ARM · Novel Algorithms Workshop 2020

8th September 2020

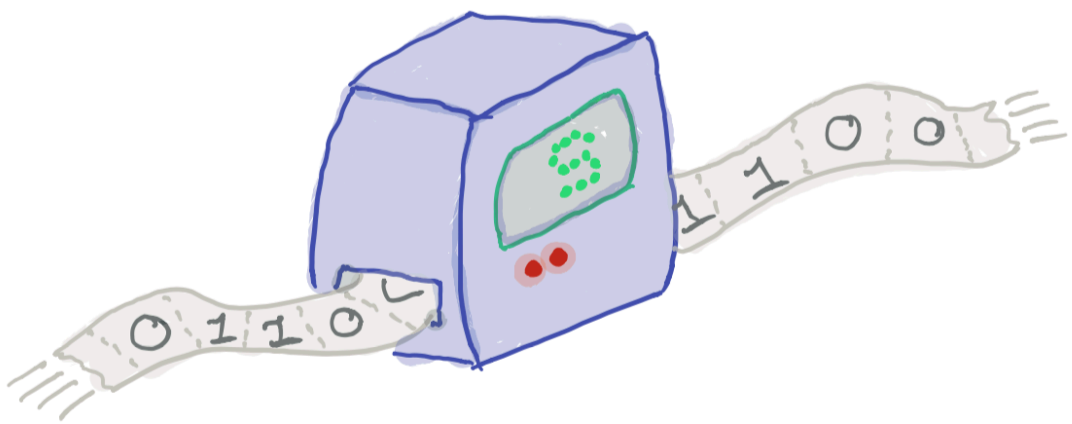
Reversible Computing:

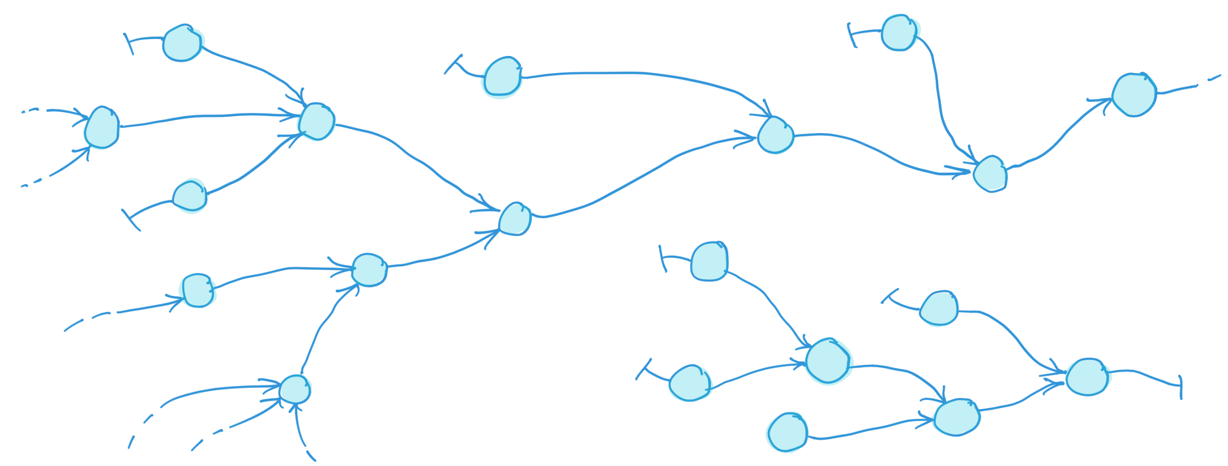
Reuniting Computers & The Laws of Physics

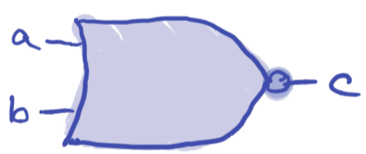
William Earley

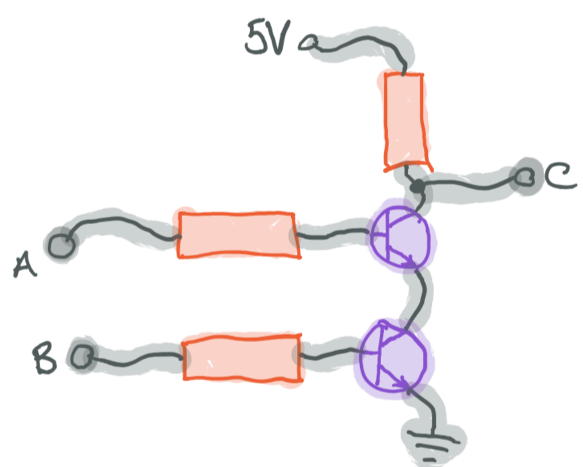
Micklem Lab · DAMTP

Irreversibility in Computing









a	b	c
0	0	1
0	1	1
1	0	1
1	1	0

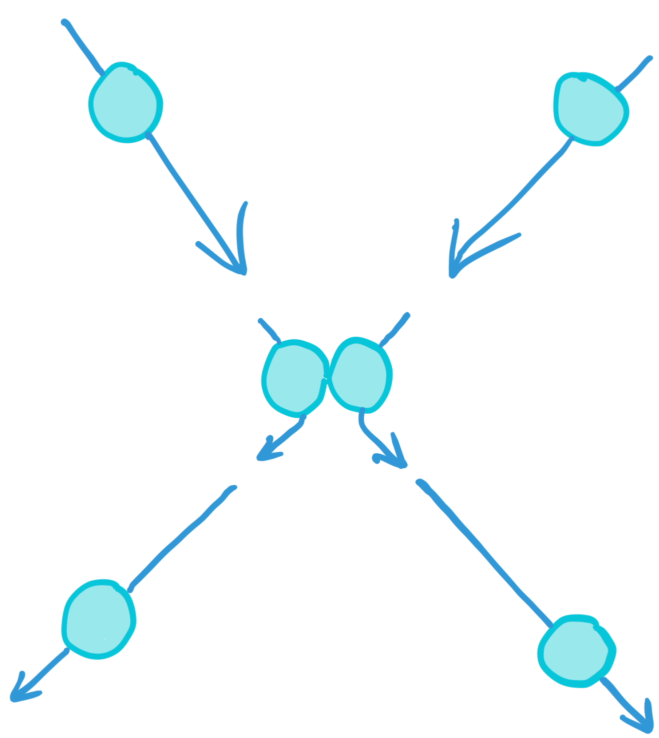
$$\mathtt{TRUE} = \lambda x.\lambda y.x$$

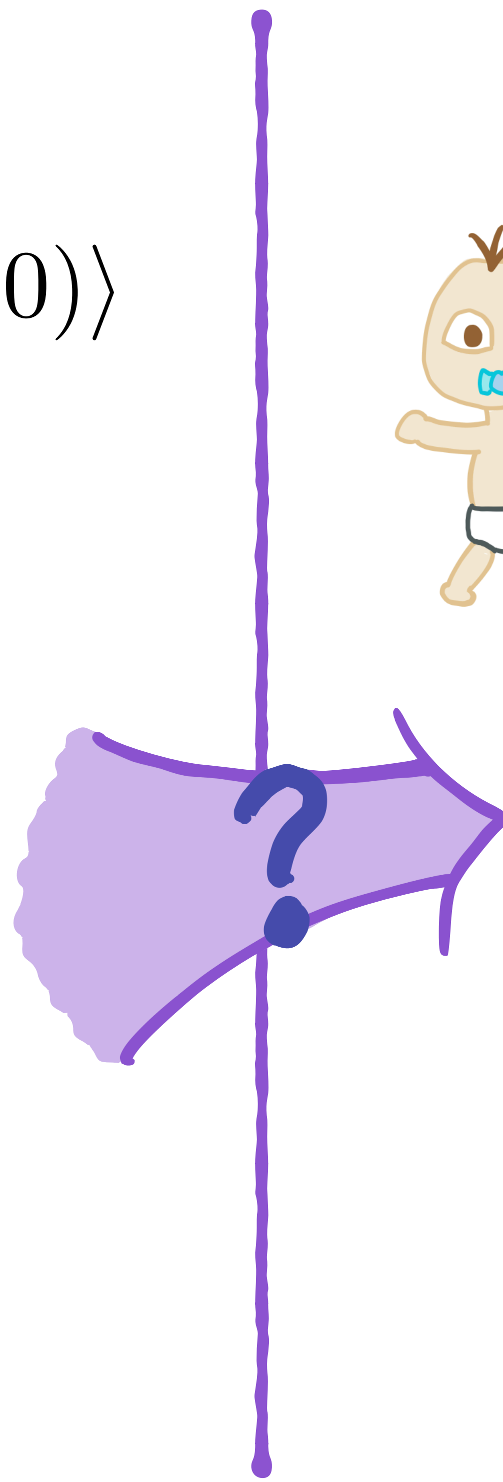
```
>>> sum(range(10))
45
```

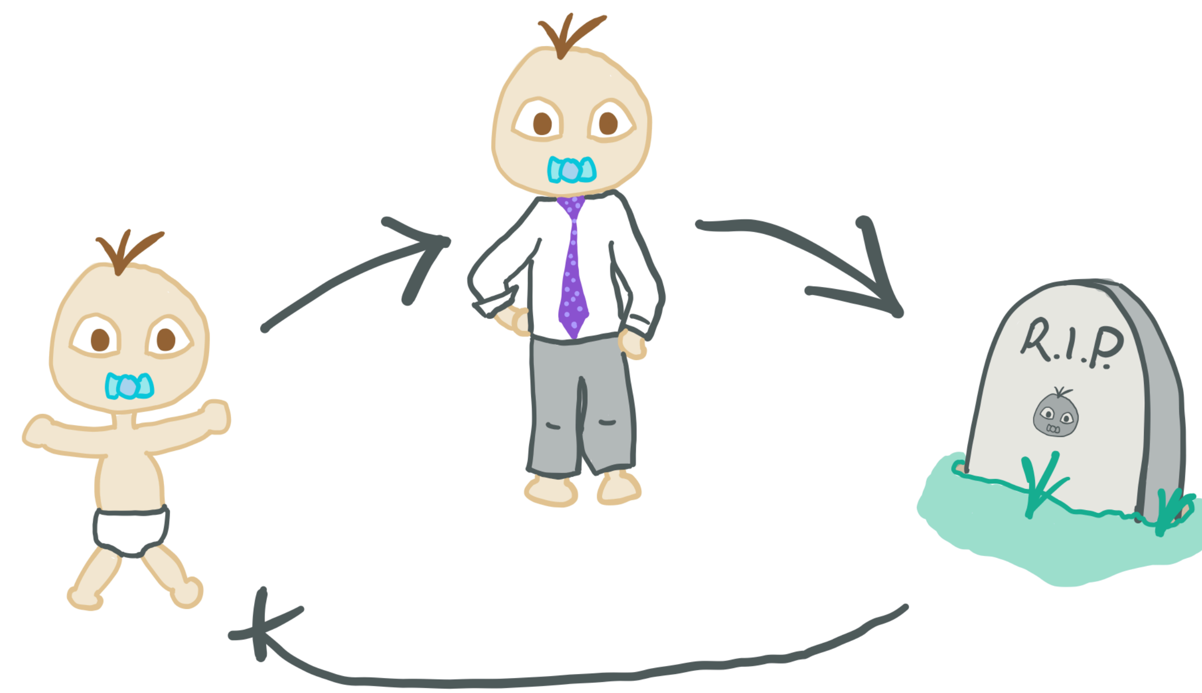
```
(curl -s 'wttr.in/Cambridge' | grep -i rain) 2>/dev/null
```

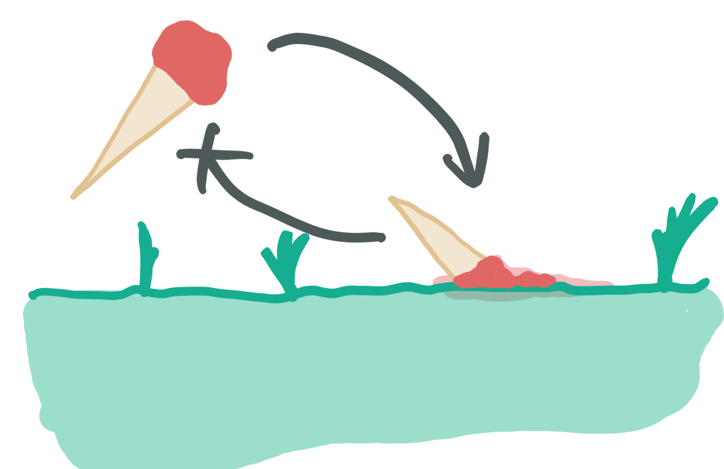
Irreversibility in Physics

$$|\psi(t)\rangle = e^{-i\hat{H}t/\hbar}\,|\psi(0)\rangle$$

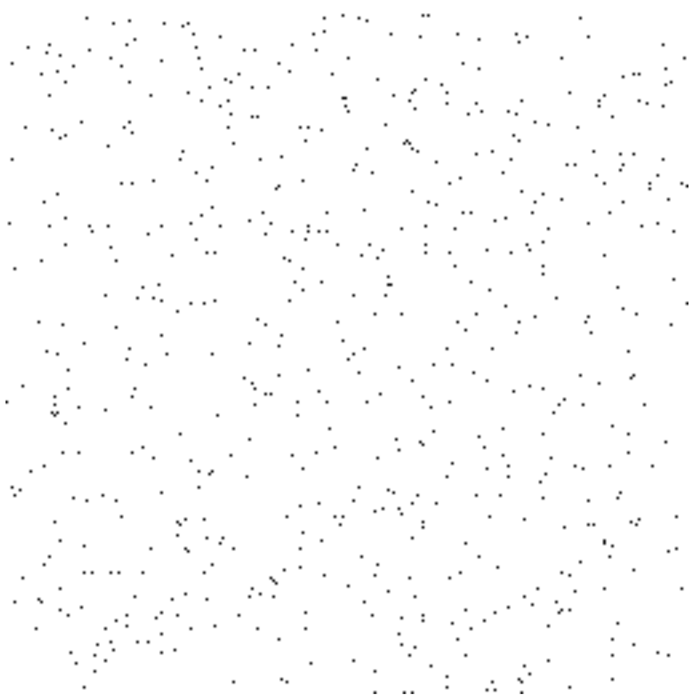




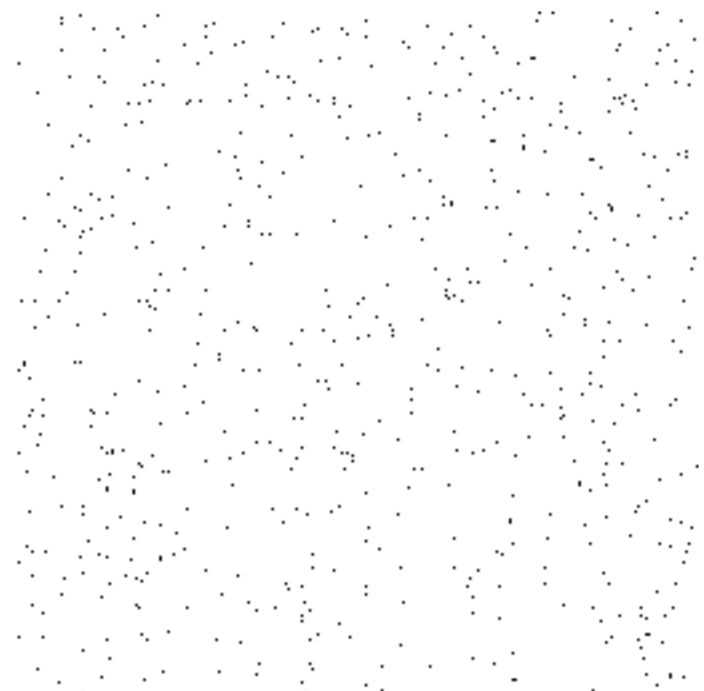




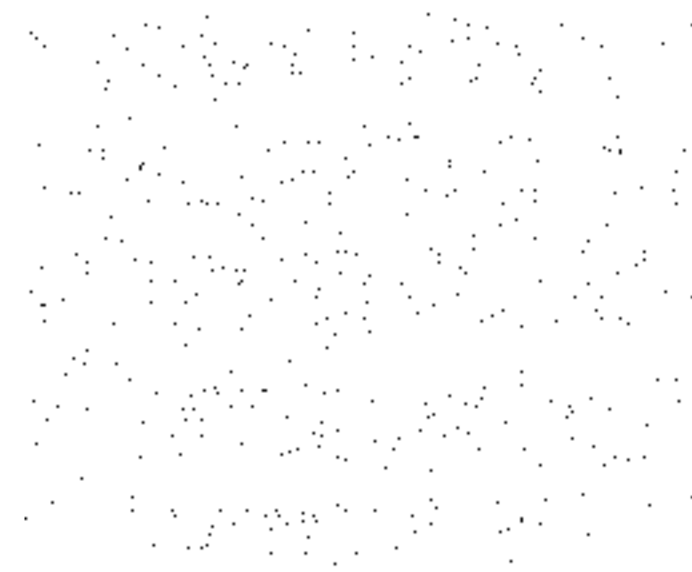
Simulating Irreversibility



[0,100]



[100,0]



[-200,-300]

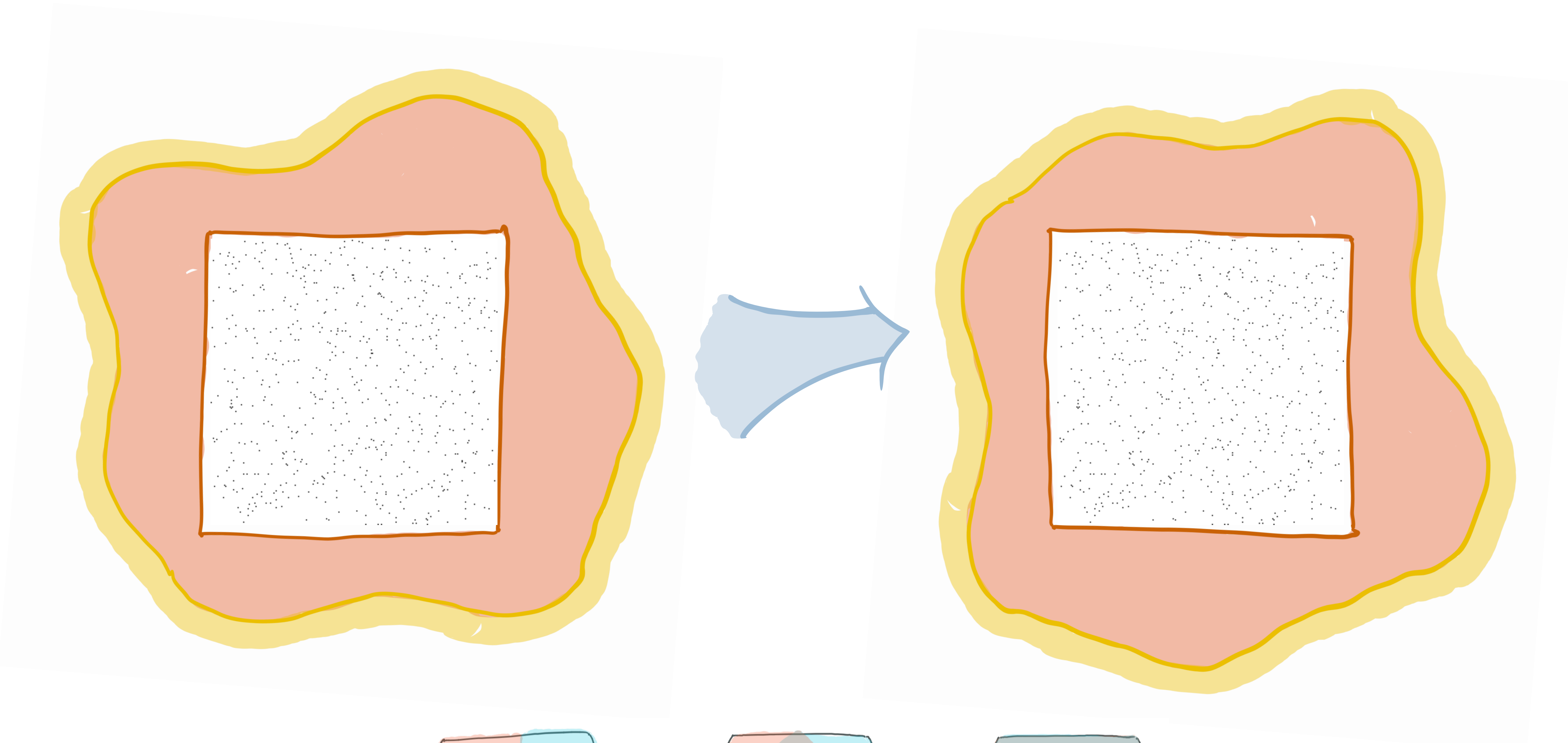
T-symmetry

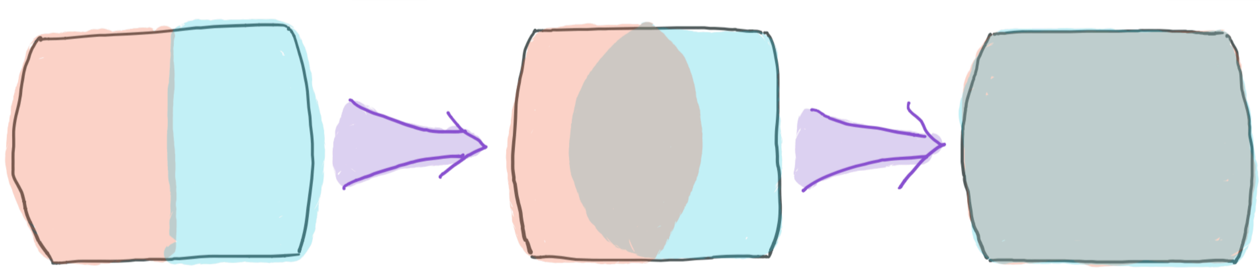
Simulating Irreversibility

Simulating Irreversibility

John DeMoss and Kevin Cahill — 2007 — 'Laminar Flow' — University of New Mexico — Dept. Physics & Astronomy

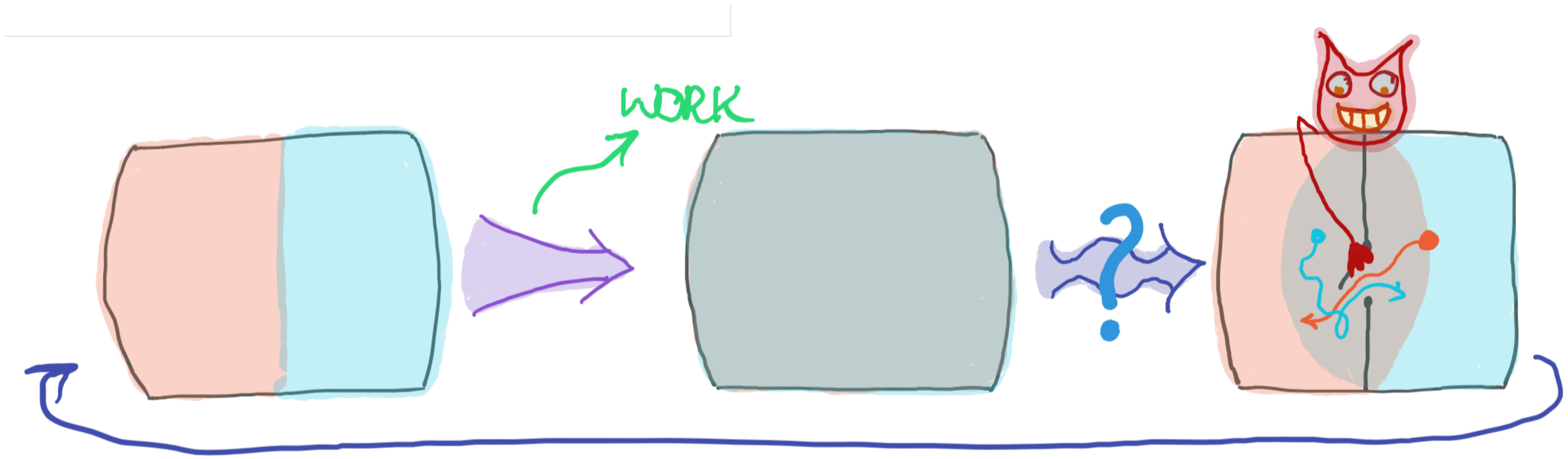
Simulating Irreversibility

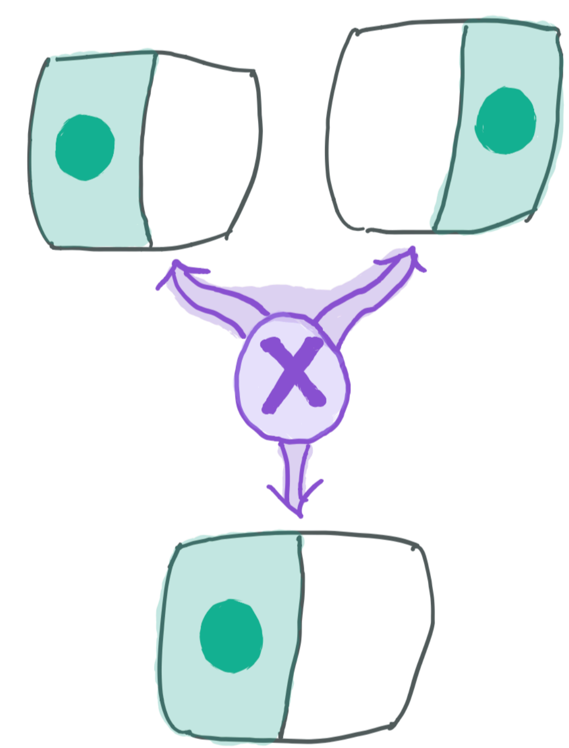


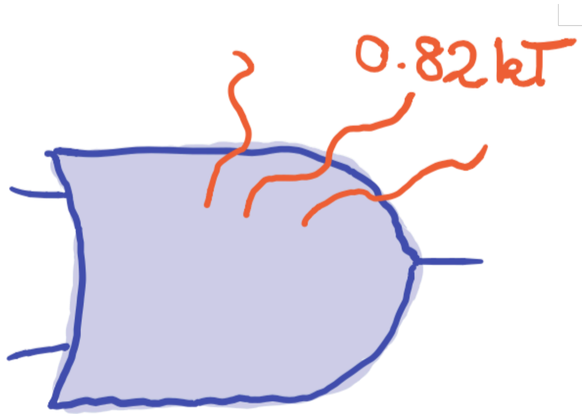


Maxwell's Dæmon

Connecting Thermodynamics to Information Theory







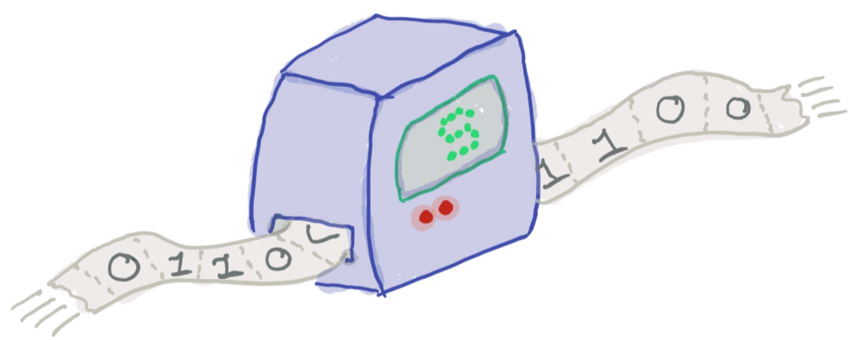
$$I = -\sum p_i \log p_i$$

$$\Delta q_h \geq k_B T \Delta I$$

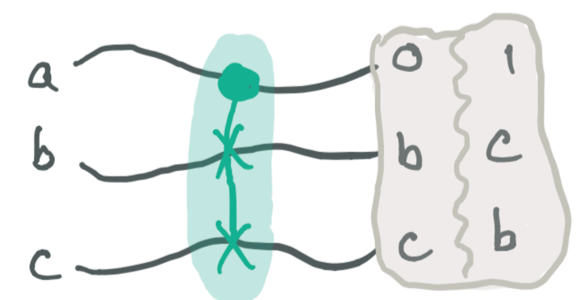
Leo Szilard — 1929 — 'On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings' — Zeitschrift für Physik
Rolf Landauer — 1961 — 'Irreversibility and heat generation in the computing process' — IBM J. Res. Dev.

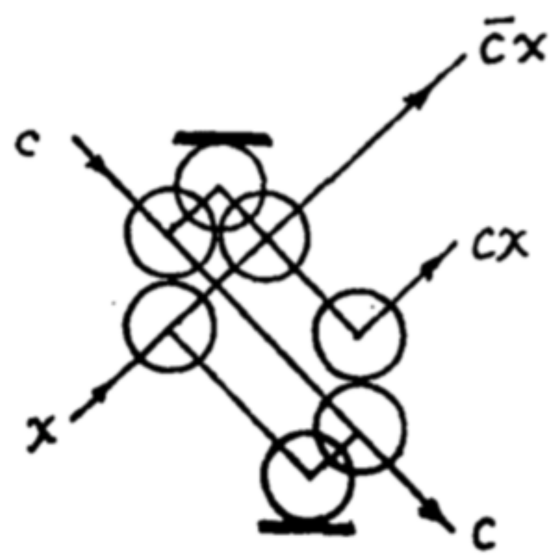
Computing Reversibly

Bennett's RTM[2]

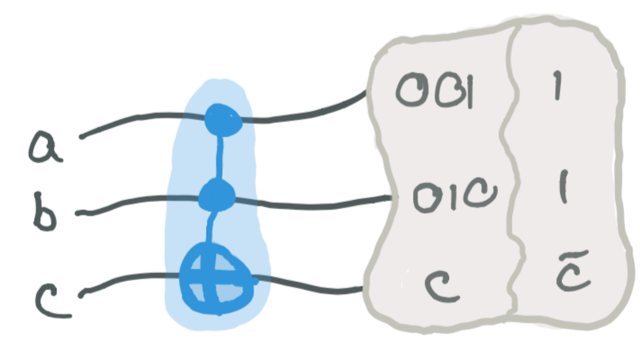


CSWAP[3] / Fredkin Gate





Switch Gate[3]



CCNOT[1] / Toffoli Gate

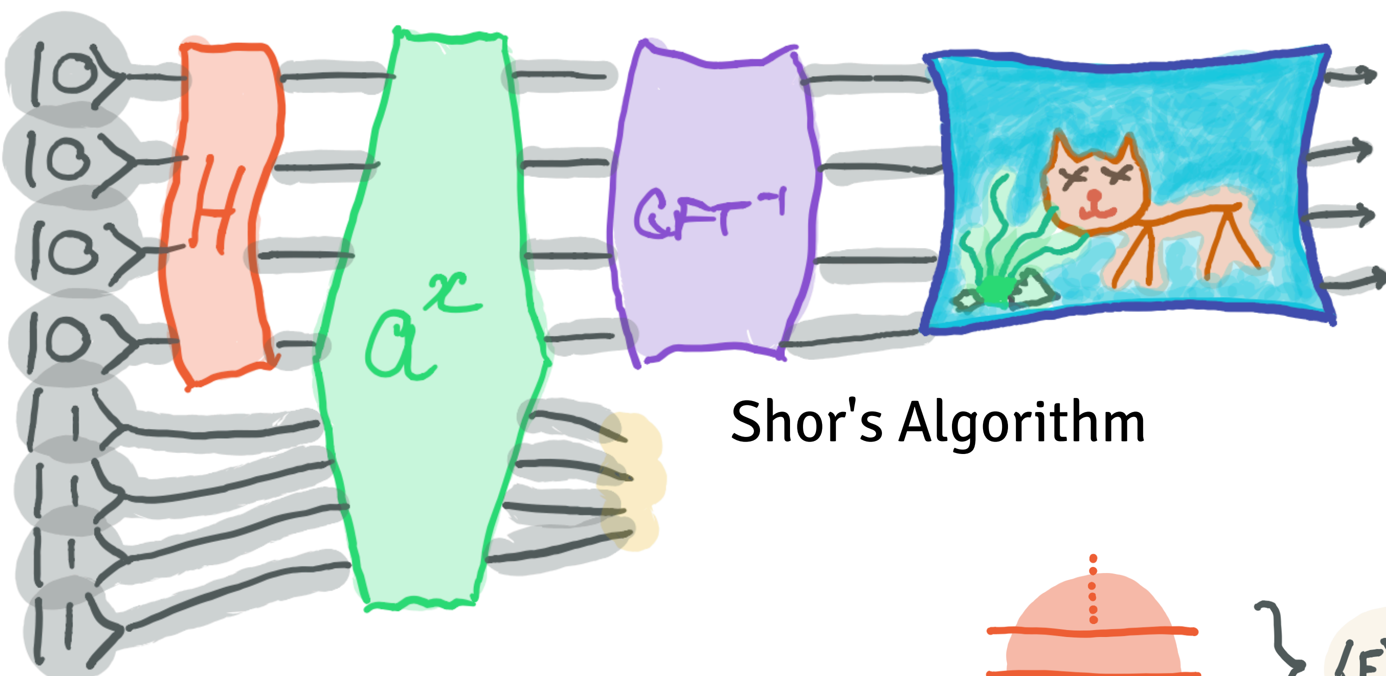
[1]Rolf Landauer — 1961 — 'Irreversibility and heat generation in the computing process' — IBM J. Res. Dev.
[2]Charles H Bennett — 1973 — 'Logical Reversibility of Computation' — IBM J. Res. Dev.
[3]Edward Fredkin and Tommaso Toffoli — 1981 — 'Conservative Logic' — Collision-Based Computing

"Even if classical balls could be shot with perfect accuracy into a perfect apparatus, fluctuating tidal forces from turbulence in the atmospheres of nearby stars would be enough to randomise their motion within a few hundred collisions. Needless to say, the trajectory would be spoiled much sooner if stronger nearby noise sources (e.g., thermal radiation and conduction) were not eliminated."

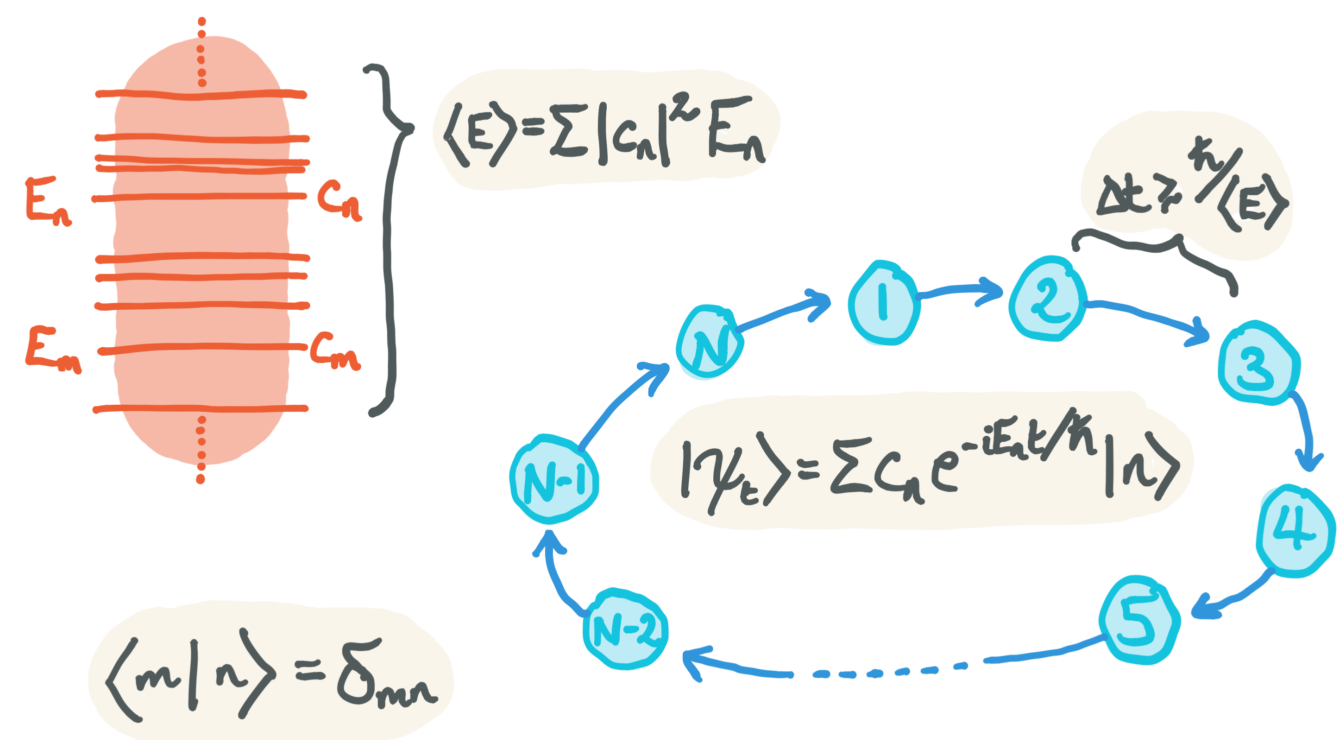
– *Charles Bennett*

Charles H Bennett — 1982 — 'The thermodynamics of computation—a review' — Int. J. Theor. Phys.

Quantum Computers



Shor's Algorithm



$$i\hbar\partial_t\,|\psi(t)\rangle = \mathbf{H}\,|\psi(t)\rangle$$

$$|\psi(n\,\delta t)\rangle = \mathbf{U}^n\,|\psi(0)\rangle$$

$$\mathbf{U} = e^{-i\mathbf{H}\delta t/\hbar}$$

$$\mathbf{U}^{-1} \equiv \mathbf{U}^*$$

Peter W Shor — 1994 — 'Algorithms for quantum computation: discrete logarithms and factoring' — Proc. 35th FOCS
Norman Margolus and Lev B Levitin — 1998 — 'The maximum speed of dynamical evolution' — Physica D

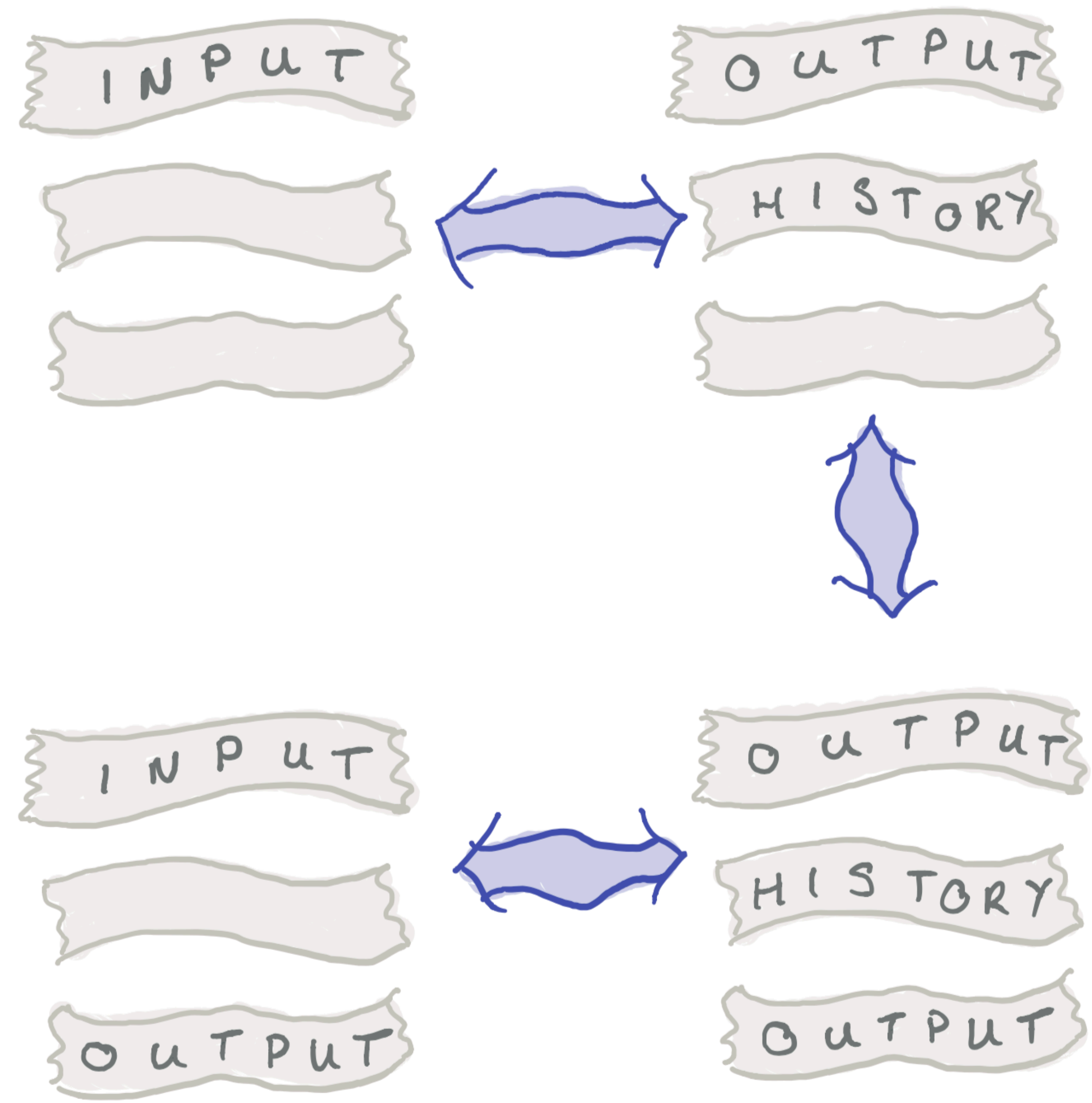
Garbage

The Simpsons — 'Trash of the Titans' — S09E22

Garbage

$$f : x \mapsto f(x)$$

$$\tilde{f} : x \leftrightarrow (x, f(x))$$



Charles H Bennett — 1973 — 'Logical Reversibility of Computation' — IBM J. Res. Dev.

Pebbling

TABLE 2
Reversible simulation in time $O(T^{\log 3/\log 2})$ and space $O(S \cdot \log T)$.

Stage	Action	Checkpoints in storage (0 = initial ID, checkpoint $j = (jm)$th step ID)								
0	Start	0								
1	Do segment 1	0	1							
2	Do segment 2	0	1	2						
3	Undo segment 1	0		2						
4	Do segment 3	0		2	3					
5	Do segment 4	0		2	3	4				
6	Undo segment 3	0		2		4				
7	Do segment 1	0	1	2		4				
8	Undo segment 2	0	1			4				
9	Undo segment 1	0				4				
10	Do segment 5	0				4	5			
11	Do segment 6	0				4	5	6		
12	Undo segment 5	0				4		6		
13	Do segment 7	0				4		6	7	
14	Do segment 8	0				4		6	7	8
15	Undo segment 7	0				4		6		8
16	Do segment 5	0				4	5	6		8
17	Undo segment 6	0				4	5			8
18	Undo segment 5	0				4				8
19	Do segment 1	0	1			4				8
20	Do segment 2	0	1	2		4				8
21	Undo segment 1	0		2		4				8
22	Do segment 3	0		2	3	4				8
23	Undo segment 4	0		2	3					8
24	Undo segment 3	0		2						8
25	Do segment 1	0	1	2						8
26	Undo segment 2	0	1							8
27	Undo segment 1	0								8

Charles H Bennett — 1989 — 'Time/Space Trade-Offs for Reversible Computation' — SIAM J. Comp.

Thinking Reversibly

- **Bennett's algorithms**
  - efficient embedding of irreversibility ✓
  - not easily composable ✗
  - injective rather than bijective

Thinking Reversibly

- True reversible programming: make use of bijections

$$+ : (a, b) \mapsto (a, b, a + b)$$ ✗

$$+_1 : (a, b) \leftrightarrow (a + b, b)$$
$$+_2 : (a, b) \leftrightarrow (a + b, a - b)$$ ✓

- By appropriately exploiting information in the output, can reduce or even eliminate the 'garbage' output

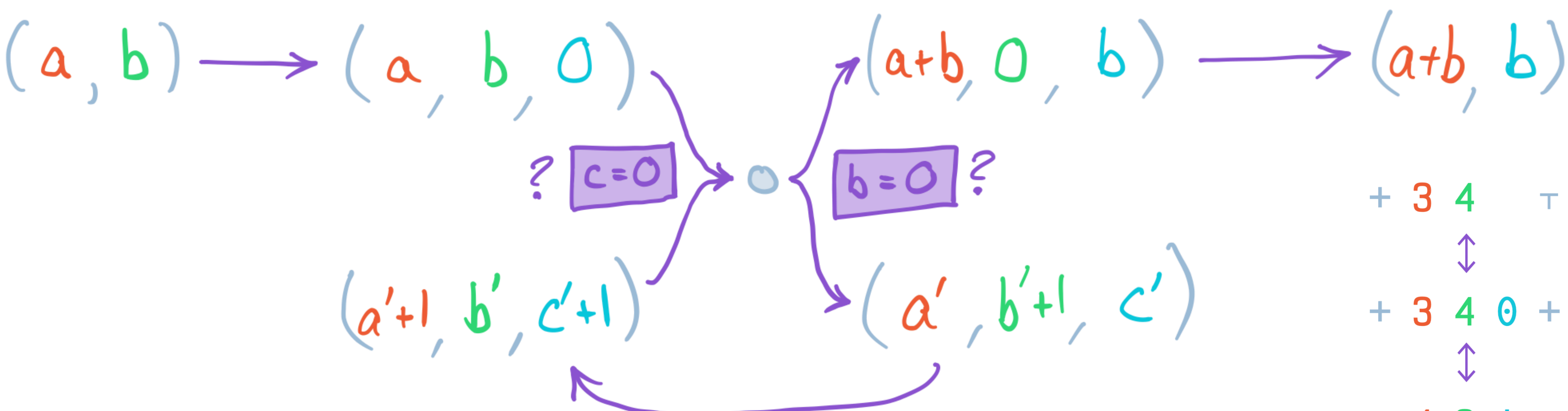
- The remaining garbage, if purposefully constructed, is often found to be useful for further computation

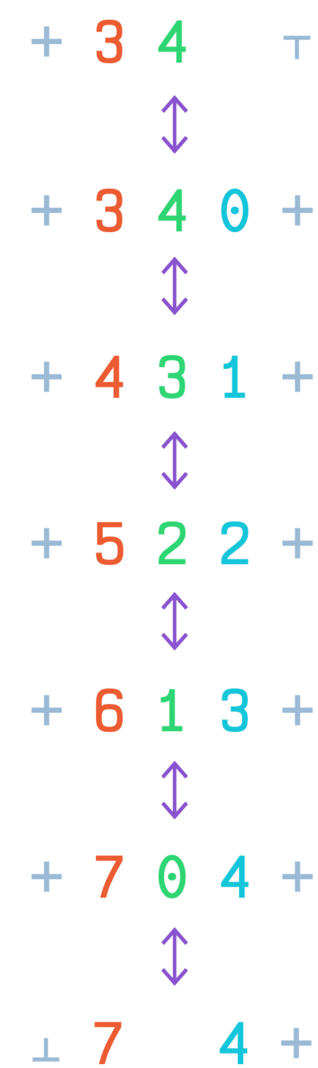
- For example, it turns out that $+_1$ and some reversible looping is sufficient for

$$\text{square} : n \leftrightarrow n^2$$

Example 1

Addition/Subtraction



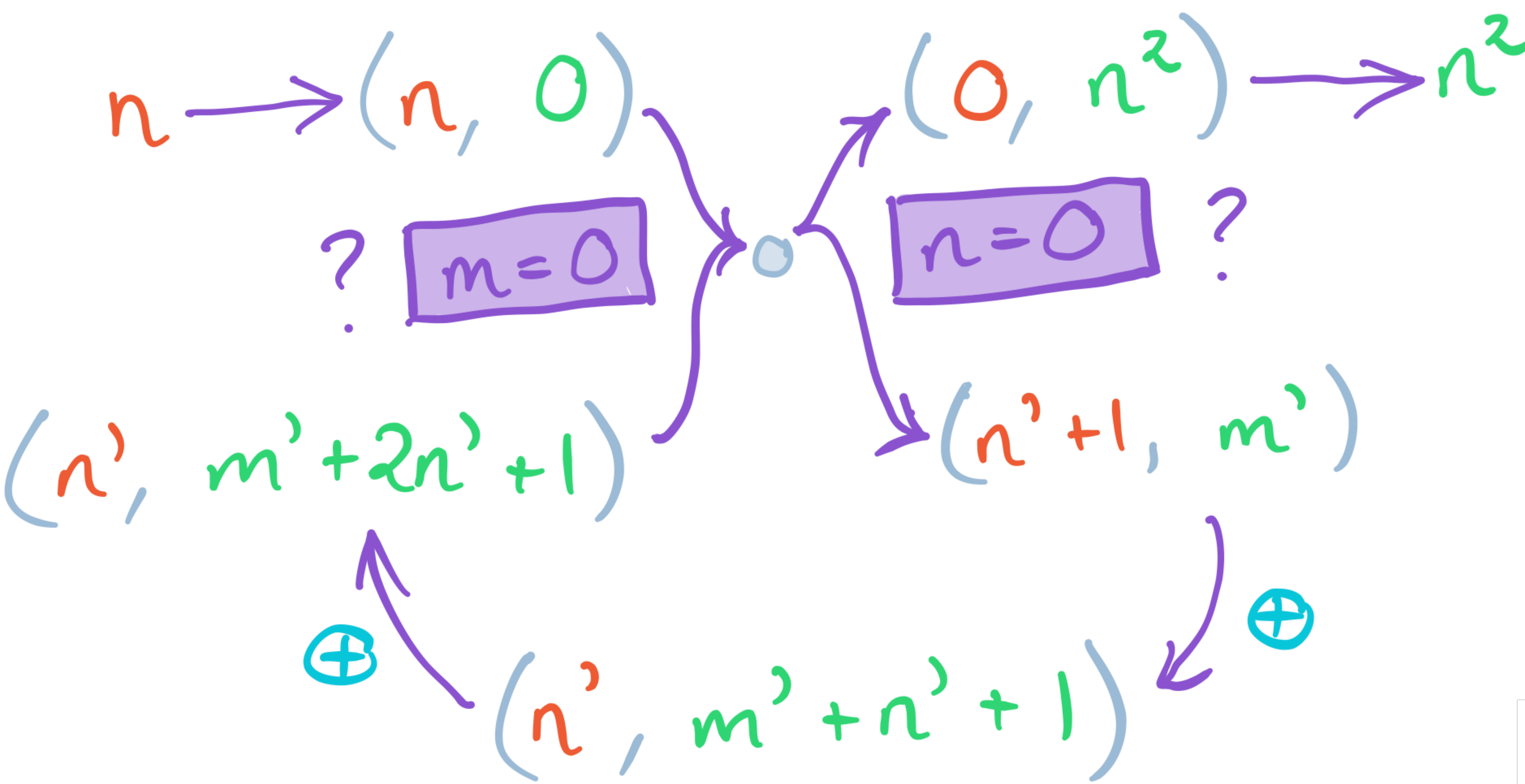


- Reversible analogue of Peano definition
- Straightforward extension to Integers
- Can also implement for Rationals and Reals

Example 2

Square/Square Root

$$n^2 = \sum_{k=0}^{n-1}(2k+1)$$



$$i\hbar\frac{\partial}{\partial t}\left|\psi\right\rangle = \underbrace{\left(-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right)}_{\mathbf{H}}\left|\psi\right\rangle$$

Example 3

Schrödinger's Equation

$$i\hbar\partial_t\left|\psi(t)\right\rangle = \mathbf{H}\left|\psi(t)\right\rangle$$

$$\left|\psi(n\,\delta t)\right\rangle = \mathbf{U}^n\left|\psi(0)\right\rangle$$

$$\mathbf{U} = e^{-i\mathbf{H}\delta t/\hbar}$$

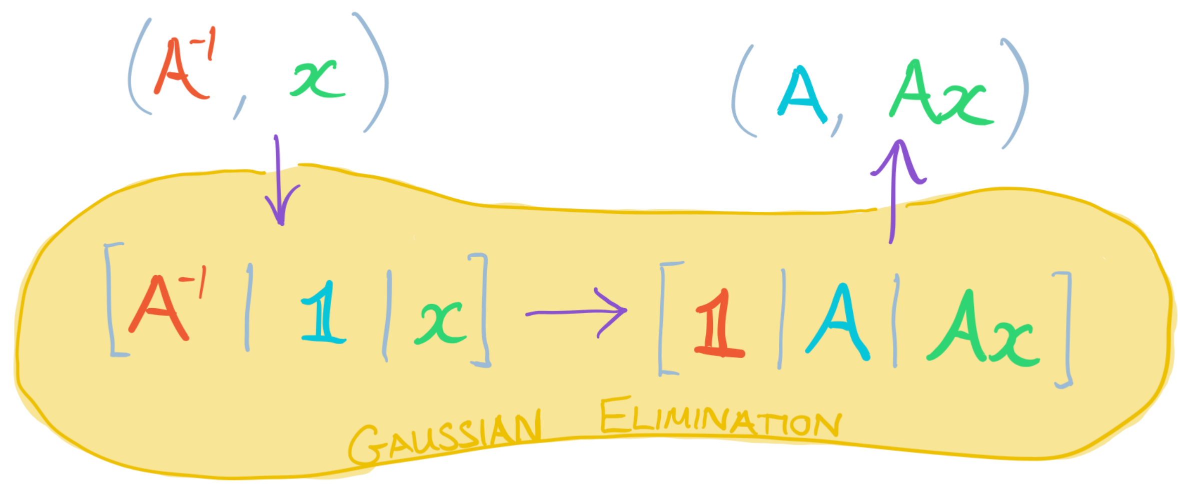
$$\mathbf{U}^{-1} \equiv \mathbf{U}^*$$

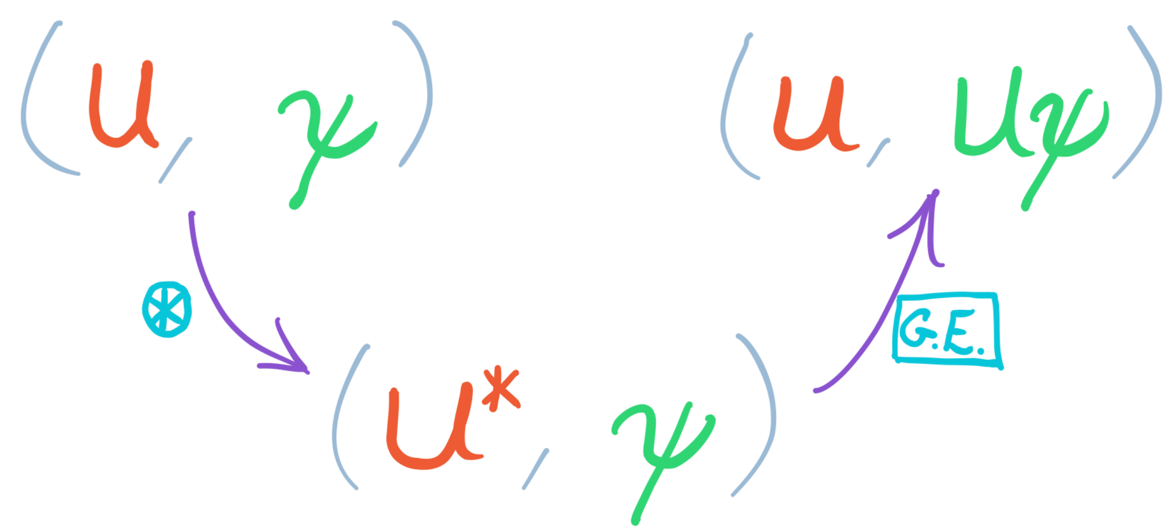
```
;; This subroutine changes a point in the real wave DEST
;; according to the curvature in the corresponding
;; neighborhood in the real wave SRC, and the potential at
;; the given point.
(defsub pfunc (dest src i alphas epsilon)
  ((dest _ i) += ((alphas _ i) */ (src _ i)))
  ((dest _ i) -= (epsilon */ (src _ ((i + 1) & 127))))
  ((dest _ i) -= (epsilon */ (src _ ((i - 1) & 127)))))

;; Take turns updating the two components of the wave in a
;; way such that they will chase each other around in
;; (higher-dimensional) circles.
(defsub schstep (psiR psiI alphas epsilon)
  ;; psiR += f(psiI)
  (for i = 0 to 127
     ;; psiR[i] += pfunc(psiI,i)
     (call pfunc psiR psiI i alphas epsilon))
  ;; psiI -= f(psiR)
  (for i = 0 to 127
     ;; psiI[i] -= pfunc(psiR,i)
     (rcall pfunc psiI psiR i alphas epsilon)))

;; Print the current wave to the output stream.
(defsub printwave (wave)
  (for i = 0 to 127
       (printword (wave _ i)))
  (println))

;; Main program, goes by the name of SCHROED.
(defmain schroed
  (for i = 1 to 1000 ;Time for electron to fall to well bottom.
     (call schstep psiR psiI alphas epsilon)
     ;; Print both wave components.
     (call printwave psiR)
     (call printwave psiI)))
```





Michael P Frank — 1997 — 'The R Programming Language and Compiler' — MIT Rev. Comp. Proj. Memo

Reversible Languages

```
(defsub mult (m1 m2 prod)
   ;; Use grade-school algorithm:
   (for pos = 0 to 31
      (if (m1 & (1 << pos)) then
         (prod += (m2 << pos)))))
```

R[1]

```
(defun fact (n) (assert (and (integerp n) (> n 0)))
  (if (onep n) #'onep n (* n (fact (1- n)))))
```

Ψ-LISP[2]

```
procedure fib                procedure main_fwd
  if n=0 then x1 += 1          n += 4
              x2 += 1          call fib
       else n -= 1
              call fib       procedure main_bwd
              x1 += x2         x1 += 5
              x1 <=> x2        x2 += 8
  fi x1=x2                     uncall fib
```

Janus[3]

```
type Nat4 = Bool * Bool * Bool * Bool

add1 :: Nat4 ↔ Nat4 :: sub1
| (a, b, c, False)          ↔  (a, b, c, True)
| (a, b, False, True)       ↔  (a, b, True, False)
| (a, False, True, True)    ↔  (a, True, False, False)
| (False, True, True, True) ↔  (True, False, False, False)
| (True, True, True, True)  ↔  (False, False, False, False)
```

Theseus[4]

[1]Michael P Frank — 1997 — 'The R Programming Language and Compile' — MIT Rev. Comp. Proj. Memo
[2]Henry G Baker — 1992 — 'NREVERSAL of Fortune—the Thermodynamics of Garbage Collection' — Intl. Workshop on Memory Management
[3]Tetsuo Yokoyama and Robert Glück — 2007 — 'A reversible programming language and its invertible self-interpreter.' — Partial evaluation and semantics-based program manipulation.
[4]Roshan P James and Amr Sabry — 2014 — 'Theseus: a high-level language for reversible computation.' — Reversible Computation

Reversible Languages

```
str(s|count) {
        s [
                s temp s temp s temp s temp
                s temp s temp s temp s temp
                count str(s|count)len | count
                temp s temp s temp s temp s
                temp s temp s temp s temp s
        ] s
} (s|count)len

revloop(in|count|out) {
        count [
                in temp in temp in temp in temp in
                   temp in temp in temp in temp in
                out temp out temp out temp out temp out
                    temp out temp out temp out temp out
                revloop(in|count|out)poolver
        ] count
} (in|count|out)poolver

reverse(in|out) {
        str(in|count)len
        revloop(in|count|out)poolver
        nel(count|out)rts
} (in|out)esrever

(ent|in) { reverse(in|out)esrever } (out|ent)
```

Kayak[5]

```
swap-fl1 swap-fl2 : {a b c : U} → PLUS a (PLUS b c) ↔ PLUS c (PLUS b a)
swap-fl1 = assocl₊ ⊙ swap₊ ⊙ (id↔ ⊕ swap₊)

swap-fl2 = (id↔ ⊕ swap₊) ⊙
           assocl₊ ⊙
           (swap₊ ⊕ id↔) ⊙
           assocr₊ ⊙
           (id↔ ⊕ swap₊)
```

Π[6]

```
-- Peano definition of natural numbers
data Z;
data S n;

-- reversible Peano addition
      + a b _ = + a b Z +;
+ (S a) b c + = + a (S b) (S c) +;
  + Z a+b a + = _ a a+b +;

-- termination conditions
! + a b   _;
! _ a a+b +;
```

```
+ 3 4 _
    ↕
+ 3 0 4 +
    ↕
+ 4 1 3 +
    ↕
+ 5 2 2 +
    ↕
+ 6 3 1 +
    ↕
+ 7 4 0 +
    ↕
_ 7 4 +
```

alethe[7]

[5] Ben Rudiak-Gould — 2002 — Esoteric Programming Language Awards
[6] Jacques Carette, Roshan P. James, and Amr Sabry — 2018 — 'Embracing the laws of physics: Three reversible models of computation.' — arXiv
[7] William Earley — 2019 — DNA25 Poster; Paper in Preparation

Quantum Languages

```
procedure grover(int n) {
  int l=floor(log(n,2))+1;        // no. of qubits
  int m=ceil(pi/8*sqrt(2^l));     // no. of iterations
  int x;
  int i;
  qureg q[l];
  qureg f[1];

  {
    reset;
    Mix(q);                 // prepare superposition
    for i= 1 to m {         // main loop
      query(q,f,n);         // calculate C(q)
      CPhase(pi,f);         // negate |n>
      !query(q,f,n);        // undo C(q)
      diffuse(q);           // diffusion operator
    }
    measure q,x;            // measurement
    print "measured",x;
  } until x==n;
}
```

QCL

```
import Quipper

spos :: Bool -> Circ Qubit
spos b = do q <- qinit b
            r <- hadamard q
            return r
```

Quipper

$$
\begin{aligned}
&\mathit{toff} : \mathbf{Q_2} \multimap \mathbf{Q_2} \multimap \mathbf{Q_2} \multimap \mathbf{Q_2} \otimes (\mathbf{Q_2} \otimes \mathbf{Q_2})\\
&\mathit{toff}\ c\ x\ y = \mathbf{if}^{\circ}\ c\\
&\qquad\qquad \mathbf{then}\ (\mathrm{qtrue},\ \mathit{cnot}\ x\ y)\\
&\qquad\qquad \mathbf{else}\ \ (\mathrm{qfalse}, (x, y))
\end{aligned}
$$

QML

Q#

```
# Solve a circuit-satisfiability problem.

!include <gates>

!use_macro not1 not_x4
not_x4.$A = x3
not_x4.$Y = $x4

!use_macro or2 or_x5
or_x5.$A = x1
or_x5.$B = x2
or_x5.$Y = $x5
```

QMASM

```
def solve[n:!ℕ](bits:!𝔹^n){
  // prepare superposition between 0 and 1
  x:=H(0:𝔹);
  // prepare superposition between bits and 0
  qs := if x then bits else (0:int[n]) as 𝔹^n;
  // uncompute x
  forget(x=qs[0]); // valid because `bits[0]==1`
  return qs;
}
```

Silq

qCGL

LIQUi|⟩

Q|SI⟩

QPL

Q

QFC

Reversible vs Irreversible

- Equipotent
- Fundamentally same resource usage
  - Reversible makes erasure explicit
  - Can reduce net resource usage
- Reversibility *encourages* a more careful and efficient approach to resource usage

Hybrid

- What might a reversibility exploitation story look like?
    - Reversible processor with entropy-dissipation co-processor
    - Expose irreversible abstraction layer over reversible operations (or reversibility aware compiler to insert erasure operations)
    - Gradual software transition
- Intermediate benefit: transition from actively dissipative transistors to low energy components

Linear Types

- Linear logic is a superset of Quantum logic
  - Capable of describing reversible computation
  - Not strictly reversible though, e.g.
    $\pounds 1 \multimap (\text{sweets} \ \& \ \text{crisps} \ \& \ \text{drink})$
- Gateway drug to reversible programming
  - e.g. new Haskell extension to specify that an argument *must* be consumed (forbids *implicit* erasure)

Research Timeline

Theory

- Information–Entropy Connection (Szilard '29, Landauer '61)
- Foundation of Reversible Computation (Bennett '73)
- Ballistic Computation (Toffoli+Fredkin '81)
- Programming Languages (Ψ-LISP: Baker '92, R: Frank '97, …)
- Analyses of Physical Limits (Frank '99, Lloyd '02, …)

Research Timeline

Application

- Helical Logic (Merkle '96)
- Passive Transistor Logic (de Vos et al '99)
- Quantum Dot Cellular Automata (Lent et al '01–'03)
- nSQUID Josephson-Junction Circuits (Semenov et al '03)
- Asynchronous Ballistic Fluxon Logic (Frank et al '17)

...along with research into effective synthesis of the relevant circuits

N.B. These technologies mostly predate the papers listed; to the best of my knowledge these are the first rigorous applications of each technology to the field of reversible computing.

Engines of Parsimony

I. How fast can any computer run, given some spacetime region and power supply and taking into account all areas of physics?

II. What happens when we try to communicate between/ synchronise reversible computers?

III. What happens when we try to share some common resource between asynchronous reversible computers?

William Earley — 2020 — 'Engines of Parsimony: Part I; Limits on Computational Rates in Physical Systems' — arXiv
William Earley — 2020–2021 — Parts II & III — In Preparation

Thank you!

UNIVERSITY OF CAMBRIDGE

EPSRC

Engineering and Physical Sciences Research Council

Department of Applied Mathematics and Theoretical Physics (DAMTP)

William Earley — 2016–2021 — 'Modelling approaches to molecular computation' — EPSRC Project Reference 1781682