Fractional Types

Anonymous Author(s)

Abstract

Text of abstract . . . .

1 Introduction

Conservation of Information. If quantum field theory is correct (as it so far seems to be) then information, during any physical process, is neither created nor destroyed. Landauer [5, 12, 13], Bennett [3, 4, 6], Fredkin [9] and others made compelling arguments that this physical principle induces a corresponding computational principle of "conservation of information." This principle is indeed one of the defining characteristics of quantum computing and its classical restriction known as reversible computing.

Quantum and Reversible Computing. The Toffoli gate is known to be universal for classical reversible circuits [15]. Adding just one gate (the Hadamard gate) produces a universal set of primitives for quantum circuits [2]. The "only" difference between the two circuit models is that quantum circuits can process superpositions of values (waves) in one step whereas classical circuits lack this form of parallelism. Most importantly, structurally the two circuits models are identical and one can derive properties valid for both families by focusing on the simpler classical model. In fact, classical reversible computations are often used as "subroutines" of quantum computations.

Instead of using the Toffoli gate as a universal primitive for reversible classical circuits, one can leverage the full force of type theory and category theory by expressing reversible classical computations as *isomorphisms over finite types* [8, 11] or *equivalences over groupoids* [7]. This perspective exposes interesting mathematical structure in reversible computations that we will exploit to solve the "ancilla problem" explained next.

Temporary Storage using Ancilla Bits. The universality of the Toffoli gate for classical reversible computing and of the combination of the Toffoli and Hadamard gates for quantum computing should not distract from efficiency and safety concerns. The theorems proving universality (i) assume that temporary storage (often called *ancilla bits*) may be used [15], and (ii) that this temporary storage is returned to its initial state before de-allocation. Indeed if no temporary storage is allowed, the Toffoli gate is not universal [1] and as we demonstrate using space-time tradeoffs in Sec. 2, the more temporary storage is allowed, the more efficient certain computations could become. The condition requiring that the temporary storage is only de-allocated when returned to its initial condition is a safety condition. Violating this condition destroys the symmetry between input and output making the circuits not reversible and, in the quantum model, causes irreversible decoherence problems. As reviewed in Sec. 2, current proposals for reversible and quantum programming languages (e.g. Quipper [10]), this safety condition is at the programmer's discretion whereas.

Negative Entropy. According to the conventional theory of communication [14], a type with N values is viewed as an abstract system that has N distinguishable states where the amount of information contained in each state is $\log N$. This entropy is a measure of information which materializes itself in memory or bandwidth requirements when storing or transmitting elements of this type. Thus a type with 8 elements needs 3 bits of memory for storage or 3 bits of bandwidth for communication. The logarithmic map implies that information contained in a composite state is the sum of the information contained in its constituents. For example, the type $A \times B$ where A has two elements and B has three elements can be thought of a composite system consisting of two independent unrelated subsystems. Each state of the composite system therefore contains $\log(2 * 3) = \log 2 + \log 3$ bits which is the sum of the information contained in each subsystem. A *fractional type* $\frac{1}{A}$ introduces negative entropy. For example, a type with cardinality $\frac{1}{8}$ has entropy $\log \frac{1}{8} = -3$. It is natural to interpret this negative entropy just like we interpret "negative money," as a resource (space or bandwidth) to be repaid (reclaimed) by some other part of the system. Indeed, we will introduce such fractional types and use them to represent "garbage collection processes" that reclaim temporary storage.

Contributions.

Outline.

2 Motivating Examples

Toffoli4 using 2 Toffoli3: core of proof of universality; simple enough

In-place matrix transpose: ease of programming, efficiency

Example(s) from Quipper etc. with focus on safety condition

3 Extra for now

- Quantum computing hot.
- Unitary evolution; many algorithms promote classical reversible functions into unitary gates (examples, including some from Quipper)

PL'18, January 01–03, 2018, New York, NY, USA
2018.

- Common case: given a permutation between finite sets, find a reversible program that implements it. Answer depends on the available reversible primitives.
- Example, say we want to transpose a matrix, i.e., write a permutation of type `A*B <-> B*A` where `A` and `B` are the dimensions. Wikipedia example (https://en.wikipedia.org/wiki/In-place_matrix_transposition):

  ```
  11 12 13 14
  21 22 23 24
  ```

  transposed to

  ```
  11 21
  12 22
  13 23
  14 24
  ```

  Here `A` is `Nat x Nat` and `B` is `Nat x Nat x Nat x Nat` and the input and output matrices are:

  ```
  M  = (11 , 21) , (12 , 22) , (13, 23) , (14 , 24)

  trM = (11 , 12 , 13 , 14) , (21 , 22 , 23 , 23)
  ```

  Say we are given a language like Pi that is sound and complete with respect to permutations on finite types, we would write the permutation like so.

  ```
  WRITE PERMUTATION
  ```

  This code does not use additional space, i.e., it performs the matrix transpose in constant space, i.e., performs an in-place matrix transposition. It is well know that with additional space, one can write more efficient matrix transpositions (explain and citations).
- So many reversible/quantum circuits allow the introduction of so-called ancilla wires, which serve as scratch space (goes back to Toffoli). The problem is that there is a discipline for using ancilla bits: they need to be created in a known state and can only be gc'ed when they are returned to the same state (explain).
- No language knows how to automatically guarantee this condition; left to the programmer (Quipper), or some (incomplete?) dynamic checks (Ricercar).
- We solve this using **fractional types**.

Complexity perspective: https://www.scottaaronson.com/papers/gates.pdf

Quipper: https://arxiv.org/pdf/1304.3390.pdf; uses ancilla; one use is to compile irreversible circuits to reversible ones; need ancilla bits; more generally several quantum algorithms need ancilla bits (see below); in quipper de-allocation left to programmer.

Ricercar tries to define rules for ancilla but not complete https://msoeken.github.io/papers/2015_rc_2.pdf

Leads to bugs as analyzed by http://drops.dagstuhl.de/opus/volltexte/2019/10196/pdf/OASIcs-PLATEAU-2018-4.pdf:

> Bug type 6: Incorrect composition of operations using mirroring Section 4.5 discussed how bugs in deallocating ancillary qubits can happen due to bad parameters. Here we see how bugs in deallocating ancillary qubits can happen due to incorrect composition of operations following a mirroring pattern. For example, in Table 7, the operations in rows 2 and 3 are respectively mirrored and undone in rows 6 and 5. These lines of code need careful reversal of every loop and every operation.
>
> A common pattern in quantum programs involves performing operations (e.g., add), contingent on a set of qubits known as control qubits. Without language support, this pattern needs many lines of code and manual allocation of ancillary qubits. In the Scaffold code example in Table 7, rows 3 and 5 are just computing the intersection of qubits q, with the help of ancillary qubits initialized in row 1, in order to realize the controlled rotation operation in row 4. Furthermore, quantum algorithms often need varying numbers of control qubits in different parts of the algorithm, leading to replicated code from multiple versions of the same subroutine differing only by the number of control qubits.

Uses of ancillas: https://quantum.country/search:

> There's a rough heuristic worth noting here, which is that you can often convert if-then style thinking into quantum circuits. You introduce an ancilla qubit to store the outcome of evaluating the if condition. And then depending on the state of the ancilla, you perform the appropriate state manipulation. Finally, when possible you reverse the initial computation, resetting the ancilla to its original state so you can subsequently ignore it.

https://www.nap.edu/read/25196/chapter/5#73

> For error correction, one needs to replicate the state of a qubit onto a number of qubits. While the no cloning theorem prevents one from copying the state of a qubit directly onto another, one can create a redundant entangled qubit state of many qubits. The key is that the qubits to be entangled must start out in a known state. Qubits with a known state (for purposes of this discussion, it will be the state `|0>`), called "ancilla qubits," may be added to a computation for this purpose. Since the state of ancilla qubits are known, it is possible to create a simple circuit that makes the output state of all these ancilla qubits match the protected qubit: run each ancilla through a controlled-NOT gate, where the control is driven by the qubit that needs to be replicated. Assume that there is a qubit with state `PSI` that we want to protect, where `|PSI>`

> represents an arbitrary superposition state |PSI> = a0 |0> + a0 |1> In the CNOT gate, the ancilla |0> state will remain a |0> state by the |0> component of |PSI>, but it will be converted to |1> by the |1> component of |PSI> The result of this operation is the newly entangled two-qubit state a0 |00> + a1 |11>, creating a system in which the ancilla qubit is now perfectly entangled with the first qubit. Adding more ancillas increases the distance of the repetition code.

https://www.sigarch.org/the-case-for-quantum-computing/

> Imagine a 3-qubit quantum majority code in which a logical "0" is encoded as "000" and a logical "1" is encoded as "111." Just as with a classical majority code, a single bit-flip error can be corrected by restoring to the majority value. Unlike a classical code, however, we can not directly measure the qubits else their quantum state will be destroyed. Instead, we measure syndromes from each possible pair of qubits by interacting them with an ancilla, then measure each ancilla. Although the errors to the qubits are actually continuous, the effect of measuring the ancilla is to discretize the errors, as well as inform us whether an error occurred so that it can be corrected. With this methodology, quantum states are restored in a modular way for even a large quantum computer. Furthermore, operations on error-corrected qubits can be viewed as digital rather than analog, and only a small number of universal operations (H, T, CNOT) are needed for universal quantum computation. Through careful design and engineering, error correction codes and this small set of precise operations will lead to machines that could support practical quantum computation.

https://homepages.cwi.nl/~rdewolf/qcnotesv2.pdf

> Using an ancilla qubit, it is possible to avoid doing any intermediate measurements in a quantum circuit. Show how this can be done. Hint: instead of measuring the qubit, apply a CNOT that "copies" it to a new |0>-qubit, which is then left alone until the end of the computation. Analyze what happens.

Reversible/Quantum Circuits and Ancilla Wires: Early use of ancillas in Toffoli's paper to implement arbitrary reversible functions using a fixed number of 3 input gates https://link.springer.com/content/pdf/10.1007%2F3-540-10003-2_104.pdf

Use examples from Ricercar

Is the below a possible example?

```
Say we already have a permutation A <-> B
we can implement a permutation X <-> Z
when there exists Y such that A/X = Y = B/Z

X -> X * Y * 1/Y
  -> A * 1/Y
  -> B * 1/Y
  -> Y * Z * 1/Y
  -> Z
```

A Comonad for Pi:

Adding variables to Pi first?

Plain Pi over finite types:

Value types	t	$::=$	$0 \mid 1 \mid t + t \mid t * t$
Values	v	$::=$	$\star \mid inl(v) \mid inr(v) \mid (v, v)$
Level-1 types			$t \leftrightarrow t$
Level-1 programs	c	$::=$	$\cdots$
Level-2 types			$c \Leftrightarrow c$
Level-2 programs	α	$::=$	$\cdots$

Now we define pointed types $[t \bullet v]$ which are a record consisting of a value type together with a value of that type. Both the level-1 and level-2 program lift naturally to the world of pointed types but instead of manually rewriting everything we use the fact that pointed types form a comonad. So for example, given a level-1 program we should be able to just use extend to compose instead of plain composition of combinators??

Then we can define η and ϵ in the comonadic world.

Adding variables to Pi first?

Denotations:

Positive rational numbers are a model. Apparently there is a categorification https://alistairsavage.ca/pubs/Copelli-Categorification_of_the_Nonnegative_Rational_Numbers.pdf

Use all the constructions name, coname, etc. and see what they do in this context!

References

[1] Scott Aaronson, Daniel Grier, and Luke Schaeffer. 2017. The Classification of Reversible Bit Operations. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*. 23:1–23:34. https://doi.org/10.4230/LIPIcs.ITCS.2017.23

[2] Dorit Aharonov. 2003. A Simple Proof that Toffoli and Hadamard are Quantum Universal. (2003). arXiv:quant-ph/0301040.

[3] C.H. Bennett. 2003. Notes on Landauer's principle, reversible computation, and Maxwell's Demon. *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics* 34, 3 (2003), 501–510.

[4] C.H. Bennett. 2010. Notes on the history of reversible computation. *IBM Journal of Research and Development* 32, 1 (2010), 16–23.

[5] C.H. Bennett and R. Landauer. 1985. The fundamental physical limits of computation. *Scientific American* 253, 1 (1985), 48–56.

[6] C. H. Bennett. 1973. Logical reversibility of computation. *IBM J. Res. Dev.* 17 (November 1973), 525–532. Issue 6.

[7] Jacques Carette and Amr Sabry. 2016. Computing with Semirings and Weak Rig Groupoids. In *ESOP (Lecture Notes in Computer Science)*, Vol. 9632. Springer, 123–148.

[8] Marcelo Fiore. 2004. Isomorphisms of generic recursive polynomial types. In *POPL*. ACM, 77–88.

[9] E. Fredkin and T. Toffoli. 1982. Conservative logic. *International Journal of Theoretical Physics* 21, 3 (1982), 219–253.

[10] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: A Scalable Quantum Programming Language. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13)*. ACM, New York, NY, USA, 333–342. https://doi.org/10.1145/2491956.2462177

[11] Roshan P. James and Amr Sabry. 2012. Information effects. In *POPL*. ACM, 73–84.

[12] R. Landauer. 1961. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.* 5 (July 1961), 183–191. Issue 3.

[13] Rolf Landauer. 1996. The physical nature of information. *Physics Letters A* (1996).

[14] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell Systems Technical Journal* 27 (1948), 379–423,623–656.

[15] Tommaso Toffoli. 1980. Reversible Computing. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*. Springer-Verlag, 632–644.