OxCaml's Modes, Adjoint Natural Deduction, and Back

ANTON LORENZEN, University of Edinburgh, UK

We relate the Adjoint Natural Deduction to the new mode system proposed for OCaml.

We relate the Mode Calculus (Figure 2 of Lorenzen et al. [2024]) to Adjoint Natural Deduction (Figure 3 of Jang et al. [2024]) and vice versa. We only show this for one mode axis, where we have the two modes *affine* A and *unrestricted* U corresponding to `many` and `once` in OCaml. Other modes are tricky, since they are either not available in Adjoint Natural Deduction (stack allocation, uniqueness) or not available in OCaml (strictness/relevance). We further exclude space credits ♣ (and with it in-place reuse and borrowing) since these features are not available in Adjoint Natural Deduction and we exclude linear "with" $A \mathbin{\&} B$ which is not available in OCaml.

A major difference between the two calculi is their treatment of submoding. In the Mode Calculus, each term at mode `many` is also valid at mode `once` without any changes to its type. For example, given a list of elements at mode `many`, and an element at mode `once`, we can cons the element on the list and obtain a result list at mode `once`. In contrast, in the Adjoint Natural Deduction calculus, we need to explicitly downcast terms, which includes changing the type to a downshifted version of the original type. On the flipside, we could easily extend the Adjoint Natural Deduction calculus by types that only exist at certain modes.[1] In the Mode Calculus, each type is available at all modes (but we can restrict introduction forms to certain modes). Due to these differences, we will ignore the issue of submoding altogether.

1 EMBEDDING THE MODE CALCULUS IN ADJOINT NATURAL DEDUCTION

For a mode judgement $\Gamma \vdash e : \tau \mathbin{@} \mu$, we transform $\mu \Rightarrow m$ and $t \mathbin{@} \mu \Rightarrow A_m$. Modes:

$$\begin{aligned}[\![\mathsf{many}]\!] &= U \\ [\![\mathsf{once}]\!] &= A\end{aligned}$$

Keep in mind that the submoding relationship is `many` < `once` in the Mode Calculus, but `A` < `U` in Adjoint Natural Deduction. Interpretation of type and mode pairs:

$$\begin{aligned}
[\![1 \mathbin{@} \mu]\!] &= 1_{[\![\mu]\!]} \\
[\![\tau_1 + \tau_2 \mathbin{@} \mu]\!] &= \oplus\{\, \mathsf{inl} : [\![\tau_1 \mathbin{@} \mu]\!],\ \mathsf{inr} : [\![\tau_2 \mathbin{@} \mu]\!] \,\} \\
[\![\tau_1 \times \tau_2 \mathbin{@} \mu]\!] &= [\![\tau_1 \mathbin{@} \mu]\!] \otimes [\![\tau_2 \mathbin{@} \mu]\!] \\
[\![(\tau_1 \mathbin{@} \mu_1 \to \tau_2 \mathbin{@} \mu_2) \mathbin{@} \mu_3]\!] &= \uparrow_{\bot}^{[\![\mu_3]\!]} \left(\downarrow_{\bot}^{[\![\mu_1]\!]} [\![\tau_1 \mathbin{@} \mu_1]\!] \multimap \downarrow_{\bot}^{[\![\mu_2]\!]} [\![\tau_2 \mathbin{@} \mu_2]\!]\right) \\
[\![\, \Box^M \tau \mathbin{@} \mu]\!] &= \downarrow_{\mu}^{[\![\mathsf{many}]\!]} [\![\, \tau \mathbin{@} \mathsf{many} \,]\!]
\end{aligned}$$

The only translation which might be surprising is the one for the arrow type. Here, we create the arrow on the side of Adjoint Natural Deduction at the bottom mode $\bot$ (`A` in this case). This is maximally restrictive, but we can then upcast the arrow to any mode μ_3, and use arguments and returns of any modes μ_1 and μ_2. In particular, these three modes are unrelated and no constraints would be generated between them.

We translate contexts by translating the types. The contexts of the mode calculus do not contain explicit locks, but the locks act as an operation of the context that deletes inaccessible variables.

$$\begin{aligned}[\![\, \cdot \,]\!] &= \cdot \\ [\![\Gamma,\ x : \tau \mathbin{@} \mu]\!] &= [\![\Gamma]\!]\,,\ x : [\![\, \tau \mathbin{@} \mu \,]\!]\end{aligned}$$

[1] In Adjoint Natural Deduction, we already assume that $m \geqslant k$ implies $\sigma(m) \supseteq \sigma(k)$. We could strengthen this to an inclusion of type universes such that $m \geqslant k$ and $x : A_m$ implies $x : A_k$. This would then allow us to add a submoding rule to the calculus. A calculus with separate type universes, modalties and submoding can be modelled as a double category [Katsumata 2018; Tang et al. 2024].

The translation for introduction terms follows directly from the translation of types above (where free variables introduced by the translation are assumed to be fresh):

$$
\begin{aligned}
[\![()]\!] &= () \\
[\![x]\!] &= \mathsf{down}\ x \\
[\![\mathsf{inl}\ e]\!] &= \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ (\mathsf{inl}(x))) \\
[\![\mathsf{inr}\ e]\!] &= \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ (\mathsf{inr}(x))) \\
[\![(e_1,\ e_2)]\!] &= \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{match}\ [\![e_2]\!]\ (\mathsf{down}\ y \Rightarrow\ \mathsf{down}\ (x,\ y))) \\
[\![\lambda x.\ e]\!] &= \mathsf{down}\ (\mathsf{susp}\ (\lambda x.\ \mathsf{match}\ x\ (\mathsf{down}\ x \Rightarrow\ [\![e]\!]))) \\
[\![\mathsf{box}_M\ e]\!] &= \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ (\mathsf{down}\ x))
\end{aligned}
$$

I do not pay special attention to synthesizability in the translation above. It may thus be necessary to add extra type annotations into the translated term; but this would be straightforward to do. Similarly for eliminations:

$$
\begin{aligned}
[\![\mathsf{case}\ e_1\ \mathsf{with}\ \{\ \mathsf{inl}\ x \Rightarrow\ e_2;\ \mathsf{inr}\ y \Rightarrow\ e_3\ \}]\!] &= \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ z \Rightarrow \\
&\qquad \mathsf{match}\ z\ (\mathsf{inl}(x) \Rightarrow\ [\![e_2]\!];\ \mathsf{inr}(y) \Rightarrow\ [\![e_3]\!])) \\
[\![\mathsf{let}\ (x,\ y)\ =\ e_1\ \mathsf{in}\ e_2]\!] &= \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ z \Rightarrow\ \mathsf{match}\ z\ ((x,\ y) \Rightarrow\ [\![e_2]\!])) \\
[\![e_1\ e_2]\!] &= \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ x \Rightarrow\ (\mathsf{force}\ x)\ [\![e_2]\!]) \\
[\![\mathsf{unbox}_M\ e]\!] &= \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{match}\ x\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ x))
\end{aligned}
$$

Notice that we change the syntax for splitting pairs slightly to avoid considering the extra space credit. With this we can prove the following lemma:

Lemma 1. (*Translation to AND*)
If $\Gamma \vdash e : \tau\ @\ \mu$, then $[\![\Gamma]\!] \vdash [\![e]\!] \Leftrightarrow\ \downarrow_{\bot}^{\mu} [\![\tau\ @\ \mu]\!]$.

2 EMBEDDING ADJOINT NATURAL DEDUCTION IN THE MODE CALCULUS

Conversely, we can embed Adjoint Natural Deduction in the Mode Calculus.

$$
\begin{aligned}
[\![U]\!] &= \mathsf{many} \\
[\![A]\!] &= \mathsf{once}
\end{aligned}
$$

Interpretation of types. For simplicity, we assume that all sums contain only two labels.

$$
\begin{aligned}
[\![1_m]\!] &= 1 \\
\oplus\{\ \mathsf{inl} : A_m,\ \mathsf{inr} : B_m\ \} &= [\![A_m]\!] + [\![B_m]\!] \\
[\![A_m \otimes B_m]\!] &= [\![A_m]\!] \times [\![B_m]\!] \\
(A_m \multimap B_m) &= [\![A_m]\!]\ @\ [\![m]\!] \to [\![B_m]\!]\ @\ [\![m]\!] \\
[\![\downarrow_m^U\ A_U]\!] &= \Box^M\ [\![A_U]\!] \\
[\![\downarrow_A^A\ A_A]\!] &= [\![A_A]\!] \\
[\![\uparrow_k^m\ A_k]\!] &= 1\ @\ \mathsf{many} \to [\![A_k]\!]\ @\ [\![k]\!]
\end{aligned}
$$

Notice that in the Mode Calculus there is no box type for once variables. In particular, while we expect to support mode polymorphism over the modes on function arrows in the future, we will not be able to express mode polymorphism over boxes.

Contexts:

$$
\begin{aligned}
[\![\cdot]\!] &= \cdot \\
[\![\Gamma,\ x : A_m]\!] &= [\![\Gamma]\!],\ x : [\![\ A\ ]\!]\ @\ [\![\ m\ ]\!]
\end{aligned}
$$

The translation for introduction terms follows directly from the translation of types above (where free variables introduced by the translation are assumed to be fresh):

$$
\begin{array}{lcl}
[\![()]\!] & = & () \\
[\![x]\!] & = & x \\
[\![\mathsf{inl}\ e]\!] & = & \mathsf{inl}\ [\![e]\!] \\
[\![\mathsf{inr}\ e]\!] & = & \mathsf{inr}\ [\![e]\!] \\
[\![(e_1,\ e_2)]\!] & = & ([\![e_1]\!],\ [\![e_2]\!]) \\
[\![\lambda x.\ e]\!] & = & \lambda x.\ [\![e]\!] \\
[\![\mathsf{down}_U\ e]\!] & = & \mathsf{box}_M\ [\![e]\!] \\
[\![\mathsf{down}_A\ e]\!] & = & [\![e]\!] \\
[\![\mathsf{susp}\ e]\!] & = & \lambda x.\ [\![e]\!]
\end{array}
$$

and similarly for eliminations:

$$
\begin{array}{lcl}
[\![\mathsf{match}\ s\ (\mathsf{inl}(x) \Rightarrow\ e_1;\ \mathsf{inr}(y) \Rightarrow\ e_2)]\!] & = & \mathsf{case}\ [\![s]\!]\ \mathsf{with}\ \{\ \mathsf{inl}\ x \Rightarrow\ [\![e_1]\!];\ \mathsf{inr}\ y \Rightarrow\ [\![e_2]\!]\ \} \\
[\![\mathsf{match}\ s\ ((x,\ y) \Rightarrow\ e)]\!] & = & \mathsf{let}\ (x,\ y)\ =\ [\![s]\!]\ \mathsf{in}\ [\![e]\!] \\
[\![\mathsf{match}\ s\ (\mathsf{down}_U\ x \Rightarrow\ e)]\!] & = & \mathsf{let}\ x\ =\ \mathsf{unbox}_M\ [\![s]\!]\ \mathsf{in}\ [\![e]\!] \\
[\![\mathsf{match}\ s\ (\mathsf{down}_A\ x \Rightarrow\ e)]\!] & = & \mathsf{let}\ x\ =\ [\![s]\!]\ \mathsf{in}\ [\![e]\!] \\
[\![\mathsf{match}\ s\ (() \Rightarrow\ e)]\!] & = & \mathsf{let}\ x\ =\ [\![s]\!]\ \mathsf{in}\ [\![e]\!] \\
[\![\mathsf{force}\ s]\!] & = & [\![s]\!]\ () \\
[\![s\ e]\!] & = & [\![s]\!]\ [\![e]\!]
\end{array}
$$

With this, we can prove the following lemma:

Lemma 2. (*Translation from AND*)
If $\Delta \vdash e \Leftrightarrow\ A_m$, then $[\![\Delta]\!] \vdash [\![e]\!] : [\![A]\!] \ @\ [\![m]\!]$.

3 PROOFS

Lemma 3. (*Context splitting*)
We have $[\![\Gamma_1\ +\ \Gamma_2]\!]\ =\ [\![\Gamma_1]\!]\ ;\ [\![\Gamma_2]\!]$.

Proof. By straightforward induction.

Lemma 4. (*Correctness of the translation*)
If $\Gamma \vdash e : \tau\ @\ \mu$, then $[\![\Gamma]\!] \vdash [\![e]\!] \Leftrightarrow\ \ \downarrow^{\mu}_{\bot} [\![\tau\ @\ \mu]\!]$.

Proof. By induction over the typing derivation. We re-state the rules of the Mode Calculus with the premises and conclusion translated, and then derive the rule in the Adjoint Natural Deduction calculus.

$$\frac{}{[\![\Gamma]\!],\ x : [\![\tau\ @\ \mu]\!],\ [\![\Gamma']\!] \vdash \mathsf{down}\ x \Leftrightarrow\ \ \downarrow^{\mu}_{\bot} [\![\tau\ @\ \mu]\!]}\ \textsc{var}$$

By the HYP and $[\downarrow I]$ rules. Notice that all modes considered here have weakening.

$$\frac{[\![\Gamma,\ \text{🔒}_{\mu_3}]\!],\ x : [\![\tau_1\ @\ \mu_1]\!] \vdash [\![e]\!] \Leftrightarrow\ \ \downarrow^{\mu_2}_{\bot} [\![\tau_2\ @\ \mu_2]\!]}{[\![\Gamma]\!] \vdash \mathsf{down}\ (\mathsf{susp}\ (\lambda x.\ \mathsf{match}\ x\ (\mathsf{down}\ x \Rightarrow\ [\![e]\!]))) \Leftrightarrow\ \ \downarrow^{\mu_3}_{\bot}\uparrow^{\mu_3}_{\bot} (\downarrow^{\mu_1}_{\bot} [\![\tau_1\ @\ \mu_1]\!] \multimap \downarrow^{\mu_2}_{\bot} [\![\tau_2\ @\ \mu_2]\!])}\ \textsc{lam}$$

The 🔒$_{\mu_3}$ operation deletes all variables from Γ that are at a higher mode than μ_3. Conversely, in the translated version, all variables at a lower mode than $[\![\mu_3]\!]$ are deleted. This allows to use the $[\downarrow I]$ rule. We can eliminate the $\downarrow^{\mu_1}_{\bot}$ of the argument, since the return value of $[\![e]\!]$ is also at the bottom mode. This translation also motivates our choice of the 🔒 symbol in the function abstraction rule: our functions act as IS4 boxes (see Example 8 of Jang et al. [2024]).

$$\frac{[\![\Gamma_1]\!] \vdash [\![e_1]\!] \Leftrightarrow\ \ \downarrow^{\mu_3}_{\bot}\uparrow^{\mu_3}_{\bot} (\downarrow^{\mu_1}_{\bot} [\![\tau_1\ @\ \mu_1]\!] \multimap \downarrow^{\mu_2}_{\bot} [\![\tau_2\ @\ \mu_2]\!]) \quad [\![\Gamma_2]\!] \vdash [\![e_2]\!] : \downarrow^{\mu_1}_{\bot} [\![\tau_1\ @\ \mu_1]\!]}{[\![\Gamma_1]\!]\ ;\ [\![\Gamma_2]\!] \vdash \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ x \Rightarrow\ (\mathsf{force}\ x)\ [\![e_2]\!]) \Leftrightarrow\ \ \downarrow^{\mu_2}_{\bot} [\![\tau_2\ @\ \mu_2]\!]}\ \textsc{app}$$

We can eliminate the $\downarrow_\bot^{\mu_3}$ of the function, since the result is at mode bottom.

$$\frac{}{[\![\Gamma]\!] \vdash () \Leftrightarrow 1_m}\ \textsc{unit}$$

Obvious.

$$\frac{[\![\Gamma]\!] \vdash [\![e]\!] \Leftrightarrow \downarrow_\bot^\mu [\![\tau_1 @ \mu]\!]}{[\![\Gamma]\!] \vdash \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ (\mathsf{inl}(x))) \Leftrightarrow\ \downarrow_\bot^\mu \oplus \mathsf{inl} : [\![\tau_1 @ \mu]\!];\ \mathsf{inr} : [\![\tau_2 @ \mu]\!]}\ \textsc{inl}$$

$$\frac{[\![\Gamma]\!] \vdash [\![e]\!] \Leftrightarrow \downarrow_\bot^\mu [\![\tau_2 @ \mu]\!]}{[\![\Gamma]\!] \vdash \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ (\mathsf{inr}(x))) \Leftrightarrow\ \downarrow_\bot^\mu \oplus \mathsf{inl} : [\![\tau_1 @ \mu]\!];\ \mathsf{inr} : [\![\tau_2 @ \mu]\!]}\ \textsc{inr}$$

$$\frac{[\![\Gamma_1]\!] \vdash [\![e_1]\!] \Leftrightarrow \downarrow_\bot^\mu [\![\tau_1 @ \mu]\!] \quad [\![\Gamma_2]\!] \vdash [\![e_2]\!] \Leftrightarrow \downarrow_\bot^\mu [\![\tau_2 @ \mu]\!]}{[\![\Gamma_1]\!]\ ;\ [\![\Gamma_2]\!] \vdash \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{match}\ [\![e_2]\!]\ (\mathsf{down}\ y \Rightarrow\ \mathsf{down}\ (x,\ y))) \Leftrightarrow\ \downarrow_\bot^\mu [\![\tau_1 @ \mu]\!] \otimes [\![\tau_2 @ \mu]\!]}\ \textsc{pair}$$

We can eliminate the $\downarrow_\bot^\mu$ of the argument at mode bottom. We can then re-introduce the downarrow since we only use the result at mode $[\![\mu]\!]$.

$$\frac{\begin{array}{c}[\![\Gamma_1]\!] \vdash [\![e_1]\!] \Leftrightarrow \downarrow_\bot^{\mu_1} \oplus \mathsf{inl} : [\![\tau_1 @ \mu_1]\!];\ \mathsf{inr} : [\![\tau_2 @ \mu_1]\!] \\ [\![\Gamma_2]\!],\ x : [\![\tau_1 @ \mu_1]\!] \vdash [\![e_2]\!] \Leftrightarrow \downarrow_\bot^{\mu_2} [\![\tau_3 @ \mu_2]\!] \\ [\![\Gamma_2]\!],\ y : [\![\tau_2 @ \mu_1]\!] \vdash [\![e_3]\!] \Leftrightarrow \downarrow_\bot^{\mu_2} [\![\tau_3 @ \mu_2]\!]\end{array}}{[\![\Gamma_1]\!]\ ;\ [\![\Gamma_2]\!] \vdash \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ z \Rightarrow\ \mathsf{match}\ z\ (\mathsf{inl}(x) \Rightarrow\ [\![e_2]\!];\ \mathsf{inr}(y) \Rightarrow\ [\![e_3]\!]) \Leftrightarrow\ \downarrow_\bot^{\mu_2} [\![\tau_3 @ \mu_2]\!]}\ \textsc{case}$$

$$\frac{\begin{array}{c}[\![\Gamma_1]\!] \vdash [\![e_1]\!] \Leftrightarrow \downarrow_\bot^{\mu_1} ([\![\tau_1 @ \mu_1]\!] \otimes [\![\tau_2 @ \mu_1]\!]) \\ [\![\Gamma_2]\!],\ x : [\![\tau_1 @ \mu_1]\!],\ y : [\![\tau_2 @ \mu_1]\!] \vdash [\![e_2]\!] \Leftrightarrow \downarrow_\bot^{\mu_2} [\![\tau_3 @ \mu_2]\!]\end{array}}{[\![\Gamma_1]\!]\ ;\ [\![\Gamma_2]\!] \vdash \mathsf{match}\ [\![e_1]\!]\ (\mathsf{down}\ z \Rightarrow\ \mathsf{match}\ z\ ((x,\ y) \Rightarrow\ [\![e_2]\!]) \Leftrightarrow\ \downarrow_\bot^{\mu_2} [\![\tau_3 @ \mu_2]\!]}\ \textsc{split}$$

As before, we eliminate the downarrow of the scrutinee. This gives us a value (sum or pair) at mode $[\![\mu_1]\!]$. When matching on this value, we need to ensure that every element in the context is larger or equal to $[\![\mu_1]\!]$, but this is clear since we only use the variables z in that context.

$$\frac{[\![\Gamma]\!] \vdash [\![e]\!] \Leftrightarrow \downarrow_\bot^U [\![t @ \mathsf{many}]\!]}{[\![\Gamma]\!] \vdash \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ (\mathsf{down}\ x)) \Leftrightarrow\ \downarrow_\bot^\mu \downarrow_\mu^U\ [\![t @ \mathsf{many}]\!]}\ \textsc{box}$$

We need to split the downshift into two downshifts, hence the seemingly redundant elimination and introduction of the downshift. This part of the translation shows that the box of the mode calculus corresponds to the downshift of Adjoint Natural Deduction.

$$\frac{[\![\Gamma]\!] \vdash [\![e]\!] \Leftrightarrow \downarrow_\bot^\mu \downarrow_\mu^U\ [\![t @ \mu]\!]}{[\![\Gamma]\!] \vdash \mathsf{match}\ [\![e]\!]\ (\mathsf{down}\ x \Rightarrow\ \mathsf{match}\ x\ (\mathsf{down}\ x \Rightarrow\ \mathsf{down}\ x)) \Leftrightarrow\ \downarrow_\bot^U [\![t @ \mathsf{many}]\!]}\ \textsc{unbox}$$

Conversely, in this rule, we have to merge two downshifts into one.

Lemma 5. (*Translation from AND*)
If $\Delta \vdash e \Leftrightarrow A_m$, then $[\![\Delta]\!] \vdash [\![e]\!] : [\![A]\!] @ [\![m]\!]$.

Proof. By induction on the typing derivation. We re-state the rules of Adjoint Natural Deduction with the premises and conclusion translated, and then derive the rule in the Mode Calculus.

$$\frac{\Gamma\ =\ x : [\![A]\!] @ [\![m]\!]}{[\![\Delta_W]\!]\ +\ \Gamma \vdash x : [\![A]\!] @ [\![m]\!]}\ \textsc{hyp}$$

By the VAR rule.

$$\frac{[\![\Delta]\!],\ x : [\![A]\!] @ [\![m]\!] \vdash [\![e]\!] : [\![B]\!] @ [\![m]\!]}{[\![\Delta]\!] \vdash \lambda x.\ [\![e]\!] : ([\![A]\!] @ [\![m]\!] \to [\![B]\!] @ [\![m]\!]) @ [\![m]\!]}\ \mathrm{I}$$

Since we can assume that $\Delta \geqslant m$ and thus $[\![\Delta]\!] \leqslant [\![m]\!]$, we can omit the lock operation on $[\![\Delta]\!]$.

$$\frac{[\![\Delta]\!] \vdash [\![s]\!] : ([\![A]\!] @ [\![m]\!] \to [\![B]\!] @ [\![m]\!]) @ [\![m]\!] \quad [\![\Delta']\!] \vdash [\![e]\!] : [\![A]\!] @ [\![m]\!]}{[\![\Delta]\!] + [\![\Delta']\!] \vdash [\![s]\!]\ [\![e]\!] : [\![B]\!] @ [\![m]\!]}\ \mathsf{E}$$

Obvious.

$$\frac{[\![\Delta]\!] \vdash [\![e]\!] : [\![A]\!] @ [\![k]\!]}{[\![\Delta]\!] \vdash \lambda x.\ [\![e]\!] : (1 @ \mathsf{many} \to [\![A]\!] @ [\![k]\!]) @ [\![m]\!]}\ \mathrm{I}$$

Since $m \geqslant k$, thus $[\![m]\!] \leqslant [\![k]\!]$. Since we can assume that $\Delta \geqslant m$, and thus $[\![\Delta]\!] \leqslant [\![m]\!] \leqslant [\![k]\!]$, we can omit the lock operation on $[\![\Delta]\!]$.

$$\frac{\Delta' \geqslant m \quad [\![\Delta']\!] \vdash [\![s]\!] : (1 @ \mathsf{many} \to [\![A]\!] @ [\![k]\!]) @ [\![m]\!]}{[\![\Delta_W]\!] + [\![\Delta']\!] \vdash [\![s]\!]\ () : [\![A]\!] @ [\![k]\!]}\ \mathsf{E}$$

Obvious. Note that we do not need the side-condition $\Delta' \geqslant m$ directly (though it has to be present to maintain the presupposition).

$$\frac{[\![\Delta]\!] \vdash [\![e_1]\!] : [\![A]\!] @ [\![m]\!] \quad [\![\Delta']\!] \vdash [\![e_2]\!] : [\![B]\!] @ [\![m]\!]}{[\![\Delta]\!] + [\![\Delta']\!] \vdash ([\![e_1]\!],\ [\![e_2]\!]) : [\![A]\!] \times [\![B]\!] @ [\![m]\!]}\ \mathrm{I}$$

$$\frac{}{[\![\Delta_W]\!] \vdash () : 1 @ [\![m]\!]}\ \mathrm{1I}$$

$$\frac{[\![\Delta]\!] \vdash [\![e]\!] : [\![A]\!] @ [\![m]\!]}{[\![\Delta]\!] \vdash \mathsf{inl}\ [\![e]\!] : [\![A]\!] + [\![B]\!] @ [\![m]\!]}\ \mathrm{I}$$

$$\frac{[\![\Delta]\!] \vdash [\![e]\!] : [\![A]\!] @ [\![m]\!]}{[\![\Delta]\!] \vdash \mathsf{inr}\ [\![e]\!] : [\![A]\!] + [\![B]\!] @ [\![m]\!]}\ \mathrm{I}$$

Obvious.

$$\frac{\begin{array}{l}[\![\Delta]\!] \vdash [\![s]\!] : [\![A]\!] \times [\![B]\!] @ [\![m]\!] \quad \Delta \geqslant m \geqslant r \\ [\![\Delta']\!],\ x : [\![A]\!] @ [\![m]\!],\ y : [\![B]\!] @ [\![m]\!] \vdash [\![e']\!] : [\![C]\!] @ [\![r]\!]\end{array}}{[\![\Delta]\!] + [\![\Delta']\!] \vdash \mathsf{let}\ (x,\ y)\ =\ [\![s]\!]\ \mathsf{in}\ [\![e']\!] : [\![C]\!] @ [\![r]\!]}\ \mathsf{E}$$

$$\frac{\begin{array}{l}[\![\Delta]\!] \vdash [\![s]\!] : 1 @ [\![m]\!] \quad \Delta \geqslant m \geqslant r \\ [\![\Delta']\!] \vdash [\![e']\!] : [\![C]\!] @ [\![r]\!]\end{array}}{[\![\Delta]\!] + [\![\Delta']\!] \vdash \mathsf{let}\ x\ =\ [\![s]\!]\ \mathsf{in}\ [\![e']\!] : [\![C]\!] @ [\![r]\!]}\ \mathrm{1E}$$

$$\frac{\begin{array}{l}[\![\Delta]\!] \vdash [\![s]\!] : [\![A]\!] + [\![B]\!] @ [\![m]\!] \quad \Delta \geqslant m \geqslant r \\ [\![\Delta']\!],\ x : [\![A]\!] @ [\![m]\!] \vdash [\![e_1]\!] : [\![C]\!] @ [\![r]\!] \quad [\![\Delta']\!],\ y : [\![B]\!] @ [\![m]\!] \vdash [\![e_2]\!] : [\![C]\!] @ [\![r]\!]\end{array}}{[\![\Delta]\!] + [\![\Delta']\!] \vdash \mathsf{case}\ [\![s]\!]\ (\mathsf{inl}\ x \to [\![e_1]\!];\ \mathsf{inr}\ y \to [\![e_2]\!]) : [\![C]\!] @ [\![r]\!]}\ \mathsf{E}$$

Obvious. Note that we do not need the side-conditions $\Delta \geqslant m \geqslant r$ directly (though they have to be present to maintain the presupposition).

$$\frac{\Delta' \geqslant n \quad [\![\Delta']\!] \vdash [\![e]\!] : [\![A]\!] \mathbin{@} \mathsf{many}}{[\![\Delta_W]\!] + [\![\Delta']\!] \vdash \mathsf{box}_M\, [\![e]\!] : \Box^M\, [\![A]\!] \mathbin{@} [\![m]\!]}\ \mathrm{I}$$

$$\frac{\Delta' \geqslant \mathsf{once} \quad [\![\Delta']\!] \vdash [\![e]\!] : [\![A]\!] \mathbin{@} \mathsf{once}}{[\![\Delta_W]\!] + [\![\Delta']\!] \vdash [\![e]\!] : [\![A]\!] \mathbin{@} \mathsf{once}}\ \mathrm{I}$$

Follows from the BOX-rule. The SUB-rule allows us to discard the $[\![\Delta_W]\!]$ context.

$$\frac{\begin{array}{c}[\![\Delta]\!] \vdash [\![s]\!] : \Box^M\, [\![A]\!] \mathbin{@} [\![m]\!] \quad \Delta \geqslant m \geqslant r \\ [\![\Delta']\!],\ x : [\![A]\!] \mathbin{@} \mathsf{many} \vdash [\![e']\!] : [\![C]\!] \mathbin{@} [\![r]\!]\end{array}}{[\![\Delta]\!] + [\![\Delta']\!] \vdash \mathsf{let}\ x = \mathsf{unbox}_M\, [\![s]\!]\ \mathsf{in}\ [\![e']\!] : [\![C]\!] \mathbin{@} [\![r]\!]}\ \mathrm{E}$$

$$\frac{\begin{array}{c}[\![\Delta]\!] \vdash [\![s]\!] : [\![A]\!] \mathbin{@} \mathsf{once} \quad \Delta \geqslant \mathsf{once} \\ [\![\Delta']\!],\ x : [\![A]\!] \mathbin{@} \mathsf{once} \vdash [\![e']\!] : [\![C]\!] \mathbin{@} [\![r]\!]\end{array}}{[\![\Delta]\!] + [\![\Delta']\!] \vdash \mathsf{let}\ x = [\![s]\!]\ \mathsf{in}\ [\![e']\!] : [\![C]\!] \mathbin{@} [\![r]\!]}\ \mathrm{E}$$

Obvious.

REFERENCES

Jang, Roshal, Pfenning, and Pientka. 2024. Adjoint Natural Deduction. In *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Katsumata. 2018. A Double Category Theoretic Analysis of Graded Linear Exponential Comonads. In *Foundations of Software Science and Computation Structures*, edited by Christel Baier and Ugo Dal Lago, 110–127. Springer International Publishing, Cham.

Lorenzen, White, Stephen, Eisenberg, and Lindley. 2024. Oxidizing OCaml with Modal Memory Management. *Proceedings of the ACM on Programming Languages* 8 (ICFP). ACM New York, NY, USA. https://antonlorenzen.de/oxidizing-ocaml-modal-memory-management.pdf.

Tang, White, Dolan, Hillerström, Lindley, and Lorenzen. 2024. Modal Effect Types. arXiv:cs.PL/2407.11816.