Linear Types for Linear Algebra, and Beyond

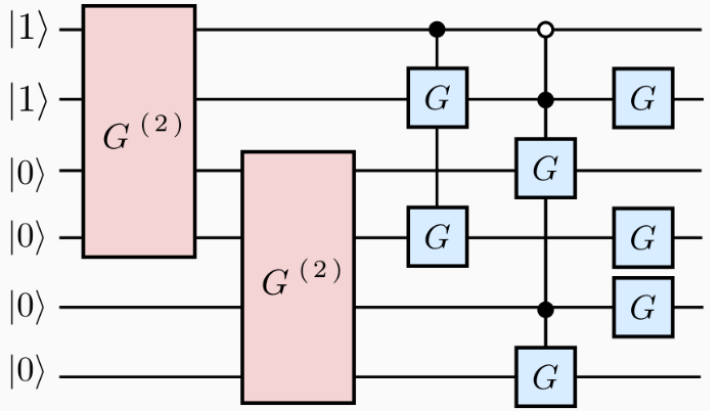
Jennifer Paykin

University of Vermont
jennifer.paykin@uvm.edu

November 14, 2026
ECLAPS, Montreal, Canada

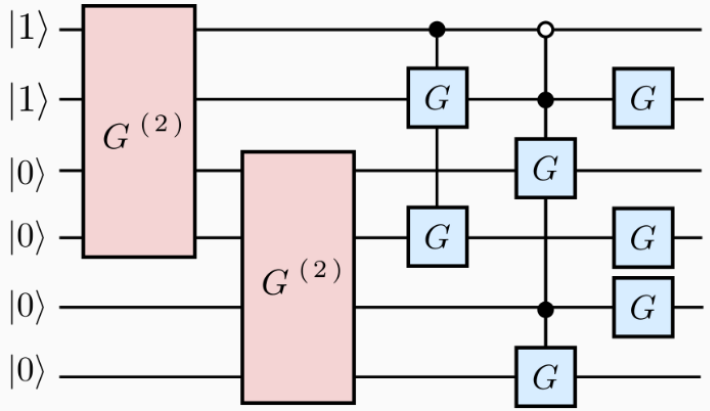
Programming with linear algebra

Programming with matrices



A quantum circuit. Each wire represents a qubit, each box represents a matrix. [2].

Programming with matrices



A quantum circuit. Each wire represents a qubit, each box represents a matrix. [2].

```
% Arrays: time and measured signals
time = (0:0.01:0.04)'; % seconds
voltage = [3.3 3.4 3.6 3.5 3.7]';
current = [0.10 0.11 0.13 0.12 0.14]';

% Create a table of measurements
measurements = table(time,voltage,current)

% Array indexing
voltage(3) % third voltage sample
voltage(voltage > 3.5) % voltages above threshold

% Table indexing
measurements(2:4, :) % time window
measurements.voltage(3) % named variable access
measurements{3, "current"} % numeric value from table
```

Matlab `mathworks.com/products/matlab/getting-started.html`

```
f :: Real(784) -> Real(784)
f = w2 . relu . w1
    where w1 x = _
          w2 x = _
```

Programming linear neurons in neural networks. [7]

Programming with linear transformations

Matrices vs functions

$$\begin{pmatrix} 0 & 0 & 3 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

$$\begin{aligned} mat : \mathbb{R}^3 &\multimap \mathbb{R}^3 \\ &= \lambda x.\mathsf{case}\ x\ \mathsf{of} \\ &\quad \mid [1,0,0] \to [0,1,0] \\ &\quad \mid [0,1,0] \to [0,0,2] \\ &\quad \mid [0,0,1] \to [3,0,0] \end{aligned}$$

The Curry-Howard correspondence

Types are **(finite-dimensional) vector spaces**

Terms are **linear transformations**

- Pablo Arrighi and Gilles Dowek. Lineal: A linear-algebraic λ-calculus. [3]
- Alejandro Díaz-Caro and Gilles Dowek. A linear linear lambda calculus. [4]
- Jennifer Paykin and Sam Winnick. Qudit Quantum Programming with Projective Cliffords. [6]

Linear types for linear algebra

Vector spaces

Definition (*)

A vector space over a field F is a non-empty set V together with:

- An abelian group $(V, +, \mathbf{0})$;
- A scalar multiplication operation $\cdot : F \times V \to V$ that distributes over $+$ in particular ways

*All of this applies to modules over a ring, not just vector spaces over a field.

Vector spaces

Definition (*)

A vector space over a field F is a non-empty set V together with:

- An abelian group $(V, +, \mathbf{0})$;
- A scalar multiplication operation $\cdot : F \times V \to V$ that distributes over $+$ in particular ways

Definition

A linear transformation $f : V_1 \to V_2$ is a function that respects addition and scalar multiplication:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f(v_1) + r_2 \cdot f(v_2)$$

*All of this applies to modules over a ring, not just vector spaces over a field.

Types are finite-dimensional vector spaces

Type (τ)	Vectors

Types are finite-dimensional vector spaces

Type (τ)	Vectors
F	F

Types are finite-dimensional vector spaces

Type (τ)	Vectors
F	F
$\tau_1 \oplus \tau_2$	$\{[v_1, v_2] \mid v_1 \in \tau_1, v_2 \in \tau_2\}$

Types are finite-dimensional vector spaces

Type (τ)	Vectors
F	F
$\tau_1 \oplus \tau_2$	$\{[v_1, v_2] \mid v_1 \in \tau_1, v_2 \in \tau_2\}$
$\tau_1 \multimap \tau_2$	$\{f : \tau_1 \to \tau_2 \mid f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f(v_1) + r_2 \cdot f(v_2)\}$ (can think of as matrices)

Types are finite-dimensional vector spaces

Type (τ)	Vectors
F	F
$\tau_1 \oplus \tau_2$	$\{[v_1, v_2] \mid v_1 \in \tau_1, v_2 \in \tau_2\}$
$\tau_1 \multimap \tau_2$	$\{f : \tau_1 \to \tau_2 \mid f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f(v_1) + r_2 \cdot f(v_2)\}$ (can think of as matrices)

*Can also handle tensor product of vector spaces $\tau_1 \otimes \tau_2$.

Terms are linear transformations

Closed terms are vectors...

$$e ::= r \in F \mid \mathbf{0}_\tau \mid [e_1, e_2] \mid e \cdot e \mid e_1 + e_2 \mid \cdots$$

Terms are linear transformations

Closed terms are vectors...

$$e ::= r \in F \mid \mathbf{0}_\tau \mid [e_1, e_2] \mid e \cdot e \mid e_1 + e_2 \mid \cdots$$

...open terms are linear transformations...

$$\cdots \mid x \mid \lambda x.e \mid e\ e \mid \cdots$$

Terms are linear transformations

Closed terms are vectors...

$$e ::= r \in F \mid \mathbf{0}_\tau \mid [e_1, e_2] \mid e \cdot e \mid e_1 + e_2 \mid \cdots$$

...open terms are linear transformations...

$$\cdots \mid x \mid \lambda x.e \mid e\ e \mid \cdots$$

...and we can destruct vectors into their basis elements...

$$\cdots \mid \text{case } e \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\}$$

Terms are linear transformations

$$
\begin{aligned}
e ::= {} & r \in F \mid \mathbf{0}_\tau \mid [e_1, e_2] \mid e \cdot e \mid e_1 + e_2 \\
& \mid x \mid \lambda x.e \mid e\ e \\
& \mid \text{case } e \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\}
\end{aligned}
$$

Values are scalars, direct sums, and functions:

$$v ::= r \mid [v_1, v_2] \mid \lambda x.e$$

Terms are linear transformations

$$
\begin{aligned}
e ::= {} & r \in F \mid \mathbf{0}_\tau \mid [e_1, e_2] \mid e \cdot e \mid e_1 + e_2 \\
& \mid x \mid \lambda x.e \mid e\ e \\
& \mid \text{case } e \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\}
\end{aligned}
$$

Addition and scalar multiplication distribute over constructors:

$$
\begin{aligned}
[v_1, v_2] + [v_1', v_2'] &\to [v_1 + v_1', v_2 + v_2'] \\
(\lambda x.e_1) + (\lambda x.e_2) &\to \lambda x.e_1 + e_2 \\
r \cdot [v_1, v_2] &\to [r \cdot v_1, r \cdot v_2] \\
r \cdot (\lambda x.e) &\to \lambda x.r \cdot e
\end{aligned}
\qquad
\begin{aligned}
\mathbf{0}_F &\to 0 \\
\mathbf{0}_{\tau_1 \oplus \tau_2} &\to [\mathbf{0}_{\tau_1}, \mathbf{0}_{\tau_2}] \\
\mathbf{0}_{\tau_1 \multimap \tau_2} &\to \lambda x.\mathbf{0}_{\tau_2}
\end{aligned}
$$

Terms are linear transformations

$$
\begin{aligned}
e ::= {} & r \in F \mid \mathbf{0}_\tau \mid [e_1, e_2] \mid e \cdot e \mid e_1 + e_2 \\
& \mid x \mid \lambda x.e \mid e\ e \\
& \mid \text{case } e \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\}
\end{aligned}
$$

Case analysis breaks up a direct sum into its basis elements:

$$
\begin{aligned}
& \text{case } [v_1, v_2] \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\} \\
& \to \\
& e_1[v_1/x_1] + e_2[v_2/x_2]
\end{aligned}
$$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda(x : F).\ x \cdot x$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda (x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda(x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda(x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

Linear typing judgment $\Gamma \vdash e : \tau$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda (x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

All variables must be used.

$$\frac{}{x : \tau \vdash x : \tau}\ \text{VAR} \qquad \frac{r \in F}{\emptyset \vdash r : F}\ F\text{-I}$$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda (x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

Scalars cannot be used more than once e.g. in scalar multiplication.

$$\frac{\Gamma_1 \vdash e_1 : F \qquad \Gamma_2 \vdash e_2 : \tau \qquad \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2) = \emptyset}{\Gamma_1, \Gamma_2 \vdash e_1 \cdot e_2 : \tau}\ F\text{-E}$$

In linear logic terms: scalars are multiplicative.

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda(x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

Vectors are additive.

$$\frac{}{\Gamma \vdash \mathbf{0} : \tau}\ 0 \qquad \frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 + e_2 : \tau}\ +$$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda(x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

Direct sum ($\oplus$) is a biproduct:

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash [e_1, e_2] : \tau_1 \oplus \tau_2}\ \oplus\text{-I}$$

Enforcing linearity for linear transformations

A function is a linear transformation iff:

$$f\,(r_1 \cdot v_1 + r_2 \cdot v_2) = r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

linear	non-linear
$\lambda x.x$	
$\lambda x.\ 2 \cdot x$	$\lambda (x : F).\ x \cdot x$
$\lambda x.\ \mathbf{0}$	$\lambda x.\ 1$
$\lambda x.x + x$	$\lambda x.\lambda y.x + y$

Direct sum ($\oplus$) is a biproduct:

$$\frac{\Gamma \vdash e : \tau_1 \oplus \tau_2 \qquad \Gamma', x_i : \tau_i \vdash e_i : \tau' \qquad \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \emptyset}{\Gamma, \Gamma' \vdash \mathrm{case}\ e\ \mathrm{of}\ \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\} : \tau'}\ \oplus\text{-E}$$

Linear types are linear algebra

Theorem ([6])

There exists a fully abstract denotational semantics in terms of freely-generated finite-dimensional vector spaces and linear transformations.

Linear types are linear algebra

Theorem ([6])

There exists a fully abstract denotational semantics in terms of freely-generated finite-dimensional vector spaces and linear transformations.

Corollary

If $\vdash f : \tau_1 \multimap \tau_2$, *then*

$$f(r_1 \cdot v_1 + r_2 \cdot v_2) \qquad \equiv \qquad r_1 \cdot f\ v_1 + r_2 \cdot f\ v_2$$

Beyond vector spaces

Example 1: Unitary matrices

Quantum states and unitary matrices

Quantum computing works over finite-dimensionaI Hilbert spaces—complex vector spaces with an inner product $\langle u \mid v \rangle = u^\dagger v \in \mathbb{C}$.

Definition

A quantum state is a vector in a Hilbert space with $\langle \varphi \mid \varphi \rangle = 1$.

Quantum states and unitary matrices

Quantum computing works over finite-dimensionaI Hilbert spaces—complex vector spaces with an inner product $\langle u \mid v \rangle = u^\dagger v \in \mathbb{C}$.

Definition

A quantum state is a vector in a Hilbert space with $\langle \varphi \mid \varphi \rangle = 1$.

Definition

Unitary operations U are linear transformations between Hilbert spaces that respect the inner product:

$$\langle U\varphi \mid U\psi \rangle = \langle \varphi \mid \psi \rangle$$

The Curry-Howard correspondence

Types are **(finite-dimensional) Hilbert spaces**

Terms are **unitary transformations**

- Thorsten Altenkirch and Jonathan Grattage. A functional quantum programming language. [1]
- Alejandro Diaz-Caro, Mauricio Guillermo and Alexandre Miquel and Benoit Valiron. Realizability in the unitary sphere. [5]

Linear transformations that preserve the inner product

$$\frac{\Gamma \vdash e : \tau_1 \oplus \tau_2 \qquad \Gamma', x_i : \tau_i \vdash e_i : \tau' \qquad \boxed{e_1 \perp e_2}}{\Gamma, \Gamma' \vdash \text{case } e \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\} : \tau'} \oplus\text{-E}$$

$e_1 \perp e_2$ means e_1 and e_2 are orthogonal: for every context C:

$$\langle C[e_1] \mid C[e_2] \rangle = \langle [x_1, \mathbf{0}] \mid [\mathbf{0}, x_2] \rangle = 0$$

Linear transformations that preserve the inner product

$$\frac{\Gamma \vdash e : \tau_1 \oplus \tau_2 \qquad \Gamma', x_i : \tau_i \vdash e_i : \tau' \qquad \boxed{e_1 \perp e_2}}{\Gamma, \Gamma' \vdash \text{case } e \text{ of } \{[x_1, \mathbf{0}] \to e_1 \mid [\mathbf{0}, x_2] \to e_2\} : \tau'} \oplus\text{-E}$$

$e_1 \perp e_2$ means e_1 and e_2 are orthogonal: for every context C:

$$\langle C[e_1] \mid C[e_2] \rangle = \langle [x_1, \mathbf{0}] \mid [\mathbf{0}, x_2] \rangle = 0$$

Open problem: complete decision procedure for orthogonality

Example 2: The Clifford group

The Clifford and Pauli groups

Important subgroups of unitary transformations (for quantum error correction and fault-tolerant quantum computing):

- Pauli group: $\mathcal{P} = \{i^r p \mid p \in \{I, X, Y, Z\}\}$

The Clifford and Pauli groups

Important subgroups of unitary transformations (for quantum error correction and fault-tolerant quantum computing):

- Pauli group: $\mathcal{P} = \{i^r p \mid p \in \{I, X, Y, Z\}\}$
- Clifford group: $\{U \mid \forall P \in \mathcal{P}, UPU^\dagger \in \mathcal{P}\}$
    - Center-fixing homomorphisms over the Pauli group $P \mapsto UPU^\dagger$

The Curry-Howard correspondence

Types are **generalized Pauli groups**

Terms are **center-fixing homomorphisms** *i.e.,* **projective Clifford operators**

- Jennifer Paykin and Sam Winnick. Qudit Quantum Programming with Projective Cliffords. [6]

Center-fixing homomorphisms

Center-fixing: P_1 commutes with P_2 iff $f(P_1)$ commutes with $f(P_2)$.

Center-fixing homomorphisms

Center-fixing: P_1 commutes with P_2 iff $f(P_1)$ commutes with $f(P_2)$.

Paulis can be represented as vectors $[x, z]$ over $\mathbb{Z}_2$ with phase:

$$i^r X^x Z^z$$

Center-fixing homomorphisms

Center-fixing: P_1 commutes with P_2 iff $f(P_1)$ commutes with $f(P_2)$.

Paulis can be represented as vectors $[x, z]$ over $\mathbb{Z}_2$ with phase:

$$i^r X^x Z^z$$

The symplectic form $\omega([x_1, z_1], [x_2, z_2]) = z_1 \cdot x_2 - x_1 \cdot z_2$ tells us if two Paulis commute or anticommute:

$$P_1 P_2 = (-1)^{\omega([x_1,z_1],[x_2,z_2])} P_2 P_1$$

Center-fixing homomorphisms

Center-fixing: P_1 commutes with P_2 iff $f(P_1)$ commutes with $f(P_2)$.

Paulis can be represented as vectors $[x, z]$ over $\mathbb{Z}_2$ with phase:

$$i^r X^x Z^z$$

The symplectic form $\omega([x_1, z_1], [x_2, z_2]) = z_1 \cdot x_2 - x_1 \cdot z_2$ tells us if two Paulis commute or anticommute:

$$P_1 P_2 = (-1)^{\omega([x_1,z_1],[x_2,z_2])} P_2 P_1$$

$$\frac{\Gamma \vdash e : \mathcal{P} \qquad \Gamma' \vdash e_i : \tau' \qquad \boxed{\omega(e_1, e_2) \equiv \omega(\mathbf{X}, \mathbf{Z}) = 1}}{\Gamma, \Gamma' \vdash \text{case } e \text{ of } \{\mathbf{X} \to e_1 \mid \mathbf{Z} \to e_2\} : \tau'} \oplus\text{-E}$$

Center-fixing homomorphisms

Center-fixing: P_1 commutes with P_2 iff $f(P_1)$ commutes with $f(P_2)$.

Paulis can be represented as vectors $[x, z]$ over $\mathbb{Z}_2$ with phase:

$$i^r X^x Z^z$$

The symplectic form $\omega([x_1, z_1], [x_2, z_2]) = z_1 \cdot x_2 - x_1 \cdot z_2$ tells us if two Paulis commute or anticommute:

$$P_1 P_2 = (-1)^{\omega([x_1,z_1],[x_2,z_2])} P_2 P_1$$

$$\frac{\Gamma \vdash e : \mathcal{P} \qquad \Gamma' \vdash e_i : \tau' \qquad \boxed{\omega(e_1, e_2) \equiv \omega(\mathbf{X}, \mathbf{Z}) = 1}}{\Gamma, \Gamma' \vdash \text{case } e \text{ of } \{\mathbf{X} \to e_1 \mid \mathbf{Z} \to e_2\} : \tau'} \oplus\text{-E}$$

*Complete but not necessarily efficient decision procedure.

Conclusion

Conclusion

- Instead of matrices or circuits, write functions!

Conclusion

- Instead of matrices or circuits, write functions!
- Linear types enforce linear transformations

Conclusion

- Instead of matrices or circuits, write functions!
- Linear types enforce linear transformations
- Can extend beyond vector spaces to:
  - Modules over rings
  - Vectors with phase (e.g. Pauli group)
  - Linear transformations that must preserve other properties (inner product, commutativity)

Conclusion

- Instead of matrices or circuits, write functions!
- Linear types enforce linear transformations
- Can extend beyond vector spaces to:
  - Modules over rings
  - Vectors with phase (e.g. Pauli group)
  - Linear transformations that must preserve other properties (inner product, commutativity)

What kinds of linear transformations are you programming?

References

[1] Thorsten Altenkirch and Jonathan Grattage. **"A functional quantum programming language".** In: LICS. 2005.

[2] Juan Miguel Arrazola et al. **"Universal quantum circuits for quantum chemistry".** In: Quantum (2022).

[3] Pablo Arrighi and Gilles Dowek. **"Lineal: A linear-algebraic lambda-calculus".** In: Logical Methods in Computer Science (2017).

[4] Alejandro Díaz-Caro and Gilles Dowek. **"A linear linear lambda-calculus".** In: Mathematical Structures in Computer Science (2024).

[5] Alejandro Díaz-Caro et al. **"Realizability in the Unitary Sphere".** In: LICS. 2019.

[6] Jennifer Paykin and Sam Winnick. **"Qudit Quantum Programming with Projective Cliffords".** In: PACMPL POPL (2026).

[7] Joey Velez-Ginorio et al. **"Compiling to Linear Neurons".** In: PACMPL POPL (2026).

Linear Types for Linear Algebra, and Beyond

Jennifer Paykin

University of Vermont
jennifer.paykin@uvm.edu

November 14, 2026
ECLAPS, Montreal, Canada