Encoding Primes in Quantum Superposition: A Type-Theoretic and Quantum Approach to Twin Primes and the Riemann Hypothesis

Matthew Long
Magneton Labs

March 8, 2025

Abstract

We explore recent advancements in quantum computing and type theory that enable encoding prime numbers in a quantum superposition. We review how representing primes as quantum states ("Prime states") can reveal number-theoretic patterns — including twin primes — and discuss implications for the Riemann Hypothesis. We then examine how modern type systems and functional programming (with Haskell as a case study) can model these concepts, leveraging strong typing to represent primes and quantum states. Finally, we outline mathematical frameworks (analytic and computational) that support this approach, and consider avenues toward constructing a proof or verification for long-standing conjectures. This work synthesizes quantum algorithmic insights, type-theoretic modeling, and analytic number theory to outline a novel interdisciplinary pathway toward understanding prime distributions and the Riemann Hypothesis.

1 Introduction

Prime numbers have long been a focal point linking number theory, computer science, and now quantum computing. The Riemann Hypothesis (RH) — which asserts that all nontrivial zeros of the Riemann zeta function have real part $\frac{1}{2}$ — underpins our understanding of prime distribution. A proof of RH would rigorously bound the error term in the prime counting function $\pi(x)$ (the number of primes $\leq x$) to on the order of $\sqrt{x}\log x$, thus tightening the connection between primes and the zeros of $\zeta(s)$. At the same time, twin primes (primes p and $p+2$) represent a mysterious subset of primes; the Twin Prime Conjecture posits infinitely many such pairs, but it remains unproven despite recent breakthroughs on bounded prime gaps. While RH governs the global distribution of primes, it is believed to offer little direct help on specific patterns like twin primes. This underscores the need for new perspectives that bridge different domains of mathematics and computation.

Quantum computing offers a novel perspective by allowing superposition of states, potentially enabling us to represent and manipulate many primes simultaneously. Recent research has shown that one can construct a quantum Prime state — a superposition of all primes below a bound — using quantum circuits. Such a state, remarkably, encodes number-theoretic information in its quantum entanglement. This opens the door to "quantum experiments" on primes: for example, measuring certain properties of an entangled prime superposition can statistically reveal the distribution of primes or twin primes. In parallel, advancements in type systems and functional

programming provide powerful tools to model mathematical structures in software. Strongly-typed functional languages like Haskell allow us to represent mathematical invariants (e.g. primality) at compile time, and even simulate aspects of quantum algorithms at a high level. By leveraging Haskell's type system and purity, we can create models that mirror quantum states of primes or enforce primality as a type constraint, offering a sandbox to test theoretical ideas safely.

In this paper, we review the state-of-the-art in these areas and propose a framework combining them to approach deep conjectures. Section 2 surveys related work: recent quantum algorithms involving primes, type-theoretic representations of mathematical concepts, and known connections between quantum physics and the Riemann Hypothesis. Section 3 discusses how advanced type systems can represent primes in superposition, outlining how Haskell's type and functional features can encode prime numbers and even simulate quantum superposition behavior. In Section 4, we examine the potential links between twin prime distribution, quantum superposition of primes, and the Riemann Hypothesis — highlighting how twin primes might manifest in a quantum context and whether this could shed light on RH. Section 5 provides a Haskell-based implementation example, illustrating both a classical simulation of a prime superposition state and the use of Haskell's type system to enforce properties like primality. Section 6 outlines mathematical formulations and tools (quantum-mechanical models of zeta, type-theoretic proof systems, etc.) that could support this interdisciplinary approach, and how they might be leveraged toward a proof or verification of RH or the Twin Prime Conjecture. We conclude with insights on how combining quantum computing, type theory, and classical number theory could pave new paths toward solving the Riemann Hypothesis and related prime conjectures.

2 Related Work and Recent Advances

Quantum Algorithms for Primes and Prime Counting

Quantum computing's relevance to prime numbers is well established by Shor's factoring algorithm, but more recent work goes further by treating primes themselves as quantum data. Latorre and Sierra (2014) introduced a quantum algorithm to construct a Prime state — an equal superposition of all primes below 2^n (using an n-qubit register) [1]. The construction uses Grover's search with a quantum oracle for primality (based on the Miller–Rabin test) to project the quantum register onto prime numbers. This Prime state is highly entangled, and intriguingly, measures of its entanglement correspond to number-theoretic functions like the distribution of twin primes and Chebyshev's bias. By augmenting this state preparation with the Quantum Fourier Transform (QFT), Latorre and Sierra showed one could estimate the prime counting function $\pi(x)$ faster than classically and "with an error below the bound that allows for the verification of the Riemann hypothesis". In essence, a sufficiently large quantum computer could compute $\pi(x)$ for huge x to check if the results deviate from the prediction of RH (any deviation beyond $O(x^{1/2}\log x)$ would falsify RH). A popular science article by Yirka [2] notes that their algorithm, if realized on ~ 80 qubits, could count primes up to $\sim 10^{25}$ — far beyond classical reach — potentially testing Riemann's prime formula at unprecedented scales. While such an experiment cannot prove RH, it showcases the synergy of quantum computing and analytic number theory in exploring RH.

Building on the Prime state idea, researchers also defined a Twin Prime state to specifically probe twin primes. This involves creating a superposition of primes, adding 2 to each (within a quantum circuit), and then checking again for primality [1]. The result is a quantum state in which

an amplitude is nonzero only for those primes p where $p+2$ is also prime. Measuring this state's ancilla qubit yields the probability of observing a "twin prime pair," which is essentially the ratio of twin primes to all primes below the bound:

$$\Pr[(p, p+2) \text{ both prime}] = \frac{\pi_2(2^n)}{\pi(2^n)}.$$

This probability is small — asymptotically on the order of $1/\ln^2 x$ — and preparing the twin prime state is proportionally harder (the success probability scales as $1/n^2$ for a 2^n range). While not offering a computational speedup over classical methods for twin primes, it is conceptually notable that quantum states can directly encode twin prime statistics. The authors also outlined a quantum state to test Goldbach's conjecture, superposing all even numbers as sums of two primes. These ideas suggest that many arithmetic conjectures could, in principle, be transformed into experiments on quantum states, providing statistical evidence or counter-evidence within quantum limits.

Connections to the Riemann Hypothesis

Beyond these specific algorithms, there is a broader body of work connecting quantum mechanics to the Riemann Hypothesis. The decades-old Hilbert–Pólya conjecture posits that the nontrivial zeros of $\zeta(s)$ correspond to eigenvalues of an unbounded self-adjoint operator (a Hamiltonian). If such a quantum Hamiltonian is found, RH would follow as a consequence of its spectral properties [4]. This idea motivated numerous attempts to model the zeta zeros as a quantum spectrum. For example, Berry and Keating (1999) conjectured a specific semiclassical Hamiltonian

$$H = \frac{1}{2}(xp + px),$$

whose quantization might yield the imaginary parts of Riemann zeros as energy levels. More rigorous developments by Sierra and others have "quantized" various classical models to produce spectra matching known zeros, and even proposed an experimental setup (a Riemann zeros interferometer) to observe the spectrum of the zeta function. Another vein of research uses random matrix theory: Montgomery's pair correlation conjecture (1973) observed that the statistical distribution of spacing between nontrivial zeros of $\zeta(s)$ matches the eigenvalue spacing in random Hermitian matrices. Recent work by McGuigan (2023) applied actual quantum algorithms (QFT on a 6-qubit device) to study toy zeta-like functions that obey analogues of RH, treating these functions as quantum ground-state wavefunctions and exploring connections to supersymmetric quantum mechanics and matrix models.

Advances in Twin Primes and Prime Gaps

On the number theory side, the 21st century has seen major progress on prime gaps. In 2013, Yitang Zhang stunned the mathematical community by proving that infinitely many primes exist with a gap $\leq 70\,000\,000$ [**?**]. Subsequent improvements by Maynard and the Polymath project rapidly shrank the bound; as of 2014, it is known that there are infinitely many prime pairs with gap at most 246. However, the gap of 2 for actual twin primes remains unbridged. These breakthroughs, achieved with combinatorial sieves and deep analysis (not assuming RH), underscore how elusive twin primes are. The distribution of twin primes is conjecturally governed by the Hardy–Littlewood heuristic:

$$\pi_2(x) \sim 2C_2 \frac{x}{(\ln x)^2},$$

with $C_2 \approx 0.66016\ldots$ being the twin prime constant. This resembles the form of the prime density $\pi(x) \sim x/\ln x$ but with an extra $\ln x$ in the denominator. Even assuming RH, one only gets relatively coarse bounds on prime gaps (on RH one can show $p_{n+1} - p_n = O(\sqrt{p_n}\log p_n)$); this is far from enough to guarantee infinitely many gap-2 occurrences. As noted by Goldston and others, "while the Riemann Hypothesis is decisive in determining the distribution of primes, it seems to be of little help with regard to twin primes." Nonetheless, any new structural insight into primes (quantum or otherwise) is valuable for both problems. For example, a recent (unrefereed) preprint by Omran (2024) claims a direct connection between twin primes and the zeta function, attempting to prove the Twin Prime Conjecture via an analytic contradiction argument involving the divergence of $\zeta(1)$.

Type Theory and Formal Methods

In parallel, the rise of formal verification and type-theoretic proof assistants (Coq, Lean, Agda, etc.) has enabled rigorous machine-checked proofs of deep mathematical theorems. Although RH and twin primes remain open, simpler results like the Prime Number Theorem have been formalized. Haskell, while not a full proof assistant, leverages advanced type features such as Generalized Algebraic Data Types (GADTs), Type Families, Data Kinds, and Liquid Haskell refinement types. These tools allow one to encode properties (such as the primality of a number) directly in the type system, effectively having the compiler certify certain mathematical invariants. This synthesis of computational verification and type theory paves the way for "executable mathematics."

Furthermore, Haskell's functional paradigm and its extensions toward dependent types (e.g., in Idris or Dependent Haskell proposals) allow us to represent notions like a type `Prime n` that is inhabited only if n is prime. While current implementations are limited, the principle is clear: the type system can encode a superposition-like structure of all primes, and functions that depend on such types will be statically checked for correctness.

Haskell for Quantum Modeling

Haskell is also an excellent platform for quantum programming. Its pure functional semantics and immutable data structures align well with quantum mechanics' linear transformations and no-cloning rule. Notably, Haskell has been used to embed quantum circuit DSLs, such as Quipper [10], which allow high-level quantum program description and translation to quantum gate circuits. This makes Haskell an attractive choice to prototype algorithms like Prime state preparation, wherein one might implement a quantum oracle for primality and Grover's algorithm to simulate the creation of a prime superposition state.

3 Type Systems for Representing Prime Superpositions

In classical computing, a superposition of states has no direct analog — it is a quantum mechanical notion where a system can be in a linear combination of basis states simultaneously. However, we can model a superposition in a type-theoretic or computational sense as a collection of possibilities. In a programming language, this might be a list of values or a nondeterministic choice; in a type system, it could be a union type or an indexed family of types.

Consider how to represent "a number that is prime" in a type system. In a dependently-typed setting (such as Coq or Agda), one defines a dependent type `Prime(n)` that has inhabitants only if

a proof of "n is prime" is provided. Such a dependent type effectively encodes the set of all primes as a type — in other words, a type of prime numbers. Any element of this type is by definition a prime number. If we ignore the proof content, the type `Prime` is an index over the naturals that picks out primes. This has the flavor of superposition: an inhabitant of `Prime` could be any prime — it is not determined until we supply a specific proof or value.

In Haskell (which is not fully dependently typed), we can leverage type-level computation using Data Kinds and Type Families. The code snippet below (adapted from public examples) illustrates a compile-time sieve that produces a type-level list of primes up to n:

```
{-# LANGUAGE DataKinds, TypeFamilies, TypeOperators, UndecidableInstances
    #-}

type family Numbers n where
  Numbers 0 = '[]
  Numbers 1 = '[]
  Numbers n = Numbers' (n + 1) 2

type family Numbers' n i where
  Numbers' n n = '[]
  Numbers' n i = i ': Numbers' n (i + 1)

type family FilterNonDiv x xs where
  FilterNonDiv x '[] = '[]
  FilterNonDiv x (y ': ys) = FilterNonDiv' (Mod y x == 0) x ys y

type family FilterNonDiv' isDiv x ys y where
  FilterNonDiv' True  x ys y = FilterNonDiv x ys       -- drop y if
      divisible by x
  FilterNonDiv' False x ys y = y ': FilterNonDiv x ys    -- keep y if not
      divisible

type family Sieve xs where
  Sieve '[]       = '[]
  Sieve (p ': xs) = p ': Sieve (FilterNonDiv p xs)

type family PrimesUpTo n where
  PrimesUpTo n = Sieve (Numbers n)

-- Example usage:
type PrimesUnder100 = PrimesUpTo 100
```

In this code, `PrimesUpTo 100` reduces, at compile time, to the type-level list of primes under 100. For example, in GHCi, the command `:kind! PrimesUnder100` would yield

```
PrimesUnder100 ::  [Nat] = '[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,8
```

This means the compiler has effectively *proven* the primality of those numbers by constructing the list. One could further define a type class `Prime n` with instances for each prime; the compiler's instance resolution then serves as a proof of primality for small n. For larger numbers, this becomes impractical without full dependent types, but the principle is established: the type system can represent a set of values (primes) and enforce membership.

To emulate a superposition of primes at the value level in Haskell, we use a data structure to hold many primes at once. For example, a simplified representation of a quantum state is a list of basis states paired with amplitudes:

```
import Data.Complex

type Amp = Complex Double
type Basis = Int  -- Basis states represented by an Int
type State = [(Basis, Amp)]  -- Quantum state as list of basis states with
    amplitudes

-- Generate the "prime state" for n qubits (i.e., primes < 2^n)
primeState :: Int -> State
primeState n =
  let maxVal = 2^n
      primes = [p | p <- [2 .. maxVal - 1], isPrime p]
      k = fromIntegral (length primes)
      amp = 1.0 :+ 0.0  -- Complex amplitude (1+0i)
      normAmp = amp / sqrt k  -- Normalize amplitude
  in [(p, normAmp) | p <- primes]
```

Here, `primeState n` produces a list of all primes less than 2^n, each with equal amplitude $\frac{1}{\sqrt{\pi(2^n)}}$. This is a classical simulation; the data structure `State` conceptually corresponds to the quantum state

$$|\text{Prime}_n\rangle = \frac{1}{\sqrt{\pi(2^n)}} \sum_{\substack{p<2^n \\ p \text{ prime}}} |p\rangle,$$

which is exactly the Prime state of Latorre & Sierra. One may then simulate operations on this state. For instance, to simulate twin prime state preparation, we modify the state by checking for each prime whether $p+2$ is also prime:

```
-- Pseudo-code for twin prime state amplitude extraction
twinPrimeAmps :: Int -> [Amp]
twinPrimeAmps n =
  [ amp * indicator (isPrime (basis + 2))
  | (basis, amp) <- primeState n
  ]
  -- where "indicator" is 1 if the condition is true, 0 otherwise.
```

After this operation, only primes with a twin (i.e. p such that $p+2$ is prime) retain nonzero amplitude. Measuring this state would yield a twin prime with probability equal to the fraction of primes that have a twin.

Additionally, type systems can enforce quantum constraints. Emerging linear types (now available in GHC extensions) ensure that quantum variables (qubits) are used exactly once, modeling the no-cloning theorem. Combining such linear types with Haskell's type system provides a rigorous framework for representing quantum states and operations.

4 Twin Primes, Quantum Superposition, and the Riemann Hypothesis

One of the fascinating aspects of the Prime state is how number-theoretic patterns manifest as quantum properties. For example, the entanglement entropy of certain qubits in the Prime state reflects biases in prime distribution. Latorre & Sierra observed that the reduced density matrix of the least significant qubit has eigenvalues related to the density of primes in different congruence classes. They found that the expectation value of a Pauli Z operator on that qubit corresponds to the difference in primes congruent to 1 mod 4 versus 3 mod 4 (the Chebyshev bias). This is remarkable: a single-qubit measurement can detect subtle number-theoretic biases.

Similarly, two-qubit correlations in the Prime state can capture prime pair correlations. In principle, an entanglement measurement on two adjacent qubits might encode the prevalence of prime pairs with certain separations. The Twin Prime state is a more direct way to capture this: it entangles a prime p with the condition that $p+2$ is prime. Analyzing such a state could provide an "experimental" confirmation of the Hardy–Littlewood twin prime conjecture up to a given range by comparing the measured probability with the theoretical prediction of

$$\frac{2C_2}{(\ln x)^2}.$$

At first glance, twin primes (pairs of primes with difference 2) and RH (a global statement on prime distribution) seem disconnected. However, both describe correlations in the sequence of primes. RH implies that primes are distributed as evenly as possible according to their density, while the existence of twin primes demonstrates local clumping beyond a naive random model. Moreover, some researchers have attempted to relate the Euler product for $\zeta(s)$ to prime pair behavior, suggesting that an appropriate modification of the product may highlight twin prime contributions.

Quantum superposition offers a fresh perspective: constructing a state that is sensitive only to twin primes effectively creates a projector onto twin primes. A quantum Fourier transform applied to such a state might extract periodicities or correlations from the twin prime distribution, analogous to how the Fourier transform of the Prime state yields information about $\pi(x)$. Although these ideas are speculative, they illustrate how different mathematical phenomena might be unified under a quantum framework.

5 Haskell Modeling and Implementation

To concretize the discussion, we now present two Haskell-based approaches:

1. **Quantum State Simulation:** We simulate a quantum state representing the superposition of primes.
2. **Type-Level Encoding:** We use Haskell's type system to enforce primality as a compile-time property.

(a) Quantum State Simulation in Haskell

Below is a Haskell snippet that simulates the creation of a prime state for n qubits. For example, for $n = 4$ (primes less than $2^4 = 16$):

```
> let state = primeState 4    -- primes less than 16: 2,3,5,7,11,13
> length state
6
> take 6 state
[(2,0.4082483),(3,0.4082483),(5,0.4082483),
 (7,0.4082483),(11,0.4082483),(13,0.4082483)]
```

Each prime is assigned an amplitude ≈ 0.4082483 (i.e., $1/\sqrt{6}$), and the state is normalized:

$$\sum |amp|^2 = 1.$$

To simulate an operation analogous to checking for twin primes, we filter the state:

```
> let twinState = [ (basis, amp)
                  | (basis, amp) <- state, isPrime (basis + 2) ]
> twinState
[(3,0.4082483),(5,0.4082483),(11,0.4082483)]
```

Here, out of the primes {2,3,5,7,11,13}, only those with a twin partner (3, 5, and 11) remain. If measured, the probability of obtaining a twin prime is $3/6 = 0.5$, matching the theoretical ratio.

For integration with a quantum DSL like Quipper, one might imagine a pseudocode outline:

```
-- Pseudocode (not actual Quipper code):
circuit :: Circ Qubit
circuit = do
    qs <- qinit (replicate n False)    -- n qubits for the number register
    anc <- qinit False                 -- ancilla qubit
    hadamard_on qs                     -- create superposition over 2^n states
    forEachNumber qs $ \q -> do
         if (classicalPrimeTest q == False)
              then qnot anc
    groverDiffusion qs anc            -- amplify states with prime numbers
    return qs
```

(b) Type-Level Primality in Haskell

Using Data Kinds and Type Families, we can enforce that a number is prime at compile time. For example:

```
{-# LANGUAGE DataKinds, KindSignatures, TypeOperators, TypeFamilies #-}

import GHC.TypeLits (Nat)
import Data.Proxy

class PrimeNat (n :: Nat)
instance (PrimesUpTo n ~ primes, Elem n primes ~ True) => PrimeNat n
  -- This constraint ensures that an instance exists only if n is prime.

factorialIfPrime :: (PrimeNat p) => Proxy p -> Integer
factorialIfPrime _ = product [1 .. natVal (Proxy @p)]
```

If one attempts to use `factorialIfPrime (Proxy @10)`, the code will not compile because 10 is not prime. In contrast, `factorialIfPrime (Proxy @7)` will compile and run. This demonstrates how the type system can serve as a tool for enforcing mathematical properties at compile time.

6 Mathematical Frameworks and Towards a Proof

The ultimate goal of these explorations is to contribute to a proof or disproof of major conjectures like the Riemann Hypothesis. While neither RH nor the Twin Prime Conjecture has been proven, many partial results and reformulations are known.

Analytic and Spectral Frameworks

The traditional approach to RH is through analytic number theory, notably via the explicit formula that connects zeros of $\zeta(s)$ to the distribution of primes. One strategy is to identify an operator H whose eigenvalues are of the form $\frac{1}{2} + it_\rho$, with each nontrivial zero ρ of $\zeta(s)$ corresponding to an eigenvalue. The Hilbert–Pólya conjecture suggests that if such an operator is self-adjoint, then RH follows. Quantum approaches, including the use of the Quantum Fourier Transform, semiclassical analysis, and random matrix theory, support this viewpoint.

Interplay with Twin Primes

Twin primes address local correlations in the primes, whereas RH is concerned with global distribution. Nevertheless, both reflect the underlying structure of the primes. Certain analytic approaches, for example those that manipulate Euler products, attempt to isolate contributions corresponding to prime pairs. Furthermore, quantum algorithms that project onto twin prime states might empirically check regularities analogous to RH's error bounds for the twin prime counting function.

Type-Theoretic and Computational Frameworks

Formal verification using proof assistants (e.g., Lean, Coq) has the potential to rigorously verify proofs of deep conjectures. Although the full proofs of RH or the Twin Prime Conjecture are far beyond current capabilities, type-theoretic approaches (as demonstrated with Haskell) provide a sandbox for experimentation. Moreover, integrating computational experiments with formal methods could help verify large-scale numerical conditions that, combined with analytic arguments, might eventually yield a proof.

Potential New Avenues

The cross-pollination of quantum computing and number theory might inspire entirely new approaches. Quantum annealing or adiabatic algorithms could be employed to search for patterns in primes or optimize functions whose optima encode number-theoretic properties. Similarly, the notion of quantum entanglement might have a number-theoretic analogue, potentially leading to new invariants of the prime number sequence.

7 Conclusion

We have surveyed how the frontier of quantum computing and the rigor of type theory can come together to shed light on the mysteries of prime numbers, twin primes, and the Riemann Hypothesis. Quantum computing contributes the concept of superposition and entanglement, allowing us to represent prime numbers collectively in ways that classical methods cannot. This has led to proposals of quantum algorithms that encode prime distribution and prime correlations into quantum states. Meanwhile, advanced type systems (exemplified by Haskell's type-level programming and emerging dependent type extensions) allow us to model these problems with high precision, verifying properties at compile time.

Our exploration finds that while twin primes and the Riemann Hypothesis remain unresolved, there are tantalizing connections between them. The *Prime state* concept bridges global and local aspects of prime distributions, and experimental simulation via Haskell offers a promising route for prototyping conjectures and searching for patterns. In summary, the intersection of quantum computing, type systems, and classical number theory is a fertile ground for innovation and may eventually pave new paths toward solving some of the greatest unsolved problems in mathematics.

References

[1] José I. Latorre and Germán Sierra. Quantum Computation of Prime Number Functions. *Quantum Inf. Comput.*, 14:0577, 2014. `https://arxiv.org/abs/1302.6245`.

[2] Bob Yirka. Researchers develop quantum computer algorithm for counting prime numbers. *Phys.org*, March 26, 2013.

[3] Michael McGuigan. Quantum Computing and the Riemann Hypothesis. *arXiv:2303.04602*, 2023.

[4] Germán Sierra. The Riemann Zeros as Spectrum and the Riemann Hypothesis. *Symmetry*, 11(4):494, 2019. `https://arxiv.org/abs/1601.01797`.

[5] Quanta Magazine. Big Question About Primes Proved in Small Number Systems. Sept 26, 2019.

[6] Vincenzo Oliva (via Math StackExchange). Answer to "Does the Riemann Hypothesis imply the Twin Prime Conjecture?", 2015.

[7] Dan Goldston quoted by Dietrich Burde. Discussion on RH and Twin Primes, Math StackExchange, 2015.

[8] Saleh Omran. A connection between twin primes and Riemann zeta function, and a possible way to prove the twin prime conjecture. *Cambridge Open Engage preprint*, 2024.

[9] Haskell Type-Level Primes Example. Available on GitHub.

[10] A. S. Green et al. Quipper: A scalable quantum programming language. In *ACM SIGPLAN Notices*, 2013.