Foundations of Quantum Computational Advantage
May 1, 2024

Programming Clifford Unitaries with Symplectic Types

Jennifer Paykin (Intel Labs)

with Sam Winnick (University of Waterloo)

intel.

Quantum Programming Languages

Gate-based programming:

- Qiskit, Circ, Q#, tket, Intel Quantum SDK

```
quantum_kernel void measZAll() {
  for (int Index = 0; Index < N; Index++)
    MeasZ(QubitReg[Index], CReg[Index]); // Apply measurement gates
}
```

Beyond gate-based programming:

- Identify mathematical abstractions
- Build a language that harnesses those abstractions
- Express algorithms naturally and enable new ideas

Cliffords as automorphisms on the Pauli group

Unitary matrices U satisfying

$$\forall P \in \mathcal{P}_n,$$
$$UPU^\dagger \in \mathcal{P}_n$$

Pauli group ($\mathcal{P}$)

$$X \times X = I$$
$$X \times Y = iZ$$
$$X \times Z = -iY$$
$$X \times I = X$$

Projective Clifford group:

Automorphisms on the Pauli group

$$P \mapsto P'$$

that fix the center

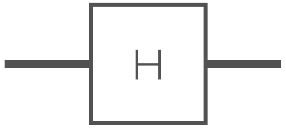
Main Idea

Clifford unitaries

expressed as functions
on qudit Pauli operators

that satisfy certain properties
(center-fixing automorphism)

Example



Idea:
Clifford unitaries
expressed as functions
on Pauli operators
that satisfy certain properties

```
h (P : PauliType) : Phase PauliType =
    case P of
        inX -> ?
        inZ -> ?
```

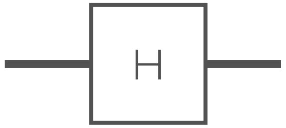
$HXH = (-1)^0 Z$

`case ? of …`
=
break up the input into basis elements

`PauliType`
=
type of single-qubit Pauli encodings

`inX/inZ`
=
syntax referring to X/Z Paulis

Example

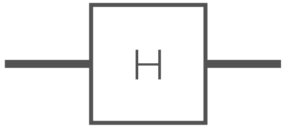


Idea:
Clifford unitaries
expressed as functions
on Pauli operators
that satisfy certain properties

```
h (P : PauliType) : Phase PauliType =
    case P of
        inX -> <0> inZ
        inZ -> ?
```

$HXH = (-1)^0 Z$

Example



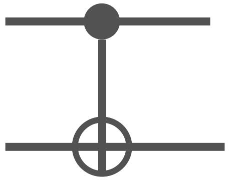
Idea:

Clifford unitaries
expressed as functions
on Pauli operators
that satisfy certain properties

```
h (P : PauliType) : Phase PauliType =
    case P of
        inX -> <0> inZ
        inZ -> <0> inX
```

$HZH = (-1)^0 X$

Example



```
cnot (P : PauliType ⊕ PauliType) : Phase (PauliType ⊕ PauliType) =
    case P of
        in1 Q -> case Q of
                    inX ->
                    inZ -> <0>(in1 inZ)
        in2 Q -> case Q of
                    inX ->
                    inZ ->
```

$$CNOT\,(Z \otimes I)\,CNOT = Z \otimes I$$

Example

```
cnot (P : PauliType ⊕ PauliType) : Phase (PauliType ⊕ PauliType) =
    case P of
        in1 Q -> case Q of
                    inX -> <0>(in1 inX) * <0>(in2 inX)
                    inZ -> <0>(in1 inZ)
        in2 Q -> case Q of
                    inX ->
                    inZ ->
```

$$CNOT\,(X \otimes I)CNOT = X \otimes X$$

Example

```
cnot (P : PauliType ⊕ PauliType) : Phase (PauliType ⊕ PauliType) =
    case P of
        in1 Q -> case Q of
                    inX -> <0>(in1 inX) * <0>(in2 inX)
                    inZ -> <0>(in1 inZ)
        in2 Q -> case Q of
                    inX -> <0>(in2 inX)
                    inZ -> <0>(in1 inZ) * <0>(in2 inZ)
```

Desiderata

1. Functions implement Cliffords:

   center-fixing automorphisms on the Pauli group

```
notClifford (P : PauliType) : Phase PauliType =
    case P of
        inX -> <0> inX
        inZ -> <0> inX
```

Type-checking Error:
The **inX** and **inZ** branches of the case statement should anticommute.

Type system for ensuring functions are indeed automorphisms.

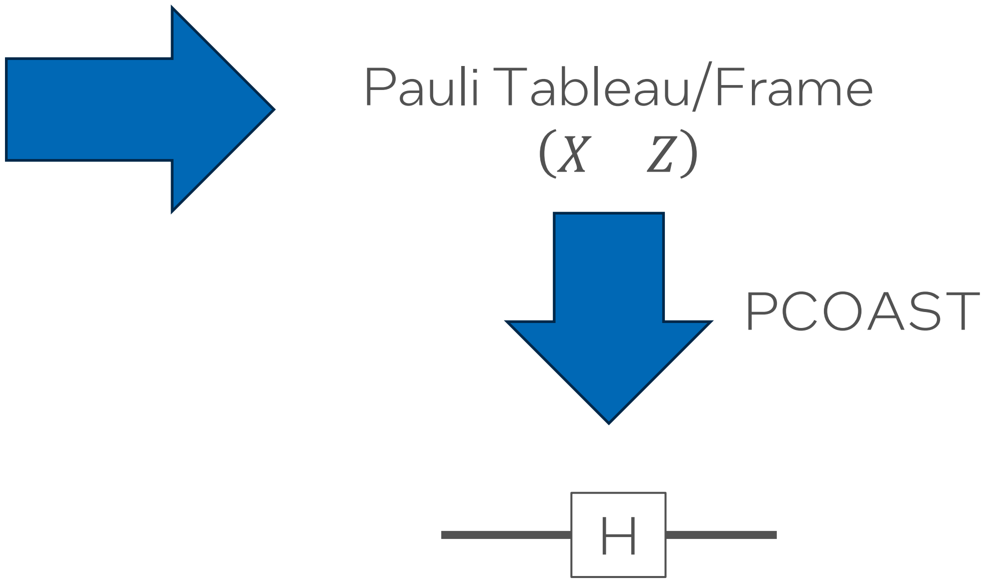
Desiderata

2. All Cliffords can be represented

Desiderata

3. (Qubit) Clifford functions can be compiled to circuits

```
h (P : PauliType) : Phase PauliType =
    case P of
        inX -> <0>inZ
        inZ -> <0>inX
```



Aaronson and Gottesman, “Improved simulation of stabilizer circuits,” 2004.

Paykin, Schmitz, et al. PCOAST: A Pauli-based quantum circuit optimization framework. *QCE 2023*.

Overview

1. Background on encodings of the Pauli group

2. Projective Cliffords as symplectic functions over Pauli encodings

3. Type system for symplectic functions

Background: the Pauli group

Any two Paulis either commute or anti-commute

Single-qubit Paulis (p)

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \mathrm{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \mathrm{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Pauli group ($\mathcal{P}$)

$$X \times X = I$$
$$X \times Y = iZ$$
$$X \times Z = -iY$$
$$X \times I = X$$

...

Symplectic Form

$$\omega : \mathcal{P} \otimes \mathcal{P} \to \mathbb{Z}_2$$

encodes commutativity of Paulis

$$P_1 \times P_2 = (-1)^{\omega(P_1,P_2)} P_2 \times P_1$$

$$\omega(X,Y) = \omega(Y,Z) = \omega(Z,X) = 1$$
$$\omega(P,P) = 0$$
$$\omega(I,P) = 0$$

Background: the Pauli algebra

Every member of the Pauli group can be written as

$i^r \Delta_{[x,z]}$

where $r \in \mathbb{Z}_4$ and $x, z \in \mathbb{Z}_2$ and

$\Delta_{[x,z]} = i^{xz} X^x Z^z$

Let's write this $\langle r \rangle [x, z]$.

Example:

"Y" $= \langle 0 \rangle [1,1]$

since $Y = iXZ = i^0 i^1 X^1 Z^1$.

Symplectic form

$$\omega(\langle r_1 \rangle [x_1, z_1], \langle r_2 \rangle [x_2, z_2]) = x_1 z_2 - z_1 x_2$$

Example:

$$\omega(\text{"X"}, \text{"Y"}) = \omega(\langle 0 \rangle [1,0], \langle 0 \rangle [1,1])$$
$$= 1 * 1 - 0 * 1 = 1$$

Background: generalizing the Pauli algebra

Generalize to n-qubit Paulis $\mathcal{P}_n$

$$p_0 \otimes \cdots \otimes p_{n-1}$$

Algebra:

$$\langle r \rangle [\vec{x}, \vec{z}]$$

$$= i^r \Delta_{[x_0, z_0]} \otimes \cdots \otimes \Delta_{[x_{n-1}, z_{n-1}]}$$

$$= i^r i^{\vec{x} \cdot \vec{z}} (X^{x_0} \otimes \cdots \otimes X^{x_{n-1}})(Z^{z_0} \otimes \cdots \otimes Z^{z_{n-1}})$$

where $r \in \mathbb{Z}_4$,

$\vec{x} = [x_0, \ldots, x_{n-1}] \in \mathbb{Z}_2^n$

$\vec{z} = [z_0, \ldots, z_{n-1}] \in \mathbb{Z}_2^n$

V = vectors in the Pauli algebra encoding over $\mathbb{Z}_2$

aka $V = \mathbb{Z}_2^n \oplus \mathbb{Z}_2^n$

Symplectic Form

$$\omega : V \otimes V \to \mathbb{Z}_2$$

$$\omega([\vec{x_1}, \vec{z_1}], [\vec{x_2}, \vec{z_2}]) = \vec{x_1} \cdot \vec{z_2} - \vec{z_1} \cdot \vec{x_2}$$

Background: generalizing the Pauli algebra

Generalize to n-qu$\boldsymbol{d}$it Paulis $P_{d,n}$

$$X|r\rangle = |(r+1) \bmod d\rangle$$
$$Z|r\rangle = \zeta^r |r\rangle \qquad \text{where } \zeta^d = 1.$$

Algebra:
$\langle r\rangle[\vec{x}, \vec{z}]$

$$= \zeta^r \Delta_{[\vec{x},\vec{z}]} = \zeta^r \zeta^{\frac{1}{2}\vec{x}\cdot\vec{z}} X^{\vec{x}} Z^{\vec{x}}$$

where $r \in ½\, \mathbb{Z}_{d'}$

$$d' = \begin{cases} d & d \text{ odd} \\ 2d & d \text{ even} \end{cases}$$

$\vec{x} = [x_0, \ldots, x_{n-1}] \in \mathbb{Z}_d^n$
$\vec{z} = [z_0, \ldots, z_{n-1}] \in \mathbb{Z}_d^n$

V = vectors in the Pauli algebra encoding over $\mathbb{Z}_d$
aka $V = \mathbb{Z}_d^n \oplus \mathbb{Z}_d^n$

Symplectic Form
$\omega: V \otimes V \to \mathbb{Z}_d$

$$\omega([\overrightarrow{x_1}, \overrightarrow{z_1}], [\overrightarrow{x_2}, \overrightarrow{z_2}]) = \overrightarrow{x_1} \cdot \overrightarrow{z_2} - \overrightarrow{z_1} \cdot \overrightarrow{x_2}$$

Theorem

V' = vectors in the Pauli algebra encoding vector space over $R' = \mathbb{Z}_{d'}$

$\frac{1}{2}\mathbb{Z}_{d'}$ = coefficients of ζ in the Pauli algebra encoding where $\zeta^{1/2}$ is a d'-th root of unity

The set of projective Cliffords $PCl'_{d'/d}$

$\cong$

The set of pairs of functions (δ, ϕ) where

- $\delta: V' \to \frac{1}{2}\mathbb{Z}_{d'}$ is a linear transformation;
- $\phi: V' \to V'$ is a symplectomorphism---a linear isomorphism that respects the symplectic form; and
- the function $\Delta_v \mapsto \zeta^{\delta(v)}\Delta_{\phi(v)}$ is right-definite.

$$\Delta_v \mapsto \zeta^{\delta(v)}\Delta_{\phi(v)}$$

Theorem

V = vectors in the Pauli algebra encoding over $\mathbb{Z}_d$
aka $V = \mathbb{Z}_d^n \oplus \mathbb{Z}_d^n$

The set of **projective Cliffords** $PCl'_{d,n}$
$\cong$
Functions over the Pauli algebra where

- $\mu: V \to \mathbb{Z}_d$ is an R-linear map; and
- $\psi: V \to V$ is a symplectomorphism---a linear isomorphism satisfying

$$\omega\big(\psi(P_1), \psi(P_2)\big) = \omega(P_1, P_2)$$

Proof sketch:

Projective Clifford → Encoding (δ, ϕ) over V'
→ Compact encoding (μ, ψ) over V
→ Encoding (δ, ϕ) over V'
→ Projective Clifford

Desiderata

Projective **Cliffords** $PCl'_{d,n}$
$\cong$
Pairs of functions (μ, ψ) where

- $\mu : V \to \mathbb{Z}_d$ is a linear transformation; and
- $\psi : V \to V$ is a symplectomorphism.

~~1. Functions implement Cliffords: automorphisms on the Pauli group~~
  - ~~Type system for ensuring functions are automorphisms~~

1a. Functions implement (μ, ψ)
  - Type system for ensuring properties are respected

~~2. All Cliffords can be represented~~

2a. All such functions can be represented

3. All functions can be compiled to circuits

Path towards a type system

Projective **Cliffords** $PCl'_{d,n}$
$\cong$
Pairs of functions (μ, ψ) where
- $\mu: V \to \mathbb{Z}_d$ is a linear transformation; and
- $\psi: V \to V$ is a symplectomorphism.

1. Type system for free modules over a ring, with biproducts

2. Type system for symplectic morphisms—linear transformations that respect the symplectic form

3. Type system for Paulis $\langle r \rangle v$

Defining a type system

1. What are types?
2. What are values of a given type?
3. What properties should well-typed expressions satisfy?
4. What are the typing rules for well-typed expressions?

$$e[c_1 \cdot v_1 + c_2 \cdot v_2] \equiv c_1 \cdot e[v_1] + c_2 \cdot e[v_2]$$

Types: free finitely-generated R-modules

Values: vectors in the R-module

Module Types τ	Values
R	Constants $r \in R$
$\tau_1 \oplus \tau_2$	Tuples $[v_1, v_2]$

Expressions: linear transformations

$$x : \tau \vdash e : \tau'$$

1. Expressions

$$\Gamma \vdash e : \tau'$$

$$\Gamma \coloneqq x_1 \colon \tau_1, \dots, x_n \colon \tau_n$$

$$\tau \coloneqq R \mid \tau_1 \oplus \tau_2$$

$e \coloneqq$	$x \mid r \mid [\mathrm{e}_1, \mathrm{e}_2]$	Variables, scalars, and vectors
	$e_1 + e_2 \mid e_1 \cdot e_2$	Operations on scalars and vectors
	case e of $\{in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2\}$	Vector case analysis

relevant type system:

- contraction: variables **can** be duplicated
- no weakening: variables **cannot** be discarded

Arrighi & Dowek. Lineal: A linear-algebraic lambda-calculus. LMCS 2017.
Díaz-Caro & Dowek. A new connective in natural deduction, and its application to quantum computing. TCS 2023.
Díaz-Caro & Dowek. A linear linear lambda-calculus. MSCS 2024.

1. Semantics

$x:\tau \vdash e : \tau'$

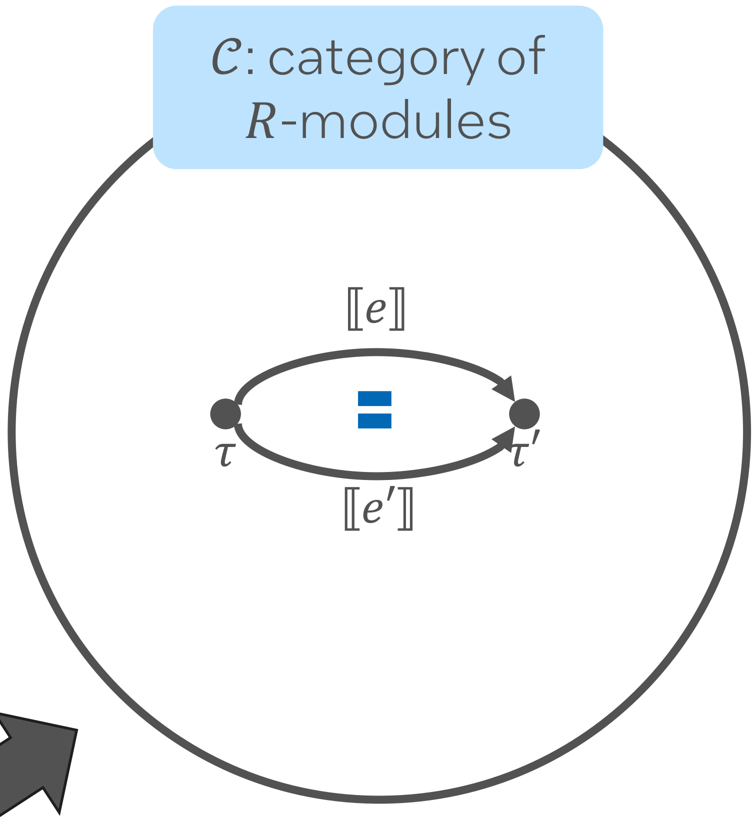
operational semantics

$x:\tau \vdash e' : \tau'$

equivalence relation

$x:\tau \vdash e \equiv e' : \tau'$

categorical semantics



Path towards a type system

Projective **Cliffords** $PCl'_{d,n}$
$\cong$
Pairs of functions (μ, ψ) where
- $\mu : V \to \mathbb{Z}_d$ is a linear transformation; and
- $\psi : V \to V$ is a symplectomorphism.

1. Type system for free modules over a ring, with biproducts

2. **Type system for symplectic morphisms—linear transformations that respect the symplectic form**

3. Type system for Paulis $\langle r \rangle v$

2. Type system for symplectic morphisms

Types: free finitely-generated $\boldsymbol{R}$-modules for which symplectic form is defined

Values: vectors in the $\boldsymbol{R}$-module

Symplectic Types $\boldsymbol{\sigma}$	Values
$\boldsymbol{Q} = R \oplus R$	Single-qudit vector $[x, z]$ encoding $\Delta_{[x,z]}$
$\sigma_1 \oplus \sigma_2$	Tuples $[v_1, v_2]$

Expressions: linear transformations that respect symplectic form

$$x : \sigma \vdash^S e : \sigma'$$

$$\omega(e[v_1], e[v_2]) \equiv \omega(v_1, v_2)$$

2. Symplectic type system

$$a_1 : \tau_1, \ldots, a_n : \tau_n \; ; b : \sigma \vdash^S e : \sigma$$

Linear transformation

Respect symplectic form

Module Types $\boldsymbol{\tau}$	Values
R	Constants $r \in R$
$\tau_1 \oplus \tau_2$	Tuples $[v_1, v_2]$

Symplectic types $\boldsymbol{\sigma}$	Vector spaces used in the Pauli algebra (dimension $\boldsymbol{2n}$)
$\boldsymbol{Q} = R \oplus R$	Single-qudit vector $[x, z]$ encoding $\Delta_{[x,z]}$
$\sigma_1 \oplus \sigma_2$	Tuple of n-qudit vectors

2. Expressions

Symplectic types $\boldsymbol{\sigma}$	Vector spaces used in the Pauli algebra (dimension $2\boldsymbol{n}$)
$\boldsymbol{Q} = \mathrm{R} \oplus \mathrm{R}$	Single-qudit vector [z,x] encoding $\Delta_{[z,x]}$
$\sigma_1 \oplus \sigma_2$	Tuple of n-qudit vectors

e ≔	$x \mid r \mid [\mathrm{e}_1, \mathrm{e}_2]$	Variables, scalars, and vectors
	$e_1 + e_2 \mid e_1 \cdot e_2$	Operations on scalars and vectors
	$\text{case } e \text{ of } \{in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2\}$	Vector case analysis
*	$\omega_\sigma(e_1, e_2)$	Symplectic form

$$\frac{\Gamma_1 \vdash e_1 : \sigma \quad \Gamma_2 \vdash e_2 : \sigma}{\Gamma_1 \cup \Gamma_2 \vdash \omega_\sigma(e_1, e_2) : R}$$

$$\omega_{\boldsymbol{Q}}([r_1, r_1'], [r_2, r_2']) \quad \to \quad r_1 r_2' - r_1' r_2$$

$$\omega_{\sigma_1 \oplus \sigma_2}([v_1, v_1'], [v_2, v_2']) \quad \to \quad \omega_{\sigma_1}(v_1, v_2) + \omega_{\sigma_2}(v_1', v_2')$$

2. Expressions

Symplectic types $\boldsymbol{\sigma}$	Vector spaces used in the Pauli algebra (dimension $2n$)
$\boldsymbol{Q} = R \oplus R$	Single-qudit vector [z,x] encoding $\Delta_{[z,x]}$
$\sigma_1 \oplus \sigma_2$	Tuple of n-qudit vectors

$e \ \coloneqq$	$x \mid r \mid [\mathrm{e}_1, \mathrm{e}_2]$	Variables, scalars, and vectors
	$e_1 + e_2 \mid e_1 \cdot e_2$	Operations on scalars and vectors
*	$\text{case } e \text{ of } \{in_\mathrm{x} \to e_\mathrm{x} \mid in_\mathrm{z} \to e_\mathrm{z}\}$	Pauli case analysis
	$\text{case } e \text{ of } \{in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2\}$	Vector case analysis
	$\omega_\sigma(e_1, e_2)$	Symplectic form

$in_z = [1,0]$
$in_x = [0,1]$

$$\frac{\Gamma; \Delta \vdash^S e : \boldsymbol{Q} \quad \Gamma'; \Delta' \vdash^S e_x : \sigma \quad \Gamma'; \Delta' \vdash^S e_z : \sigma \quad \omega_\sigma(e_x, e_z) \equiv 1}{\Gamma \cup \Gamma'; \Delta, \Delta' \vdash^S \text{case } e \text{ of } \{ in_X \to e_\mathrm{x} \mid in_Z \to e_\mathrm{z} \} : \sigma}$$

2. Expressions

Symplectic types $\boldsymbol{\sigma}$	Vector spaces used in the Pauli algebra (dimension $2n$)
$\boldsymbol{Q} = \mathrm{R} \oplus \mathrm{R}$	Single-qudit vector [z,x] encoding $\Delta_{[z,x]}$
$\sigma_1 \oplus \sigma_2$	Tuple of n-qudit vectors

```
h (P : QType) : QType =
    case P of
        inX -> inZ
        inZ -> inX
```

$$\omega(in_Z, in_X) = \omega([0,1],[1,0]) = 0 - 1 = 1\ (mod\ 2)$$

$$\frac{\Gamma; \Delta \vdash^S e : \boldsymbol{Q} \quad \Gamma'; \Delta' \vdash^S e_x : \sigma \quad \Gamma'; \Delta' \vdash^S e_z : \sigma \quad \omega_\sigma(e_x, e_z) \equiv 1}{\Gamma \cup \Gamma'; \Delta, \Delta' \vdash^S \text{case } e \text{ of } \{\, in_X \to e_x \mid in_Z \to e_z \,\} : \sigma}$$

2. Expressions

Symplectic types $\boldsymbol{\sigma}$	Vector spaces used in the Pauli algebra (dimension $2n$)
$\boldsymbol{Q} = \mathrm{R} \oplus \mathrm{R}$	Single-qudit vector [z,x] encoding $\Delta_{[z,x]}$
$\sigma_1 \oplus \sigma_2$	Tuple of n-qudit vectors

```
notSymplectic(x : QType) : QType =
    case x of
        inX -> inZ
        inZ -> inZ
```

$$\omega(in_Z, in_Z) = \omega([0,1],[0,1]) = 0 - 0 \neq 1$$

$$\frac{\Gamma; \Delta \vdash^S e : \boldsymbol{Q} \quad \Gamma'; \Delta' \vdash^S e_x : \sigma \quad \Gamma'; \Delta' \vdash^S e_z : \sigma \quad \omega_\sigma(e_x, e_z) \equiv 1}{\Gamma \cup \Gamma'; \Delta, \Delta' \vdash^S \text{case } e \text{ of } \{ in_X \to e_\mathrm{x} \mid in_Z \to e_\mathrm{z} \} : \sigma}$$

2. Expressions

Symplectic types $\boldsymbol{\sigma}$	Vector spaces used in the Pauli algebra (dimension $\mathbf{2n}$)
$\boldsymbol{Q} = \mathrm{R} \oplus \mathrm{R}$	Single-qudit vector [z,x] encoding $\Delta_{[z,x]}$
$\sigma_1 \oplus \sigma_2$	Tuple of n-qudit vectors

$e \;\coloneqq$	$x \mid r \mid [\mathrm{e}_1, \mathrm{e}_2]$	Variables, scalars, and vectors
	$e_1 + e_2 \mid e_1 \cdot e_2$	Operations on scalars and vectors
	$\mathrm{case}\ e\ \mathrm{of}\ \{in_{\mathrm{x}} \to e_{\mathrm{x}} \mid in_{\mathrm{z}} \to e_{\mathrm{z}}\}$	Pauli case analysis
*	$\mathrm{case}\ e\ \mathrm{of}\ \{in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2\}$	Vector case analysis
	$\omega_\sigma(e_1, e_2)$	Symplectic form

$$\frac{\Gamma; \Delta \vdash^S e : \sigma_1 \oplus \sigma_2 \quad \Gamma'; \Delta', x_1{:}\sigma_1 \vdash^S e_1{:}\sigma \quad \Gamma'; \Delta', x_2{:}\sigma_2 \vdash^S e_2 : \sigma \quad \boxed{\omega_\sigma(e_1, e_2) \equiv 0}}{\Gamma \cup \Gamma'; \Delta, \Delta' \vdash^S \mathrm{case}\ e\ \mathrm{of}\ \{\, in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2 \,\}{:}\,\sigma}$$

2. Theorem

If $\Gamma; z : \sigma \vdash^S e : \sigma'$

then, for all $v_1, v_2 : \sigma$,

$$\omega(e\{z \mapsto v_1\}, e\{z \mapsto v_2\}) \equiv \omega(v_1, v_2)$$

Path towards a type system

Projective **Cliffords** $PCl'_{d,n}$

$\cong$

Pairs of functions (μ, ψ) where

- $\mu: V \rightarrow \mathbb{Z}_d$ is a linear transformation; and
- $\psi: V \rightarrow V$ is a symplectomorphism.

1. Type system for free modules over a ring, with biproducts
2. Type system for symplectic morphisms—linear transformations that respect the symplectic form
3. **Type system for Paulis $\langle \boldsymbol{r} \rangle \boldsymbol{v}$**

3. Type system for Pauli algebra

Pauli types $\mathbf{T}$	Values
$\text{Phase}(\sigma)$	Pairs $\langle r \rangle v$ for $r \in R$, $v : \sigma$

Functions: (δ, ϕ) where

- $\delta: \sigma \to R$ is a linear transformation;
- $\phi: \sigma \to \sigma'$ is a symplectic morphism

3. Expressions

$e \coloneqq$	$x \mid r \mid [\mathrm{e}_1, \mathrm{e}_2]$	Variables, scalars, and vectors
	$e_1 + e_2 \mid e_1 \cdot e_2$	Operations on scalars and vectors
	$\omega(e_1, e_2)$	Symplectic form
	case e of $\{in_{\mathrm{x}} \rightarrow e_{\mathrm{x}} \mid in_{\mathrm{z}} \rightarrow e_{\mathrm{z}}\}$	Pauli case analysis
	case e of $\{in_1(x_1) \rightarrow e_1 \mid in_2(x_2) \rightarrow e_2\}$	Vector case analysis
*	$\langle e \rangle e' \mid \mathrm{e}_1 \times e_2$	Pauli operations

3. Type system for Pauli algebra

$$e : \sigma \multimap \mathrm{Phase}(\sigma)$$

$$\downarrow$$

$$[\![e]\!] : \sigma \to R \oplus \sigma$$

such that

$$\mu = [\![e]\!] \circ \mathbf{first} : \sigma \to R$$
$$\psi = [\![e]\!] \circ \mathbf{second} : \sigma \to \sigma$$

satisfy

- $\mu : \sigma \to R$ is a linear transformation;
- $\psi : \sigma \to \sigma$ is a symplectomorphism.

…so what?

- Functions over Paulis as a programming abstraction
  - Data structures, recursion, polymorphism
  - Interactive feedback on what makes a Clifford
  - Quantum algorithms in terms of change-of-basis
  - Alternate bases other than `inX`/`inZ`

- Beyond Cliffords
  - The Clifford hierarchy as functions on Paulis?
  - Pauli matrices as a basis for Hilbert spaces?

Conclusion

- Programming Cliffords as functions over Paulis:
  - Clever encodings and typing rules isolate the functions corresponding to Cliffords
  - Operational and denotational semantics show it is sound
  - Need examples and implementations to show if it is useful

- Type systems can harness mathematical structures into programming abstractions

Foundations of Quantum Computational Advantage

May 1, 2024

Programming Clifford Unitaries with Symplectic Types

Jennifer Paykin (Intel Labs)
jennifer.paykin@intel.com

with Sam Winnick (University of Waterloo)

Bonus Slides

Beyond Cliffords

Approach #1: Paulis form a basis for *all* square complex matrices.

$$Rot(P,\theta) : \mathbb{C}[\boldsymbol{Q}]$$
$$= \cos\left(\frac{\theta}{2}\right) I + i \sin\left(\frac{\theta}{2}\right) P$$

$$MeasZ : \boldsymbol{Q} \multimap \mathbb{C}[\boldsymbol{Q}]$$
$$= Q \mapsto \frac{1}{4}(I+Z)Q(I+Z) + \frac{1}{4}(I-Z)Q(I-Z)$$

What properties characterize unitaries, channels, etc?

How would we compile these to circuits?

Beyond Cliffords: the Clifford Hierarchy

$$\mathcal{C}^1 = Pauli\ group$$
$$\mathcal{C}^{k+1} = \{U | \forall P \in \mathcal{C}^1, UPU^\dagger \in \mathcal{C}^k\}$$

Intuitively: the $(k+1)$-th level of the Clifford hierarchy ~ symplectic function from Paulis to k-th level?

$$\mathcal{C}^1 = \boldsymbol{Q}^n$$
$$\mathcal{C}^{k+1} = \boldsymbol{Q}^n \multimap \mathcal{C}^{k+1}$$

Beyond Cliffords: the Clifford Hierarchy

$$\mathcal{C}^1 = \boldsymbol{Q}^n$$
$$\mathcal{C}^{k+1} = \boldsymbol{Q}^n \multimap \mathcal{C}^{k+1}$$

```
// t : 𝒞³ = Pauli ⊸ Pauli ⊸ Pauli
t (P : Pauli) (Q : Pauli) : Pauli =
    case P of
        inX -> case Q of
                  inX -> <0>[1,1]
                  inZ -> <1>inZ
        inZ -> not Q
```

Beyond Cliffords: the Clifford Hierarchy

Challenges:

- $\mathcal{C}^k$ is not, in general, a group. Not even closed under composition.
- The entire Clifford hierarchy $\neq$ all unitaries

- How does the Clifford hierarchy interact with the Pauli algebra encoding?

Example: lift Paulis to Cliffords

```
pauli_to_clifford (P : Q) : (Q -> Phase Q) =
    fun Q =>
      <omega(P,Q)> Q.
```

$$PQP^{\dagger} = (-1)^{\omega(P,Q)}\, Q$$

Compilation to circuits

Expressions

↓

Pauli frames/tableaus

↓ PCOAST

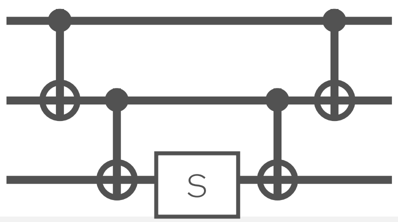
Circuits

$$\vdash^T f : \underbrace{Q \oplus \cdots \oplus Q}_{n} \multimap \underbrace{Q \oplus \cdots \oplus Q}_{n}$$

↓

$$\begin{pmatrix} f(in_0(in_Z)) & f(in_0(in_X)) \\ \vdots & \vdots \\ f(in_{n-1}(in_Z)) & f(in_{n-1}(in_Z)) \end{pmatrix}$$

↓



Paykin, Schmitz, et al. PCOAST: A Pauli-based quantum circuit optimization framework. *QCE 2023*.