Programming Clifford Unitaries with Symplectic Types

Jennifer Paykin (Intel Labs)

with Sam Winnick (University of Waterloo)
Quantum Programming Languages

Gate-based programming:
- Qiskit, Circ, Q#, tket, Intel Quantum SDK

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

\[ \forall P \in \mathcal{P}_n, \quad UPU^\dagger \in \mathcal{P}_n \]

Pauli group ($\mathcal{P}$)

\[
\begin{align*}
X \times X &= I \\
X \times Y &= iZ \\
X \times Z &= -iY \\
X \times I &= X
\end{align*}
\]

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

\[ h(P : \text{PauliType}) : \text{Phase PauliType} = \]
\[
\text{case } P \text{ of}
\]
\[
\text{inX} \rightarrow ?
\]
\[
\text{inZ} \rightarrow ?
\]

\[ HXH = (-1)^0 Z \]

\[
\text{case } ? \text{ of } ...
\]
\[
\text{break up the input into basis elements}
\]

\[
\text{PauliType} = \text{type of single-qubit Pauli encodings}
\]

\[
\text{inX/inZ} = \text{syntax referring to } X/Z \text{ Paulis}
\]
Example

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

\[ h(P : \text{PauliType}) : \text{Phase PauliType} = \]
\[
\begin{cases}
\text{case } P \text{ of} \\
\text{inX } \rightarrow \langle 0 \rangle \text{ inZ} \\
\text{inZ } \rightarrow ?
\end{cases}
\]

\[ HXH = (-1)^0 Z \]
Example

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

$$h (P : \text{PauliType}) : \text{Phase PauliType} =$$

```
  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 ⊗ I) CNOT = Z ⊗ I_
Example

cnot\((P : \text{PauliType} \oplus \text{PauliType}) : \text{Phase} (\text{PauliType} \oplus \text{PauliType}) =\)

\[
\begin{align*}
\text{case } P \text{ of} \\
\text{in1 } Q \rightarrow \text{ case } Q \text{ of} \\
\quad \text{inX} \rightarrow \langle 0 \rangle (\text{in1 } \text{inX}) \ast \langle 0 \rangle (\text{in2 } \text{inX}) \\
\quad \text{inZ} \rightarrow \langle 0 \rangle (\text{in1 } \text{inZ}) \\
\text{in2 } Q \rightarrow \text{ case } Q \text{ of} \\
\quad \text{inX} \rightarrow \\
\quad \text{inZ} \rightarrow \\
\end{align*}
\]

\text{\textbf{CNOT}} (X \otimes I)\text{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

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

   Type system for ensuring functions are indeed automorphisms.

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

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

Single-qubit Paulis ($\mathcal{P}$)

\[
X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\
Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}
\]

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

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

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

Any two Paulis either commute or anti-commute
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:

\[
\omega("X","Y") = \omega(\langle 0 \rangle [1,0], \langle 0 \rangle [1,1]) = 1 \ast 1 - 0 \ast 1 = 1
\]
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 $r \in \mathbb{Z}_4$,

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

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

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

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 \( n \)-qudit Paulis \( P_{d,n} \)

\[
X|r\rangle = |(r + 1) \mod d\rangle \\
Z|r\rangle = \zeta^r|r\rangle \quad \text{where } \zeta^d = 1.
\]

Algebra:
\[
\langle r\rangle[\tilde{x}, \tilde{z}] = \zeta^r \Delta[\tilde{x}, \tilde{z}] = \zeta^r \zeta^\frac{1}{2} \tilde{x} \cdot \tilde{z} X \tilde{x} Z \tilde{z}
\]

where
- \( r \in \frac{1}{2} \mathbb{Z}_{d'} \)
- \( d' = \begin{cases} d & \text{d odd} \\ 2d & \text{d even} \end{cases} \)
- \( \tilde{x} = [x_0, \ldots, x_{n-1}] \in \mathbb{Z}_d^n \)
- \( \tilde{z} = [z_0, \ldots, z_{n-1}] \in \mathbb{Z}_d^n \)

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

Symplectic Form
\( \omega: V \otimes V \rightarrow \mathbb{Z}_d \)
\[
\omega([\tilde{x}_1, \tilde{z}_1], [\tilde{x}_2, \tilde{z}_2]) = \tilde{x}_1 \cdot \tilde{z}_2 - \tilde{z}_1 \cdot \tilde{x}_2
\]
Theorem

The set of projective Cliffords $PCl_{d'/d}'$ is isomorphic to the set of pairs of functions $(\delta, \phi)$ where:

- $\delta: V' \rightarrow \frac{1}{2}Z_{d'}$ is a linear transformation;
- $\phi: V' \rightarrow V'$ 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.

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

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

The set of **projective Cliffords** $PCI_{d,n}'$

$\cong$

Functions over the Pauli algebra where

- $\mu: V \rightarrow \mathbb{Z}_d$ is an $R$-linear map; and
- $\psi: V \rightarrow V$ is a symplectomorphism—a linear isomorphism satisfying
  \[ \omega(\psi(P_1), \psi(P_2)) = \omega(P_1, P_2) \]

**Proof sketch:**

Projective Clifford $\rightarrow$ Encoding $(\delta, \phi)$ over $V'$
  $\rightarrow$ Compact encoding $(\mu, \psi)$ over $V$
  $\rightarrow$ Encoding $(\delta, \phi)$ over $V'$
  $\rightarrow$ Projective Clifford

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

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

1. Functions implement Cliffords: automorphisms on the Pauli group
   - Type system for ensuring functions are automorphisms
1a. Functions implement \((\mu, \psi)\)
   - 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

Projective Cliffords \(PCl'_{d,n}\)

Pairs of functions \((\mu, \psi)\) where
- \(\mu: V \to \mathbb{Z}_d\) is a linear transformation; and
- \(\psi: V \to V\) is a symplectomorphism.
Path towards a type system

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 \( \langle r \rangle \nu \)

Projective **Cliffords** \( PCl'_{d,n} \)

\[ \cong \]

Pairs of functions \((\mu, \psi)\) where
- \(\mu: V \to \mathbb{Z}_d\) is a linear transformation; and
- \(\psi: V \to V\) is a symplectomorphism.
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?

**Types:** free finitely-generated $R$-modules

**Values:** vectors in the $R$-module

<table>
<thead>
<tr>
<th>Module Types $\tau$</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>Constants $r \in R$</td>
</tr>
<tr>
<td>$\tau_1 \oplus \tau_2$</td>
<td>Tuples $[v_1, v_2]$</td>
</tr>
</tbody>
</table>

Expressions: linear transformations

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

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

\[ \Gamma \vdash e : \tau' \]
\[ \Gamma := x_1 : \tau_1, \ldots, x_n : \tau_n \]
\[ \tau := R \mid \tau_1 \oplus \tau_2 \]

<table>
<thead>
<tr>
<th>( e )</th>
<th>( \Gamma )</th>
<th>( e ) ( : ) ( \tau' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \begin{align*} x &amp; \mid r \mid [e_1, e_2] \end{align*} )</td>
<td>Variables, scalars, and vectors</td>
<td></td>
</tr>
<tr>
<td>( e_1 + e_2 \mid e_1 \cdot e_2 )</td>
<td>Operations on scalars and vectors</td>
<td></td>
</tr>
<tr>
<td>case ( e ) of ( { in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2 } )</td>
<td>Vector case analysis</td>
<td></td>
</tr>
</tbody>
</table>

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

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

\[ C : \text{category of } R\text{-modules} \]
Path towards a type system

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 $\langle r \rangle \nu$

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

Pairs of functions $(\mu, \psi)$ where
- $\mu: V \to \mathbb{Z}_d$ is a linear transformation; and
- $\psi: V \to V$ is a symplectomorphism.
2. Type system for symplectic morphisms

Types: free finitely-generated $R$-modules for which symplectic form is defined

Values: vectors in the $R$-module

<table>
<thead>
<tr>
<th>Symplectic Types $\sigma$</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = R \oplus R$</td>
<td>Single-qudit vector $[x, z]$ encoding $\Delta_{[x,z]}$</td>
</tr>
<tr>
<td>$\sigma_1 \oplus \sigma_2$</td>
<td>Tuples $[v_1, v_2]$</td>
</tr>
</tbody>
</table>

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 $\tau$

<table>
<thead>
<tr>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
</tr>
<tr>
<td>Constants $r \in R$</td>
</tr>
<tr>
<td>$\tau_1 \oplus \tau_2$</td>
</tr>
<tr>
<td>Tuples $[v_1, v_2]$</td>
</tr>
</tbody>
</table>

### Symplectic types $\sigma$

<table>
<thead>
<tr>
<th>Vector spaces used in the Pauli algebra (dimension $2n$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = R \oplus R$</td>
</tr>
<tr>
<td>Single-qudit vector $[x, z]$ encoding $4_{[x,z]}$</td>
</tr>
<tr>
<td>$\sigma_1 \oplus \sigma_2$</td>
</tr>
<tr>
<td>Tuple of n-qudit vectors</td>
</tr>
</tbody>
</table>
2. Expressions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e := x \mid r \mid [e_1, e_2]$</td>
<td>Variables, scalars, and vectors</td>
</tr>
<tr>
<td>$e_1 + e_2 \mid e_1 \cdot e_2$</td>
<td>Operations on scalars and vectors</td>
</tr>
<tr>
<td>case $e$ of ${in_1(x_1) \rightarrow e_1 \mid in_2(x_2) \rightarrow e_2}$</td>
<td>Vector case analysis</td>
</tr>
<tr>
<td>$\ast \omega_\sigma(e_1, e_2)$</td>
<td>Symplectic form</td>
</tr>
</tbody>
</table>

\[ \Gamma_1 \vdash e_1 : \sigma \quad \Gamma_2 \vdash e_2 : \sigma \]
\[ \Gamma_1 \cup \Gamma_2 \vdash \omega_\sigma(e_1, e_2) : R \]

\[ \omega_Q([r_1, r_1'], [r_2, r_2']) \rightarrow r_1 r_2' - r_1' r_2 \]
\[ \omega_{\sigma_1 \oplus \sigma_2}([v_1, v_1'], [v_2, v_2']) \rightarrow \omega_{\sigma_1}(v_1, v_2) + \omega_{\sigma_2}(v_1', v_2') \]
2. Expressions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e := x \mid r \mid [e_1, e_2]$</td>
<td>Variables, scalars, and vectors</td>
</tr>
<tr>
<td>$e_1 + e_2 \mid e_1 \cdot e_2$</td>
<td>Operations on scalars and vectors</td>
</tr>
<tr>
<td>* case $e$ of ${in_x \to e_x \mid in_z \to e_z}$</td>
<td>Pauli case analysis</td>
</tr>
<tr>
<td>case $e$ of ${in_1(x_1) \to e_1 \mid in_2(x_2) \to e_2}$</td>
<td>Vector case analysis</td>
</tr>
<tr>
<td>$\omega_\sigma(e_1, e_2)$</td>
<td>Symplectic form</td>
</tr>
</tbody>
</table>

Symplectic types $\sigma$ | Vector spaces used in the Pauli algebra (dimension $2n$) |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = R \oplus R$</td>
<td>Single-qudit vector $[z, x]$ encoding $\Delta_{[z, x]}$</td>
</tr>
</tbody>
</table>
| $\sigma_1 \oplus \sigma_2$ | Tuple of n-qudit vectors

\[
\Gamma; \Delta \vdash^S e : Q \quad \Gamma'; \Delta' \vdash^S e_x : \sigma \quad \Gamma'; \Delta' \vdash^S e_z : \sigma \quad \omega_\sigma(e_x, e_z) \equiv 1
\]

\[
\Gamma \cup \Gamma'; \Delta, \Delta' \vdash^S \text{case } e \text{ of } \{in_x \to e_x \mid in_z \to e_z\} : \sigma
\]
2. Expressions

\[
\begin{align*}
\text{h (P : QType) : QType =} \\
\quad \text{case P of} \\
\quad \quad \text{inX -> inZ} \\
\quad \quad \text{inZ -> inX}
\end{align*}
\]

\[
\omega (\text{in}_Z, \text{in}_X) = \omega ([0,1], [1,0]) = 0 - 1 = 1 \pmod{2}
\]

\[
\begin{align*}
\Gamma; \Delta \vdash_S e : Q & \quad \Gamma'; \Delta' \vdash_S e_x : \sigma \\
& \quad \Gamma'; \Delta' \vdash_S e_z : \sigma \\
& \quad \omega_\sigma (e_x, e_z) \equiv 1 \\
\end{align*}
\]

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

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

notSymplectic(x : QType) : QType =

\[
\text{case } x \text{ of } \\
\text{  inX } \rightarrow \text{ inZ } \\
\text{  inZ } \rightarrow \text{ inZ }
\]

\[
\Gamma; \Delta \vdash^S e : Q \quad \Gamma'; \Delta' \vdash^S e_x : \sigma \quad \Gamma'; \Delta' \vdash^S e_z : \sigma \quad \omega_\sigma(e_x, e_z) \equiv 1
\]

\[
\Gamma \cup \Gamma'; \Delta, \Delta' \vdash^S \text{ case } e \text{ of } \{ \text{in}_X \rightarrow e_x \mid \text{in}_Z \rightarrow e_z \} : \sigma
\]
2. Expressions

<table>
<thead>
<tr>
<th>$\Gamma; \Delta \vdash_S e : \sigma_1 \oplus \sigma_2$</th>
<th>$\Gamma' \vdash S_1 e_1 : \sigma$ \hspace{1cm} $\Gamma' \vdash S_2 e_2 : \sigma$</th>
<th>$\omega_\sigma(e_1, e_2) \equiv 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \cup \Gamma', \Delta, \Delta' \vdash_S$ case $e$ of ${in_1(x_1) \rightarrow e_1 \mid in_2(x_2) \rightarrow e_2}: \sigma$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table: Expressions

<table>
<thead>
<tr>
<th>$e$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \mid r \mid [e_1, e_2]$</td>
<td>Variables, scalars, and vectors</td>
</tr>
<tr>
<td>$e_1 + e_2 \mid e_1 \cdot e_2$</td>
<td>Operations on scalars and vectors</td>
</tr>
<tr>
<td>case $e$ of ${in_x \rightarrow e_x \mid in_z \rightarrow e_z}$</td>
<td>Pauli case analysis</td>
</tr>
<tr>
<td>$\omega_\sigma(e_1, e_2)$</td>
<td>Symplectic form</td>
</tr>
</tbody>
</table>

#### Symplectic types $\sigma$

| $Q = \mathbb{R} \oplus \mathbb{R}$ | Single-qudit vector $[z, x]$ encoding $\Delta_{[z, x]}$ |
| $\sigma_1 \oplus \sigma_2$ | Tuple of n-qudit vectors |
2. Theorem

\[ \text{If } \Gamma; z: \sigma \vdash^S e : \sigma' \]

\[ \text{then, for all } v_1, v_2: \sigma, \]

\[ \omega(e\{z \mapsto v_1\}, e\{z \mapsto v_2\}) \equiv \omega(v_1, v_2) \]
Path towards a type system

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 \( \langle r \rangle v \)

Projective Clifford \( PCl'_{d,n} \)

\[ \cong \]

Pairs of functions \( (\mu, \psi) \) where

- \( \mu: V \to \mathbb{Z}_d \) is a linear transformation; and
- \( \psi: V \to V \) is a symplectomorphism.
3. Type system for Pauli algebra

<table>
<thead>
<tr>
<th>Pauli types $T$</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase($\sigma$)</td>
<td>Pairs $(r)v$ for $r \in R, v : \sigma$</td>
</tr>
</tbody>
</table>

Functions: $(\delta, \phi)$ where
- $\delta: \sigma \rightarrow R$ is a linear transformation;
- $\phi: \sigma \rightarrow \sigma'$ is a symplectic morphism
3. Expressions

<table>
<thead>
<tr>
<th>$e$</th>
<th>$x \mid r \mid [e_1, e_2]$</th>
<th>Variables, scalars, and vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$e_1 + e_2 \mid e_1 \cdot e_2$</td>
<td>Operations on scalars and vectors</td>
</tr>
<tr>
<td></td>
<td>$\omega(e_1, e_2)$</td>
<td>Symplectic form</td>
</tr>
<tr>
<td></td>
<td>case $e$ of { $in_x \rightarrow e_x \mid in_z \rightarrow e_z$ }</td>
<td>Pauli case analysis</td>
</tr>
<tr>
<td></td>
<td>case $e$ of { $in_1(x_1) \rightarrow e_1 \mid in_2(x_2) \rightarrow e_2$ }</td>
<td>Vector case analysis</td>
</tr>
<tr>
<td></td>
<td>$\langle e \rangle e' \mid e_1 \times e_2$</td>
<td>Pauli operations</td>
</tr>
</tbody>
</table>
3. Type system for Pauli algebra

\[ e : \sigma \rightarrow \text{Phase}(\sigma) \]

\[ \llbracket e \rrbracket : \sigma \rightarrow R \oplus \sigma \]

such that

\[ \mu = \llbracket e \rrbracket \circ \text{first} : \sigma \rightarrow R \]
\[ \psi = \llbracket e \rrbracket \circ \text{second} : \sigma \rightarrow \sigma \]

satisfy

- \( \mu : \sigma \rightarrow R \) is a linear transformation;
- \( \psi : \sigma \rightarrow \sigma \) 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 $\text{inX/inZ}$

- 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
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.

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

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

What properties characterize unitaries, channels, etc? How would we compile these to circuits?
Beyond Cliffords: the Clifford Hierarchy

\[ C^1 = \text{Pauli group} \]

\[ C^{k+1} = \{ U | \forall P \in C^1, UPU^\dagger \in C^k \} \]

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

\[ C^1 = Q^n \]

\[ C^{k+1} = Q^n \rightarrow C^{k+1} \]
Beyond Cliffords: the Clifford Hierarchy

\[
\begin{align*}
\mathcal{C}^1 &= Q^n \\
\mathcal{C}^{k+1} &= Q^n \circ \mathcal{C}^{k+1}
\end{align*}
\]

// t : \mathcal{C}^3 = Pauli \rightarrow Pauli \rightarrow Pauli

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

FoQaCIA 2024
Beyond Cliffords: the Clifford Hierarchy

Challenges:

- $C^k$ 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

⊢^T f : Q ⊕ \cdots ⊕ Q \rightarrow Q ⊕ \cdots ⊕ Q

\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}