Verification of Quantum Computing

and the Bloch sphere representation

Chris Henson

Drexel University

Current Research/Motivation

- proof assistants are programming languages that allow formal proofs of mathematical theorems
- quantum computing is a great candidate for formal verification: it is both conceptually unintuitive and practically expensive
- algorithms such as Grover, Shor, etc. have been formally verified in languages such as Coq, Isabelle/HOL, etc.
- methods for optimizing general quantum circuits can also be verified
- some formalization frameworks provide the ability to export to formats such as OpenQASM and provide bindings in more accessible languages like Python

The Bloch Sphere

- Why do we expect any connection between qubits and a sphere?
- In a sense the relationship between a qubit and a complex number is the same as complex and real numbers

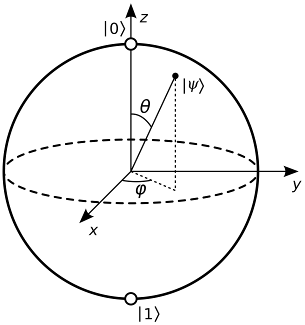

Figure: Bloch sphere representation of a Qubit

Groups

Definition (Groups)

A group is a set G with operation $\cdot$, such that:

- $\forall a, b \in G : a \cdot b \in G$
- $\forall a, b, c \in G : (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- $\forall a \in G, \exists \mathbb{1} \in G : a \cdot \mathbb{1} = a = \mathbb{1} \cdot a$
- $\forall a \in G, \exists a^{-1} \in G : a \cdot a^{-1} = \mathbb{1}$

Definition (Group Homomorphism)

Given two groups $(G, \cdot)$ and $(H, \times)$, a function $f : G \to H$ is a homomorphism if $\forall g_1, g_2 \in G : f(g_1 \cdot g_2) = f(g_1) \times f(g_2)$

Definition (Group Isomorphism)

A homomorphism $f : G \to H$ that is bijective (one-to-one and onto) is called an isomorphism

Groups

Definition (Special Unitary Group)

SU(2) is the group of all 2×2 complex matrices that are unitary ($A^\dagger A = I_2$) and have determinant 1. It can be parameterized by

$$A = \begin{pmatrix} \alpha & \beta \\ -\overline{\beta} & \overline{\alpha} \end{pmatrix}, \; |\alpha|^2 + |\beta|^2 = 1$$

Definition (Special Orthogonal Group)

SO(3) is the group of all 3×3 real matrices that are orthogonal ($A^\top A = I_3$) and have determinant 1.

Quaternions

Definition (Quaternions)

A quaternion may be desribed by the expression

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} \quad a, b, c, d \in \mathbb{R}$$

where the basis quaternions 1, **i**, **j**, **k** additionally satisfy

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1,$$
$$\mathbf{i}\,\mathbf{j} = -\mathbf{j}\,\mathbf{i} = \mathbf{k}, \qquad \mathbf{j}\,\mathbf{k} = -\mathbf{k}\,\mathbf{j} = \mathbf{i}, \qquad \mathbf{k}\,\mathbf{i} = -\mathbf{i}\,\mathbf{k} = \mathbf{j}.$$

and have 1 as a left and right identity.

Quaternions

- We can also view quaternions as pairs of complex numbers $(a + bi,\ c + di) \equiv a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$
- This idea of repeatedly taking a product of an algebra with itself is refered to as the *Cayley–Dickson construction*, where at each step we loose some property of the algebra
- Another way of viewing the quaternions that connects with quantum computing is through an isomorphism to the Pauli matrices. One such isomorphism is given by:

$$\mathbf{1} \mapsto I_2, \quad \mathbf{i} \mapsto i\,Z\,, \quad \mathbf{j} \mapsto i\,Y\,, \quad \mathbf{k} \mapsto i\,X$$

- By identifying the *pure imaginary* quaternions ($a = 0$) with $\mathbb{R}^3$, we can model rotations in a numerically stable way
- This is the intuition behind the Bloch sphere, that both qubits and quaternions can be represented by pairs of complex numbers

$\mathrm{SU}(2) \cong \mathbb{S}^3$

More formally, we have that the groups SU(2) and $\mathbb{S}^3$ of unit quaternions are isomorphic. The isomorphism can be written quite neatly:

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} \to \begin{pmatrix} a+bi & c+di \\ -c+di & a-bi \end{pmatrix}$$

$\mathbb{S}^3 \to \mathrm{SO}(3)$

Definition (Conjugation Map)

Consider a three dimensional vector represented as a pure imaginary quaternion $r \in \mathbb{R}^3 \cong \mathbb{R}\mathbf{i} + \mathbb{R}\,\mathbf{j} + \mathbb{R}\,\mathbf{k}$ and a unit quaternion $q \in \mathbb{S}^3$.

The conjugation of r is given by the map $\mathbb{R}^3 \to \mathbb{R}^3$ given by

$$r \to qrq^{-1}$$

and can be shown to be a rotation. Additionally we have that $qrq^{-1} = (-q)r(-q)^{-1}$

$\mathbb{S}^3 \to SO(3)$

For a unit quaternion $a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$, the homomorphism can also be expressed directly as:

$$\begin{pmatrix} 1-2c^2-2d^2 & 2bc-2da & 2bd+2ca \\ 2bc+2da & 1-2b^2-2d^2 & 2cd-2ba \\ 2bd-2ca & 2cd+2ba & 1-2b^2-2c^2 \end{pmatrix}$$

This was know as early as Euler:

PROBLEMATIS INITIO PROPOSITI SOLUTIO GENERALIS IN NUMERIS RATIONALIBUS

33. Coronidis loco solutionem problematis nostri e methodo DIOPHANTEA petitam subiungam, quae sequenti modo satis concinne exhiberi potest.

Sumantur pro lubitu quatuor numeri p, q, r, s ac posita quadratorum eorum summa

$$pp + qq + rr + ss = u$$

novem numeri quaesiti ita determinati reperiuntur[1])

$$A = \frac{pp+qq-rr-ss}{u}, \quad B = \frac{2qr+2ps}{u}, \quad C = \frac{2qs-2pr}{u},$$

$$D = \frac{2qr-2ps}{u}, \quad E = \frac{pp-qq+rr-ss}{u}, \quad F = \frac{2pq+2rs}{u},$$

$$G = \frac{2qs+2pr}{u}, \quad H = \frac{2rs-2pq}{u}, \quad I = \frac{pp-qq-rr+ss}{u}.$$

FIGURE: Euler's representation of rotations

Representing Groups

- groups are represented as a *typeclass*
- essentially a typeclass is a dependently-typed record, with some extra type inference capabilities
- a subtle point is that this allows *different equivalence relations*

```
Class Group := {
        id :  G
      ; inverse: G → G
      ; rel_equiv: Equivalence Grel
      ; id_left: forall x: G, (id • x) •= x
      ; id_right: forall x: G, (x • id) •= x
      ; assoc: forall x y z:  G, (x • y) • z •= x • (y • z)
      ; right_inv: forall x: G, x • (inverse x) •= id
}.
```

Representing Morphisms

Likewise, homomorphism and isomorphism can be represented with typeclasses

```
Class GroupHomomorphism G H
  (Gop: G → G → G)
  (Hop: H → H → H)
  (Grel: relation G)
  (Hrel: relation H)
  (hom_f: G → H)
: Type
:= {
    hom_left_group: Group G Gop Grel
  ; hom_right_group: Group H Hop Hrel
  ; hom_mul {a1 a2}: Hrel
                         (hom_f (Gop a1 a2))
                         (Hop (hom_f a1) (hom_f a2))
}.
```

Representing Morphisms

Another subtle point about having multiple equivalence relations is that we must prove they are “compatible”

```
Section hom_trans.
Variables A B C : Type.

Variable Arel: relation A.
Variable Brel: relation B.
Variable Crel: relation C.

Variable A_Eq: Equivalence Arel.
Variable B_Eq: Equivalence Brel.
Variable C_Eq: Equivalence Crel.

Variable AtoB: A → B.
Variable BtoC: B → C.

Variables Aop: A → A → A.
Variables Bop: B → B → B.
Variables Cop: C → C → C.

Variable BtoC_rw  :  Proper (Brel ==> Crel) BtoC.

Lemma GroupHomomorphism_trans:
  GroupHomomorphism A B Aop Bop Arel Brel AtoB →
  GroupHomomorphism B C Bop Cop Brel Crel BtoC →
  GroupHomomorphism A C Aop Cop Arel Crel (fun a ⇒ BtoC (AtoB a)).
Proof.
...
```

Representing Matrices, complex numbers, and quaternions

- we can represent groups abstractly, but it somewhat interesting to directly work with the underlying types
- complex numbers and quaternions are represented as pairs of real numbers
- a matrix is represented as a function that takes two natural numbers and returns a complex number

```
Definition C := (R * R)%type

Definition Quaternion := (R * R * R * R)%type.

Definition Matrix (m n : nat) := nat → nat → C.
```

Subset Types

- *Subset types* carry both a value and a proof that the values satisfies some predicate
- any function involving these types also carries a proof of closure

```
Definition Versor := { q | Qnorm q = 1}.

Definition SU2 := {
   U: Matrix 2 2
   | WF_Matrix U ∧
    (U 0 0 = Cconj (U 1 1) ∧ U 0 1 = − Cconj (U 1 0) ) ∧
    d2_det U = C1 }.

Definition SO3 := {
   U: Matrix 3 3
   | WF_Matrix U ∧ Real_matrix U ∧
     U × (U) ⊤ ≡  I 3 ∧ (U) ⊤ ×  U ≡ I 3 ∧
     d3_det U = C1 }.
```

Subset Types

```
Definition Versor_to_SU2 (v: Versor): SU2.
Proof.
  destruct v as [(((x, y), z), w) E].
  unfold SU2.
  exists (
    fun row ⇒ fun col ⇒
    match row, col with
    | 0, 0 ⇒ (x, y)
    | 0, 1 ⇒ (z, w)
    | 1, 0 ⇒ (Ropp z, w)
    | 1, 1 ⇒ (x, Ropp y)
    | _, _ ⇒ C0
    end
  ).
  unfold Qnorm in E.
  apply pow_eq with (n := 2%nat) in E.
  rewrite pow2_sqrt in E.
  replace ((1^2)%R) with 1 in E by lra.
  repeat split.
  — show_wf.
  — lca.
  — lca.
  — unfold d2_det, Cmult, Cminus, Cplus, C1.
     simpl. f_equal.
     rewrite ← E.
     all: lra.
  — repeat apply Rplus_le_le_0_compat.
     all: apply pow2_ge_0.
Defined.
```

Subset Types

- I define equality of these subset types as equality of their values (first projection)
- this is general, as this also takes an equivalence relation

```
Definition sigma_proj1_rel
  {X: Type}
  {A: X → Prop}
  {rel: relation X}
  (e: Equivalence rel)
  (s1 s2: sig A)
  : Prop
:= rel (proj1_sig s1) (proj1_sig s2).
```

$\mathrm{SU}(2) \to \mathrm{SO}(3)$


```
Theorem SU2_Homomorphism_SO3:
  GroupHomomorphism
    SU2
    SO3
    SU2_mul
    SO3_mul
    SU2_equiv
    SO3_equiv
    (fun U ⇒ Versor_to_SO3 (SU2_to_Versor U)).
Proof.
  apply GroupHomomorphism_trans with versor_equiv Vmul.
  — apply (sigma_proj1_rel_equivalence eq_equivalence).
  — apply (sigma_proj1_rel_equivalence mat_equiv_equivalence).
  — unfold Morphisms.Proper, Morphisms.respectful.
    intros.
    unfold versor_equiv, SO3_equiv, sigma_proj1_rel, proj1_sig in *.
    destruct x as [(((a1, b1), c1), d1) E1].
    destruct y as [(((a2, b2), c2), d2) E2].
    by_cell; inversion H; subst; reflexivity.
  — apply SU2_Iso_to_hom.
  — apply Versor_Homomorphism_SO3.
Qed.
```

Questions?

References I

J.H. Gallier and J. Quaintance, *Linear algebra and optimization with applications to machine learning*, no. v. 1, World Scientific Publishing Company Pte. Limited, 2020.

Kesha Hietala, Robert Rand, Shih-Han Hung, Liyi Li, and Michael Hicks, *Proving Quantum Programs Correct*, 12th International Conference on Interactive Theorem Proving (ITP 2021) (Dagstuhl, Germany) (Liron Cohen and Cezary Kaliszyk, eds.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 193, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, June 2021, pp. 21:1–21:19.

Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu, and Michael Hicks, *A verified optimizer for Quantum circuits*, Proceedings of the ACM on Programming Languages **5** (2021), no. POPL, 37, POPL '21.

References II

Robert Rand, *Verified Quantum Computing*, https://rand.cs.uchicago.edu/vqc/index.html, 2019.

Alistair Savage, *Introduction to Lie Groups*, https://alistairsavage.ca/mat4144/notes/MAT4144-5158-LieGroups.pdf, 2015.

Who discovered the covering homomorphism between su(2) and so(3)?, History of Science and Mathematics Stack Exchange https://hsm.stackexchange.com/questions/11057/who-discovered-the-covering-homomorphism-between-su2, 2019.

James Wells, *Lectures on Standard Model Particle Physics*, https://indico.cern.ch/event/243629/, 2013.