A theory for defining and handling matrices in Event-B

Guillaume Dupont

IRIT, Toulouse INP – ENSEEIHT

Rodin Workshop – May 2026

Context

In general:

- matrices are a very, very common tool in mathematics

  $\Rightarrow$ linear algebra, control theory, no really they are *everywhere*
- they have a rich ecosystem of results, properties, theorems...
- they have not been formalised in Event-B!

Specifically for me:

- hybrid systems $\Rightarrow$ defining ODE using matrices is standard
- working on quantum computing $\Rightarrow$ quantum gates are complex Hermitian matrices
- the challenge is interesting

Let's define matrices in an Event-B theory! [Abr+09; BM13]

Basic formalisation

- A matrix is (abstractly) a rectangular array of *objects* (usually in a field but not mandatory)
- It is characterised by a *height* (number of lines, m) and a *width* (number of columns, n)
- It is given by specifying the elements for each line $i \in 1..m$ and column $j \in 1..n$

$$M = (M_{ij})_{i \in 1..m, j \in 1..n} = \begin{pmatrix} M_{11} & M_{12} & \dots & M_{1n} \\ M_{21} & M_{22} & \dots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \dots & M_{mn} \end{pmatrix}$$

- The set of m by n matrices is denoted $\mathcal{M}(m, n)$

Basic formalisation

```
THEORY Matrix
DATATYPES
  Matrix
    constructors matrixOf(width : ℤ, height : ℤ,
      at : ℙ((ℤ × ℤ) × ℂ))
OPERATORS
  MatrixWD predicate (m: Matrix)
    direct definition width(m) > 0 ∧ height(m) > 0∧
     at(m) ∈ 1..width(m) × 1..height(m) → ℂ
  matrix predicate (m: ℕ, n: ℕ, f: 1..m × 1..n → ℂ)
    well-definedness m > 0, n > 0
    direct definition matrixOf(m, n, f)
```

We define a matrix $M \in \mathcal{M}(m,n)$ by giving the mapping:

$$\begin{array}{rccc} \hat{M} & : & 1..m \times 1..n & \longrightarrow & K \\ & & (i,j) & \longmapsto & M_{ij} \end{array}$$

- As often, we have to define a broad datatype (on maximal types) + a WD predicate used in conditions throughout the theory
- **matrix** is a "specialized" constructor that enforces WD

$$M = \mathbf{matrix}(m,n,f) \Rightarrow \mathbf{MatrixWD}(M)$$

- at is supposed to be the mapping function; it may be used as

$$at(M)(i \mapsto j) \equiv M_{ij}$$

Particular matrices

```
M0 expression (m: ℕ, n: ℕ)
 well-definedness condition m > 0, n > 0
 direct definition
  matrix(m, n, λi ↦ j · i ∈ 1..m ∧ j ∈ 1..n | C0)
Mdiag expression (n: ℕ, d: 1..n → ℂ)
 well-definedness condition n > 0
 direct definition matrix(n, n,
  (λi ↦ j · i ∈ 1..n ∧ j ∈ 1..n ∧ i = j | d(i))∪
  (λi ↦ j · i ∈ 1..n ∧ j ∈ 1..n ∧ i ≠ j | C0)
MId expression (n: ℕ)
 well-definedness condition n > 0
 direct definition Mdiag(n, λk · k ∈ 1..n | C1)
```

(note that all these matrices are well-defined for positive dimensions)

- **M0**(m, n) is the m by n matrix that contains only 0s[1]
- **Mdiag**(n, d_i) builds the *diagonal* (square) matrix of size n with d_i $(= d(i))$ on its diagonal and 0 everywhere else; that is

$$D_{ij} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

- **MId**(n) is the diagonal matrix of size n with 1s on its diagonal[2], called the *identity matrix*

[1] Provided K defines such an element

[2] Same remark

Operators

The two main operators on matrices are addition and multiplication[3]:

$$\forall A, B \in \mathcal{M}(m,n), A + B = (A_{ij} + B_{ij})_{i \in 1..m, j \in 1..n} \quad (\in \mathcal{M}(m,n))$$

$$\forall A \in \mathcal{M}(p,m), B \in \mathcal{M}(m,n), A \cdot B = \left(\sum_{k \in 1..m} A_{ik} B_{kj} \right)_{i \in 1..p, j \in 1..m} \quad (\in \mathcal{M}(p,n))$$

(notice that matrix product necessitates a *generalized sum*)

You also have multiplication by a scalar, transposition, inner and outer products for vectors, etc. but once you have the principle they are relatively easy to define

[3] Provided K is itself associated with addition and multiplication

Operators – Implementation

Mplus *expression infix associative commutative* ($m1$: **Matrix**, $m2$: **Matrix**)
 well-definedness condition $\mathbf{MatrixWD}(m1) \land \mathbf{MatrixWD}(m2) \land$
  $height(m1) = height(m2) \land width(m1) = width(m2)$
 direct definition $\mathbf{matrix}(height(m1), width(m), \lambda i \mapsto j \cdot i \in 1..height(m1) \land j \in 1..width(m1) \mid$
  $at(m1)(i \mapsto j) + at(m2)(i \mapsto j))$
Mtimes *expression infix* ($m1$: **Matrix**, $m2$: **Matrix**)
 well-definedness condition $\mathbf{MatrixWD}(m1) \land \mathbf{MatrixWD}(m2) \land width(m1) = height(m2)$
 direct definition $\mathbf{matrix}(height(m1), width(m2), \lambda i \mapsto j \cdot i \in 1..height(m1) \land j \in 1..width(m2) \mid$
  $\mathbf{cSum}(\lambda k \cdot k \in 1..width(m) \mid at(m1)(i \mapsto k) \times at(m2)(k \mapsto j)))$

(for those interested in the sausage making:)

fold *expression* (b: B, f: $E \times B \to B$)
 direct definition $\mathbf{FrSB1}(\{g \mapsto x \mid x \geq 0 \land g \in \mathbb{N} \nrightarrow E \land 1..x \subseteq \mathrm{dom}(g)\},$
  $\{g, x \cdot x \geq 0 \land g \in \mathbb{N} \nrightarrow E \land 1..x + 1 \subseteq \mathrm{dom}(g) \mid (g \mapsto x) \mapsto (g \mapsto x + 1)\},$
  $\lambda g \mapsto x \cdot x \geq 0 \land g \in \mathbb{N} \nrightarrow E \land 1..x \subseteq \mathrm{dom}(g) \mid z,$
  $\lambda(g \mapsto x) \mapsto v \cdot x > 0 \land g \in \mathbb{N} \nrightarrow E \land v \in B \land 1..x \subseteq \mathrm{dom}(g) \mid f(g(x) \mapsto v))$
  -- $\mathbf{fold}(b, f) \in (\mathbb{Z} \nrightarrow (\mathbb{Z} \times E)) \times \mathbb{Z}$
cSum *expression* (b: $\mathbb{N} \nrightarrow \mathbb{C}$) -- $\sum_{k=1}^{\mathrm{card}(b)} b(k)$
 well-definedness condition $\mathrm{finite}(b) \land b \in 1..\mathrm{card}(b) \to \mathbb{C}$
 direct definition $\mathbf{fold}(C0, \lambda(n \mapsto e) \mapsto acc \cdot n \in \mathbb{Z} \land e \in \mathbb{C} \land acc \in \mathbb{C} \mid acc + e)$
  $((\lambda n \cdot n \in 1..\mathrm{card} \mid n \mapsto b(n)) \mapsto \mathrm{card}(b))$

(with **FrSB1** defined in [Can24][4])

[4] http://dominique.cansell.free.fr/JRAInstantiation.html

A few theorems

Most theorems relate to WD and dimensions

Mplus_WD: $\forall m1, m2 \cdot \mathbf{MatrixWD}(m1) \wedge \mathbf{MatrixWD}(m2) \wedge height(m1) = height(m2) \wedge width(m1) = width(m2) \Rightarrow \mathbf{MatrixWD}(m1\ \mathbf{Mplus}\ m2)$

...

Mplus_width: $\forall m1, m2 \cdot \mathbf{MatrixWD}(m1) \wedge \mathbf{MatrixWD}(m2) \wedge height(m1) = height(m2) \wedge width(m1) = width(m2) \Rightarrow width(m1\ \mathbf{Mplus}\ m2) = width(m1)$

...

The others are about basic algebraic properties:

Mplus_left_neutral: $\forall m \cdot \mathbf{MatrixWD}(m) \Rightarrow m\ \mathbf{Mplus}\ M0(width(m), height(m)) = m$

...

Mtimes_left_neutral: $\forall m \cdot \mathbf{MatrixWD}(m) \wedge width(m) = height(m) \Rightarrow m\ \mathbf{Mtimes}\ \mathbf{MId}(width(m)) = m$

...

And of course we define a bunch of proof rules to handle easy cases

Metavariables $m1, m2 \in \mathbf{Matrix}$

Rewrite Rules

Mplus_wd_rew: $\mathbf{MatrixWD}(m1\ \mathbf{Mplus}\ m2)$

rhs1: $\top \Rightarrow \mathbf{MatrixWD}(m1) \wedge \mathbf{MatrixWD}(m2) \wedge width(m1) = width(m2) \wedge height(m1) = height(m2)$

...

Mplus_width_rew: $width(m1\ \mathbf{Mplus}\ m2)$

rhs1: $\top \Rightarrow width(m1)$

...

Using the theory

```
CONTEXT Example
CONSTANTS
 PauliX
AXIOMS
 axm1: PauliX = matrix(2, 2, {
     1 ↦ 1 ↦ C0, 1 ↦ 2 ↦ C1,
     2 ↦ 1 ↦ C1, 2 ↦ 2 ↦ C0
 }
 thm1: PauliX Mtimes PauliX = MId(2) theorem
END
```

- Pauli's X matrix is defined as

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- σ_x is involutive (idempotent)

$$\sigma_x^2 = \sigma_x \times \sigma_x = I_2$$

Associated WD:

- **matrix**: function is a total function $1..2 \times 1..2 \to \mathbb{C}$
- **Mtimes**: sizes are compatible (width of LHS = height of RHS)

Theory content

Formalised:

- sum, product, scalar product, tensor product
- *bra-ket*, inner and outer products
- transpose, conjugate, adjoint, trace
- eigenvalues, eigenvectors (not constructive)
- Hilbertian, unitary, positive (semi-)definite, Hermitian
- a few things on bases

NOT formalised (yet):

- determinant (hard)
- inverse (very hard)

Assessment and limitations

- Once you get the gist of it, formalisation it not that hard
- Except in a few places, proof is relatively easy
  BUT: a lot of typing, WD and dimension-related sub-goals
  (partially alleviated by PR)
- The elephant in the room: the pieces of code provided are about matrices **of complex numbers**

  $\Rightarrow$ matrices over an other K = you have to redo everything

  $\Rightarrow$ writing a "generic" (parametric) matrix theory is doable but: 1) it is cumbersome (to define and to use) and 2) it means operations **cannot be operators** (they have to be functions...)

I wish there was some kind of *functor*-like construction in the theory plug-in (à la OCaml): give K, $Kplus$, $Ktimes$, $K0$, $K1$ $\Rightarrow$ "generate" $KMatrix$ (and properties)

Theory is almost ready for release (missing a few proofs and improvements on names and such)

Part I

Bibliography

References I

[Abr+09] Jean-Raymond Abrial et al. *Proposals for Mathematical Extensions for Event-B*. Tech. rep. 2009.

[BM13] Michael Butler and Issam Maamria. "Practical Theory Extension in Event-B". In: *Theories of Programming and Formal Methods*. Ed. by Zhiming Liu, Jim Woodcock, and Huibiao Zhu. Vol. 8051. LNCS. Springer, 2013, pp. 67–81.

[Can24] Dominique Cansell. "Schemata of Recursive Functions and Iterative Algorithms". In: *The 11th Rodin User and Developer Workshop, 25th June, 2024, Bergamo, Italy*. 2024. URL: https://wiki.event-b.org/index.php/Rodin_Workshop_2024.