Programs:
From Monads to the Quantum Computer
From Contemporary CS to Contemporary Physics

Cem Bozşahin, Cognitive Science, METU

for a Course on Philosophy of Computer Science

October 27, 2022

- We have seen that algorithms are nice and succint ways to express ideas to other people.
- We can leave some bits to the interpreter, because, after all, they can independently operate to see for themselves.
- You would be amazed to find out how much of so-called 'formal notation' leaves a lot to the interpreter, see Hadamard (1945), Ganesalingam (2013).
- But it is at the level of a program that a computer scientist puts herself in computer's shoes.
    - We can talk about a list structure at the level of an algorithm without needing details.
    - We can talk about a conditional at the level of an algorithm without being specific about how it is carried out.
    - In a program, we can do neither. We need to be specific.

- This shoe is designed by us.
- Therefore we are trying to see what kind of control and data structures can be passed on to the computer for itself to independently operate.
- There is no homunculi inside to interpret things which the designer has not thought of.
- When it operates, it is attempting a human problem. At least today's computers do so (see Bozşahin 2018 for discussion.)

- We have seen that a digital computer has to be designed to operate within certain limits (encoding, decoding, instruction set, etc.)
- Everything in a digital computer has to translate to its native instruction set for it to operate.
- The instruction set is designed by us, with careful engineering.

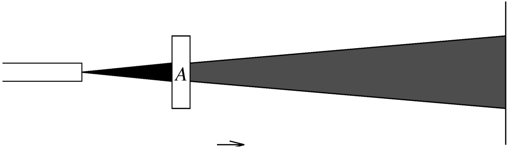
\- so that a programmer can know what is wrong if something goes wrong.

- The same goes for the analog computer. They have to be designed to interpret the analog device (e.g. a conducting foam) within certain limits.
- Otherwise, an analog program (usually called an analog, rather than program) cannot be fixed if there's something wrong with it. See Rubel (1993), Mills (2008).

- What about the quantum computer? Is it designed, or is it 'the quantum nature computes'?
- I am not the right person to talk about the quantum computer in detail, but here we go about the 'kitchen view',
- I am summarizing from Rieffel and Polak (2000), Arora and Barak (2009).

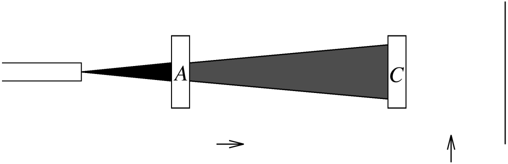
Photon polarization

- Photons are the only particles we can observe directly.
- Polarizers are optical filters which let light waves of certain polarization to pass through, blocking others.
- One experiment: a strong light source (e.g. laser pointer), three polaroids (filters): Filters A, B and C.
- A: horizontal polarizer, B: 45 degrees, C: vertical
- First, insert Filter A between the laser source and a projection screen.



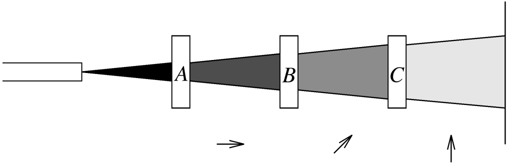
- Assuming that incoming light is randomly polarized, the intensity of the output (i.e. number of photons passed) will be half the intensity of incoming light.
- All outgoing photons are horizontally polarized.
- Filter A cannot be a simple sieve, letting only already horizontally polarized photons.
- Because, if that were the case, only a few of the randomly polarized incoming electrons would be horizontally polarized.
- We would expect much larger attenuation (cutoff) if that were the case. The filter is DOING something. Or nature does something to the filter.

Now insert Filter C (vertical polarizer)



- The intensity of the output drops to zero. The output of A is all horizontally polarized, all blocked by C.
- So far not much of a surprise.

Now insert Filter B between A and C:

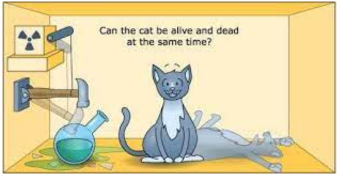


- The output intensity is NOT zero. What is going on?
- It turns out that the intensity of output is 1/8th of the incoming light.
- This is a counterintuitive result. We would expect intensity to decrease, not increase, if we add more filters.
- To understand the quantum computer, we need to understand the representational potential of this counterintuitive result.

Representation in quantum mechanics

- The state space of a quantum system, consisting of positions, momentums, polarization, spin, angular momentum etc. is modeled by Hilbert space of wave functions. These are not finite dimensional.
- Let's look at finite-dimensional complex vector spaces for the purpose of quantum computing.
- Dirac's bra-ket notation for quantum states is a very asymmetric notation which continues to horrify some computer scientists.

- The abstract wave function is represented as a ket, e.g.: $|{\rightarrow}\rangle$ for the horizontal polarizer A. You make up the vector name.
- Schrödinger was very creative, he had $|alive\rangle$ and $|dead\rangle$ for cats.



- Kets are column vectors. For example $|0\rangle$ can be $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle$ can be $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- Other representations are possible, as long as we are consistent. E.g. we can swap definitions of $|0\rangle$ and $|1\rangle$.

- Bras are conjugate transposes of kets. For example $\langle 0| = (1 \;\; 0)$ and $\langle 1| = (0 \;\; 1)$
- They are row vectors
- We can multiply bras and kets as in matrix algebra, e.g. $|0\rangle\langle 1| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (0 \;\; 1) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ $\langle 0||0\rangle = (1 \;\; 0) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1$
- $\langle 0|\,|1\rangle = (1 \;\; 0) \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0$ also written $\langle 0|1\rangle$
- $\langle 1|\,|1\rangle = (0 \;\; 1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1$ $\langle 1|1\rangle$
- ket-bra is a matrix, bra-ket is a number. Ket and bra are vectors. Ket-ket is tensor product ($\otimes$). Bra-bra is not tensor-unique. (lessons for CS: what to encode/decode)

- Any matrix-vector algebra is fine, e.g. $|0\rangle\langle 0|1\rangle = |0\rangle\, 0$, which we write by convention as vector with a coefficient 0: $0\,|0\rangle$
- $0\,|0\rangle = 0\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$
- Representation and numbers: Coefficients (e.g. 0 above) can be complex numbers. That seems to be required given our current understanding of real and complex numbers, and, nature, with them.
- NB. Bras are not simply transposes, they are conjugate transposes.
- Why do we need the conjugate transpose? The imaginary coefficient of a complex number corresponds to circular polarization. (i.e. it is there in the input light).
- We are here studying what we must encode and decode.

Complex coefficients and real results

- Let $|a\rangle = \begin{pmatrix} 2+3i \\ 6-5i \\ 9+i \end{pmatrix}$ $\quad \langle a|$ is complex conjugate, transposed
- then $\langle a|a\rangle = (2-3i \quad 6+5i \quad 9-i) \begin{pmatrix} 2+3i \\ 6-5i \\ 9+i \end{pmatrix} =$
  $(2-3i)(2+3i)+(6+5i)(6-5i)+(9-i)(9+i) =$
  $4+6i-6i-9i^2+36-30i+30i-25i^2+81+9i-9i-i^2 =$
  $4+9+36+25+81+1 \qquad (i^2=-1)$
- More lessons for CS: intensions (e.g. i) are nice. Extensions are what we interpret.

- A quantum system consists of one particle: states of the system are wave functions (vectors), and observables are operators (matrices).
- There are infinitely many states, and arbitrarily largely many are computable.
- Multiple systems (bits, particles) need one more representational support: tensor product of matrices-vectors

Tensor product

$$\left(\begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array}\right)_A^{m\times n} \otimes \left(\begin{array}{cc} b_{11} & b_{12} \\ b_{21} & b_{22} \end{array}\right)_B^{p\times q} = \left(\begin{array}{cc} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{array}\right)_{A\otimes B}^{mp\times nq}$$

$$= \left(\begin{array}{cccc} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{array}\right)$$

This is an idealization. We are assuming that quantum systems are closed, and continue to be closed when combined.

Engineering of that is a huge challenge in computer engineering.

We assume it holds in nature. Maybe it doesn't.

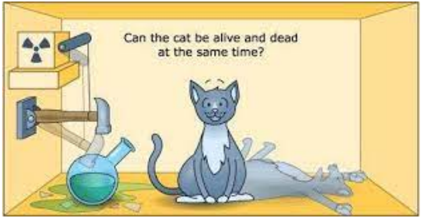
- ket-ket example: we get another ket, with combined dimension
$$|0\rangle\,|1\rangle = |0\rangle \otimes |1\rangle = |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$
- bra-bra is not tensor-unique:
$\langle 0|\,\langle 1| =? \langle 0| \otimes \langle 1| =? \langle 01| =??(1 \;\; 0) \otimes (0 \;\; 1) =??(0 \;\; 1 \;\; 0 \;\; 0)$
$??(0 \;\; 0 \;\; 1 \;\; 0)$
- Unless we interpret tensor product as Kronecker product.
- Puritans take note: That's because a tensor product is an equivalence class over matrices, whereas the Kronecker product always yields a unique matrix.
- So, we've been doing Kronecker product when we said tensor product. Welcome to math confusion.

- Lessons for CS: The representational support is there to facilitate encoding and decoding.
- Hmm. Is $\langle 01|01\rangle$ bad then, if $\langle 01|$ is weird?
- Didn't we just say we have $\langle a|$ and $\langle a|a\rangle$, for any a, if we have $|a\rangle$?
- And physicists and mathematicians do talk about stuff like $\langle 01|$: Nannicini (2020).
- $\langle 01|$ is actually the Kronecker product $\odot$, not tensor product $\otimes$.
- $\langle 01| = \langle 0| \odot \langle 1| = (1 \;\; 0) \odot (0 \;\; 1) = (0 \;\; 1 \;\; 0 \;\; 0)$

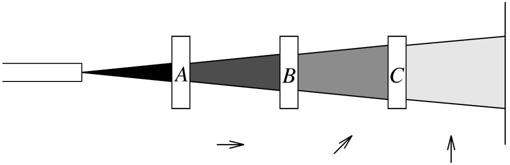
- $\otimes$ and $\odot$ are related but different, both increasing dimension (unlike matrix and Hadamard products).
- Both are compositional.
- Kronecker gives a unique result.
- Write $\otimes$ when we mean $\odot$? Better not write anything to baffle future generations. CS nitpicking;)
- And, KroneckerFarm won't catch on but TensorFarm is cool.
- Just like Schönfinkeling is unheard of (except in secret sects of CS) but Currying is very popular.

Back to the counterintuitive result

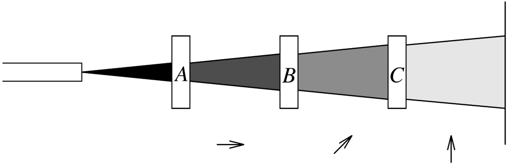
- We have done all this to make sense of the counterintuitive result.
- And I was kidding about vectors doing something by their names, cf. Schrödinger. What they do is determined by their interpretation.
- For the cat, we have $\frac{1}{\sqrt{2}}\,|\mathit{dead}\rangle + \frac{1}{\sqrt{2}}\,|\mathit{alive}\rangle$



What about photon polarization?



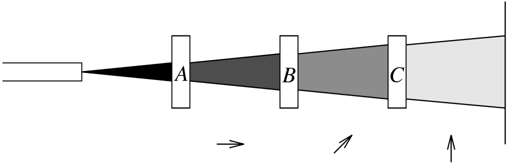
- Arbitrary input polarization, given A, B, C filters, requires measurement bases for them. After all, they are instruments with certain properties.



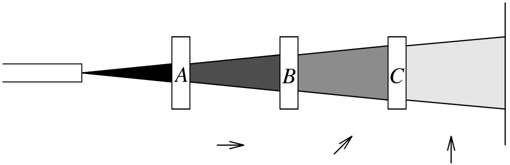
- Let's assume a polarizer measures the quantum state of photons in its (1) polarization direction; (2) and its orthogonal polarization, i.e.
- A is $a_1 \, |{\rightarrow}\rangle + a_2 \, |{\uparrow}\rangle$ recall: a_j, b_j, c_j are numbers
- B is $b_1 \, |{\nearrow}\rangle + b_2 \, |{\nwarrow}\rangle$
- C is $c_1 \, |{\uparrow}\rangle + c_2 \, |{\rightarrow}\rangle$
- We care about direction of polarization only, i.e.:

  $| \, a_1 \, |^2 + | \, a_2 \, |^2 = 1 \quad | \, b_1 \, |^2 + | \, b_2 \, |^2 = 1 \quad | \, c_1 \, |^2 + | \, c_2 \, |^2 = 1$

for example $a_j, b_j, c_j = \frac{1}{\sqrt{2}}$



- Filter B's $|\nearrow\rangle$ and $|\nwarrow\rangle$ can be described using others, respectively as
- $\frac{1}{\sqrt{2}}\,|\uparrow\rangle + \frac{1}{\sqrt{2}}\,|\rightarrow\rangle$ and $\frac{1}{\sqrt{2}}\,|\uparrow\rangle - \frac{1}{\sqrt{2}}\,|\rightarrow\rangle$
- Check that the math adds up to 1. We have $\sum_{1}^{4}(\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}})^2$



- Now, the counterintuitive result of measuring 1/8th of incoming photons on screen is:
  - Photons passing thru A are in state $|{\rightarrow}\rangle$. That's 1/2 of incoming photons.
  - Photons passing thru B are in state $|{\nearrow}\rangle$. That's 1/2 of photons coming from A, or 1/4 of incoming photons.
  - Photons passing thru C are in state $|{\uparrow}\rangle$. That's 1/2 of photons coming from B, or 1/8 of incoming photons.

Are we reasoning backwards?

- Did we just make up gadgets to match an observed result?
- Pessimists and agnostics: yes.
- Virtually all others: No.
- We extended our theoretical vocabulary, and solidified our encoding/decoding (hopefully), to take on hitherto untackled problems. We can avoid overfitting the result, unless we cheat. Quantum healers beware!
- That's what good theories do.
- And it takes a bit of daredevil approach to science.

On to quantum computing

- We know that quantum computation is provably better than classical computation in some problems.
  - Uninformed search in a classical computer is $O(n)$, n being problem size.
  - Uninformed search in a quantum computer is $O(\sqrt{n})$, with Grover's algorithm.
  - That's a polynomial improvement in all polynomial problems, because the problem is P-complete.
- We don't know whether QC is provably better in all hard problems.
  - Schor's algorithm does not target an NP-complete problem.
- But it does solve one classically intractable problem polynomially.
- Where does the power come from?

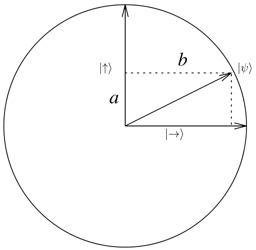


Fig. 1. Measurement is a projection onto the basis.

- Classical on-off bit can only be in state 0 or state 1.
- Quantum bit (qubit) is a superposition like $|\psi\rangle$ on the left.
- Think of $|\uparrow\rangle$ as $|1\rangle$, and $|\rightarrow\rangle$ as $|0\rangle$ we have $|\psi\rangle = a\,|1\rangle + b\,|0\rangle$
- such that they are bit-like, i.e. $\mid a \mid^2 + \mid b \mid^2 = 1$
- It can take any value in the rectangle.
- That's a lot more than just two values.
- Amazingly enough though, it is still digital, because any measuring device has finite orthonormal bases.

- That's the quantum advantage in a SINGLE system.
- COMBINED systems (multiple qubits) have one more advantage:
  - It is not always the case that their constituent states are separable.
    i. A separable state: $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle = |0\rangle \otimes (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)$
    ii. A non-separable state: $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$
- Non-separable states are entangled. This is not expressible in classical computation.
- This intension has physical correlate, which we can manipulate in physics of computing.

- Extended expressivity with the quantum computer
  - Superposed qubit of the single system
  - Non-separable (entangled) qubits of the combined system
- These are not available in a classical computer
- Does that sound like 'quantum nature computes' to you?
- It doesn't sound like that at all to me
- Somebody has to harness these properties to be more expressive
- More expressive for what? To attempt solving more human problems

* There is a cotton industry built around misunderstanding the analog, digital and the quantum computer.

- One task for the computer scientist, should we choose to accept it, is to expose it before it brings more harm to human knowledge construction.

How does a quantum computer operate?

- Python has a quantum library, called `qiskit`.
- Lisp, as usual, defines a new quantum language using its macro facility, called `qlisp`.
- In case you want to program it at the level like of a QTM:
- To compute function f on input x, that is, to do $f(x)$.

- Input is prepared in a superposition, from $|0\ldots0\rangle$ to $|x0\cdots0\rangle$.
- Once the quantum computer runs f, using its quantum gates etc. it ends in $|xf(x)0\cdots0r\rangle$. r is residue of quantum gates.

  (I am caricaturizing the noise handling here as if it is isolated, for the sake of introduction; see Arora and Barak p.215)

- Copy (not clone) result to unused bits by transformations, to get: $|xf(x)f(x)0\cdots 0r\rangle$
- Run the gates in reverse, we get $|x0\cdots 0f(x)0\cdots r_1\cdots r_2\cdots\rangle$
- Run input transformation in reverse (Hadamard): $|0\cdots 0f(x)0\cdots 0\cdots r_1\cdots r_2\cdots\rangle$ (and live with the noise)

- We have seen that intensions are important to describe to others what we have in mind about nature.
- If you are passing this knowledge to a human being, she is herself an interpreter. But, they tend to change over time.
- If you are passing it to a computer, you better not leave it to the interpreter to connect to it results, if you consider this to be your practice.
- So we must make a clear distinction between functions on values (intensions) and values themselves.
- Ancient Egyptians knew 2>1. Did they know that number of protons in Helium's nucleus is greater than that of Hydrogen?
- That's what the monads symbolize. We use them sometimes without knowing, in fact in ALL data and program abstraction.

- The extensional view of computer programs: All functional programs have behaviorally equivalent lambda-terms.
  - Values: $1 = \lambda x.(x^2)1$. We also have $1{=}(\lambda x.1)a$ for any a.
  - Functions: $\lambda x.fx{=}f$ because $fa = (\lambda x.fx)a{=}fa$ for any a.
  - $f{=}\lambda f \lambda x.fx$ is eta conversion, assuming no free occurence of x in f.
  - The Halting Problem of TM : $(\lambda x.xx)(\lambda y.yy)$
- Turing's TM makes it evident that you need representational support for that.
- Monads: Programs are intensions on values, not values themselves. (Moggi, 1988, 1991)
- This is the intensional view of the computer program.

\- Intension with 's', not 't'.

Two views of computing

- Lambda-calculus: Everything is a value, including functions.
- Monads: Everything is a function, including values.
- TM: Either way, you need representation to be able to program.
- That's one reason everyone talks about the TM.
- Does the monadic view sound familiar?
  - QM says all wave functions are vectors. The wave function characterizes a particle (a value), and observables are matrices operating over such vectors. They are functions too.
  - We are treating everything as a function of kets and bras, including the scalars.

If you keep four things in mind about monads, they would no longer be mystery objects or divine revelations:

1. A monad always composes, and always the ultimate function is head function.
2. Type Ma means having access to a value of type a inside the monad. Another reading: The type Ma represents a computation that yields a value of type a (Wadler, 1993). No more no less. What can you do with that inside? Well it's a monad: you compose.
3. Monads deal with endofunctions, functions from some domain to same domain as co-domanic (range).
4. Something has to kick-start the monad. That's the inject (unit) function.

And you can keep doing this, combining monads. Much like chaining your program calls.
Monadic computation is hermetic computation.

- The type of any Monad M is $Ma \to ((a \to Mb) \to Mb) = Ma \to (a \to Mb) \to Mb$.
- This is also counterintuitive. Where does it come from?
- Composition of two functions f and g is $[\, g \circ f \,](x)$
- which is $f(gx)$
- It has the type $(a \xrightarrow{g} b) \to (b \xrightarrow{f} c) \to (a \xrightarrow{h} c)$, where h is the result function of composition, and g and f are input functions entering composition.

- Composition: $(a \xrightarrow{g} b) \rightarrow (b \xrightarrow{f} c) \rightarrow (a \xrightarrow{h} c)$
   - The first b, from $(a \xrightarrow{g} b)$, is a computationally structured value, not just a value,
   - we can rewrite as $(a \xrightarrow{g} Ma)$, where M is the monadic type constructor.
   - Similarly, c is a structured value; we can rewrite as $(b \xrightarrow{f} Mb)$ and $(a \xrightarrow{h} Mb)$, maintaining the c type.
- We now appear to have the type $(a \xrightarrow{g} Ma) \rightarrow (Ma \xrightarrow{f} Mb) \rightarrow (a \xrightarrow{h} Mb)$ for function composition.

- We've reached: $(a \xrightarrow{g} Ma) \rightarrow (b \xrightarrow{f} Mb) \rightarrow (a \xrightarrow{h} Mb)$
- However, f's input type b is a (i.e. the result type) of g, so that we get the type $(a \xrightarrow{g} Ma) \rightarrow (a \xrightarrow{f} Mb) \rightarrow (a \xrightarrow{h} Mb)$.
- We must create the result function from the inputs, which are g and f, which we can do by abstracting over the not-yet-computed a.
- The 'bind' operation of the monad does that, giving the type $Ma \rightarrow (a \rightarrow Mb) \rightarrow Mb$.

- Having to work with two functions in a monad is intuitive when we have two elements which are evidently functional:
  - ex: mother-of(father-of(someone)), and father-of(mother-of(someone))
- It is not so intuitive when there is only one function, or no function at all:
  - 4 and 5? square(4)?
- If monad's endofunctions are functions themselves, what are the input and output functions here?

- Think about squaring 4. Second input is *sq*, already a function; first input can be functionalized as $\lambda x.4$ (because the result must be determined by the ultimate function)

1. $M4$ injects 4 into M as an intension. Extensionally, you can think of it as $(\lambda x.x)(\lambda x.4)$.
2. Inside M what is happening is extensionally $\lambda y.sq((\lambda x.4)y)((\lambda x.x)(\lambda x.4))$

   - Yes this is heresy, talking about extensions in a monad. We are trying to see what goes inside.

   All values are functions. We are composing *sq* with $(\lambda x.4)$
3. Ma to a is realized as $(\lambda x.x)(\lambda x.4) = \lambda x.4$
4. Then $a \rightarrow Mb$ is realized as $sq((\lambda x.4)(\lambda x.4))$
5. Then Mb is realized as $sq(4)$. Think of *sq* as $\lambda w.w^2$

- And why on earth can't we just write $(\lambda x.sq(x))4$?
- After all, this is what we CODE as a program, taking 4 as input.

  Compare the monadic version: $\lambda y.sq((\lambda x.4)y)((\lambda x.x)(\lambda x.4))$
- In monadic view (and it is a view), all values operated on by computation are functions.
- Therefore, in monadic computation, we never have just the value, but intensions on the value.
- Even state change, that is, procedural programming, can be treated as a monad (Launchbury and Peyton Jones, 1994).

- What about two constants, with no functions in sight? say 4 and 5.
- IF you intend to treat them monadically, both must be functions, e.g. $\lambda x.4$ and $\lambda y.5$. The last one will survive.
- This is quite counterintuitive too.
- This is what we get extensionally inside the monad:
  $\lambda z.(\lambda x.5)((\lambda x.4)z)((\lambda x.x)4)=$
  $\lambda z.(\lambda x.5)((\lambda x.4)z)4=$
  $(\lambda x.5)((\lambda x.4)4)=$
  $(\lambda x.5)4=5$
- Without abnormal stop (it can compose), the monad is saying it has no clue, and spits out the ultimate value.
- Point to take home: If we want meaningful computation, we better pass in at least one value which is intended to be a function. Two or more can be similarly sequenced.

- What about constants? Are they monadic functions?
- say: I want a monad to always give 4.
- We expect to get 4 when first input is *a* second input is *b*. Both are functions.
- Here is the monadic version: $\lambda y.((\lambda x \lambda z.4)b)(a\ y)((\lambda x.x)a)$
- We are still composing, this time $(\lambda x \lambda z.4)b$ with a
- But it says a bit more about being a constant TO the interpreter: It is a function that not even many functions can map to another value. (multiple cases generalize from Currying binary cases.)

- Monads make intensions specific, hoping this will help clarify expressing intention, with 't'.
- Values by themselves do not make these aspects clear.
- These are some points to take home for philosophy.

Arora, S. and B. Barak (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Bozşahin, C. (2018). Computers aren't syntax all the way down or content all the way up. *Minds and Machines 28*(3), 543–567.

Ganesalingam, M. (2013). *The language of mathematics*. Springer.

Hadamard, J. (1945). *An essay on the psychology of invention in the mathematical field*. Dover.

Launchbury, J. and S. L. Peyton Jones (1994). Lazy functional state threads. *ACM SIGPLAN Notices 29*(6), 24–35.

Mills, J. W. (2008). The nature of the extended analog computer. *Physica D: Nonlinear Phenomena 237*(9), 1235–1256.

Moggi, E. (1988). *Computational Lambda-Calculus and Monads*. LFCS, University of Edinburgh.

Moggi, E. (1991). Notions of computation and monads. *Information and computation 93*(1), 55–92.

Nannicini, G. (2020). An introduction to quantum computing, without the physics. *SIAM Review 62*(4), 936–981.

Rieffel, E. and W. Polak (2000). An introduction to quantum computing for non-physicists. *ACM Computing Surveys (CSUR) 32*(3), 300–335.

Rubel, L. A. (1993). The extended analog computer. *Advances in Applied Mathematics 14*(1), 39–50.

Wadler, P. (1993). Monads and composable continuations. *Lisp and symbolic computation 7*(1), 39–56.