \mathbf{return}_1\ (x_1, x_2) : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})$.

$(\gamma_2, 0) \cdot ([\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}) \vdash_{coeff} [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau_2]\!]_\mathrm{V}$ by assumption, and $(\gamma_2, 1) \leq_{co} 1 \cdot (\gamma_2, 0) + (\overline{0}, 1)$, so by rule coeff-letin, $(\gamma_2, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}) \vdash_{coeff} x_2 \leftarrow^1 [\![e_2]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{return}_1\ (x_1, x_2) : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})$

$\gamma_1 \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} [\![e_1]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau_1]\!]_\mathrm{V}$ by assumption, and $\gamma \leq_{co} 1 \cdot \gamma_1 + \gamma_2$, so by rule letin, $\gamma \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} x_1 \leftarrow^1 [\![e_1]\!]_\mathrm{V}\ \mathbf{in}\ x_2 \leftarrow^1 [\![e_2]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{return}_1\ (x_1, x_2) : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})$ ◻

Lemma 3.66 (CBV translation: split). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\tau_1 \otimes \tau_2]\!]_V$, $(\gamma_2, q, q) \cdot [\![\Gamma, x_1 : \tau_1, x_2 : \tau_2]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$, *and* $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{case}_q\ e_1\ \mathbf{of}\ (x_1, x_2) \rightarrow e_2]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$.

Proof. $[\![\mathbf{case}_q\ e_1\ \mathbf{of}\ (x_1, x_2) \rightarrow e_2]\!]_\mathrm{V} = x \leftarrow^1 [\![e_1]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ (x_1, x_2) \rightarrow [\![e_2]\!]_\mathrm{V}$. $[\![\tau_1 \otimes \tau_2]\!]_\mathrm{V} = [\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V}$. $[\![\Gamma, x_1 : \tau_1, x_2 : \tau_2]\!]_\mathrm{V} = [\![\Gamma]\!]_\mathrm{V}, x_1 : [\![\tau_1]\!]_\mathrm{V}, x_2 : [\![\tau_2]\!]_\mathrm{V}$.

By assumption, $(\gamma_2, 0, q, q) \cdot (((([\![\Gamma]\!]_\mathrm{V}, x : [\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V}), x_1 : [\![\tau_1]\!]_\mathrm{V}), x_2 : [\![\tau_1]\!]_\mathrm{V}) \vdash_{coeff} [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau]\!]_\mathrm{V}$. $(\overline{0}, 1) \cdot ([\![\Gamma]\!]_\mathrm{V}, x : [\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V}) \vdash_{coeff} x : [\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V}$ by rule coeff-var, and $(\gamma_2, q) \leq_{co} q \cdot (\overline{0}, 1) + (\gamma_2, 0)$, so by rule coeff-split, $(\gamma_2, q) \cdot ([\![\Gamma]\!]_\mathrm{V}, x : ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})) \vdash_{coeff} \mathbf{case}_q\ x\ \mathbf{of}\ (x_1, x_2) \rightarrow [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})$.

By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} [\![e_1]\!]_\mathrm{V} : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} \times [\![\tau_2]\!]_\mathrm{V})$ and $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} x \leftarrow^q [\![e_1]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ (x_1, x_2) \rightarrow [\![e_2]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau]\!]_\mathrm{V}$ ◻

Lemma 3.67 (CBV translation: inl). *If* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau_1]\!]_V$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{inl}\ e]\!]_V : \mathbf{F}_1\ [\![\tau_1 + \tau_2]\!]_V$.

Proof. $[\![\mathbf{inl}\ e]\!]_\mathrm{V} = x \leftarrow^1 [\![e]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{return}_1\ (\mathbf{inl}\ x)$. $[\![\tau_1 + \tau_2]\!]_\mathrm{V} = [\![\tau_1]\!]_\mathrm{V} + [\![\tau_2]\!]_\mathrm{V}$.

By rule coeff-var, $(\emptyset, 1) \cdot x : [\![\tau_1]\!]_\mathrm{V} \vdash_{coeff} x : [\![\tau_1]\!]_\mathrm{V}$, so by rule coeff-inl, $(\emptyset, 1) \cdot x : [\![\tau_1]\!]_\mathrm{V} \vdash_{coeff} \mathbf{inl}\ x : [\![\tau_1]\!]_\mathrm{V} + [\![\tau_2]\!]_\mathrm{V}$

$(\emptyset, 1) \leq_{co} 1 \cdot (\emptyset, 1)$, so by rule coeff-ret $(\emptyset, 1) \cdot x : [\![\tau_1]\!]_\mathrm{V} \vdash_{coeff} \mathbf{return}_1\ (\mathbf{inl}\ x) : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} + [\![\tau_2]\!]_\mathrm{V})$.

By assumption, $\gamma \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} [\![e]\!]_\mathrm{V} : \mathbf{F}_1\ [\![\tau_1]\!]_\mathrm{V}$, and $\gamma \leq_{co} 1 \cdot \gamma$, so by rule coeff-letin, $\gamma \cdot [\![\Gamma]\!]_\mathrm{V} \vdash_{coeff} x \leftarrow^1 [\![e]\!]_\mathrm{V}\ \mathbf{in}\ \mathbf{return}_1\ (\mathbf{inl}\ x) : \mathbf{F}_1\ ([\![\tau_1]\!]_\mathrm{V} + [\![\tau_2]\!]_\mathrm{V})$. ◻

Lemma 3.68 (CBV translation: inr). *If* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{inr}\ e]\!]_V : \mathbf{F}_1\ [\![\tau_1 + \tau_2]\!]_V$.

Proof. $[\![\mathbf{inr}\ e]\!]_{\mathrm{V}} = x \leftarrow^{1} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{return}_1\ (\mathbf{inr}\ x)$ $[\![\tau_1 + \tau_2]\!]_{\mathrm{V}} = [\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}}$

By rule COEFF-VAR, $(\emptyset, 1) \cdot x : [\![\tau_2]\!]_{\mathrm{V}} \vdash_{coeff} x : [\![\tau_2]\!]_{\mathrm{V}}$, so by rule COEFF-INR, $(\emptyset, 1) \cdot x : [\![\tau_2]\!]_{\mathrm{V}} \vdash_{coeff} \mathbf{inr}\ x : [\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}}$

$(\emptyset, 1) \leq_{co} 1 \cdot (\emptyset, 1)$, so by rule COEFF-RET $(\emptyset, 1) \cdot x : [\![\tau_2]\!]_{\mathrm{V}} \vdash_{coeff} \mathbf{return}_1\ (\mathbf{inr}\ x) : \mathbf{F}_1\ ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})$.

By assumption, $\gamma \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} [\![e]\!]_{\mathrm{V}} : \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{V}}$, and $\gamma \leq_{co} 1 \cdot \gamma$, so by rule COEFF-LETIN, $\gamma \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} x \leftarrow^{1} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{return}_1\ (\mathbf{inr}\ x) : \mathbf{F}_1\ ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})$. □

Lemma 3.69 (CBV translation: case). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau_1 + \tau_2]\!]_V$, $(\gamma_2, q) \cdot [\![\Gamma, x_1 : \tau_1]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$, $(\gamma_2, q) \cdot [\![\Gamma, x_2 : \tau_2]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$, $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$, *and* $q \leq_{co} 1$ *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{case}_q\ e\ \mathbf{of}\ \mathbf{inl}\ x_1 \to e_1; \mathbf{inr}\ x_2 \to e_2]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$.

Proof. $[\![\mathbf{case}\ e\ \mathbf{of}\ x_1 \to e_1; x_2 \to e_2]\!]_{\mathrm{V}} = x \leftarrow^{q} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \to [\![e_1]\!]_{\mathrm{V}}; \mathbf{inr}\ x_2 \to [\![e_2]\!]_{\mathrm{V}}$. $[\![\tau_1 + \tau_2]\!]_{\mathrm{V}} = [\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}}$. $[\![\Gamma, x_1 : \tau_1]\!]_{\mathrm{V}} = [\![\Gamma]\!]_{\mathrm{V}}, x_1 : [\![\tau_1]\!]_{\mathrm{V}}$. $[\![\Gamma, x_2 : \tau_2]\!]_{\mathrm{V}} = [\![\Gamma]\!]_{\mathrm{V}}, x_2 : [\![\tau_2]\!]_{\mathrm{V}}$.

By assumption, $(\gamma_2, 0, q) \cdot (([\![\Gamma]\!]_{\mathrm{V}}, x : ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})), x_1 : [\![\tau_1]\!]_{\mathrm{V}}) \vdash_{coeff} [\![e_1]\!]_{\mathrm{V}} : \mathbf{F}_1\ [\![\tau]\!]_{\mathrm{V}}$ and $(\gamma_2, 0, q) \cdot (([\![\Gamma]\!]_{\mathrm{V}}, x : ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})), x_2 : [\![\tau_2]\!]_{\mathrm{V}}) \vdash_{coeff} [\![e_2]\!]_{\mathrm{V}} : \mathbf{F}_1\ [\![\tau]\!]_{\mathrm{V}}$

By rule COEFF-VAR, $(\overline{0}, 1) \cdot ([\![\Gamma]\!]_{\mathrm{V}}, x : ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})) \vdash_{coeff} x : ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})$. $(\gamma_2, q) \leq_{co} q \cdot (\overline{0}, 1) + (\gamma_2, 0)$, and $q \leq_{co} 1$, so by rule COEFF-CASE, $(\gamma_2, q) \cdot ([\![\Gamma]\!]_{\mathrm{V}}, x : ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})) \vdash_{coeff} \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \to [\![e_1]\!]_{\mathrm{V}}; \mathbf{inr}\ x_2 \to [\![e_2]\!]_{\mathrm{V}} : \mathbf{F}_1\ [\![\tau]\!]_{\mathrm{V}}$

By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} [\![e]\!]_{\mathrm{V}} : \mathbf{F}_1\ ([\![\tau_1]\!]_{\mathrm{V}} + [\![\tau_2]\!]_{\mathrm{V}})$, and $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$, so by rule COEFF-LETIN, $\gamma \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} x \leftarrow^{q} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{case}_q\ x\ \mathbf{of}\ \mathbf{inl}\ x_1 \to [\![e_1]\!]_{\mathrm{V}}; \mathbf{inr}\ x_2 \to [\![e_2]\!]_{\mathrm{V}} : \mathbf{F}_1\ [\![\tau]\!]_{\mathrm{V}}$. □

Lemma 3.70 (CBV translation: box). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$ *and* $\gamma \leq_{co} q \cdot \gamma_1$ *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{box}_q\ e]\!]_V : \mathbf{F}_1\ [\![\Box_q\ \tau]\!]_V$.

Proof. We have

$$[\![\mathbf{box}_q\ e]\!]_{\mathrm{V}} = \mathbf{return}_1\ \{x \leftarrow^{q} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{return}_q\ x\}$$

and

$$[\![\Box_q\ \tau]\!]_{\mathrm{V}} = \mathbf{U}\,(\mathbf{F}_q\ [\![\tau]\!]_{\mathrm{V}})$$

by rule COEFF-VAR, $(\overline{0}, 1) \cdot ([\![\Gamma]\!]_{\mathrm{V}}, x : [\![\tau]\!]_{\mathrm{V}}) \vdash_{coeff} x : [\![\tau]\!]_{\mathrm{V}}$ and $(\overline{0}, q) \leq_{co} q \cdot (\overline{0}, 1)$, we can use rule COEFF-RET, $(\overline{0}, q) \cdot ([\![\Gamma]\!]_{\mathrm{V}}, x : [\![\tau]\!]_{\mathrm{V}}) \vdash_{coeff} \mathbf{return}_q\ x : \mathbf{F}_q\ [\![\tau]\!]_{\mathrm{V}}$ to conclude.

By assumption, $\gamma_1 \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} [\![e]\!]_{\mathrm{V}} : \mathbf{F}_1\ [\![\tau]\!]_{\mathrm{V}}$ and $\gamma \leq_{co} q \cdot \gamma_1 + \overline{0}$, so by rule COEFF-LETIN, $\gamma \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} x \leftarrow^{q} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{return}_q\ x : \mathbf{F}_q\ [\![\tau]\!]_{\mathrm{V}}$

By rule COEFF-THUNK, $\gamma \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} \{x \leftarrow^{q} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{return}_q\ x\} : \mathbf{U}\,(\mathbf{F}_q\ [\![\tau]\!]_{\mathrm{V}})$ and $\gamma \leq_{co} 1 \cdot \gamma$, so by rule COEFF-RET, $\gamma \cdot [\![\Gamma]\!]_{\mathrm{V}} \vdash_{coeff} \mathbf{return}_1\ \{x \leftarrow^{q} [\![e]\!]_{\mathrm{V}}\ \mathbf{in}\ \mathbf{return}_q\ x\} : \mathbf{F}_1\ (\mathbf{U}\,(\mathbf{F}_q\ [\![\tau]\!]_{\mathrm{V}}))$. □

Lemma 3.71 (CBV translation: unbox). *If* $\gamma_1 \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\Box_{q_1}\ \tau]\!]_V$, $(\gamma_2, q_1 \cdot q_2) \cdot [\![\Gamma, x : \tau]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau']\!]_V$, *and* $\gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2$, *then* $\gamma \cdot [\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{unbox}_{q_2}\ x = e_1\ \mathbf{in}\ e_2]\!]_V : \mathbf{F}_1\ [\![\tau']\!]_V$.

Proof. We have

$$[\![\Box_{q_1}\ \tau]\!]_{\mathrm{V}} = \mathbf{U}\,(\mathbf{F}_{q_1}\ [\![\tau]\!]_{\mathrm{V}})$$

and

$$[\![\Gamma, x : \tau]\!]_{\mathrm{V}} = [\![\Gamma]\!]_{\mathrm{V}}, x : [\![\tau]\!]_{\mathrm{V}}$$

and

$$[\![\mathbf{unbox}_{q_2}\ x = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{V}} = y \leftarrow^{q_2} [\![e_1]\!]_{\mathrm{V}}\ \mathbf{in}\ x \leftarrow^{q_2} y!\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{V}}$$

.

By rule COEFF-VAR, $(\overline{0}, 1) \cdot ([\![\Gamma]\!]_{\mathrm{V}}, y : \mathbf{U}\,(\mathbf{F}_{q_1}\ [\![\tau]\!]_{\mathrm{V}})) \vdash_{coeff} y : \mathbf{U}\,(\mathbf{F}_{q_1}\ [\![\tau]\!]_{\mathrm{V}})$, so by rule COEFF-FORCE, $(\overline{0}, 1) \cdot ([\![\Gamma]\!]_{\mathrm{V}}, y : \mathbf{U}\,(\mathbf{F}_{q_1}\ [\![\tau]\!]_{\mathrm{V}})) \vdash_{coeff} y! : \mathbf{F}_{q_1}\ [\![\tau]\!]_{\mathrm{V}}$.

By assumption, $(\gamma_2, 0, q_1 \cdot q_2) \cdot ((\llbracket\Gamma\rrbracket_{\mathrm{v}}, y : \mathbf{U}\,(\mathbf{F}_{q_1}\ \llbracket\tau\rrbracket_{\mathrm{v}})), x : \llbracket\tau\rrbracket_{\mathrm{v}}) \vdash_{coeff} \llbracket e_2\rrbracket_{\mathrm{v}} : \mathbf{F}_1\ \llbracket\tau'\rrbracket_{\mathrm{v}}$, and $(\gamma_2, q_2) \leq_{co} q_2 \cdot (\overline{0}, 1) + (\gamma_2, 0)$, so by rule COEFF-LETIN, $(\gamma_2, q_2) \cdot (\llbracket\Gamma\rrbracket_{\mathrm{v}}, y : \mathbf{U}\,(\mathbf{F}_{q_1}\ \llbracket\tau\rrbracket_{\mathrm{v}})) \vdash_{coeff} x \leftarrow^{q_2} y!\ \mathbf{in}\ \llbracket e_2\rrbracket_{\mathrm{v}} : \mathbf{F}_1\ \llbracket\tau'\rrbracket_{\mathrm{v}}$.

By assumption, $\gamma_1 \cdot \llbracket\Gamma\rrbracket_{\mathrm{v}} \vdash_{coeff} \llbracket e_1\rrbracket_{\mathrm{v}} : \mathbf{F}_1\ (\mathbf{U}\,(\mathbf{F}_{q_1}\ \llbracket\tau\rrbracket_{\mathrm{v}}))$ and $\gamma \leq_{co} q_2 \cdot \gamma_1 + \gamma_2$, so by rule COEFF-LETIN $\gamma \cdot \llbracket\Gamma\rrbracket_{\mathrm{v}} \vdash_{coeff} y \leftarrow^{q_2} \llbracket e_1\rrbracket_{\mathrm{v}}\ \mathbf{in}\ x \leftarrow^{q_2} y!\ \mathbf{in}\ \llbracket e_2\rrbracket_{\mathrm{v}} : \mathbf{F}_1\ \llbracket\tau'\rrbracket_{\mathrm{v}}$ □

3.3.3 *Encapsulated coeffects.*

Computation with effects can be presented with a type-and-effect system or a type system augmented with a graded monad type constructor [Wadler and Thiemann 2003]. Similarly, computation with coeffects can be presented with a type-and-coeffect system, as above, or with a type system augmented by a graded *comonad* type constructor to capture the notion of computation with coeffects. In this *comonadic* calculus, coeffects are encapsulated within the comonadic type, rather than being a part of the typing judgement. We adapt this calculus from the comonad-inspired extension to a functional language (section 6.7) from Petricek [2017]. As above, we can translate this system to CBPV using either the CBN or CBV translations.

The core of this type system is a linear type system. This system also includes two constructs for comonadic operations (*extract* and *extend*) and two constructs for managing contexts with comonadic terms (*discard* and *divide*).

$\boxed{\Gamma \vdash_{com} e : \tau}$ *(coeffect typing)*

$$\textsc{lam-com-var}\quad \frac{}{x : \tau \vdash_{com} x : \tau} \qquad \textsc{lam-com-abs}\quad \frac{\Gamma, x : \tau_1 \vdash_{com} e : \tau_2}{\Gamma \vdash_{com} \lambda x.e : \tau_1 \multimap \tau_2}$$

$$\textsc{lam-com-app}\quad \frac{\Gamma_1 \vdash_{com} e_1 : \tau_1 \multimap \tau_2 \qquad \Gamma_2 \vdash_{com} e_2 : \tau_1}{\Gamma_1, \Gamma_2 \vdash_{com} e_1\ e_2 : \tau_2} \qquad \textsc{lam-com-unit}\quad \frac{}{\varnothing \vdash_{com} () : \mathbf{unit}}$$

$$\textsc{lam-com-extract}\quad \frac{\Gamma \vdash_{com} e : \square_q\ \tau \qquad q \leq_{co} 1}{\Gamma \vdash_{com} \mathbf{extract}\, e : \tau}$$

$$\textsc{lam-com-extend}\quad \frac{\Gamma_1 \vdash_{com} e_1 : \square_{q'_1}\ \tau_1 \ldots \Gamma_k \vdash_{com} e_k : \square_{q'_k}\ \tau_k \qquad q'_1 \leq_{co} q \cdot q_1 \ldots q'_k \leq_{co} q \cdot q_k \qquad x_1 : \square_{q_1}\ \tau_1, \ldots, x_k : \square_{q_k}\ \tau_k \vdash_{com} e' : \tau'}{\Gamma_1, \ldots, \Gamma_k \vdash_{com} \mathbf{extend}_q\ {x_1}^{q_1} = e_1, \ldots, {x_k}^{q_k} = e_k\ \mathbf{in}\ e' : \square_q\ \tau'}$$

$$\textsc{lam-com-discard}\quad \frac{\Gamma_1 \vdash_{com} e_1 : \square_q\ \tau_1 \qquad \Gamma_2 \vdash_{com} e_2 : \tau_2 \qquad q \leq_{co} 0}{\Gamma_1, \Gamma_2 \vdash_{com} \mathbf{discard}\ \_ = e_1\ \mathbf{in}\ e_2 : \tau_2}$$

$$\textsc{lam-com-divide}\quad \frac{\Gamma_1 \vdash_{com} e_1 : \square_q\ \tau_1 \qquad q \leq_{co} q_1 + q_2 \qquad \Gamma_2, x : \square_{q_1}\ \tau_1, x : \square_{q_2}\ \tau_2 \vdash_{com} e_2 : \tau_2}{\Gamma_1, \Gamma_2 \vdash_{com} \mathbf{divide}\ {x_1}^{q_1}, {x_2}^{q_2} = e_1\ \mathbf{in}\ e_2 : \tau_2}$$

The *extract* construct is dual to a monadic return construct. While monadic return takes a value and places it in the trivial effect with that value, comonadic extract takes the trivial coeffect with a value and returns the value itself.

We also have an *extend* construct, dual to the monadic bind. In monadic bind, we take a value of the monadic type and use it as a bare type to produce a term of the monadic type, sequencing the computations. In comonadic extend, we take a term of the comonadic type and use it with the comonadic type to produce a bare type. We then extend the grade that we had on the original input to box the final output. In our construction, we allow for multiple binders so that the "function" can depend on multiple "arguments."

As an example, consider the following judgement:

$$x_1 : \square_2\ \mathbf{int}, x_2 : \square_2\ \mathbf{int} \vdash_{com} \mathbf{extend}_2\ {x_3}^1 = x_1, {x_4}^1 = x_2\ \mathbf{in}\ (\mathbf{extract}\, x_3) + (\mathbf{extract}\, x_4) : \square_2\ \mathbf{int}$$

This term requires two variables, x_1 and x_2, which can each be used twice. We are trying to produce a term which represents their sum and can be used twice. To construct something that can be used twice, we need to use an *extend* annotated with a 2. This tells us that we need double of whatever resources are needed to produce it once. The binding list of the extend specifies which resources we need to produce the sum – one use of each of the summands. In the body of the extend, we *extract* the integer out of each box to be added. We are allowed to extract it because, in the context of the body, the variables have grade 1, denoting that it can be used.

Extract and extend are similar to box and unbox from the coeffectful calculus in that they both operate on the multiplicative structure of the semiring. That is, they both control how resources are used and how result values that depend on resources can be used. There are important differences. Most noticeably, in this calculus, the bindings are in the introduction form for the box type, rather than on the elimination form. To introduce a box, we choose which resources contribute to it. In the coeffectful calculus, every variable in the context has a grade associated with it, and so the introduction form can simply operate on the entire usage vector.

In the coeffectful calculus, we freely make use of structural rules to operate on the context. A key operation is context addition – this allows us to allocate resources between two subexpressions. In this calculus, resources are encapsulated in a single boxed assumption. The linear type system would then force us to put all of these resources into one sub-expression. This is, of course, not the desired behavior, so we include the *divide* construct. For example, consider this judgement:

$$x : \Box_2\ \mathbf{int} \vdash_{com} \mathbf{divide}\ {x_1}^1, {x_2}^1 = x\ \mathbf{in}\ (\mathbf{extract}\ x_1) + (\mathbf{extract}\ x_2) : \mathbf{int}$$

In this term, we have a variable which we can use twice, and we want to add it to itself. This requires one use of the variable in each operand to the addition, so we use divide.

The final construct is *discard*. Because the linear type system does not admit weakening, we may be stuck with an assumption $x : \Box_0\ \tau$ in the context. The value of this assumption cannot be accessed, but the assumption itself must be discharged , so we do this with discard. A natural question, then, is why use a linear type system, rather than one which admits weakening? The linearity enforces that assumptions with grade 0 and *only* these assumptions can be weakened. Any assumptions with grade not approximable by 0 must be used as the coeffect system intends.

Both CBN and CBV semantics for this calculus admit a type preserving translation to CBPV. Because coeffects are encapsulated in the types, this also guarantees that coeffects are preserved.

For simplicity, we only show a single binding in the translation of extend. The CBN translation is:

Type translation

$$
\begin{aligned}
[\![\tau_1 \multimap \tau_2]\!]_{\mathrm{N}} &= (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}})^{1} \to [\![\tau_2]\!]_{\mathrm{N}} \\
[\![\Box_q\ \tau]\!]_{\mathrm{N}} &= \mathbf{F}_q\ (\mathbf{U}\,[\![\tau]\!]_{\mathrm{N}})
\end{aligned}
$$

Context translation

$$
[\![\Gamma\,, x : \tau]\!]_{\mathrm{N}} = [\![\Gamma]\!]_{\mathrm{N}}\,, x :^{1} \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}
$$

Terms

$$
\begin{aligned}
[\![x]\!]_{\mathrm{N}} &= x! \\
[\![\lambda x.e]\!]_{\mathrm{N}} &= \lambda x^{1}.[\![e]\!]_{\mathrm{N}} \\
[\![e_1\ e_2]\!]_{\mathrm{N}} &= [\![e_1]\!]_{\mathrm{N}}\,\{[\![e_2]\!]_{\mathrm{N}}\} \\
[\![\mathbf{extract}\ e]\!]_{\mathrm{N}} &= x \leftarrow^{1} [\![e]\!]_{\mathrm{N}}\ \mathbf{in}\ x! \\
[\![\mathbf{extend}_q\ x^{q'} = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{N}} &= x' \leftarrow [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ \mathbf{return}_q\ \{x \leftarrow \mathbf{return}_1\ \{\mathbf{return}_{q'}\ x'\}\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{N}}\} \\
[\![\mathbf{divide}\ x_1{}^{q_1}, x_2{}^{q_2} = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{N}} &= x \leftarrow^{1} [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ x_1 \leftarrow^{1} \mathbf{return}_1\ \{\mathbf{return}_{q_1}\ x\}\ \mathbf{in}\ M \\
&\quad \text{where } M := x_2 \leftarrow^{1} \mathbf{return}_1\ \{\mathbf{return}_{q_2}\ x\}\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{N}} \\
[\![\mathbf{discard}\ \_\ = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{N}} &= x \leftarrow^{1} [\![e_1]\!]_{\mathrm{N}}\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{N}}
\end{aligned}
$$

The following property holds of this translation:

Lemma 3.72 (Translation preserves types). *If $\Gamma \vdash_{com} e : \tau$ then $[\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau]\!]_N$.*

Proof. By induction over the typing derivation $\Gamma \vdash_{com} e : \tau$, using the case lemmas below. □

Lemma 3.73 (Comonad CBN Translation: var). $[\![x : \tau]\!]_N \vdash_{coeff} [\![x]\!]_N : [\![\tau]\!]_N$

Proof. $[\![x : \tau]\!]_{\mathrm{N}} = x :^{1} \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}$ and $[\![x]\!]_{\mathrm{N}} = x!$. By rule coeff-var, setting $\Gamma_1, \Gamma_2 = \varnothing$ and $q = 1$, we have $x : \mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}$. Then by rule coeff-force, $x! : [\![\tau]\!]_{\mathrm{N}}$ □

Lemma 3.74 (Comonad CBN Translation: abs). *If $[\![\Gamma, x : \tau_1]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\tau_2]\!]_N$ then $[\![\Gamma]\!]_N \vdash_{coeff} [\![\lambda x.e]\!]_N : [\![\tau_1 \multimap \tau_2]\!]_N$*

Proof. We have $[\![\Gamma, x : \tau_1]\!]_{\mathrm{N}} = [\![\Gamma]\!]_{\mathrm{N}}\,, x :^{1} (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}})$, $[\![\tau_1 \multimap \tau_2]\!]_{\mathrm{N}} = (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}})^{1} \to [\![\tau_2]\!]_{\mathrm{N}}$, and $[\![\lambda x.e]\!]_{\mathrm{N}} = \lambda x^{1}.[\![e]\!]_{\mathrm{N}}$, so this is immediate from the assumption and rule coeff-abs. □

Lemma 3.75 (Comonad CBN Translation: app). *If $[\![\Gamma_1]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\tau_1 \multimap \tau_2]\!]_N$ and $[\![\Gamma_2]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau_1]\!]_N$ then $[\![\Gamma_1\,, \Gamma_2]\!]_N \vdash_{coeff} [\![e_1\ e_2]\!]_N : [\![\tau_2]\!]_N$*

Proof. We have $[\![\tau_1 \multimap \tau_2]\!]_{\mathrm{N}} = (\mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}})^{1} \to [\![\tau_2]\!]_{\mathrm{N}}$ and $[\![e_1\ e_2]\!]_{\mathrm{N}} = [\![e_1]\!]_{\mathrm{N}}\,\{[\![e_2]\!]_{\mathrm{N}}\}$. Then $(\overline{0} \cdot [\![\Gamma_1]\!]_{\mathrm{N}})\,, [\![\Gamma_2]\!]_{\mathrm{N}} \vdash_{coeff} \{[\![e_2]\!]_{\mathrm{N}}\} : \mathbf{U}\,[\![\tau_1]\!]_{\mathrm{N}}$ by assumption and rule coeff-thunk. Notice that here (and in a number of other cases), we use the fact that we can weaken by 0 graded assumptions. Then the claim holds by rule coeff-app. □

Lemma 3.76 (Comonad CBN Translation: extract). *If $[\![\Gamma]\!]_N \vdash_{coeff} [\![e]\!]_N : [\![\Box_q\ \tau]\!]_N$ and $q \leq_{co} 1$, then $[\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{extract}\ e]\!]_N : [\![\tau]\!]_N$*

Proof. We have $[\![\Box_q\ \tau]\!]_{\mathrm{N}} = \mathbf{F}_q\ (\mathbf{U}\,[\![\tau]\!]_{\mathrm{N}})$ and $[\![\mathbf{extract}\ e]\!]_{\mathrm{N}} = x \leftarrow^{1} [\![e]\!]_{\mathrm{N}}\ \mathbf{in}\ x!$. In the body of the let, we have $(\overline{0} \cdot [\![\Gamma]\!]_{\mathrm{N}})\,, x :^{q} (\mathbf{U}\,[\![\tau]\!]_{\mathrm{N}}) \vdash_{coeff} x! : [\![\tau]\!]_{\mathrm{N}}$ by rule coeff-var (using the assumption on q) and rule coeff-force. Then the claim holds rule coeff-letin. □

Lemma 3.77 (Comonad CBN Translation: extend). *If $[\![\Gamma]\!]_N \vdash_{coeff} [\![e_1]\!]_N : [\![\Box_{q_1'}\ \tau_1]\!]_N$ and $q_1' \leq_{co} q \cdot q_1$ and $[\![x : \Box_{q_1}\ \tau]\!]_N \vdash_{coeff} [\![e_2]\!]_N : [\![\tau_2]\!]_N$, then $[\![\Gamma]\!]_N \vdash_{coeff} [\![\mathbf{extend}_q\ x_1{}^{q_1} = e_1\ \mathbf{in}\ e_2]\!]_N : [\![\Box_q\ \tau_2]\!]_N$*

Proof. Recall that $\llbracket \Box_q\ \tau \rrbracket_{\mathrm{N}} = \mathbf{F}_q\ (\mathbf{U}\,\llbracket \tau \rrbracket_{\mathrm{N}})$. Then in the body of the outer let, our context is $x' :^{q'_1} \mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}}$. Using rule COEFF-LETIN, rule COEFF-RET, and our assumption about q_1, it suffices to show that, in context $x' :^{q_1} \mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}}$, the thunk can be given type $\mathbf{U}\,\llbracket \tau_2 \rrbracket_{\mathrm{N}}$. Notice that the grade on the assumption is now q_1 instead of q'_1 because of the multiplication from the return construct. Then by rule COEFF-THUNK, rule COEFF-LETIN, and our assumption about $\llbracket e_2 \rrbracket_{\mathrm{N}}$, we need to show that $x' :^{q_1} \mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}} \vdash_{coeff} \mathbf{return}_1\ \{\mathbf{return}_{q_1}\ x'\} : \mathbf{F}_1\ (\mathbf{U}\,(\mathbf{F}_{q_1}\ (\mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}})))$. The outer $\mathbf{F}$ constructor comes from the requirement that the bound expression of a let be a computation, and the outer $\mathbf{U}$ constructor comes from the requirement that assumptions have value types. This judgement holds by rule COEFF-RET, rule COEFF-THUNK, and rule COEFF-VAR. □

Lemma 3.78 (Comonad CBN Translation: Divide). *If* $\llbracket \Gamma_1 \rrbracket_N \vdash_{coeff} \llbracket e_1 \rrbracket_N : \llbracket \Box_q\ \tau_1 \rrbracket_N$ *and* $q \leq_{co} q_1 + q_2$ *and* $\llbracket \Gamma_2, x_1 : \Box_{q_1}\ \tau_1, x_2 : \Box_{q_2}\ \tau_1 \rrbracket_N \vdash_{coeff} \llbracket e_2 \rrbracket_N : \llbracket \tau_2 \rrbracket_N$*, then* $\llbracket \Gamma_1 , \Gamma_2 \rrbracket_N \vdash_{coeff} \llbracket \mathbf{divide}\ x_1{}^{q_1}, x_2{}^{q_2} = e_1\ \mathbf{in}\ e_2 \rrbracket_N : \llbracket \tau_2 \rrbracket_N$

Proof. By our third assumption and rule COEFF-LETIN, it suffices to show that the context of the body of the innermost let is $\llbracket \Gamma_2 \rrbracket_{\mathrm{N}} , x_1 :^1 \mathbf{U}\,(\mathbf{F}_{q_1}\ (\mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}})) , x_2 :^1 \mathbf{U}\,(\mathbf{F}_{q_2}\ (\mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}}))$. Then by rule COEFF-LETIN, the body of the outermost let has context $x :^q \mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}}$ (ignoring 0 graded assumptions). Then by our assumption about q, we can let the context for the bound expression of x_1 be $x :^{q_1} \mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}}$, and similarly with q_2 for x_2. Then by rule COEFF-RET, rule COEFF-THUNK, and rule COEFF-VAR, x_1 and x_2 have exactly the desired type in context for $\llbracket e_2 \rrbracket_{\mathrm{N}}$. □

Lemma 3.79 (Comonad CBN Translation: Discard). *If* $\llbracket \Gamma_1 \rrbracket_N \vdash_{coeff} \llbracket e_1 \rrbracket_N : \llbracket \Box_q\ \tau_1 \rrbracket_N$ *and* $\llbracket \Gamma_2 \rrbracket_N \vdash_{coeff} \llbracket e_2 \rrbracket_N : \llbracket \tau_2 \rrbracket_N$ *and* $q \leq_{co} 0$*, then* $\llbracket \Gamma_1 , \Gamma_2 \rrbracket_N \vdash_{coeff} \llbracket \mathbf{discard}\ \_ = e_1\ \mathbf{in}\ e_2 \rrbracket_N : \llbracket \tau_2 \rrbracket_N$

Proof. By our assumption on $\llbracket e_1 \rrbracket_{\mathrm{N}}$ and rule COEFF-LETIN, it suffices to show that $\llbracket \Gamma_2 \rrbracket_{\mathrm{N}} , x :^q \mathbf{U}\,\llbracket \tau_1 \rrbracket_{\mathrm{N}} \vdash_{coeff} \llbracket e_2 \rrbracket_{\mathrm{N}} : \llbracket \tau_2 \rrbracket_{\mathrm{N}}$. But by our assumption about q and the fact that weakening by 0 graded assumptions is admissible in this system, our assumption on $\llbracket e_2 \rrbracket_{\mathrm{N}}$ is sufficient. □

We can also give this calculus a CBV semantics. For convenience, define the notation

$$x \leftarrow\!\!\!\sim M\ \mathbf{in}\ N := x' \leftarrow^1 M\ \mathbf{in}\ x \leftarrow^1 x'!\ \mathbf{in}\ N$$

This operation unwraps layers to put a graded value in context in the way the CBPV type system expects. That is, when M has type $\mathbf{F}_1\ (\mathbf{U}\,(\mathbf{F}_q\ A))$, then we have $x :^q A$ in context for N. We use this operation in the CBV translation below:

$$
\begin{array}{ll}
\textit{Types} & \\
[\![\tau_1 \multimap \tau_2]\!]_{\mathrm{v}} & = \mathbf{U}\,([\![\tau_1]\!]^{1}_{\mathrm{v}} \to \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{v}}) \\
[\![\Box_q\ \tau]\!]_{\mathrm{v}} & = \mathbf{U}\,(\mathbf{F}_q\ (\mathbf{U}\,(\mathbf{F}_1\ ([\![\tau]\!]_{\mathrm{v}})))) \\
& \\
\textit{Contexts} & \\
[\![\Gamma\,, x : \tau]\!]_{\mathrm{v}} & = [\![\Gamma]\!]_{\mathrm{v}}\,, x :^{1} [\![\tau]\!]_{\mathrm{v}} \\
& \\
\textit{Terms} & \\
[\![x]\!]_{\mathrm{v}} & = \mathbf{return}_1\ x \\
[\![\lambda x.e]\!]_{\mathrm{v}} & = \mathbf{return}_1\ \{\lambda x^{1}.[\![e]\!]_{\mathrm{v}}\} \\
[\![e_1\ e_2]\!]_{\mathrm{v}} & = x \leftarrow^{1} [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ y \leftarrow^{1} [\![e_2]\!]_{\mathrm{v}}\ \mathbf{in}\ x!\ y \\
[\![\mathbf{extract}\ e]\!]_{\mathrm{v}} & = x \leftsquigarrow [\![e]\!]_{\mathrm{v}}\ \mathbf{in}\ x! \\
[\![\mathbf{extend}_q\ x^{q'} = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{v}} & = x' \leftsquigarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ \mathbf{return}_1\ \{\mathbf{return}_q\ \{x \leftarrow M\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{v}}\}\} \\
& \text{where } M := \mathbf{return}_1\ \{\mathbf{return}_{q'}\ \{\mathbf{return}_1\ x'\}\} \\
[\![\mathbf{divide}\ {x_1}^{q_1}, {x_2}^{q_2} = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{v}} & = x \leftsquigarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ x_1 \leftarrow^{1} \mathbf{return}_1\ \{\mathbf{return}_{q_1}\ x\}\ \mathbf{in}\ M \\
& \text{where } M := x_2 \leftarrow^{1} \mathbf{return}_1\ \{\mathbf{return}_{q_2}\ x\}\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{v}} \\
[\![\mathbf{discard}\ \_\ = e_1\ \mathbf{in}\ e_2]\!]_{\mathrm{N}} & = x \leftsquigarrow [\![e_1]\!]_{\mathrm{v}}\ \mathbf{in}\ [\![e_2]\!]_{\mathrm{v}}
\end{array}
$$

The typical translation of a comonad in an adjunction is as **FU**. However, under a CBV semantics, we translate types as value types. Thus, we are required to bracket the **FU** with an additional **U** on the outside and **F** on the inside (we give the inner one a trivial grade). As such, accessing the comonad requires multiple layers of wrapping and unwrapping.

This CBV translation also preserves types:

Lemma 3.80 (Translation preserves types). *If* $\Gamma \vdash_{com} e : \tau$, *then* $[\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$

Proof. By induction on the derivation of $\Gamma \vdash_{com} e : \tau$, using the case lemmas below. □

Lemma 3.81 (Comonad CBV Translation: unwraping let). *If* $\gamma_1 \cdot \Gamma \vdash_{coeff} M : \mathbf{F}_1\ (\mathbf{U}\,(\mathbf{F}_q\ A))$ *and* $\gamma_2 \cdot \Gamma\,, x :^{q} A \vdash_{coeff} N : B$, *then* $(\gamma_1 + \gamma_2) \cdot \Gamma \vdash_{coeff} x \leftsquigarrow M\ \mathbf{in}\ N : B$.

Proof. This follows directly from rule coeff-var, rule coeff-letin, and rule coeff-force. □

Lemma 3.82. $[\![x : \tau]\!]_V \vdash_{coeff} [\![x]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$

Proof. This is direct from rule coeff-var and rule coeff-ret. □

Lemma 3.83. *If* $[\![\Gamma, x : \tau]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$, *then* $[\![\Gamma]\!]_V \vdash_{coeff} [\![\lambda x.e]\!]_V : \mathbf{F}_1\ [\![\tau_1 \multimap \tau_2]\!]_V$

Proof. Recall that $[\![\tau_1 \multimap \tau_2]\!]_{\mathrm{v}} = \mathbf{U}\,([\![\tau_1]\!]^{1}_{\mathrm{v}} \to \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{v}})$. Then by rule coeff-abs and our assumption, we have $[\![\Gamma]\!]_{\mathrm{v}}\,, x :^{1} [\![\tau_1]\!]_{\mathrm{v}} \vdash_{coeff} \lambda x^{1}.[\![e]\!]_{\mathrm{v}} : [\![\tau_1]\!]_{\mathrm{v}} \to \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{v}}$. Then the claim holds by rule coeff-thunk and rule coeff-ret. □

Lemma 3.84. *If* $[\![\Gamma_1]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\tau_1 \multimap \tau_2]\!]_V$ *and* $[\![\Gamma_2]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_1]\!]_V$, *then* $[\![\Gamma_1\,, \Gamma_2]\!]_V \vdash_{coeff} [\![e_1\ e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$

Proof. By rule coeff-letin and our assumptions, it suffices to prove that $x :^{1} \mathbf{U}\,([\![\tau_1]\!]^{1}_{\mathrm{v}} \to \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{v}})\,, y :^{1} [\![\tau_1]\!]_{\mathrm{v}} \vdash_{coeff} x!\ y : \mathbf{F}_1\ [\![\tau_2]\!]_{\mathrm{v}}$. Then this holds by rule coeff-var, rule coeff-force, and rule coeff-app. □

Lemma 3.85. *If* $[\![\Gamma]\!]_V \vdash_{coeff} [\![e]\!]_V : \mathbf{F}_1\ [\![\Box_q\ \tau]\!]_V$ *and* $q \leq_{co} 1$, *then* $[\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{extract}\ e]\!]_V : \mathbf{F}_1\ [\![\tau]\!]_V$

Proof. From 3.81, it suffices to show that $x :^{1} \mathbf{U}\,(\mathbf{F}_1\ [\![\tau]\!]_{\mathrm{v}}) \vdash_{coeff} x! : \mathbf{F}_1\ [\![\tau]\!]_{\mathrm{v}}$, which is true by rule coeff-var and rule coeff-force. □

Lemma 3.86 (Comonad CBV Translation: extend). *If* $[\![\Gamma]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\Box_{q_1'}\ \tau_1]\!]_V$ *and* $q_1' \leq_{co} q \cdot q_1$ *and* $[\![x : \Box_{q_1}\ \tau_1]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$*, then* $[\![\Gamma]\!]_V \vdash_{coeff} [\![\mathbf{extend}_q\ {x_1}^{q_1} = e_1\ \mathbf{in}\ e_2]\!]_V : \mathbf{F}_1\ [\![\Box_q\ \tau_2]\!]_V$

Proof. First, notice that $x' :^{q_1} [\![\tau_1]\!]_V \vdash_{coeff} \mathbf{return}_1\ \{\mathbf{return}_{q_1}\ \{\mathbf{return}_1\ x'\}\} : \mathbf{F}_1\ [\![\Box_{q_1}\ \tau_1]\!]_V$. Then by rule COEFF-LETIN and the assumption about e_2, the body of the inner thunk has type $\mathbf{F}_1\ [\![\tau_2]\!]_V$. Then by rule COEFF-THUNK, rule COEFF-RET, and the assumption about q_1, the body of the outer thunk has type $\mathbf{F}_q\ (\mathbf{U}\,(\mathbf{F}_1\ [\![\tau_2]\!]_V))$. Then the body of the unwrapping let has type $\mathbf{F}_1\ (\mathbf{U}\,(\mathbf{F}_q\ (\mathbf{U}\,(\mathbf{F}_1\ [\![\tau_2]\!]_V)))) = \mathbf{F}_1\ [\![\Box_q\ \tau_2]\!]_V$. Thus, the claim holds by 3.81. □

Lemma 3.87 (Comonad CBV Translation: Divide). *If* $[\![\Gamma_1]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\Box_q\ \tau_1]\!]_V$ *and* $q \leq_{co} q_1 + q_2$ *and* $[\![\Gamma_2, x_1 : \Box_{q_1}\ \tau_1, x_2 : \Box_{q_2}\ \tau_1]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$*, then* $[\![\Gamma_1, \Gamma_2]\!]_V \vdash_{coeff} [\![\mathbf{divide}\ {x_1}^{q_1}, {x_2}^{q_2} = e_1$ in $\mathbf{F}_1\ [\![\tau_2]\!]_V$

Proof. Notice that, if the body of the unwrapping let has an assumption $x :^q [\![\tau_1]\!]_V$, then by rule COEFF-RET, rule COEFF-THUNK, and the assumption about q, then the bound expression for x_1 has type $\mathbf{F}_1\ [\![\Box_{q_1}\ \tau_1]\!]_V$, and similarly for x_2. Then by the assumption on e_2, we have that the body of the innermost let has type $\mathbf{F}_1\ [\![\tau_2]\!]_V$. Then the claim holds by rule COEFF-LETIN, assumption on e_1, and 3.81. □

Lemma 3.88 (Comonad CBN Translation: Discard). *If* $[\![\Gamma_1]\!]_V \vdash_{coeff} [\![e_1]\!]_V : \mathbf{F}_1\ [\![\Box_q\ \tau_1]\!]_V$ *and* $[\![\Gamma_2]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$ *and* $q \leq_{co} 0$*, then* $[\![\Gamma_1, \Gamma_2]\!]_V \vdash_{coeff} [\![\mathbf{discard}\ \_\ = e_1\ \mathbf{in}\ e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$

Proof. By 3.81 and assumption about e_1, it suffices to show that $[\![\Gamma_2]\!]_V, x :^q [\![\tau_1]\!]_V \vdash_{coeff} [\![e_2]\!]_V : \mathbf{F}_1\ [\![\tau_2]\!]_V$. But by assumption on q and the fact that weakening by 0 graded assumptions is admissible, this is exactly our assumption about e_2. □

3.4 Products, Products, Products

CBPV is an example of a polarized type system. Value types are positive and computation types are negative. As a result, CBPV includes both positive and negative products. The former are values, and correspond to tuples in a call-by-value programming language. Their introduction form is strict and their elimination is most naturally defined via pattern matching. Negative products are computations — they can contain effectful computation as subcomponents — and are most naturally eliminated via projection.

Polarization is also found in linear type systems, which are related to coeffects. In such systems, there are also two forms of products: multiplicative products (also called tensor products) are positive and additive products (also called "with" products) are negative. Multiplicative products are formed from disjoint resources and must be eliminated via pattern matching so that those resources are not discarded. In contrast, additive products are formed from shared resources and must be eliminated via projection so that resources are not duplicated.

By design, the polarity in CBPV allows it to model the duality between call-by-value and call-by-name semantics. With the addition of coeffect tracking, we can use this same structure to observe the duality between shared and disjoint demands on resources. However, these two forms of polarity do not have to align. We've seen that we can have positive products that are values and have disjoint resources, and negative products that are computations and share resources. In this section, we explore the other two forms of products: shared value products and disjoint computational products. These two additions are not the same: the former adds expressiveness, whereas the latter can already be simulated by existing features.

Shared value products. For shared value products, we introduce the syntax $A_1 \,\&\, A_2$ for this new type, and new values $\langle V_1, V_2\rangle$, $V.1$, and $V.2$. The typing rules for are as follows:

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} V_1 : A_1 \quad \gamma \cdot \Gamma \vdash_{coeff} V_2 : A_2}{\gamma \cdot \Gamma \vdash_{coeff} \langle V_1, V_2\rangle : A_1 \,\&\, A_2}\ \textsc{coeff-vwith} \qquad \frac{\gamma \cdot \Gamma \vdash_{coeff} V : A_1 \,\&\, A_2}{\gamma \cdot \Gamma \vdash_{coeff} V.1 : A_1}\ \textsc{coeff-vfst} \qquad \frac{\gamma \cdot \Gamma \vdash_{coeff} V : A_1 \,\&\, A_2}{\gamma \cdot \Gamma \vdash_{coeff} V.2 : A_2}\ \textsc{coeff-vsnd}$$

Unlike $A_1 \times A_2$, the existing "tensor" product of the value language, the components of these tuples share resources. As a result, the elimination form must be a projection operation. However, unlike $B_1 \,\&\, B_2$, the components of this type are values. Because of this strictness, it is sound, although unusual, to include the elimination form for this type in the value language. These projections must be effect-free because they can only apply to values.

For the operational semantics, we must introduce a new form of closure for this type, written $\mathbf{clo}(\gamma' \cdot \rho', \langle V_1, V_2\rangle)$, that stores both parts of the pair with a saved environment. This environment can then be used when evaluating the projected value.

$$\frac{\gamma \leq_{co} \gamma'}{\gamma \cdot \rho \vdash_{coeff} \langle V_1, V_2\rangle \Downarrow \mathbf{clo}(\gamma \cdot \rho, \langle V_1, V_2\rangle)}\ \textsc{eval-coeff-val-vwith} \qquad \frac{\gamma \cdot \rho \vdash_{coeff} V \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \langle V_1, V_2\rangle) \quad \gamma' \cdot \rho' \vdash_{coeff} V_1 \Downarrow W}{\gamma \cdot \rho \vdash_{coeff} V.1 \Downarrow W}\ \textsc{eval-coeff-val-vfst} \qquad \frac{\gamma \cdot \rho \vdash_{coeff} V \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \langle V_1, V_2\rangle) \quad \gamma' \cdot \rho' \vdash_{coeff} V_2 \Downarrow W}{\gamma \cdot \rho \vdash_{coeff} V.2 \Downarrow W}\ \textsc{eval-coeff-val-vsnd}$$

We can extend the generic coeffect soundness proof by first adding a new definition to the value relation for with products.

$$\mathcal{W}[\![A_1 \,\&\, A_2]\!] = \{\, \mathbf{clo}(\gamma' \cdot \rho', \langle V_1, V_2\rangle) |\ (\gamma' \cdot \rho', V_1) \in \mathcal{V}[\![A_1]\!] \textit{ and } (\gamma' \cdot \rho', V_2) \in \mathcal{V}[\![A_2]\!]\}$$

Then we can prove semantic analogues for each of the new syntactic rules. These three lemmas and their proofs exactly mirror the corresponding rules for computational products.

Lemma 3.89 (Semantic coeff-cpair). *If $\gamma \cdot \Gamma \vDash_{coeff} V_1 : A_1$ and $\gamma \cdot \Gamma \vDash_{coeff} V_2 : A_2$ then $\gamma \cdot \Gamma \vDash_{coeff} \langle V_1, V_2\rangle : A_1 \,\&\, A_2$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that $(\gamma, \rho, V_1) \in \mathcal{V}[\![A_1]\!]$ and $(\gamma, \rho, V_2) \in \mathcal{V}[\![A_2]\!]$.
So by definition, $\mathbf{clo}(\gamma \cdot \rho, \langle V_1, V_2\rangle) \in \mathcal{W}[\![A_1 \,\&\, A_2]\!]$.
$\gamma \leq_{co} \gamma$, so by rule eval-coeff-val-vpair, we get that $\gamma \cdot \rho \vdash_{coeff} \langle V_1, V_2\rangle \Downarrow \mathbf{clo}(\gamma \cdot \rho, \langle V_1, V_2\rangle)$
So $(\gamma, \rho, \langle V_1, V_2\rangle) \in \mathcal{V}[\![A_1 \,\&\, A_2]\!]$. □

Lemma 3.90 (Semantic coeff-vfst). *If $\gamma \cdot \Gamma \vDash_{coeff} V : A_1 \,\&\, A_2$ then $\gamma \cdot \Gamma \vDash_{coeff} V.1 : A_1$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $W \in \mathcal{W}[\![A_1 \,\&\, A_2]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$.
By definition, W must have the form $\mathbf{clo}(\gamma' \cdot \rho', \langle V_1, V_2\rangle)$ such that $(\gamma', \rho', V_1) \in \mathcal{V}[\![A_1]\!]$, i.e., there exists $W_1 \in \mathcal{W}[\![A_1]\!]$ such that $\gamma' \cdot \rho' \vdash_{coeff} V_1 \Downarrow W_1$.
So by rule eval-coeff-val-vfst, we have that $\gamma \cdot \rho \vdash_{coeff} V.1 \Downarrow W_1$.
So $(\gamma, \rho, V.1) \in \mathcal{V}[\![A_1]\!]$. □

Lemma 3.91 (Semantic coeff-vsnd). *If $\gamma \cdot \Gamma \vDash_{coeff} V : A_1 \,\&\, A_2$ then $\gamma \cdot \Gamma \vDash_{coeff} V.2 : A_2$.*

Proof. Given $\Gamma \vDash \rho$, we have by assumption that there exists $W \in \mathcal{W}[\![A_1 \,\&\, A_2]\!]$ such that $\gamma \cdot \rho \vdash_{coeff} V \Downarrow W$.
By definition, W must have the form $\mathbf{clo}(\gamma' \cdot \rho', \langle V_1, V_2\rangle)$ such that $(\gamma', \rho', V_2) \in \mathcal{V}[\![A_2]\!]$, i.e., there exists $W_2 \in \mathcal{W}[\![A_2]\!]$ such that $\gamma' \cdot \rho' \vdash_{coeff} V_2 \Downarrow W_2$.

So by rule EVAL-COEFF-VAL-VSND, we have that $\gamma \cdot \rho \vdash_{coeff} V.2 \Downarrow W_2$.
So $(\gamma, \rho, V.2) \in \mathcal{V}[\![A_2]\!]$. □

Disjoint computational products. We can also consider another form of product: nonstrict products that are composed from distinct resources. These products are more familiar: both Abel and Bernardy [2020] and Choudhury et al. [2021] include this product in their CBN languages.

To formalize this extension, we use the syntax $B_1 \times B_2$ to denote the type of these products, introduce them as pairs of computations and eliminate them via pattern matching.

$$\textsc{coeff-ctensor}\quad \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} M_1 : B_1 \qquad \gamma_2 \cdot \Gamma \vdash_{coeff} M_2 : B_2 \qquad \gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} (M_1, M_2) : B_1 \times B_2}$$

$$\textsc{coeff-csplit}\quad \frac{\gamma_1 \cdot \Gamma \vdash_{coeff} M : B_1 \times B_2 \qquad \gamma_2 \cdot \Gamma, x_1 :^q \mathbf{U}\, B_1, x_2 :^q \mathbf{U}\, B_2 \vdash_{coeff} N : B \qquad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ M\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N : B}$$

This type is isomorphic to the computation type $\mathbf{F}\,(\mathbf{U}\, B_1 \times \mathbf{U}\, B_2)$. Indeed, we use thunks in its typing rules and operational semantics. For example, the variables inserted in the context by pattern matching must have value types, so we thunk them. Also, the evaluation rule for tensors requires a new terminal form (W_1, W_2), where each component is a closed value thunk for each subterm. This thunk must divide up the available resources between the two subcomputations. When this pair is eliminated, the two thunks are added to the environments.

$$\textsc{eval-coeff-comp-ctensor}\quad \frac{\gamma \leq_{co} \gamma_1 + \gamma_2}{\gamma \cdot \rho \vdash_{coeff} (M_1, M_2) \Downarrow (\mathbf{clo}(\gamma_1 \cdot \rho, \{M_1\}), \mathbf{clo}(\gamma_2 \cdot \rho, \{M_2\}))}$$

$$\textsc{eval-coeff-comp-csplit}\quad \frac{\gamma_1 \cdot \rho \vdash_{coeff} (M_1, M_2) \Downarrow (W_1, W_2) \qquad \gamma_2 \cdot \rho,\ x_1 \mapsto^q W_1,\ x_2 \mapsto^q W_2 \vdash_{coeff} N \Downarrow T \qquad \gamma \leq_{co} q \cdot \gamma_1 + \gamma_2}{\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_q\ M\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N \Downarrow T}$$

As above, we can extend the soundness proof to include this new type. This time we extend the set of closed terminals to include:

$$\mathcal{T}[\![B_1 \times B_2]\!] = \{\ (\mathbf{clo}(\gamma_1 \cdot \rho, M_1), \mathbf{clo}(\gamma_2 \cdot \rho, M_2)) |\ (\gamma_1, \rho, M_1) \in \mathcal{M}[\![B_1]\!]\ \textit{and}\ (\gamma_2, \rho, M_2) \in \mathcal{M}[\![B_2]\!]\}$$

With this definition, we can show the semantic typing rules corresponding to the syntactic typing rules above:

Lemma 3.92 (Semantic coeff-ctensor). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} M_1 : B_1$ *and* $\gamma_2 \cdot \Gamma \vDash_{coeff} M_2 : B_2$ *and* $\gamma \leq_{co} \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} (M_1, M_2) : B_1 \times B_2$.

Proof. Given $\Gamma \vDash \rho$, by assumption we know that $(\gamma_1 \cdot \rho, M_1) \in \mathcal{M}[\![B_1]\!]$ and and $(\gamma_2 \cdot \rho, M_2) \in \mathcal{M}[\![B_2]\!]$.
This is exactly what is needed to show that $(\mathbf{clo}(\gamma_1 \cdot \rho, M_1), \mathbf{clo}(\gamma_2 \cdot \rho, M_2)) \in \mathcal{T}[\![B_1 \times B_2]\!]$. □

Lemma 3.93 (Semantic coeff-split). *If* $\gamma_1 \cdot \Gamma \vDash_{coeff} M : B_1 \times B_2$ *and* $\gamma_2 \cdot \Gamma, x_1 :^q \mathbf{U}\, B_1, x_2 :^q \mathbf{U}\, B_2 \vDash_{coeff} N : B$
$\gamma_2 \cdot \Gamma, x_1 :^q A_1 \vDash_{coeff} N : B$
$\gamma_2 \cdot \Gamma \vDash_{coeff} N : B$ *and* $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ *then* $\gamma \cdot \Gamma \vDash_{coeff} \mathbf{case}_q\ M\ \mathbf{of}\ (x_1, x_2) \ \rightarrow\ N : B$.

Proof. Given $\Gamma \vDash \rho$, there exists by assumption $T \in \mathcal{T}[\![B_1 \times B_2]\!]$ such that $\gamma_1 \cdot \rho \vdash_{coeff} M \Downarrow T$.
By definition, T must have the form $(\mathbf{clo}(\gamma_1 \cdot \rho, M_1), \mathbf{clo}(\gamma_2 \cdot \rho, M_2))$ for som $(\gamma_1, \rho, M_1) \in \mathcal{M}[\![B_1]\!]$ and $(\gamma_2, \rho, M_2) \in \mathcal{M}[\![B_2]\!]$.

Set $W_1 = \mathbf{clo}(\gamma_1 \cdot \rho, M_1)$ and $W_2 = \mathbf{clo}(\gamma_1 \cdot \rho, M_2)$. By definition, $W_1 \in \mathcal{W}[\![\mathbf{U}\, B_1]\!]$ and $W_2 \in \mathcal{W}[\![\mathbf{U}\, B_2]\!]$.

So $\Gamma, x_1 : \mathbf{U}\, B_1, x_2 : \mathbf{U}\, B_2 \vDash \rho, x_1 \mapsto W_1, x_2 \mapsto W_2$, so by assumption there exists $T \in \mathcal{T}[\![B]\!]$ such that $\gamma_2, q, q \cdot \rho, x_1 \mapsto W_1, x_2 \mapsto W_2 \vdash_{coeff} N \Downarrow T$.

Because $\gamma \leq_{co} q \cdot \gamma_1 + \gamma_2$ by assumption, we get that $\gamma \cdot \rho \vdash_{coeff} \mathbf{case}_q\ M\ \mathbf{of}\ (x_1, x_2) \rightarrow N \Downarrow T$.

So $(\gamma, \rho, \mathbf{case}_q\ M\ \mathbf{of}\ (x_1, x_2) \rightarrow N) \in \mathcal{M}[\![B]\!]$. □

Together, the four product types provide insight into the nature of computation with resources—that there are fundamental choices in how to allocate resources between subcomputations and how their results may be used.

4 RELATED WORK

Call-by-push-value (CBPV) was originally developed by Levy [2001]. Forster et al. [2019] mechanized proofs of metatheoretic properties including strong and weak normalization and translation soundness; their work inspired our mechanized proofs. Current applications of CBPV include modeling compiler intermediate languages [Downen et al. 2020; Rizkallah et al. 2018], understanding the role that polarity plays in bidirectional typing [Dunfield and Krishnaswami 2021] and subtyping [Lakhani et al. 2022], and incorporating effects into dependent type theories [Pédrot and Tabareau 2019; Pédrot et al. 2019].

CBPV and effects. Call-by-value languages with effect tracking go back to FX by Lucassen and Gifford [1988]. Wadler and Thiemann [2003] showed the connection between graded monads and effects by translating the effect system of Talpin and Jouvelot [1994] to a language that isolates effects using graded monads. Our monadic effect language is inspired by this paper, though generalized following Katsumata [2014]. In this paper, our translation is the reverse of Wadler and Thiemann, mapping a language with graded monads to an effect-style extension of CBPV. Like us, Rajani et al. [2021] use a logical relation to show the soundness of their monadic cost analysis.

Although CBPV has often been used to model the semantics of effects, its type system has only rarely been extended with effect tracking. The type system that we present in Section 2 is most closely similar to MAM, for multi-adjunctive metalanguage, from Forster et al. [2017], which builds on earlier work by Kammar and Plotkin [2012]. Forster et al. use MAM to compare the relative expressiveness of effect handlers, monadic reflection and delimited control. As in our system, MAM annotates the thunk type with an effect annotation. However, MAM uses kinds to distinguish between value and computation types and includes type polymorphism in the base language, before considering the three extensions that are the main subject of the paper.

Wuttke [2021] defines a cost-annotated version of CBPV. To do so, it augments the thunk type in CBPV with a bound $[a < I]$ that limits the number of times that thunks can be forced. This work includes both call-by-value and call-by-name translations from cost-annotated PCF terms to cost-annotated CBPV. For expressiveness, their system also includes subtyping and indexed types.

Some extensions annotate effects using the returner type instead of the thunk type. Such systems need not annotate the computation typing judgement with its effect: instead all effects are tracked in types. Extended Call-by-Push-Value (ECBPV) [McDermott and Mycroft 2019] adds call-by-need evaluation to CBPV and then layers an effect system to augment equational reasoning. This system uses an operation $\langle \phi \rangle B$ to extend the effect annotation to other computation types, combining effects in returner types and pushing effects to the result type of functions and inside with-products. Rioux and Zdancewic [2020] augment the type system of CBPV to track divergence. In this system, the sequencing operation requires that the annotation on the returner be less than or equal to any annotation on the result of the continuation.

CBPV and coeffects. Type systems that track coeffects were introduced by Brunel et al. [2014]; Ghica and Smith [2014]; Petricek et al. [2014] and developed by Abel and Bernardy [2020]; Orchard and et al. [2022]; Orchard et al. [2019]. An early applications of these type systems were for bounded linearity: coeffects track how many times resources are used during computation. However, these systems are related to tracking information flow in differential privacy [Reed and Pierce 2010], dynamic binding [Nanevski 2003] and have also been applied for resource usage in Haskell [Bernardy et al. 2017] and irrelevance in dependently-typed languages [Abel et al. 2023; Atkey 2018; Choudhury et al. 2021]. Petricek et al. [2014] give a number of additional examples, including dataflow (the number of past values needed in a stream processing language) and data liveness (whether references to a variable are still needed).

Various works give proofs of soundness of theories involving coeffects. Like us, this Abel et al. [2023] and Choudhury et al. [2021] use a heap-based operational semantics to show coeffect soundness. However, both of these proofs use a call-by-name small-step semantics, while this work demonstrates the use of a big-step semantics, and moreover applies to call-by-value systems.

Our extension of CBPV with coeffect typing is novel to this work and inspired by the duality with effects. The most related systems are those involving linearity, especially in the context of low-level or compiler intermediate languages. Schöpp [2015] develops a low-level language, similar to CBPV, that includes linear operations in its type system. The *enriched effect calculus* [Egger et al. 2009, 2012] extends a type theory for computational effects, similar to CBPV, with primitives from linear logic. Ahmed et al. [2007] augment a variant of typed assembly language with linear types. Uniqueness types enforce that their values have at most one reference at any given time. Marshall et al. [2022] investigate uniqueness in the context of a linear calculus and implement them in the Granule language.

5 CONCLUSION AND FUTURE WORK

In this paper we have explored a core language for CBPV and have extended it in two ways: once with structure to track effects and a specific effectful operation (**tick**) and once with coeffects. In this setting, we have demonstrated that the monad and comonad types of CBPV can be graded to track effects and coeffects. We have also developed an instrumented operational semantics for both effects and coeffects, and refined the latter so that it is appropriate for resource tracking. By exploring both effects and coeffects in the same manner, we are also able to observe similarities between these dual notions, and more importantly, identify their differences.

However, this work is only the starting point for investigation in this space. The natural next step is to develop a more general structure for extensions of CBPV, perhaps based on algebraic effects [Plotkin and Pretnar 2008] or effect signatures [Katsumata 2014]. This structure would allow us to verify that our rules stay general in the presence of other effects, such as nontermination and state, or other coeffects, such as information-flow tracking and differential privacy.

Another next step is to track effects and coeffects in the same system. While it is straightforward to design a system where they are treated orthogonally, Gaboardi et al. [2016] demonstrate that we can reason about how graded monads and comonads may distribute over one another, and provide coherence conditions on such behavior. Future work may explore if these same conditions apply in CBPV, as well as the operational semantics of such distributive terms.

Furthermore, we can also extend this work by adding language features that interact with effect and coeffect tracking. Potential features include polymorphism, indexed or dependent types, and quantification over effects and coeffects. Subtyping can also capture the idea that the type $\mathbf{U}_{\phi_1}\ B$ is a subtype of $\mathbf{U}_{\phi_2}\ B$ when $\phi_1 \leq_{\mathit{eff}} \phi_2$, and that the type $\mathbf{F}_{q_1}\ A$ is a subtype of $\mathbf{F}_{q_2}\ A$ when $q_2 \leq_{\mathit{co}} q_1$.

Finally, we could explore the practical concerns of these type systems in more depth. While our type systems are syntax directed, we have set aside most issues related to the implementation of a type checker. A practical system would also be concerned with how users or compilers might construct these typing derivations and how they might make effective use of the information contained within the type system.

REFERENCES

Andreas Abel and Jean-Philippe Bernardy. 2020. A Unified View of Modalities in Type Systems. *Proceedings of the ACM on Programming Languages* 4, ICFP (2020).

Andreas Abel, Nils Anders Danielsson, and Oskar Eriksson. 2023. A Graded Modal Dependent Type Theory with a Universe and Erasure, Formalized. *Proc. ACM Program. Lang.* 7, ICFP, Article 220 (aug 2023), 35 pages. https://doi.org/10.1145/3607862

Amal Ahmed, Matthew Fluet, and Greg Morrisett. 2007. L^3: A Linear Language with Locations. *Fundam. Informaticae* 77, 4 (2007), 397–449. http://content.iospress.com/articles/fundamenta-informaticae/fi77-4-06

Robert Atkey. 2018. The Syntax and Semantics of Quantitative Type Theory. In *LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, July 9–12, 2018, Oxford, United Kingdom.*

Jean-Philippe Bernardy, Mathieu Boespflug, Ryan R. Newton, Simon Peyton Jones, and Arnaud Spiwack. 2017. Linear Haskell: Practical Linearity in a Higher-Order Polymorphic Language. *Proc. ACM Program. Lang.* 2, POPL, Article 5 (dec 2017), 29 pages. https://doi.org/10.1145/3158093

Edwin Brady. 2021. Idris 2: Quantitative Type Theory in Practice. In *35th European Conference on Object-Oriented Programming (ECOOP 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 194)*, Anders Møller and Manu Sridharan (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 9:1–9:26. https://doi.org/10.4230/LIPIcs.ECOOP.2021.9

Aloïs Brunel, Marco Gaboardi, Damiano Mazza, and Steve Zdancewic. 2014. A Core Quantitative Coeffect Calculus. In *Programming Languages and Systems*, Zhong Shao (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 351–370.

Pritam Choudhury, Harley D. Eades III, Richard A. Eisenberg, and Stephanie Weirich. 2021. A Graded Dependent Type System with a Usage-Aware Semantics. *Proc. ACM Program. Lang.* 5, POPL (Jan. 2021). https://doi.org/10.1145/3434331 Artifact available.

Paul Downen, Zena M. Ariola, Simon Peyton Jones, and Richard A. Eisenberg. 2020. Kinds Are Calling Conventions. *Proc. ACM Program. Lang.* 4, ICFP, Article 104 (aug 2020), 29 pages. https://doi.org/10.1145/3408986

Jana Dunfield and Neel Krishnaswami. 2021. Bidirectional Typing. *ACM Comput. Surv.* 54, 5, Article 98 (may 2021), 38 pages. https://doi.org/10.1145/3450952

Jeff Egger, Rasmus Ejlers Møgelberg, and Alex Simpson. 2009. Enriching an Effect Calculus with Linear Types. In *Computer Science Logic*, Erich Grädel and Reinhard Kahle (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 240–254.

Jeff Egger, Rasmus Ejlers Møgelberg, and Alex Simpson. 2012. The enriched effect calculus: syntax and semantics. *Journal of Logic and Computation* 24, 3 (06 2012), 615–654. https://doi.org/10.1093/logcom/exs025 arXiv:https://academic.oup.com/logcom/article-pdf/24/3/615/2785623/exs025.pdf

Yannick Forster, Ohad Kammar, Sam Lindley, and Matija Pretnar. 2017. On the Expressive Power of User-Defined Effects: Effect Handlers, Monadic Reflection, Delimited Control. *Proc. ACM Program. Lang.* 1, ICFP, Article 13 (aug 2017), 29 pages. https://doi.org/10.1145/3110257

Yannick Forster, Steven Schäfer, Simon Spies, and Kathrin Stark. 2019. Call-by-Push-Value in Coq: Operational, Equational, and Denotational Theory. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs* (Cascais, Portugal) *(CPP 2019)*. Association for Computing Machinery, New York, NY, USA, 118–131. https://doi.org/10.1145/3293880.3294097

Marco Gaboardi, Shin-ya Katsumata, Dominic A Orchard, Flavien Breuvart, and Tarmo Uustalu. 2016. Combining effects and coeffects via grading. In *ICFP*. 476–489.

Dmitri Garbuzov, William Mansky, Christine Rizkallah, and Steve Zdancewic. 2018. Structural Operational Semantics for Control Flow Graph Machines. arXiv:1805.05400 [cs.PL]

Dan R Ghica and Alex I Smith. 2014. Bounded linear types in a resource semiring. In *European Symposium on Programming Languages and Systems*. Springer, 331–350.

Ohad Kammar and Gordon D. Plotkin. 2012. Algebraic Foundations for Effect-Dependent Optimisations. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Philadelphia, PA, USA) *(POPL '12)*. Association for Computing Machinery, New York, NY, USA, 349–360. https://doi.org/10.1145/2103656.2103698

Shin-ya Katsumata. 2014. Parametric Effect Monads and Semantics of Effect Systems. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*. ACM, New York, NY, USA, 633–645.

Zeeshan Lakhani, Ankush Das, Henry DeYoung, Andreia Mordido, and Frank Pfenning. 2022. Polarized Subtyping. In *Programming Languages and Systems*, Ilya Sergey (Ed.). Springer International Publishing, Cham, 431–461.

Daan Leijen. 2023. *The Koka Programming Language.*

Xavier Leroy, Damien Doligez, Alain Frisch, Jacques Garrigue, Didier Rémy, and Jérôme Vouillon. 2023. *The OCaml system release 5.1*. INRIA.

Paul Blain Levy. 1999. Call-by-Push-Value: A Subsuming Paradigm. In *Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications (TLCA '99)*. Springer-Verlag, Berlin, Heidelberg, 228–242.

Paul Blain Levy. 2001. *Call-by-push-value*. Ph. D. Dissertation. Queen Mary University of London, UK. https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.369233

Paul Blain Levy. 2006. Call-by-Push-Value: Decomposing Call-by-Value and Call-by-Name. *Higher Order Symbol. Comput.* 19, 4 (dec 2006), 377–414. https://doi.org/10.1007/s10990-006-0480-6

Paul Blain Levy. 2022. Call-by-Push-Value. *ACM SIGLOG News* 9, 2 (may 2022), 7–29. https://doi.org/10.1145/3537668.3537670

J. M. Lucassen and D. K. Gifford. 1988. Polymorphic Effect Systems. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (San Diego, California, USA) *(POPL '88)*. Association for Computing Machinery, New York, NY, USA, 47–57. https://doi.org/10.1145/73560.73564

Daniel Marshall, Michael Vollmer, and Dominic Orchard. 2022. Linearity and Uniqueness: An Entente Cordiale. In *Programming Languages and Systems*, Ilya Sergey (Ed.). Springer International Publishing, Cham, 346–375.

Luke Maurer, Paul Downen, Zena M. Ariola, and Simon Peyton Jones. 2017. Compiling without Continuations. *SIGPLAN Not.* 52, 6 (jun 2017), 482–494. https://doi.org/10.1145/3140587.3062380

Dylan McDermott and Alan Mycroft. 2019. Extended Call-by-Push-Value: Reasoning About Effectful Programs and Evaluation Order. In *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11423)*, Luís Caires (Ed.). Springer, 235–262. https://doi.org/10.1007/978-3-030-17184-1_9

Aleksandar Nanevski. 2003. From Dynamic Binding to State via Modal Possibility. In *Proceedings of the 5th ACM SIGPLAN International Conference on Principles and Practice of Declaritive Programming* (Uppsala, Sweden) *(PPDP '03)*. Association for Computing Machinery, New York, NY, USA, 207–218. https://doi.org/10.1145/888251.888271

Dominic Orchard and Harley Eades III et al. 2022. The Granule Project. https://granule-project.github.io/

Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. 2019. Quantitative Program Reasoning with Graded Modal Types. *Proc. ACM Program. Lang.* 3, ICFP, Article 110 (July 2019), 30 pages. https://doi.org/10.1145/3341714

Pierre-Marie Pédrot and Nicolas Tabareau. 2019. The Fire Triangle: How to Mix Substitution, Dependent Elimination, and Effects. *Proc. ACM Program. Lang.* 4, POPL, Article 58 (dec 2019), 28 pages. https://doi.org/10.1145/3371126

Pierre-Marie Pédrot, Nicolas Tabareau, Hans Jacob Fehrmann, and Éric Tanter. 2019. A Reasonably Exceptional Type Theory. *Proc. ACM Program. Lang.* 3, ICFP, Article 108 (jul 2019), 29 pages. https://doi.org/10.1145/3341712

Tomas Petricek. 2017. *Context-aware programming languages*. Ph. D. Dissertation. https://doi.org/10.48456/tr-906

Tomas Petricek, Dominic Orchard, and Alan Mycroft. 2014. Coeffects: A Calculus of Context-Dependent Computation. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming* (Gothenburg, Sweden) *(ICFP '14)*. Association for Computing Machinery, New York, NY, USA, 123–135. https://doi.org/10.1145/2628136.2628160

Gordon Plotkin and Matija Pretnar. 2008. A Logic for Algebraic Effects. In *2008 23rd Annual IEEE Symposium on Logic in Computer Science*. 118–129. https://doi.org/10.1109/LICS.2008.45

Vineet Rajani, Marco Gaboardi, Deepak Garg, and Jan Hoffmann. 2021. A Unifying Type-Theory for Higher-Order (Amortized) Cost Analysis. *Proc. ACM Program. Lang.* 5, POPL, Article 27 (jan 2021), 28 pages. https://doi.org/10.1145/3434308

Jason Reed and Benjamin C. Pierce. 2010. Distance Makes the Types Grow Stronger: A Calculus for Differential Privacy. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming* (Baltimore, Maryland, USA) *(ICFP '10)*. Association for Computing Machinery, New York, NY, USA, 157–168. https://doi.org/10.1145/1863543.1863568

Nick Rioux and Steve Zdancewic. 2020. Computation Focusing. *Proc. ACM Program. Lang.* 4, ICFP, Article 95 (aug 2020), 27 pages. https://doi.org/10.1145/3408977

Christine Rizkallah, Dmitri Garbuzov, and Steve Zdancewic. 2018. A Formal Equational Theory for Call-By-Push-Value. In *Interactive Theorem Proving*, Jeremy Avigad and Assia Mahboubi (Eds.). Springer International Publishing, Cham, 523–541.

Ulrich Schöpp. 2015. *Computation-by-Interaction for Structuring Low-Level Computation*. Ph. D. Dissertation. Habilitation thesis, Ludwig-Maximilians-Universität München.

Jean-Pierre Talpin and Pierre Jouvelot. 1994. The Type and Effect Discipline. *Inf. Comput.* 111, 2 (1994), 245–296. https://doi.org/10.1006/inco.1994.1046

Verse development team. 2023. *Verse Language Reference*. Epic Games. https://dev.epicgames.com/documentation/en-us/uefn/verse-language-reference.

Philip Wadler and Peter Thiemann. 2003. The Marriage of Effects and Monads. *ACM Trans. Comput. Logic* 4, 1 (jan 2003), 1–32. https://doi.org/10.1145/601775.601776

Maxi Wuttke. 2021. *Sound and Relatively Complete Coeffect and effect refinement type systems for call-by-push-value PCF*. Master's thesis. Universität des Sarrlandes.