Introduction to Linear Algebra for Quantum Computing[1]

Fritz Henglein
henglein@diku.dk

Department of Computer Science
University of Copenhagen
(DIKU)

November 29, 2024

[1]Disclaimer: These notes are under development. They have not been checked thoroughly for possible errors.

Linear maps and their representations

- A *linear map* is a function on vector spaces that respects the underlying structure of the vector spaces involved.
- A *unitary transformation* is a linear map on Hilbert spaces that preserves distances and is surjective. Unitary transformations are interesting because they are what quantum computers can implement efficiently.
- How to *represent* a unitary transformation and, more generally, linear maps? As
  1. Matrices
  2. Quantum circuits
  3. Tensor networks
  4. λ-expression (functional programs)
  5. Domain-specific language
- Key point: These are *data structures* for *denoting* (representing) linear maps, which are abstract mathematical objects.
- We are interested in *efficient* data structures for efficiently computing with linear maps.

Matrices

Representation of a linear map as a matrix.

Theorem

Let $B_V = [v_0, \dots v_{n-1}]$ and $B_W = [w_0, \dots, w_{m-1}]$ be ordered bases of V and W, respectively. Then matrix $M(f) : \mathbb{C}^{\boldsymbol{m \times n}}$ is the unique representation of linear map $f : Hom(V, W)$ wrt. bases B_V and B_W if $f(v_j) = \sum_{i=0}^{m-1}(a_{ij} \cdot w_i)$, where $a_{ij} = M(f)_{ij}$.

- Great for theory:
  - Sequential (functional) composition is implemented by matrix multiplication.
  - The adjoint of a linear map is implemented by the conjugate transpose of a matrix.
  - Tensor product is implemented by Kronecker product.
  - Various algorithms for finding solutions to $f(v) = 0$ or $f(v) = b$, for eigenvalues, etc.
- Great hardware support on GPUs and TPUs, specifically for matrix multiplication.

Matrices

Representation of a linear map as a matrix.

Theorem

Let $B_V = [v_0, \dots v_{n-1}]$ and $B_W = [w_0, \dots, w_{m-1}]$ be ordered bases of V and W, respectively. Then matrix $M(f) : \mathbb{C}^{\boldsymbol{m \times n}}$ is the unique representation of linear map $f : Hom(V, W)$ wrt. bases B_V and B_W if $f(v_j) = \sum_{i=0}^{m-1}(a_{ij} \cdot w_i)$, where $a_{ij} = M(f)_{ij}$.

- Terrible representation for high-dimensional vector spaces.
- Consider $V = W = \mathcal{H}^{\otimes \boldsymbol{n}}$ where $\mathcal{H} = \mathbb{C}^{\mathbf{2}}$.
  - *Every* element of V has length 2^n.
  - *Every* linear map $f : Hom(V, V)$ has matrix size $2^{2n} = 4^n$; e.g. and the matrix representing any 20-qbit circuit contains $1,099,511,627,776$ scalars, and an input contains 1,048,576 scalars.
  - The Kronecker product is an excruciatingly slow implementation of the tensor product of two linear maps.

Quantum circuits

A quantum circuit is a diagram for denoting unitary transformations $f : \mathcal{H}^{\otimes \boldsymbol{n}} \cong \mathcal{H}^{\otimes \boldsymbol{n}}$.

Pro:

- Primitive operators can be implemented by physical quantum effects more or less efficiently depending on technology.
- Only unitary transformations are denoted.

Cons:

- Interpretation via matrices is extremely inefficient as it boils down to matrix representation.
- Only unitary transformations of type $\mathcal{H}^{\otimes \boldsymbol{n}} \cong \mathcal{H}^{\otimes \boldsymbol{n}}$ can be denoted.
- Low-level language: (q)bit-level programming with (quantum) machine code instructions.

Tensor networks

Tensor networks (in various guises) are diagrams for representing multi-linear maps.

Pros:

- Exploit isomorphism $Hom(V, W) \cong W \otimes V^*$ for finite-dimensional V, W to represent linear maps more compactly than with matrices.
- Notation subsumes useful equalities by mapping them to the same diagram, e.g. $(g_2 \otimes f_2) \circ (g_1 \otimes f_1) = (g_2 \circ g_1) \otimes (f_2 \circ f_1)$.

Cons:

- How do you turn visual diagrams into an efficient data structure?

Functional programs

Represent a linear map as a functional program in a general-purpose programming language such as the machine language of a computer: "linear map as code".

Pros:

- Maximally compact representation; e.g. (compiled version of) $id = \lambda v.v$.
- Can be very efficient to apply to (representations of) input vectors.

Cons:

- How do you make sure the function is linear?
- How do you perform other operations on linear maps than application, such as computing the adjoint?
- How do you optimize the code?

Used in high-performance deep learning: The linear map resulting from automatic differentiation (backpropagation) and its adjoint are represented as code.

Domain-specific languages

A (combinatory) *domain-specific language (DSL)* is a data type whose elements denote semantic objects with some property (such as linear maps). It has constructors that denote operations on semantic objects that are guaranteed to preserve the property.

Pros:

- Guarantees that nothing but objects with the property can be constructed.
  - May come with a theorem that every object with the property can be denoted.
  - Sometimes we don't care about that, and we are only interested in denotable semantic objects.
- Supports pattern matching on symbolic constructors for efficient evaluation, rewriting, transformation, analysis, and optimization.

Cons:

- Can be hard to read.

Domain-specific languages: Examples

- Martin's DSL for quantum circuits.
- Matrices and vectors with symbolic constructors for $\circ, \otimes, .^{\dagger}$.
  - Exploit algebraic properties at run time, e.g.
    - $(M \circ N)(v) = M(N(v))$. (Avoids multiplying M with N, which is less efficient.)
    - $(M \otimes N)(v \otimes w) = M(v) \otimes N(w)$. (Avoids computing Kronecker product and outer product, which is *much* less efficient.)
- DSL for specifying analytic functions on Hilbert spaces including a differentiation operator, with a sublanguage for linear maps including an adjoint operator, which in turn contains a sublanguage of unitary transformations.
  - Only the unitary sublanguage will be presented here.
  - For hybrid quantum-classical computing such as quantum machine learning, the full language is relevant.

Inner product space, Hilbert space

Definition

An *inner product space* over $\mathbb{C}$ is a vector space V over $\mathbb{C}$ equipped with an *inner product*

$$* : V \times V \to \mathbb{C}$$

that is

- *linear* in its second argument;
- *conjugate symmetric*, $v_1 * v_2 = \overline{v_2 * v_1}$, and
- *positive definite*, $v * v > 0$ for all $v \neq 0$, that is $v * v$ always yields a nonnegative real number that is 0 if and only if v itself is the 0-vector.

A *Hilbert space* is a metrically complete inner product space.

All our inner product spaces will be finite dimensional and Hilbert spaces.

Linear maps

Definition (Linear map)

Let V, W be Hilbert spaces over $\mathbb{C}$. A function $f : V \to W$ is a *linear map* if

$$\begin{aligned} f(v +_V w) &= f(v) +_W f(w) \\ f(a \cdot_V v) &= a \cdot_W f(v) \end{aligned}$$

It is *anti-linear* if

$$\begin{aligned} f(v +_V w) &= f(v) +_W f(w) \\ f(a \cdot_V v) &= \bar{a} \cdot_W f(v) \end{aligned}$$

It is *isometric* if $f(v *_V w) = f(v) *_W f(w)$. It is *monomorphic, epimorphic, isomorphic* if it is injective, surjective, bijective, respectively, as a function on sets.

Unitary transformations

Proposition

A linear map $f : V \to W$ is isometric if and only if $\|f(v)\|_W^2 = \|v\|_V^2$ for all $v \in V$ where $\|v\|^2 = v * v$.

In other words, $f(v) * f(w) = v * w$ already holds for all pairs v, w is it just holds for equal pairs v, v.

Definition (unitary transformation)

Let V, W be Hilbert spaces over $\mathbb{C}$. A function $f : V \to W$ is a *unitary transformation* if it is linear, isometric, and epimorphic. It is *anti-unitary* if it is anti-linear, isometric, and epimorphic.

Proposition

Every unitary and anti-unitary transformation is an isomorphism.

Atomic spaces

- The trivial Hilbert space 0 consists of a single element, which is necessarily its zero-element, 0. Its operations are forced.
- $\mathbb{C}$ is a Hilbert space with vector operations inherited from $\mathbb{C}$, and $*$ is conjugate multiplication:

$$
\begin{aligned}
k \cdot x &= k \cdot_{\mathbb{C}} x && \text{(multiplication of } K \text{ as a field)} \\
x + y &= x +_{\mathbb{C}} y && \text{(addition of } K \text{ as a field)} \\
0 &= 0_{\mathbb{C}} && \text{(the 0 of } K \text{ as a field)} \\
a *_K b &= \overline{a} \cdot_{\mathbb{C}} b
\end{aligned}
$$

Direct sum space

Definition

Let V and W be Hilbert spaces. The *(external) direct sum of V and W* is the Hilbert space $V \oplus W$ with underlying set $V \times W$ whose elements are written $v \oplus w$ and with the following operations:

$$
\begin{aligned}
0 &= 0_V \oplus 0_W \\
(v_1 \oplus w_1) + (v_2 \oplus w_2) &= (v_1 +_V v_2) \oplus (w_1 +_W w_2)) \\
a \cdot (v \oplus w) &= (a \cdot_V v) \oplus (a \cdot_W w) \\
(v_1 \oplus w_1) * (v_2 \oplus w_2) &= (v_1 *_V v_2) +_{\mathbb{C}} (v_2 *_W w_2)
\end{aligned}
$$

with antilinear/linear extension on the first/second argument of $*$.

We may also write $v \oplus w$ as a column $\begin{pmatrix} v \\ w \end{pmatrix}$.

Tensor product space

$W = U \otimes V$ is the tensor product space of U and V. Its finite elements are the formal terms generated by

$$w \quad ::= \quad 0 \mid k \cdot w \mid w_1 + w_2 \mid u \otimes v$$

where $k \in \mathbb{C}, u \in U, v \in V$ that are identified modulo the vector space axioms and the equalities

$$\begin{aligned} (k \cdot v) \otimes w = k \cdot (v \otimes w) &= v \otimes (k \cdot w) \\ (v_1 + v_2) \otimes w &= (v_1 \otimes w) + (v_2 \otimes w) \\ v \otimes (w_1 + w_2) &= (v \otimes w_1) + (v \otimes w_3). \end{aligned}$$

We write $[w]_\otimes$ for the congruence class of w. Define

$$\begin{aligned} 0_W &= [0]_\otimes \\ v_1 +_W v_2 &= [v_1 + v_2]_\otimes \\ k \cdot_W v &= [k \cdot v]_\otimes \\ (u_1 \otimes v_1) * (u_2 \otimes v_2) &= (u_1 * u_2) \cdot_{\mathbb{C}} (v_1 * v_2). \end{aligned}$$

with antilinear/linear extension on the first/second argument.

Function spaces

Every vector space W can be *lifted* to a vector space of functions from a *set* V to W by defining

$$\begin{aligned} 0(v) &= 0 \\ (f+g)(v) &= f(v)+g(v) \\ (c \cdot f)(v) &= c \cdot f(v) \end{aligned}$$

Since being a linear map is preserved by lifting we have:

Proposition

Let V be a set W a vector space. Then the set $Hom(V, W) = \{f : V \to W \mid f \text{ is a linear map}\}$ forms a vector space by lifting the vector operations on W to linear maps from V to W.

The set $V \cong W$ of unitary transformations, however, does not form a vector space, since $f + g$ of two unitary transformations is generally not a unitary transformation.

Currying

- Given a binary function $\diamond : U \times V \to W$ that is usually written in infix notation and $u \in U, v \in V$ we write
  - $(u\diamond) : V \to W$ for the function defined by $(u\diamond)(v') = u \diamond v'$;
  - $(\diamond v) : U \to W$ for the function defined by $(\diamond v)(u') = u' \diamond v$.
  - This is called *section notation*, originally introduced in the programming language Miranda, a predecessor of Haskell.
- The function $u \mapsto (u\diamond) : U \to (V \to W)$ is called the *left-curried version* of $\diamond$; the function $v \mapsto (\diamond v) : V \to (U \to W)$ is called the *right-curried version* of $\diamond$.
- Turning a binary function into left- or right-curried version is called *currying*, named after logician Haskell Curry (1900-1982).

Definition

Let U, V, W be vector spaces. $\diamond : U \times V \to W$ is *bilinear* if, for all $u \in U, v \in V$, $(u\diamond)$ and $(\diamond v)$ are linear maps.

Dual space

Let H be a Hilbert space with inner product $*_H$. Its *dual space* is the vector space of *linear forms* (or *covectors*) $H^* = Hom(H, \mathbb{C})$.

Theorem (Riesz Representation Theorem)

For every linear form $f \in H^*$ there is a unique $v \in H$ such that $f = (v*)$.

Define the operations on H^* by

$$\begin{aligned} 0 &= (0*_H) \\ (v*_H) + (w*_H) &= (v +_H w) *_H \\ a \cdot (v*_H) &= (\overline{a} \cdot_H v) *_H \\ (v*_H) * (w*_H) &= \overline{(v *_H w)} \end{aligned}$$

Corollary

The linear map $iso : Hom(H, H^*)$ defined by $iso(v) = (v*_H)$ is an anti-unitary transformation.

Point space

Let H be a Hilbert space with scalar multiplication $\cdot_H$. Its *point space* is the vector space of linear maps $\dot{}H = Hom(\mathbb{C}, H)$.

It is easy to see that every element of $\dot{}H$ is represented by $(\cdot_H v)$ for a unique $v \in H$, which permits equipping $\dot{}H$ with an inner product that makes it a Hilbert space.

The operations on $\dot{}H$ are defined by

$$\begin{aligned} 0 &= (\cdot_H 0) \\ (\cdot_H v) + (\cdot_H w) &= (\cdot_H (v +_H w)) \\ a \cdot (\cdot_H v) &= (\cdot_H (a \cdot_H v)) \\ (\cdot_H v) * (\cdot_H w) &= v *_H w \end{aligned}$$

Proposition

The linear map iso : $Hom(H, \dot{}H)$ defined by $iso(v) = (\cdot_H v)$ is a unitary transformation.

H and $\dot{}H$ are typically "identified" in mathematical discourse.

Adjoints

Definition

Let G, H be Hilbert spaces. A linear map $g : Hom(H, G)$ is a *(Hermitian) adjoint* of $f : Hom(G, H)$ if $f(v) *_W w = v *_V g(w)$ for all $v \in G, w \in H$.

Theorem

Every linear map $f : Hom(G, H)$ has a unique adjoint, denoted $f^\dagger$.

- This holds also for infinite-dimensional Hilbert spaces. It follows from the Riesz Representation Theorem (see below).
- Example: The adjoint of $+ : Hom(H \oplus H, H)$ is $dup : Hom(H, H \oplus H)$ defined by

$$dup(v) = (v \oplus v)$$

in any Hilbert space H.

Adjoints: Characterization in terms of cps

- In functional programming, the *continuation-passing style (cps)* version of a function $f : S \to T$ is the function that takes a *continuation*, a function $\kappa : T \to A$ that represents the "rest of the computation" in a program and returns values of type A, as its first argument; applies f to its second argument; and finally passes the result of that to κ.
- The cps version of $f : Hom(V, W)$, where $A = \mathbb{C}$ gives rise to $f^* : Hom(W^*, V^*)$ defined by

$$f^*(\kappa)(v) \quad = \quad \kappa(f(v)), \tag{1}$$

which can be written

$$f^*(\kappa) \quad = \quad \kappa \circ f \tag{2}$$

using functional composition.
- f^* is called the *algebraic adjoint* of f. It clearly exists and is unique for all linear maps.

Adjoints: Characterization in terms of cps

Theorem

Let G, H be a Hilbert spaces. Define $f^\dagger : Hom(H, G)$ for $f : Hom(G, H)$ as the unique linear map that satisfies

$$f^*((w*)) = (v*) \quad \Leftrightarrow \quad f^\dagger(w) = v$$

Then $f^\dagger$ is the adjoint of f.

- Intuitively, the adjoint encodes the cps version of f by representing a continuation κ by the vector w instead of $(w*)$.
- This is computationally much better than representing the continuation as (arbitrary) code implementing a function.
  - It is the basis of efficient reverse-mode automatic differentiation, the core technique in deep learning.

Bra, ket, and adjoint rules

Let $v, w \in H$. We can now define notations

$$\begin{aligned} \langle v| &= (v*) \in H^* \text{ (bra)} \\ |w\rangle &= (\cdot w) \in {}^{\cdot}H \text{ (ket)} \\ \langle v|w\rangle &= (\cdot(v * w)) \in {}^{\cdot}\mathbb{C} \text{ (braket)} \end{aligned}$$

Note: $|w\rangle : {}^{\cdot}H \cong H$.
Bras and kets are designed to be adjoints of each other:

$$\begin{aligned} (g \circ f)^{\dagger} &= f^{\dagger} \circ g^{\dagger} \\ \langle v|^{\dagger} &= |v\rangle \\ |v\rangle^{\dagger} &= \langle v| \end{aligned}$$

Algebra with bras and kets

Bra-ket notation suppresses functional composition and implicit conversions between H and $\dot{}H$ to arrive at evaluating inner products by turning two parallel lines into single one:

$$\begin{aligned}
|w\rangle\langle v||v'\rangle\langle u| &= |w\rangle \circ \langle v| \circ |v'\rangle \circ \langle u| \\
&= |w\rangle \circ (\langle v| \circ |v'\rangle) \circ \langle u| \\
&= |w\rangle \circ (\cdot(v * v')) \circ \langle u| \\
&= |w\rangle \circ \langle v|w\rangle \circ \langle u| \\
&= |w\rangle\langle v|w\rangle\langle u|
\end{aligned}$$

Note: Instead of a vector, an element x of the generating set X of a *free vector space* is often written inside bra and ket. This refers implicitly to the vector $i(x)$ (often written e_x), the canonical basis vector corresponding to x. For example, $\mathbb{C}^{\mathbf{2}}$ is generated by $\mathbf{2} = \{0, 1\}$. Writing $|1\rangle$ stands for $|i(1)\rangle$.

Basis, dimension

Definition

Let V be a vector space.

- $U \subseteq V$ is a *subspace* of V if it is closed under vector space operations of V.
- The *span* of $S \subset V$ is the smallest subspace of V containing S.
- $B \subseteq V$ is a *basis* of V if it spans V and no proper subset of B is a basis.
- V is *finite-dimensional* if it has a finite basis.

Theorem

Every vector space has a basis. All bases have the same cardinality.

Definition

The *dimension* of V, $\dim V$, is the cardinality of a basis of V.

Orthogonal and orthonormal bases

Definition

Let H be a Hilbert space. A basis $B = \{v_1, \ldots, v_n\}$ of H is *orthogonal* if $v_i * v_j = 0 \Leftrightarrow i = j$. It is *orthonormal* if, additionally, $v_i * v_i = 1$ for all i.

Three orthonormal bases of $\mathbb{C}^{\mathbf{2}}$ used in quantum computing, called X-, Y- and Z-basis, respectively:

$$
\begin{array}{|rcl|rcl|}
\hline
|+\rangle & = & \frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) & |-\rangle & = & \frac{1}{\sqrt{2}} \cdot (|0\rangle - |1\rangle) \\
|i\rangle & = & \frac{1}{\sqrt{2}} \cdot (|0\rangle + i \cdot |1\rangle) & |-i\rangle & = & \frac{1}{\sqrt{2}} \cdot (|0\rangle - i \cdot |1\rangle) \\
|0\rangle & = & (1 \cdot |0\rangle + 0 \cdot |1\rangle) & |1\rangle & = & (0 \cdot |0\rangle + 1 \cdot |1\rangle) \\
\hline
\end{array}
$$

The Z-basis is the standard basis of $\mathbb{C}^{\mathbf{2}}$, also called the *computational basis*.

Orthonormal bases

Proposition

- *$\emptyset$ is the (only) basis of 0; it is orthonormal.*
- *B is an orthonormal basis of $\mathbb{C}$ if and only if $B = \{e^{i\rho}\}$ for some $\rho \in \mathbb{R}$. In particular, $\{1\}$ and $\{-1\}$ are the (only) real orthonormal bases.*
- *If B_1, B_2, B are orthonormal bases of H_1, H_2, H, respectively, then*
  - *the disjoint union $\{(b_1 \oplus 0) \mid b_1 \in B_1\} \cup \{(0 \oplus b_2) \mid b_2 \in B_2\}$ is an orthonormal basis of $H_1 \oplus H_2$;*
  - *the Cartesian product $\{b_1 \otimes b_2 \mid b_1 \in B_1, b_2 \in B_2\}$ is an orthonormal basis of $H_1 \otimes H_2$;*
  - *the copy $\{(b*) \mid b \in B\}$ is an orthonormal basis of H^*;*
  - *the copy $\{(\cdot b) \mid b \in B\}$ is an orthonormal basis of ${}^{\cdot}H$.*

Ket-bra decomposition

- Let H a Hilbert space. Let $v \in H$. Recall:

$$\begin{array}{ccccc} |v\rangle & = & (\cdot v) & \in & {}^{\cdot}H \\ \langle v| & = & (v*) & \in & H^* \end{array}$$

Theorem

For finite-dimensional Hilbert spaces G, H every linear map $f : Hom(G, H)$ can be decomposed into a sum of ket-bra's:

$$f = \sum_{i=0}^{n-1} |w_i\rangle \langle v_i|$$

for some $n \geq 0$ and $v_i \in G, w_i \in H$.

Note that $\sum_{i=0}^{n-1} |w_i\rangle \langle v_i|$ is short-hand for $\sum_{i=0}^{n-1}(|w_i\rangle \circ \langle v_i|)$.

Tensor representation of linear maps

- Recall: $f = \sum_i |w_i\rangle\langle v_i|$ where $v_i \in G, w_i \in H$.
- This "looks" like $\sum_i (w_i \otimes (v_i *)) \in H \otimes G^*$.
- Indeed it corresponds to it!

Definition

Let G, H be Hilbert spaces. Define *tensor application* $@ : (H \otimes G^*) \times G \to H$ by $(w \otimes (v*))@v' = (v * v') \cdot w$.

Proposition

Tensor application is bilinear.

Theorem

Let G, H be finite-dimensional Hilbert spaces. Tensor application $(@) : Hom(H \otimes G^*, Hom(G, H))$ is an isomorphism.

Tensor representation of linear maps

Theorem

Tensor application $(@) : Hom(H \otimes G^*, Hom(G, H))$ *is an isomorphism.*

This theorem makes it possible to equip linear maps on finite-dimensional Hilbert spaces with an inner product and thus turn them into Hilbert spaces themselves.

Definition

Let G, H be finite dimensional Hilbert spaces. Given $f, g \in Hom(G, H)$, let $t_f, t_g \in W \otimes V^*$ be such that $f = (t_f @)$ and $g = (t_g @)$. Define the inner product on $Hom(G, H)$ by

$$f * g = t_f * t_g .$$

Tensor contraction

- Let G, H be Hilbert spaces.
- We have $@ : Hom(G, H) \cong (H \otimes G^*)$; that is, we can represent all elements of $Hom(G, H)$ faithfully by corresponding elements of $H \otimes G^*$.
- What corresponds to function composition of linear maps?

Definition

Let F, G, H be Hilbert spaces. Define *tensor contraction*

$$\star : (H \otimes G^*) \times (G \otimes F^*) \to (H \otimes F^*)$$

by

$$(w \otimes (v*)) \star (v' \otimes (u*)) = w \otimes (v * v') \cdot (u*).$$

Natural unitary isomorphisms

- Let T, U, V, W be Hilbert spaces.
- Let $t \in T, u \in U, v \in V, w \in W, f : T \cong U, g : V \cong W$.
- We have the following *natural unitary transformations*:

$$
\begin{array}{rcl}
id & : & V \cong V \\
g \circ f & : & T \cong W \text{ if } U = V \\
f \oplus g & : & T \oplus V \cong U \oplus W \\
f \otimes g & : & T \otimes V \cong U \otimes W \\
absorb_{\oplus} & : & U \oplus 0 \cong U \\
swap & : & U \oplus V \cong V \oplus U \\
assoc_{\oplus} & : & U \oplus (V \oplus W) \cong (U \oplus V) \oplus W \\
absorb_{\otimes} & : & \mathbb{C} \otimes U \cong U \\
transp & : & U \otimes V \cong V \otimes U \\
assoc_{\otimes} & : & U \otimes (V \otimes W) \cong (U \otimes V) \otimes W \\
distl & : & (U \oplus V) \otimes W \cong (U \otimes W) \oplus (V \otimes W) \\
distr & : & U \otimes (V \oplus W) \cong (U \otimes V) \oplus (U \otimes W)
\end{array}
$$

Natural unitary isomorphisms

- They are defined by these *characteristic properties*:

$$
\begin{array}{rcl}
id(v) & = & v \\
(g \circ f)(t) & = & g(f(t)) \\
(f \oplus g)(t \oplus v) & = & f(t) \oplus g(v) \\
(f \otimes g)(t \otimes v) & = & f(t) \otimes g(v) \\
absorb_{\oplus}(u \oplus 0) & = & u \\
swap(u \oplus v) & = & v \oplus u \\
assoc_{\oplus}((u \oplus v) \oplus w) & = & u \oplus (v \oplus w) \\
absorb_{\otimes}(k \otimes u) & = & k \cdot u \\
transp(u \otimes v) & = & v \otimes u \\
assoc_{\otimes}((u \otimes v) \otimes w) & = & u \otimes (v \otimes w) \\
distl((u \oplus v) \otimes w) & = & (u \otimes w) \oplus (v \otimes w) \\
distr(u \otimes (v \oplus w)) & = & (u \otimes v) \oplus (u \otimes w)
\end{array}
$$

Quantum gates

- A *quantum operator* is a unitary transformation $f : \mathcal{H}^{\otimes n} \cong \mathcal{H}^{\otimes n}$, where $\mathcal{H} = \mathbb{C}^{\mathbf{2}}$ and $H^{\otimes n} = \overbrace{H \otimes \ldots \otimes H}^{n}$.
- Standard *quantum gates* and their semantics as quantum operators, represented as matrices over the *computational basis* $\{i(0), i(1)\} = \{e_0, e_1\} = \{(1,0), (0,1)\}$ of $\mathcal{H} = \mathbb{C}^{\mathbf{2}}$:

$$
\begin{array}{rclrcl}
I & = & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & X & = & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\
Y & = & \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & Z & = & \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\
S & = & \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} & T & = & \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \\
P(\rho) & = & \begin{bmatrix} 1 & 0 \\ 0 & e^{i\rho} \end{bmatrix} & H & = & \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
\end{array}
$$

Combinatory definition of quantum gates

The semantics of quantum gates as quantum operators can be defined in combinatory form using our natural unitary transformations. Let $\phi \in \mathbb{R}$, $h : W \cong W$.

$$
\begin{array}{rcl}
H & = & (|+\rangle \circ \langle 0|) + (|-\rangle \circ \langle 1|) \\
I & = & id \\
X & = & swap \\
Y & = & ((i\cdot) \oplus (-i\cdot)) \circ swap \\
Z & = & id \oplus (-1\cdot) \\
P(\phi) & = & id \oplus (e^{i\phi}\cdot) \\
Z & = & P(\pi) \\
S & = & P(\frac{\pi}{2}) \\
T & = & P(\frac{\pi}{4}) \\
SWAP & = & transp \\
C(h) & = & distl^{-1} \circ (id \oplus (id \otimes h)) \circ distl \\
CNOT & = & C(X) \\
CZ & = & C(Z) \\
CC(h) & = & C(C(h)) \\
CCNOT & = & C(CNOT)
\end{array}
$$

DSL representations

- Representing linear maps from $Hom(G, H)$ as elements of $H \otimes G^*$: Leads to using constructors (symbolic operators) for $\otimes$ and a few other operations, which avoids premature and unnecessarily costly evaluation to a normal form such as a matrix.
- Idea: Employ even more constructors, and evaluate them only when forced to do so by an evaluation context.
- Recipe:
  - Define $LinMap(G, H)$ to be a *shallow embedded domain-specific language* in a suitable general-purpose programming language for representing linear maps from G to H as a data type with constructors for $\otimes, \oplus, \circ, id, +, \cdot$, the natural unitary transformations above and more.
  - Program en *evaluation function*

$$eval : LinMap(G, H) \to LinMap(\mathbb{C}, G) \to LinMap(\mathbb{C}, H)$$

    that computes the result of applying the denoted linear map to a (representation of) an input efficiently by exploiting algebraic equalities at run time.

Linear maps and their representations reconsidered

- Let G, H be n-dimensional, respectively m-dimensional Hilbert spaces over $\mathbb{C}$.
- We have seen multiple mathematical representations of linear maps $Hom(G, H)$, which give rise to different data structures.

Matrices	$\mathbb{C}^{\boldsymbol{m} \times \boldsymbol{n}}$	$\bullet$ (matrix multiplication)
Tensors	$H \otimes G^*$	$\star$ (tensor contraction)
DSL	$LinMap(G, H)$	$\circ$ (symbolic composition)

- They are characterized by an increasing use of *constructors* for operations, notably for $\otimes, \circ, +, \cdot, 0, id$, which are evaluated only when forced by evaluation, using efficient algebraic rewriting.
  - For linear maps on vector spaces of small dimension, say < 1000, it is likely that representing them as matrices is beneficial due to eminent hardware support for matrix multiplication on modern hardware.

Summary

- Build Hilbert spaces from atomic spaces, direct sums and tensor product.
- Use unnormalized symbolic terms to represent vectors and linear maps; in particular, do not multiply out tensor products.
- Design language of combinators for expressing (subclasses of) linear maps, in particular natural unitary isomorphismsms between Hilbert spaces.
- Close language under sequential (functional) composition and parallel (both tensor and direct sum) combinators.
- Exploit algebraic properties of combinators for efficient evaluation.