Institute of SCIENCE TOKYO

Matrix Coeffect Algebra for Interdependent Context Requirements

FLOPS2026

Osamu Miyazawa, Shin-ya Nishizaki
Institute of Science Tokyo

2026/05/26

Main Idea

- Program requirements may depend on each other.
- Matrices naturally represent such dependencies.
- We propose Matrix Coeffect Algebra for structural coeffect calculus.

1. Problem: Why are scalar/componentwise coeffects not enough?
2. Solution: How do matrices represent interdependent requirements?
3. Justification: Why is the system sound?

1. Problem: Why are scalar/componentwise coeffects not enough?
2. Solution: How do matrices represent interdependent requirements?
3. Justification: Why is the system sound?

Coeffects: The Dual of Effects

- Effect System: Tracking **how a program affects** the outside world

$$\langle x : \tau \rangle \vdash \text{print } x : \text{unit } \& \ \{\text{console}\}$$

- Coeffect System: Tracking **how the context affects** a program

$$\langle x : \tau \rangle \ @ \ \langle 3 \rangle \vdash x + x + x : \tau$$

Coeffects let us track context requirements using a type system.

Structural Coeffect Calculus (Petricek et al., 2014)

Annotating programs with **per-variable** context requirements tracked by types (i.e., coeffects):

- Free-variable occurrence counts

$$\langle x : \tau, y : \tau \rangle \;@\; \langle \overset{x}{\sqcap}\, 3, \overset{y}{\sqcap}\, 2 \rangle \vdash (x + x + x) + (y + y) : \tau$$

- Past-value requirements in dataflow languages (caching depth)

$$\langle x : \tau, y : \tau \rangle \;@\; \langle \overset{x}{\sqcap}\, 2, \overset{y}{\sqcap}\, 1 \rangle \vdash \mathsf{prev}(\mathsf{prev}\ x) + \mathsf{prev}\ y : \tau$$

Multiple Coeffects

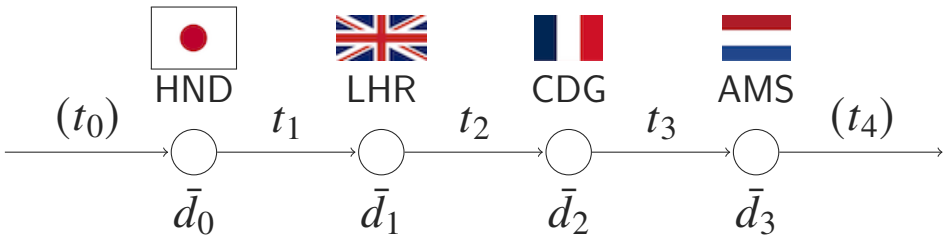
Petricek's framework can handle multiple coeffects, but only when they are **independent**.

$$\langle x : \tau, y : \tau \rangle \ @ \ \langle \overbrace{\langle 1, 2 \rangle}^{x}, \overbrace{\langle 3, 0 \rangle}^{y} \rangle \vdash \mathsf{prev}(\mathsf{prev}\ x) + (y + y + y) : \tau$$

The occurrence-count coeffect and the caching-depth coeffect are independent of each other.

However, scalar and (componentwise) vector annotations do not naturally express **interdependent** coeffects.

Motivating Example: Flight Timetables (1)



t_i : the duration of the i-th leg

$\bar{d}_i$: the scheduled departure time in the published timetable (a lower bound on departure time)

Motivating Example: Flight Timetables (2)

A traveler cannot depart from airport i before

- the scheduled departure time $\bar{d}_i$
- the arrival time from the previous leg $d_{i-1} + t_i$

So the earliest feasible departure time is

$$d_i = \max(\bar{d}_i,\ d_{i-1} + t_i)$$

- If $\bar{d}_i \geq d_{i-1} + t_i$, then $d_i = \bar{d}_i$
- If $\bar{d}_i < d_{i-1} + t_i$, then $d_i = d_{i-1} + t_i$

Motivating Example: Flight Timetables (3)

- In this domain, the *duration* t_i and the *scheduled time* $\bar{d}_i$ are intertwined and propagate across legs.
- Their contributions mix through $\max$ and $+$, updating d_i.

$$\Downarrow$$

Travel times t_i and scheduled times $\bar{d}_i$ are **interdependent**.

Key Idea: Mixing Requirements with Matrices

- Composing legs mixes the contributions of $\bar{d}_i$ and t_i (via $\max$ and $+$).
- We need a composition operator that mixes components (not component-wise).
- **Matrices** naturally combines and propagates requirements.
- Contributions:
  - Define the **Matrix Coeffect Algebra** (**MCA**)
  - Instantiate Petricek's structural coeffect calculus using MCA
  - Prove type safety via a type-preserving translation

1. Problem: Why are scalar/componentwise coeffects not enough?
2. Solution: How do matrices represent interdependent requirements?
3. Justification: Why is the system sound?

Matrix Coeffect Algebra (1)

Definition 1 (Matrix Coeffect Algebra)

A *matrix coeffect algebra* $(C, \circledast, \oplus, \mathsf{use}, \mathsf{ign}, \leqslant)$ is defined by:

- C is the set of $n \times n$ matrices over a fixed underlying domain
- $(C, \circledast, \mathsf{use})$ is a **monoid**
- $(C, \oplus, \mathsf{ign})$ is a **monoid**
- $(C, \leqslant)$ is a **preorder** (reflexivity and transitivity)

In addition, the following **distributivity** axioms hold:

$$(R \oplus S) \circledast T = (R \circledast T) \oplus (S \circledast T)$$
$$T \circledast (R \oplus S) = (T \circledast R) \oplus (T \circledast S)$$

Matrix Coeffect Algebra (2)

Roles of the operators:

- $\circledast$: **Propagation** of requirements
- $\oplus$: (Pointwise) **join** of requirements
- $\leqslant$: **Subcoeffecting** (a preorder)

MCA-Parametric Structural Coeffect Calculus (1)

Syntax: $e ::= x \mid n \mid \lambda x{:}\tau.e \mid e_1\ e_2 \mid \text{let } x = e_1 \text{ in } e_2$

Types: $\tau ::= \text{num} \mid \tau_1 \xrightarrow{s} \tau_2$ (where s is a coeffect annotation)

The operations and constants of MCA correspond to typing and structural rules as follows:

$\circledast$: (App, Let)

$\oplus$: (Contr)

use : (Var)

ign : (Weak)

$\leqslant$: (Sub)

MCA-Parametric Structural Coeffect Calculus (2)

- Typing derivations track coeffects in a compositional way (as MCA elements, i.e., matrices).
- We retain the rules of structural coeffect calculus but replace scalar/vector annotations with MCA elements (matrices).

MCA-Parametric Structural Coeffect Calculus (3)

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\,(\textsc{Var}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{s} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} +\!\!+ (s \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\,(\textsc{App})$$

$$\frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} +\!\!+ \langle s \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{s} \tau_2}\,(\textsc{Abs}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} +\!\!+ \langle s \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (s \circledast \mathbf{p}) +\!\!+ \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\,(\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} +\!\!+ \langle \mathsf{ign} \rangle \vdash e : \tau}\,(\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s} \rangle \vdash e : \tau}\,(\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s}, \mathsf{t} \rangle +\!\!+ \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} +\!\!+ \langle \mathsf{s} \oplus \mathsf{t} \rangle +\!\!+ \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\,(\textsc{Contr})$$

MCA-Parametric Structural Coeffect Calculus (3)

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\,(\textsc{Var}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{s} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} +\!\!+ (s \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\,(\textsc{App})$$

$$\frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} +\!\!+ \langle s \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{s} \tau_2}\,(\textsc{Abs}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} +\!\!+ \langle s \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (s \circledast \mathbf{p}) +\!\!+ \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\,(\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} +\!\!+ \langle \mathsf{ign} \rangle \vdash e : \tau}\,(\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s} \rangle \vdash e : \tau}\,(\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s}, \mathsf{t} \rangle +\!\!+ \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} +\!\!+ \langle \mathsf{s} \oplus \mathsf{t} \rangle +\!\!+ \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\,(\textsc{Contr})$$

MCA-Parametric Structural Coeffect Calculus (3)

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\,(\textsc{Var}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{s} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} \nplus (s \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\,(\textsc{App})$$

$$\frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} \nplus \langle s \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{s} \tau_2}\,(\textsc{Abs}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} \nplus \langle s \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (s \circledast \mathbf{p}) \nplus \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\,(\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} \nplus \langle \mathsf{ign} \rangle \vdash e : \tau}\,(\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} \nplus \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} \nplus \langle \mathsf{s} \rangle \vdash e : \tau}\,(\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} \nplus \langle \mathsf{s}, \mathsf{t} \rangle \nplus \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} \nplus \langle \mathsf{s} \oplus \mathsf{t} \rangle \nplus \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\,(\textsc{Contr})$$

MCA-Parametric Structural Coeffect Calculus (3)

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\,(\textsc{Var}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{s} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} \nplus (s \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\,(\textsc{App})$$

$$\frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} \nplus \langle s \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{s} \tau_2}\,(\textsc{Abs}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} \nplus \langle s \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (s \circledast \mathbf{p}) \nplus \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\,(\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} \nplus \langle \mathsf{ign} \rangle \vdash e : \tau}\,(\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} \nplus \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} \nplus \langle \mathsf{s} \rangle \vdash e : \tau}\,(\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} \nplus \langle \mathsf{s}, \mathsf{t} \rangle \nplus \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} \nplus \langle \mathsf{s} \oplus \mathsf{t} \rangle \nplus \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\,(\textsc{Contr})$$

MCA-Parametric Structural Coeffect Calculus (3)

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\,(\textsc{Var}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{s} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} \mathbin{+} (s \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\,(\textsc{App})$$

$$\frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} \mathbin{+} \langle s \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{s} \tau_2}\,(\textsc{Abs}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} \mathbin{+} \langle s \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (s \circledast \mathbf{p}) \mathbin{+} \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\,(\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} \mathbin{+} \langle \mathsf{ign} \rangle \vdash e : \tau}\,(\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} \mathbin{+} \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} \mathbin{+} \langle \mathsf{s} \rangle \vdash e : \tau}\,(\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} \mathbin{+} \langle \mathsf{s}, \mathsf{t} \rangle \mathbin{+} \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} \mathbin{+} \langle \mathsf{s} \oplus \mathsf{t} \rangle \mathbin{+} \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\,(\textsc{Contr})$$

MCA-Parametric Structural Coeffect Calculus (3)

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\,(\textsc{Var}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{s} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} +\!\!+ (s \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\,(\textsc{App})$$

$$\frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} +\!\!+ \langle s \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{s} \tau_2}\,(\textsc{Abs}) \qquad \frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} +\!\!+ \langle s \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (s \circledast \mathbf{p}) +\!\!+ \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\,(\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} +\!\!+ \langle \mathsf{ign} \rangle \vdash e : \tau}\,(\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s} \rangle \vdash e : \tau}\,(\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} +\!\!+ \langle \mathsf{s}, \mathsf{t} \rangle +\!\!+ \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} +\!\!+ \langle \mathsf{s} \oplus \mathsf{t} \rangle +\!\!+ \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\,(\textsc{Contr})$$

Mini Demo: Occurrence-Count Instance

Instance: use = 1, ign = 0, $\circledast = \times$, $\oplus = +$

$$\frac{\langle\rangle @ \langle\rangle \vdash \lambda x{:}\tau.\, x + x + x : \tau \xrightarrow{3} \tau \qquad \langle y{:}\tau\rangle @ \langle 2\rangle \vdash y + y : \tau}{\langle y{:}\tau\rangle @ \langle \underbrace{3 \circledast 2}_{3\times 2=6}\rangle \vdash (\lambda x{:}\tau.\, x + x + x)\,(y + y) : \tau}\ (\textsc{App})$$

Mini Demo: Occurrence-Count Instance

Instance: use = 1, ign = 0, $\circledast = \times$, $\oplus = +$

$$\frac{\langle\rangle@\langle\rangle \vdash \lambda x{:}\tau.\, x + x + x : \tau \xrightarrow{3} \tau \qquad \langle y{:}\tau\rangle@\langle 2\rangle \vdash y + y : \tau}{\langle y{:}\tau\rangle@\langle \underbrace{3 \circledast 2}_{3\times 2=6}\rangle \vdash (\lambda x{:}\tau.\, x + x + x)\,(y + y) : \tau}\ (\textsc{App})$$

$$\frac{\langle y{:}\tau, z{:}\tau\rangle@\langle 3, 2\rangle \vdash (y + y + y) + (z + z) : \tau}{\langle x{:}\tau\rangle@\langle \underbrace{3 \oplus 2}_{3+2=5}\rangle \vdash \underbrace{\big((y + y + y) + (z + z)\big)}_{(x+x+x)+(x+x)}[y \leftarrow x, z \leftarrow x] : \tau}\ (\textsc{Contr})$$

Timetable Matrices: A Concrete Example (1)

Definition 2 (Timetable Matrix)

The *timetable matrix* is defined by:

Basic domain : $\hat{\mathbb{N}} := \mathbb{N}_0 \cup \{-\infty\}$

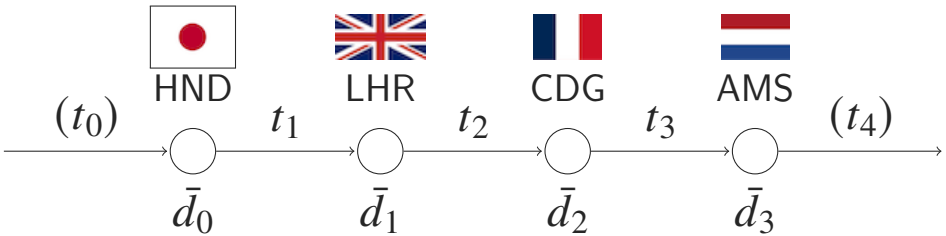
Entry operations : $a \sqcup b := \max(a, b)$, $a \sqcap b := a + b$

Matrix constants : $\mathsf{ign} = \begin{bmatrix} -\infty & -\infty \\ -\infty & 0 \end{bmatrix}$, $\mathsf{use} = \begin{bmatrix} 0 & -\infty \\ -\infty & 0 \end{bmatrix}$

Matrix operations : $(A \oplus B)_{ij} := A_{ij} \sqcup B_{ij}$, $(A \circledast B)_{ij} := \bigsqcup_k (A_{ik} \sqcap B_{kj})$

Ordinary matrix operations, but with + replaced by $\sqcup$ and $\times$ replaced by $\sqcap$.

Timetable Matrices: A Concrete Example (2)



Each leg i is represented by the following matrix:

$$R_i = \begin{bmatrix} t_i & \bar{d}_i \\ -\infty & 0 \end{bmatrix}$$

Timetable Matrices: A Concrete Example (3)

Example: composing two consecutive leg matrices

$$R_1 \circledast R_0 = \begin{bmatrix} t_1 & \bar{d}_1 \\ -\infty & 0 \end{bmatrix} \circledast \begin{bmatrix} t_0 & \bar{d}_0 \\ -\infty & 0 \end{bmatrix} = \begin{bmatrix} t_1 + t_0 & \boxed{\max(t_1 + \bar{d}_0, \bar{d}_1)} \\ -\infty & 0 \end{bmatrix}$$

$$\begin{aligned} d_1 &= \max(t_1 + d_0, \bar{d}_1) \\ &= \boxed{\max(t_1 + \bar{d}_0, \bar{d}_1)} = (R_1 \circledast R_0)_{12} \quad \text{(base case: } d_0 = \bar{d}_0\text{)} \end{aligned}$$

The $(1,2)$-Entry Gives d_i

$$(R_i \circledast \cdots \circledast R_0)_{12} = d_i$$

The Timetable Language

To represent flight itinerary problems, the MCA-parametric structural coeffect calculus needs the following additional syntax and type rule:

$$\text{Syntax:} \quad e ::= \cdots \mid \mathsf{leg}\ R\ e$$

$$\text{Typing Rule:} \quad \frac{\Gamma \ @\ \mathbf{s} \vdash e : \tau}{\Gamma \ @\ R \circledast \mathbf{s} \vdash \mathsf{leg}\ R\ e : \tau}\ (\textsc{Leg})$$

leg R e represents the itinerary obtained by adding the leg represented by timetable matrix R to itinerary e.

1. Problem: Why are scalar/componentwise coeffects not enough?
2. Solution: How do matrices represent interdependent requirements?
3. Justification: Why is the system sound?

TLMC: Target Language for Matrix Coeffects

Source Language (MCA-parametric structural coeffect calculus)

free variables with matrix coeffect annotations

⇓ **Translation**

TLMC (Target Language for Matrix Coeffects)

values of **indexed comonadic** types

Type Safety

Theorem 1 (Well-typedness of the translation)

If a closed source-language judgment is translated into a TLMC term f, then f is well-typed in TLMC.

Theorem 4 (Type Safety of TLMC)

If $\Gamma \vdash e : \tau$ and $e \rightsquigarrow^* e'$, then:

1. $\Gamma \vdash e' : \tau$
2. *either e' is a value, or there exists e'' such that $e' \rightsquigarrow e''$*

By the standard preservation and progress theorems for TLMC

Limitations and Future Work

- Higher-dimensional generalizations of MCA ($(n+1) \times (n+1)$ matrices)
- Other application domains
- Practical implementations
  - type inference
  - type checking
  - systematic methods for designing domain-specific MCAs
  - integration as an extension of an existing coeffect-aware language (e.g., Granule)

Conclusion

- **Interdependent** requirements require not only "pointwise" combination, but also "**composition**".
- **Matrices** can naturally represent requirement composition.
- Structural coeffect calculus with **MCA** can represent requirement composition naturally.

Can We Use Vectors Instead?

By defining a special **non-componentwise operation**, timetable matrices can also be encoded as vectors:

$$r_1 \mathbin{\hat{\circledast}} r_0 = \begin{bmatrix} t_1 \\ \bar{d}_1 \end{bmatrix} \mathbin{\hat{\circledast}} \begin{bmatrix} t_0 \\ \bar{d}_0 \end{bmatrix} = \begin{bmatrix} t_1 + t_0 \\ \max(t_1 + \bar{d}_0,\ \bar{d}_1) \end{bmatrix}$$

However, it is no longer an ordinary vector operation; it is essentially a matrix-style transformation encoded ad hoc.

Why the $(1,2)$-Entry Gives d_i (1)

(p.19)
Let

$$M_i = R_i \circledast \cdots \circledast R_0 \quad (i \geq 1)$$

We prove by induction that

$$(M_i)_{12} = d_i$$

Base case:

$$(R_0)_{12} = \bar{d}_0 = d_0.$$

Inductive step: Assume $(M_i)_{12} = d_i$ and $(M_i)_{22} = 0$

Why the $(1,2)$-Entry Gives d_i (2)

Then,

$$\begin{aligned}(M_{i+1})_{12} &= (R_{i+1} \circledast M_i)_{12} \\ &= \max\big((R_{i+1})_{11} + (M_i)_{12}, (R_{i+1})_{12} + (M_i)_{22}\big) \\ &= \max(t_{i+1} + d_i, \bar{d}_{i+1}) \\ &= d_{i+1}\end{aligned}$$

Key point: $\bar{d}_i$ is placed in the $(1,2)$-entry so that matrix composition reproduces the timetable recurrence

Why Are use and ign Defined This Way?

Let $A = \begin{bmatrix} t & \bar{d} \\ -\infty & 0 \end{bmatrix}$:

$$A \oplus \text{ign} = \begin{bmatrix} t & \bar{d} \\ -\infty & 0 \end{bmatrix} \oplus \begin{bmatrix} -\infty & -\infty \\ -\infty & 0 \end{bmatrix} = \begin{bmatrix} \max(t, -\infty) & \max(\bar{d}, -\infty) \\ \max(-\infty, -\infty) & \max(0, 0) \end{bmatrix} = \begin{bmatrix} t & \bar{d} \\ -\infty & 0 \end{bmatrix} = A$$

$$A \circledast \text{use} = \begin{bmatrix} t & \bar{d} \\ -\infty & 0 \end{bmatrix} \circledast \begin{bmatrix} 0 & -\infty \\ -\infty & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \max(t + 0, \bar{d} + (-\infty)) & \max(t + (-\infty), \bar{d} + 0) \\ \max(-\infty + 0, 0 + (-\infty) & \max(-\infty + (-\infty), 0 + 0) \end{bmatrix} = \begin{bmatrix} t & \bar{d} \\ -\infty & 0 \end{bmatrix} = A$$

Similarly, $\text{ign} \oplus A = A$, $\text{use} \circledast A = A$

Full Typing/Structural Rules

Typing Rules:

$$\frac{}{x : \tau \ @\ \langle \mathsf{use} \rangle \vdash x : \tau}\ (\textsc{Var}) \qquad \frac{}{\langle\rangle \ @\ \langle\rangle \vdash n : \mathsf{num}}\ (\textsc{Const})$$

$$\frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \xrightarrow{\mathsf{s}} \tau_2 \quad \Gamma_2 \ @\ \mathbf{q} \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ \mathbf{p} \nplus (\mathsf{s} \circledast \mathbf{q}) \vdash e_1\ e_2 : \tau_2}\ (\textsc{App}) \qquad \frac{\Gamma, x : \tau_1 \ @\ \mathbf{r} \nplus \langle \mathsf{s} \rangle \vdash e : \tau_2}{\Gamma \ @\ \mathbf{r} \vdash \lambda x{:}\tau_1.e : \tau_1 \xrightarrow{\mathsf{s}} \tau_2}\ (\textsc{Abs})$$

$$\frac{\Gamma_1 \ @\ \mathbf{p} \vdash e_1 : \tau_1 \quad \Gamma_2, x : \tau \ @\ \mathbf{q} \nplus \langle \mathsf{s} \rangle \vdash e_2 : \tau_2}{\Gamma_1, \Gamma_2 \ @\ (\mathsf{s} \circledast \mathbf{p}) \nplus \mathbf{q} \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : \tau_2}\ (\textsc{Let})$$

Structural Rules:

$$\frac{\Gamma \ @\ \mathbf{r} \vdash e : \tau}{\Gamma, x : \tau \ @\ \mathbf{r} \nplus \langle \mathsf{ign} \rangle \vdash e : \tau}\ (\textsc{Weak}) \quad (x \notin \Gamma) \qquad \frac{\Gamma \ @\ \mathbf{p} \nplus \langle \mathsf{s}' \rangle \vdash e : \tau}{\Gamma \ @\ \mathbf{p} \nplus \langle \mathsf{s} \rangle \vdash e : \tau}\ (\textsc{Sub}) \quad (\mathsf{s}' \leqslant \mathsf{s})$$

$$\frac{\Gamma_1, y : \tau_1, z : \tau_1, \Gamma_2 \ @\ \mathbf{p} \nplus \langle \mathsf{s}, \mathsf{t} \rangle \nplus \mathbf{q} \vdash e : \tau}{\Gamma_1, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} \nplus \langle \mathsf{s} \oplus \mathsf{t} \rangle \nplus \mathbf{q} \vdash e[z, y \leftarrow x] : \tau}\ (\textsc{Contr})$$

$$\frac{\Gamma_1, x : \tau_1, y : \tau_2, \Gamma_2 \ @\ \mathbf{p} \nplus \langle \mathsf{s}, \mathsf{t} \rangle \nplus \mathbf{q} \vdash e : \tau}{\Gamma_1, y : \tau_2, x : \tau_1, \Gamma_2 \ @\ \mathbf{q} \nplus \langle \mathsf{t}, \mathsf{s} \rangle \nplus \mathbf{q} \vdash e : \tau}\ (\textsc{Exch}) \qquad (\text{This is standard})$$

Type Safety (Extended Version) (1)

Theorem 1 (Well-typedness of the translation)

If the closed source-language judgment $\langle\rangle \ @ \ \langle\rangle \vdash e : \tau$ *is translated into a TLMC term* f*, then* f *is well-typed:* $\vdash f : [\![\langle\rangle \ @ \ \langle\rangle]\!] \to [\![\tau]\!]$

By induction on the derivation of the translation.

Theorem 2 (Type Preservation of TLMC)

If $\Gamma \vdash e : \tau$ *and* $e \rightsquigarrow e'$ *then* $\Gamma \vdash e' : \tau$

By induction over the reduction $e \rightsquigarrow e'$ of TLMC.

Type Safety (Extended Version) (2)

Theorem 3 (Progress of TLMC)

If $\Gamma \vdash e : \tau$, then either e is a value, or there exists e' such that $e \rightsquigarrow e'$

By induction on the typing derivation of $\Gamma \vdash e : \tau$.

Theorem 4 (Type Safety of TLMC)

If $\Gamma \vdash e : \tau$ and $e \rightsquigarrow^* e'$, then:

1. $\Gamma \vdash e' : \tau$
2. *either e' is a value, or there exists e'' such that $e' \rightsquigarrow e''$*

1. By induction over the reduction $e \rightsquigarrow e'$ of TLMC.
2. By induction on the typing derivation of $\Gamma \vdash e : \tau$.

TLMC: Target Language for Matrix Coeffects (1)

- TLMC is an extended typed lambda calculus
- Coeffect-annotated values are represented as values of **indexed comonadic** types
  - Each source variable is translated into a value paired with its matrix annotation

$$\langle x_1 : \tau_1, \ldots, x_n : \tau_n \rangle \ @ \ \langle R_1, \ldots, R_n \rangle$$
$$\downarrow$$
$$\mathrm{Mc}\langle (R_1, v_1), \ldots, (R_n, v_n) \rangle : C^{\langle R_1, \ldots, R_n \rangle}(\llbracket \tau_1 \rrbracket \times \cdots \times \llbracket \tau_n \rrbracket)$$

Mc is an indexed comonadic constructor

TLMC: Target Language for Matrix Coeffects (2)

TLMC consists of the following:

Functional Constructs : ordinary constructs of functional languages (e.g., integers, tuples, lambda abstraction and application, ...)

Comonadic Operations : operations for **indexed comonads** (e.g., counit, cobind, merge, ...)

Domain-Specific Operations : operations corresponding to domain-specific operations in the **source langage**, i.e., the MCA-parametric structural coeffect calculus (e.g., leg)

TLMC is introduced to give a translational semantics and prove type safety via translation rules

Translation: Source Language → TLMC (1)

The translation rules map source-language terms to TLMC terms

Example: Translation Rule for Abstraction

The following rule shows that if $[\![\Gamma, x : \tau_1 \ @ \ \mathbf{r} +\!\!+ \langle s \rangle \vdash e : \tau_2]\!] = f$, then the judgment $\Gamma \ @ \ \mathbf{r} \vdash \lambda x : \tau_1 . e : \tau_1 \xrightarrow{s} \tau_2$ is translated into the following TLMC term:

$$\frac{[\![\Gamma, x : \tau_1 \ @ \ \mathbf{r} +\!\!+ \langle s \rangle \vdash e : \tau_2]\!] \ = \ f}{[\![\Gamma \ @ \ \mathbf{r} \vdash \lambda x : \tau_1 . e : \tau_1 \xrightarrow{s} \tau_2]\!] \ = \ \lambda \rho . \lambda \nu . f(\mathsf{merge}_{\mathbf{r}, \langle s \rangle}(\rho, \nu))}$$

Translation: Source Language $\longrightarrow$ TLMC (2)

Example: Translation Rule for Leg

The following rule shows that if $[\![\Gamma \mathbin{@} \mathbf{s} \vdash e : \tau]\!] = f$, then the judgment $\Gamma \mathbin{@} R \circledast \mathbf{s} \vdash \mathsf{leg}\ R\ e : \tau$ is translated into the following TLMC term:

$$\frac{[\![\Gamma \mathbin{@} \mathbf{s} \vdash e : \tau]\!] = f}{[\![\Gamma \mathbin{@} R \circledast \mathbf{s} \vdash \mathsf{leg}\ R\ e : \tau]\!] = \lambda\rho.f(\mathsf{peel}_{R,\mathbf{s}}\ \rho)}$$

Contributions

- Matrix Coeffect Algebra
- Instantiation of structural coeffect calculus with MCA
- Extended target language, translational semantics, and type safety
- Timetable case study

Motivating Example: Flight Timetables (2)

A traveler cannot depart from airport i before

- the scheduled departure time $\bar{d}_i$
- the arrival time from the previous leg $d_{i-1} + t_i$

So the earliest feasible departure time is

$$d_i = \max(\bar{d}_i,\ d_{i-1} + t_i)$$

Equivalently:

- If $\bar{d}_i \geq d_{i-1} + t_i$, then $d_i = \bar{d}_i$

  (the traveler arrives early enough; the scheduled time determines the departure)
- If $\bar{d}_i < d_{i-1} + t_i$, then $d_i = d_{i-1} + t_i$

  (the traveler arrives late; the arrival time from the previous leg determines the departure)