Quantum Circuits for Coherent Conditional Iteration

Peter M. Hines — University of York

QISW – Oxford – March 2012

With thanks to:

P. Scott (Ottawa)

Peter.Hines@cs.york.ac.uk

If we had a quantum computer ...

What could we do with it?

Quantum Computing has many general-purpose techniques:

- QM Period-finding,
- Grover fixed-point search,
- Quantum Counting,
- etc. ...

Crucial questions:

What can we do with these?

Why is it so hard to find useful applications?

Peter.Hines@cs.york.ac.uk

How quantum algorithms are arranged

Never mind -how- they work ... what do they look like?

A quantum computer:

1. Takes, as input, a *classical computation*.
2. Performs some *quantum magic*. (a technical term!)
3. Returns some *global information* about the input.

In a quantum algorithm:

How can a *classical computation* be an input?

We need a **quantum oracle** for the classical computation.

These are:

- Unitary maps that 'look classical' on some fixed basis (the *computational basis*.)
- In general, very tedious to construct.

Peter.Hines@cs.york.ac.uk

What is a quantum oracle?

In the classical world:

F is a reversible function on n bits.

For any bit-string b, both $|b\rangle$ and $|F(b)\rangle$ are basis vectors.

In the quantum world:

An **oracle** for F is simply *' F lifted to the quantum setting'.*

It is a quantum operation U_F satisfying

$$U_F\,|b\rangle = |F(b)\rangle \quad \text{for any bitstring } b$$

Peter.Hines@cs.york.ac.uk

How do oracles behave on arbitrary inputs?

These oracles are **unitary**, and hence linear.

For an input that is a superposition of basis vectors

$$|\Psi\rangle = \alpha |b_1\rangle + \beta |b_2\rangle$$

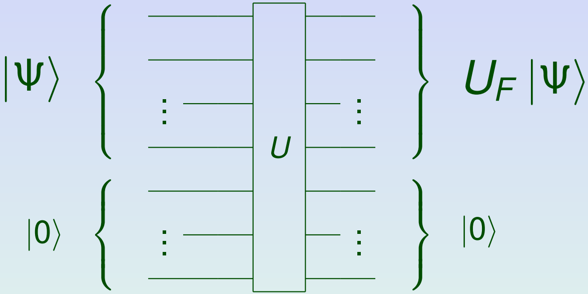
It returns the corresponding superposition of output vectors

$$U_F |\Psi\rangle = \alpha |F(b_1)\rangle + \beta |F(b_2)\rangle$$

This is the key requirement!.

Oracles with ancillas

Oracles often use *ancillas* — additional registers that play a part in the computation:



The ancilla must *start* and *finish* in some fixed state, to ensure it is *not entangled* with the result we want.

Peter.Hines@cs.york.ac.uk

How to create quantum oracles?

The usual approach

1. implement F classically, using:
   - reversible binary logic gates,
   - without any loops, or feedback,
   - ensuring all 'extra bits' start & finish in the same state.
2. Replace each gate by its 'quantum counterpart'.

The problem ...

This rules out any use of *feedback*, *conditional iteration*, *recursion*, *fixed point operations*, *the von Neumann architecture*, etc.

Peter.Hines@cs.york.ac.uk

How to create quantum oracles?

The usual approach

1. implement *F* classically, using:
   - reversible binary logic gates,
   - without any loops, or feedback,
   - ensuring all 'extra bits' start & finish in the same state.

2. Replace each gate by its 'quantum counterpart'.

The problem ...

This rules out any use of *feedback*, *conditional iteration*, *recursion*, *fixed point operations*, *the von Neumann architecture*, etc.

Peter.Hines@cs.york.ac.uk

How to create quantum oracles?

The usual approach

1. implement F classically, using:
   - reversible binary logic gates,
   - without any loops, or feedback,
   - ensuring all 'extra bits' start & finish in the same state.

2. Replace each gate by its 'quantum counterpart'.

The problem ...

This rules out any use of *feedback*, *conditional iteration*, *recursion*, *fixed point operations*, *the von Neumann architecture*, etc.

Peter.Hines@cs.york.ac.uk

Oracles from Turing machines?

How about computations defined by a Turing machine?

Quantum Turing machines

- Introduced by D. Deutsch (1985).

  Quantum theory, the Church-Turing principle and the universal quantum computer

- Criticised by J. Myers (1997)

  Can a Universal Computer be Fully Quantum?

Peter.Hines@cs.york.ac.uk

Oracles from Turing machines?

How about computations defined by a Turing machine?

Quantum Turing machines

- Introduced by D. Deutsch (1985).

  Quantum theory, the Church-Turing principle and the universal quantum computer

- Criticised by J. Myers (1997)

  Can a Universal Computer be Fully Quantum?

Peter.Hines@cs.york.ac.uk

Oracles from Turing machines?

How about computations defined by a Turing machine?

Quantum Turing machines

- Introduced by D. Deutsch (1985).

  Quantum theory, the Church-Turing principle and the universal quantum computer

- Criticised by J. Myers (1997)

  Can a Universal Computer be Fully Quantum?

Peter.Hines@cs.york.ac.uk

The problem with Quantum Turing machines

Consider a **fully quantum Turing machine**

- It has a state space, instead of a state set.
- We can also have superpositions of states ...
  including 'halting states' and non-halting states'.

Myer's question:

What happens 'after halting'?

When the qTm is in the state

$$|halting\ state\rangle + |non - halting\ state\rangle$$

What should happen next?

- (1997) Bernstein and Vazirani: QTMs are fully quantum when the *number of steps to halting* is independent of the input.
- (1998) Ozawa: Arbitrary QTMs can be made *fully quantum*.
- (1998) Linden & Popescu: “*Ozawa's halting scheme is not fully quantum, except in the trivial case in which the computer never halts.*”
- (1998) Ozawa publishes a rebuttal of Linden & Popescu
- (1998) This is shown to be incorrect by Kieu & Danos

Every proposed solution relied on an ancilla

In every case, this became irretrievably entangled with the output.

Peter.Hines@cs.york.ac.uk

- (1997) Bernstein and Vazirani: QTMs are fully quantum when the *number of steps to halting* is independent of the input.
- (1998) Ozawa: Arbitrary QTMs can be made *fully quantum*.
- (1998) Linden & Popescu: "*Ozawa's halting scheme is not fully quantum, except in the trivial case in which the computer never halts.*"
- (1998) Ozawa publishes a rebuttal of Linden & Popescu
- (1998) This is shown to be incorrect by Kieu & Danos

Every proposed solution relied on an ancilla

In every case, this became irretrievably entangled with the output.

- (1997) Bernstein and Vazirani: QTMs are fully quantum when the *number of steps to halting* is independent of the input.
- (1998) Ozawa: Arbitrary QTMs can be made *fully quantum*.
- (1998) Linden & Popescu: "*Ozawa's halting scheme is not fully quantum, except in the trivial case in which the computer never halts.* "
- (1998) Ozawa publishes a rebuttal of Linden & Popescu
- (1998) This is shown to be incorrect by Kieu & Danos

Every proposed solution relied on an ancilla

In every case, this became irretrievably entangled with the output.

- (1997) Bernstein and Vazirani: QTMs are fully quantum when the *number of steps to halting* is independent of the input.
- (1998) Ozawa: Arbitrary QTMs can be made *fully quantum*.
- (1998) Linden & Popescu: "*Ozawa's halting scheme is not fully quantum, except in the trivial case in which the computer never halts.* "
- (1998) Ozawa publishes a rebuttal of Linden & Popescu
- (1998) This is shown to be incorrect by Kieu & Danos

Every proposed solution relied on an ancilla

In every case, this became irretrievably entangled with the output.

- (1997) Bernstein and Vazirani: QTMs are fully quantum when the *number of steps to halting* is independent of the input.
- (1998) Ozawa: Arbitrary QTMs can be made *fully quantum*.
- (1998) Linden & Popescu: "*Ozawa's halting scheme is not fully quantum, except in the trivial case in which the computer never halts.* "
- (1998) Ozawa publishes a rebuttal of Linden & Popescu
- (1998) This is shown to be incorrect by Kieu & Danos

Every proposed solution relied on an ancilla

In every case, this became irretrievably entangled with the output.

Peter.Hines@cs.york.ac.uk

- (1997) Bernstein and Vazirani: QTMs are fully quantum when the *number of steps to halting* is independent of the input.
- (1998) Ozawa: Arbitrary QTMs can be made *fully quantum*.
- (1998) Linden & Popescu: "*Ozawa's halting scheme is not fully quantum, except in the trivial case in which the computer never halts.* "
- (1998) Ozawa publishes a rebuttal of Linden & Popescu
- (1998) This is shown to be incorrect by Kieu & Danos

Every proposed solution relied on an ancilla

In every case, this became irretrievably entangled with the output.

A suitable model of computation

To define oracles, which model of computation should we use?

Turing Machines

Too ambitious: the **quantum halting problem** is insurmountable.

Space-bounded Turing machines

- A reasonable compromise.
- Space bounds are physically justifiable.
- Their computations can be described within the circuit model.

The circuit model

A ready translation into quantum operations - but very difficult to use.

A suitable model of computation

To define oracles, which model of computation should we use?

Turing Machines

Too ambitious: the **quantum halting problem** is insurmountable.

Space-bounded Turing machines

- A reasonable compromise.
- Space bounds are physically justifiable.
- Their computations can be described within the circuit model.

The circuit model

A ready translation into quantum operations - but very difficult to use.

A suitable model of computation

To define oracles, which model of computation should we use?

Turing Machines

Too ambitious: the **quantum halting problem** is insurmountable.

Space-bounded Turing machines

- A reasonable compromise.
- Space bounds are physically justifiable.
- Their computations can be described within the circuit model.

The circuit model

A ready translation into quantum operations - but very difficult to use.

Peter.Hines@cs.york.ac.uk

A suitable model of computation

To define oracles, which model of computation should we use?

Turing Machines

Too ambitious: the **quantum halting problem** is insurmountable.

Space-bounded Turing machines

- A reasonable compromise.
- Space bounds are physically justifiable.
- Their computations can be described within the circuit model.

The circuit model

A ready translation into quantum operations - but very difficult to use.

Defining bounded Turing machines

In common with Turing machines:

A BTM has:

- A set of alphabet symbols & a set of labels.
- A tape, with an alphabet symbol in each cell.
- A labelled pointer.
- A transition function $\mathcal{T}$.

Unlike Turing machines:

- The tape is a fixed, finite length.
- Each label is either **left-moving** or **right-moving**.
- The pointer is positioned *between* cells on the tape.

Defining bounded Turing machines

In common with Turing machines:

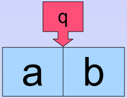
A BTM has:

- A set of alphabet symbols & a set of labels.
- A tape, with an alphabet symbol in each cell.
- A labelled pointer.
- A transition function $\mathcal{T}$.

Unlike Turing machines:

- The tape is a fixed, finite length.
- Each label is either **left-moving** or **right-moving**.
- The pointer is positioned *between* cells on the tape.

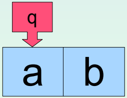
The dynamics:



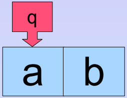
Step 1.

The pointer moves onto a cell:

- The left cell, when q is left-moving.
- The right cell, when q is right-moving.



The dynamics:

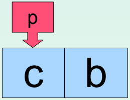


Step 2.

The transition function determines a new symbol / label pair:

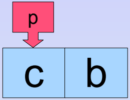
$$(q, a) \stackrel{\mathcal{T}}{\mapsto} (p, c)$$

These are written into the pointer / cell respectively.



Peter.Hines@cs.york.ac.uk

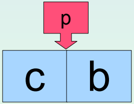
The dynamics:



Step 3.

The pointer moves back to the cell boundary:

- The left boundary, when p is left-moving.
- The right boundary, when p is right-moving.



Peter.Hines@cs.york.ac.uk

Bounded Turing machine states

A **state** for a bounded Turing machine is a:

"complete instantaneous description".

This requires:

- The contents of the tape.
- The position of the pointer.
- The label of the pointer.

Peter.Hines@cs.york.ac.uk

What are the halting states?

A BTM is in a halting state when either:

The pointer is on the far left of the tape, with a left-moving label.

OR

The pointer is on the far right of the tape, with a right-moving label.

Peter.Hines@cs.york.ac.uk

The dual concept — starting states

A BTM is in a starting state when either:

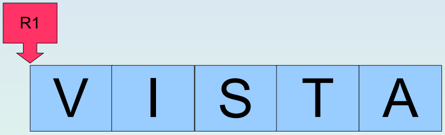
The pointer is on the far right of the tape, with a left-moving label.

OR

The pointer is on the far left of the tape, with a right-moving label.

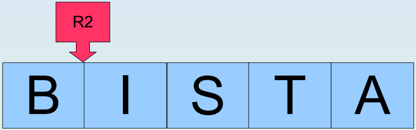
An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



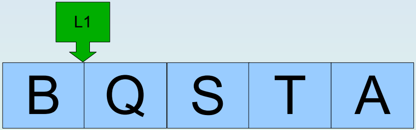
Peter.Hines@cs.york.ac.uk

- Alphabet symbols: $\{A, B, C, \dots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



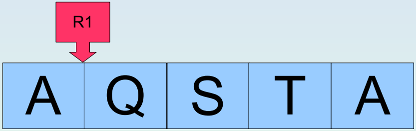
An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



An example bounded Turing machine

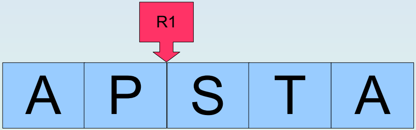
- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



Peter.Hines@cs.york.ac.uk

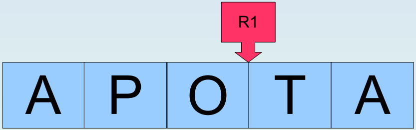
An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



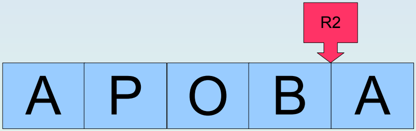
An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



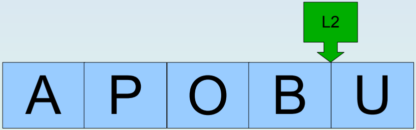
An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



An example bounded Turing machine

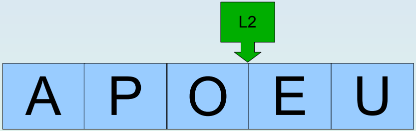
- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



Peter.Hines@cs.york.ac.uk

An example bounded Turing machine

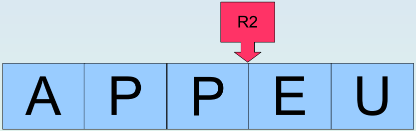
- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



Peter.Hines@cs.york.ac.uk

An example bounded Turing machine

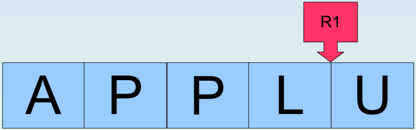
- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



Peter.Hines@cs.york.ac.uk

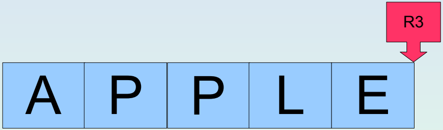
An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



An example bounded Turing machine

- Alphabet symbols: $\{A, B, C, \ldots, X, Y, Z\}$
- Pointer labels:
  - Left-moving labels $\{L1, L2, L3\}$
  - Right-moving labels $\{R1, R2, R3\}$
- Tape length = 5.



Peter.Hines@cs.york.ac.uk

The function computed by a BTM

A Bounded Turing machine $\mathcal{T}$ defines a *(partial) function* $[\mathcal{T}]$, that takes *starting states* to *halting states*.

Reversible bounded Turing machines

When the *transition function* is reversible,

1. The dynamics of $\mathcal{T}$ are (partial) reversible.
2. The function $[\mathcal{T}]$ is a *bijection*.

Our goal: To produce a quantum oracle, for this bijection.

What happens after halting?

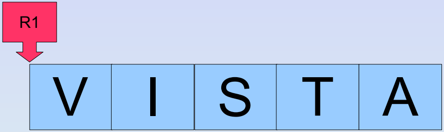
To answer J. Myer's question, *without using an infinite ancilla*, there is only one possibility:

After halting ...

the Bounded Turing machine reverses its operation
& 'uncomputes' what it has just computed.

A machine with starting / halting states is transformed into a machine that is entirely *cyclic*.

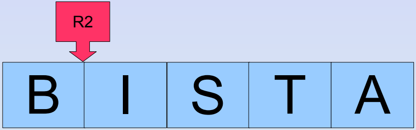
Forward Gear



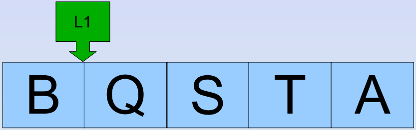
Peter.Hines@cs.york.ac.uk

From termination, to cyclic behaviour

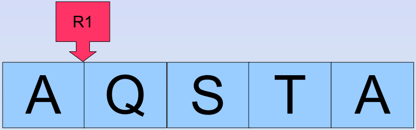
Forward Gear



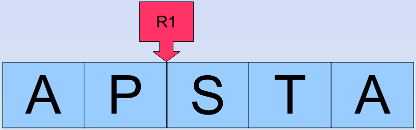
Forward Gear



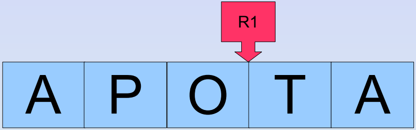
Forward Gear



Forward Gear

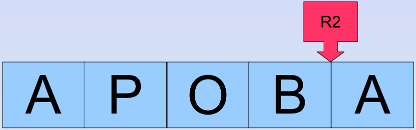


Forward Gear

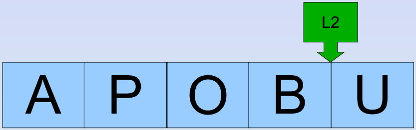


From termination, to cyclic behaviour

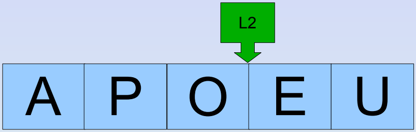
Forward Gear



Forward Gear



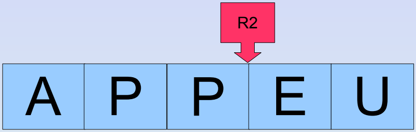
Forward Gear



Peter.Hines@cs.york.ac.uk

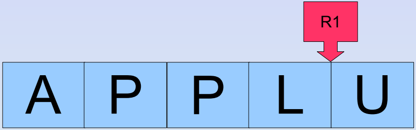
From termination, to cyclic behaviour

Forward Gear

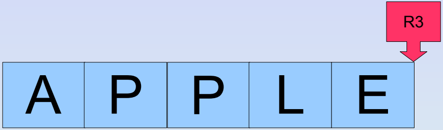


Peter.Hines@cs.york.ac.uk

Forward Gear

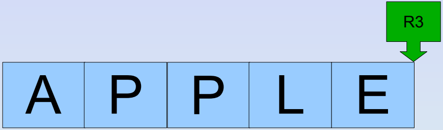


Forward Gear



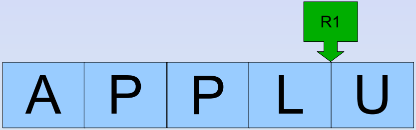
Peter.Hines@cs.york.ac.uk

Reverse Gear



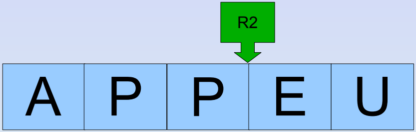
Peter.Hines@cs.york.ac.uk

Reverse Gear



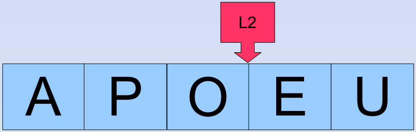
Peter.Hines@cs.york.ac.uk

Reverse Gear



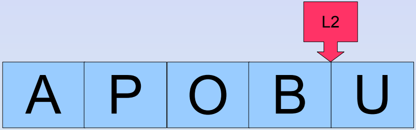
Peter.Hines@cs.york.ac.uk

Reverse Gear



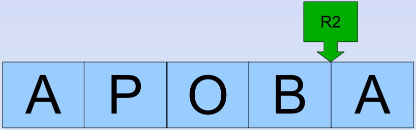
Peter.Hines@cs.york.ac.uk

Reverse Gear



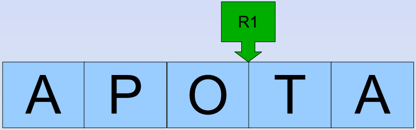
Peter.Hines@cs.york.ac.uk

Reverse Gear

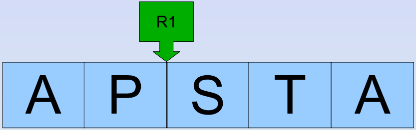


From termination, to cyclic behaviour

Reverse Gear

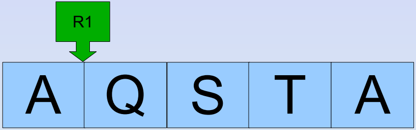


Reverse Gear



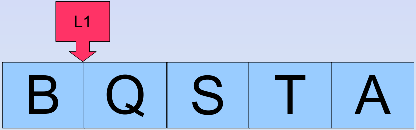
Peter.Hines@cs.york.ac.uk

Reverse Gear

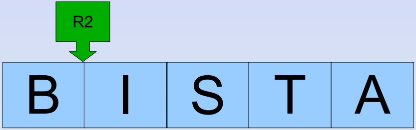


Peter.Hines@cs.york.ac.uk

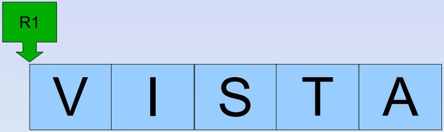
Reverse Gear



Reverse Gear



Reverse Gear



Peter.Hines@cs.york.ac.uk

What is the point of all this?

The key idea:

Quantum computers are:

1. Remarkably poor at dealing with conditional halting.
2. Remarkably good at dealing with cyclic behaviour.

By transforming conditional halting into cyclic behaviour,
we can build quantum circuits for bounded Turing machines.

Peter.Hines@cs.york.ac.uk

What is the point of all this?

The key idea:

Quantum computers are:

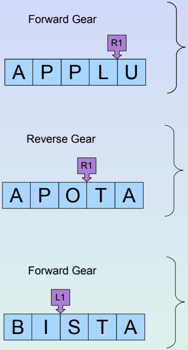
1. Remarkably poor at dealing with conditional halting.
2. Remarkably good at dealing with cyclic behaviour.

By transforming conditional halting into cyclic behaviour,
we can build quantum circuits for bounded Turing machines.

Peter.Hines@cs.york.ac.uk

Cyclic bounded Turing machines, in binary ...

Each state is represented by a binary number:



11011011

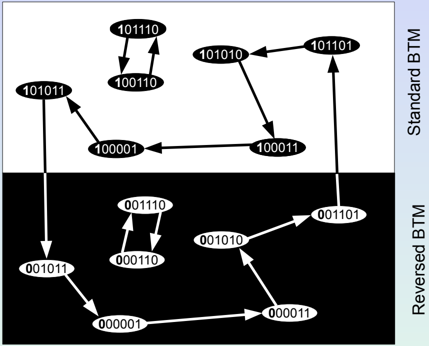
01110101

10011010

The **most significant bit** specifies whether *forward* or *reverse* gear is selected.

Peter.Hines@cs.york.ac.uk

Each state is mapped to the next state by a function $\mathcal{P}$.



This function, the **Primitive Evolution**, is a bijection.

Peter.Hines@cs.york.ac.uk

The exact problem

What we have:

There exists an oracle $U_{\mathcal{P}}$ for the Primitive Evolution $\mathcal{P}$.

If necessary, we can construct this from the transition function

What we want:

An oracle for $[\mathcal{T}]$, the function computed by the B.T.M.

This will be a **quantum circuit**, based around $U_{\mathcal{P}}$

Peter.Hines@cs.york.ac.uk

The exact problem

What we have:

There exists an oracle $U_{\mathcal{P}}$ for the Primitive Evolution $\mathcal{P}$.

If necessary, we can construct this from the transition function

What we want:

An oracle for $[\mathcal{T}]$, the function computed by the B.T.M.

This will be a **quantum circuit**, based around $U_{\mathcal{P}}$

Peter.Hines@cs.york.ac.uk

A useful tool

We use the "**Resolution Formula**", applied to the primitive evolution $\mathcal{P}$.

This:

- Takes a function defined on a set, and
- returns a (possibly partial) function defined on a subset.

The Resolution formula

1. Arose in logical models,
2. Has a strongly categorical interpretation,
3. Is useful for *state machines*.

The Resolution, computationally

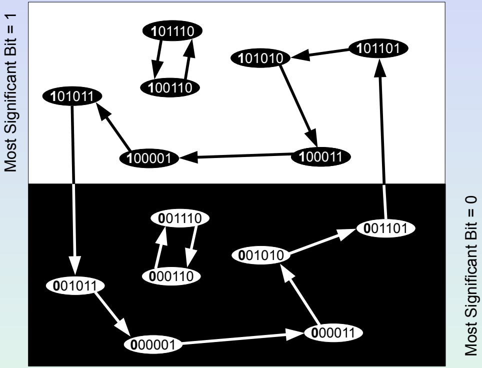
- Start with a binary string whose Most Significant Bit is 0.
- Apply $\mathcal{P}$.
- Repeat, until the M.S.B. of the new string is again 0.

This defines a function $Res_0(\mathcal{P})$ on

the strings whose M.S.B. is 0.

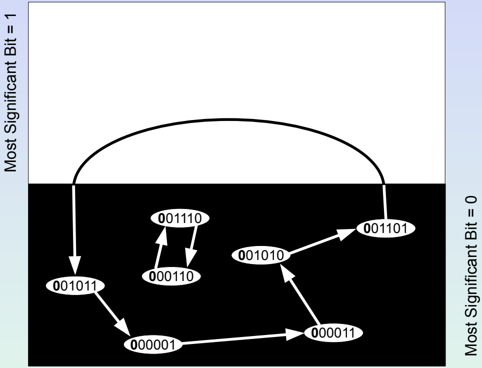
Peter.Hines@cs.york.ac.uk

The "Primitive Evolution" $\mathcal{P}$

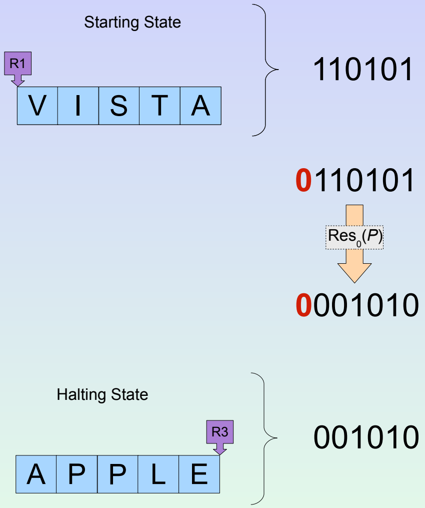


The Resolution, graphically

The “Resolution at 0” of $\mathcal{P}$



The Resolution, and BTM computations



Peter.Hines@cs.york.ac.uk

A program of work

The next steps:

1. Give an explicit mathematical formula for $Res(P)$.
2. Translate this into a formula for the oracle.
3. Give quantum circuits that compute exactly this!

The outcome: an oracle for the bounded Turing machine.

Peter.Hines@cs.york.ac.uk

A program of work

The next steps:

1. Give an explicit mathematical formula for $Res(P)$.
2. Translate this into a formula for the oracle.
3. Give quantum circuits that compute exactly this!

The outcome: an oracle for the bounded Turing machine.

Peter.Hines@cs.york.ac.uk

The next steps:

1. Give an explicit mathematical formula for $Res(P)$.
2. Translate this into a formula for the oracle.
3. Give quantum circuits that compute exactly this!

The outcome: an oracle for the bounded Turing machine.

The next steps:

1. Give an explicit mathematical formula for $Res(P)$.
2. Translate this into a formula for the oracle.
3. Give quantum circuits that compute exactly this!

The outcome: an oracle for the bounded Turing machine.

The Resolution, mathematically

- The bijection $\mathcal{P}$ acts on the **set**: 000...00 , . . . , 111...11
- The oracle $U_{\mathcal{P}}$ acts on the **space** with basis:

$$|000...00\rangle \ , \ \ldots \ , \ |111...11\rangle$$

We can divide this basis set into

$$\{ \text{ vectors starting with } 0 \} \ \cup \ \{ \text{ vectors starting with } 1 \}$$

Peter.Hines@cs.york.ac.uk

The space $\mathbb{S}$ naturally splits up as $\mathbb{S} = \mathbb{R} \oplus \mathbb{F}$

The space $\mathbb{R}$

has basis

$$|000...00\rangle \;,\; \ldots \;,\; |011...11\rangle$$

The space $\mathbb{F}$

has basis

$$|100...00\rangle \;,\; \ldots \;,\; |111...11\rangle$$

We may now write $U_{\mathcal{P}}$ as a block matrix:

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

An explicit formula

The oracle

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"' near start] & \mathbb{R} \\ \mathbb{F} \arrow[ur, "B" near start] \arrow[r, "D"'] & \mathbb{F} \end{tikzcd}$$

Its "resolution at $\mathbb{R}$"

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"' near start] & \mathbb{R} \\ \mathbb{F} \arrow[ur, "B" near start] \arrow[r, "D"'] & \mathbb{F} \end{tikzcd}$$

$$Res_0(U_{\mathcal{P}}) = A + \sum_{k=0}^{\infty} BD^kC$$

Peter.Hines@cs.york.ac.uk

An explicit formula

The oracle

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"' near start] & \mathbb{R} \\ \mathbb{F} \arrow[r, "D"'] \arrow[ur, "B" near start] & \mathbb{F} \end{tikzcd}$$

Its "resolution at $\mathbb{R}$"

$$Res_0(U_{\mathcal{P}}) = A + \sum_{k=0}^{\infty} BD^k C$$

Peter.Hines@cs.york.ac.uk

An explicit formula

The oracle

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"'] & \mathbb{R} \\ \mathbb{F} \arrow[r, "D"'] \arrow[ur, "B"] & \mathbb{F} \end{tikzcd}$$

Its "resolution at $\mathbb{R}$"

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"'] & \mathbb{R} \\ \mathbb{F} \arrow[r, "D"'] \arrow[ur, "B"] & \mathbb{F} \end{tikzcd}$$

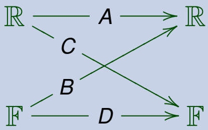
$$Res_0(U_{\mathcal{P}}) = A + \sum_{k=0}^{\infty} BD^kC$$

Peter.Hines@cs.york.ac.uk

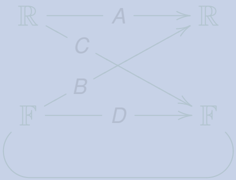
An explicit formula

The oracle

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$



Its “resolution at ℝ”



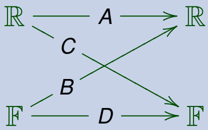
$$Res_0(U_{\mathcal{P}}) = A + \sum_{k=0}^{\infty} BD^kC$$

Peter.Hines@cs.york.ac.uk

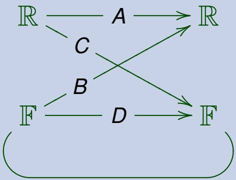
An explicit formula

The oracle

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$



Its “resolution at $\mathbb{R}$”



$$Res_0(U_{\mathcal{P}}) = A + \sum_{k=0}^{\infty} BD^kC$$

Peter.Hines@cs.york.ac.uk

An explicit formula

The oracle

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"'] & \mathbb{R} \\ \mathbb{F} \arrow[r, "D"'] \arrow[ur, "B"] & \mathbb{F} \end{tikzcd}$$

Its "resolution at $\mathbb{R}$"

$$\begin{tikzcd} \mathbb{R} \arrow[r, "A"] \arrow[dr, "C"'] & \mathbb{R} \\ \mathbb{F} \arrow[r, "D"'] \arrow[ur, "B"] & \mathbb{F} \arrow[l, bend left=60, dash] \end{tikzcd}$$

$$Res_0(U_{\mathcal{P}}) = A + \sum_{k=0}^{\infty} BD^k C$$

Peter.Hines@cs.york.ac.uk

Some standard results:

The Resolution $A + \sum_{k=0}^{\infty} BD^k C$ is:

1. Well-defined!
   — this infinite sum converges.
2. A unitary map.
   — i.e. a quantum operation.
3. An oracle for a classical computation.
   — it maps basis states to basis states.

Peter.Hines@cs.york.ac.uk

Some standard results:

The Resolution $A + \sum_{k=0}^{\infty} BD^k C$ is:

1. Well-defined!
   — this infinite sum converges.
2. A unitary map.
   — i.e. a quantum operation.
3. An oracle for a classical computation.
   — it maps basis states to basis states.

Peter.Hines@cs.york.ac.uk

Some standard results:

The Resolution $A + \sum_{k=0}^{\infty} BD^k C$ is:

1. Well-defined!
   — this infinite sum converges.
2. A unitary map.
   — i.e. a quantum operation.
3. An oracle for a classical computation.
   — it maps basis states to basis states.

Peter.Hines@cs.york.ac.uk

Some standard results:

The Resolution $A + \sum_{k=0}^{\infty} BD^k C$ is:

1. Well-defined!
   — this infinite sum converges.
2. A unitary map.
   — i.e. a quantum operation.
3. An oracle for a classical computation.
   — it maps basis states to basis states.

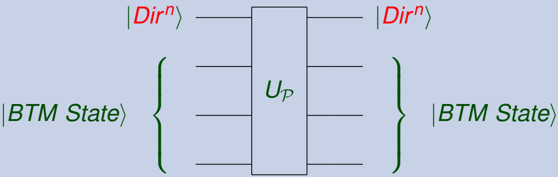
Peter.Hines@cs.york.ac.uk

Some Simple Circuits.

What gates do we require?

The obvious one ...

The *oracle* for the **primitive evolution**



Peter.Hines@cs.york.ac.uk

Some more multi-qubit gates

We also need some basic arithmetic.

The successor map

Acts as

$$Succ\,|n\rangle = |n+1\rangle$$

(modulo “overflow”).

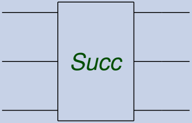
The cyclic shift

Acts as

$$|b_1 b_2 \ldots b_{n-1} b_n\rangle = \Big\downarrow {\scriptstyle Shift}\; |b_n b_1 b_2 \ldots b_{n-1}\rangle$$

We also need some basic arithmetic.

The successor map



Acts as

$$Succ\,|n\rangle = |n+1\rangle$$

(modulo "overflow").

The cyclic shift

Acts as

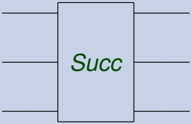
$$|b_1 b_2 \ldots b_{n-1} b_n\rangle = \Big\downarrow \text{Shift} \quad |b_n b_1 b_2 \ldots b_{n-1}\rangle$$

Peter.Hines@cs.york.ac.uk

Some more multi-qubit gates

We also need some basic arithmetic.

The successor map

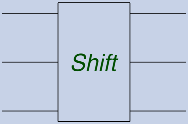


Acts as

$$Succ\,|n\rangle = |n+1\rangle$$

(modulo "overflow").

The cyclic shift



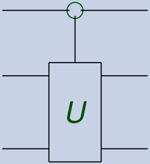
Acts as

$$|b_1 b_2 \ldots b_{n-1} b_n\rangle = \\ \Big\downarrow \mathit{Shift} \\ |b_n b_1 b_2 \ldots b_{n-1}\rangle$$

Peter.Hines@cs.york.ac.uk

We also need ‘controlled’ or ‘conditional’ gates.

“ Control on 0 ”



- The first qubit does not change.
- For the remaining qubits:
  - If the first qubit is $0\rangle$, apply U.
  - If the first qubit is $|1\rangle$, do nothing.

Matrices for control-on-0

Writing U as a block matrix:

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$$

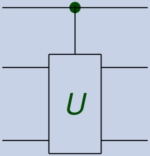
$CRTL_0 U$ has the matrix

$$\begin{pmatrix} U_{00} & U_{01} & \mathbf{0} & \mathbf{0} \\ U_{10} & U_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

The alternative form:

“ Control on 1 ”



- The first qubit does not change.
- For the remaining qubits:
  - If the first qubit is $|0\rangle$, do nothing.
  - If the first qubit is $|1\rangle$, apply U.

Matrices for control-on-1

Writing U as a block matrix:

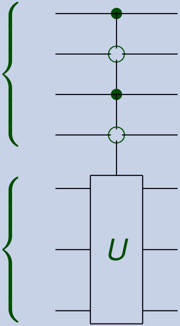
$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$$

$CTRL_1 U$ has the matrix

$$\begin{pmatrix} I & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & U_{00} & U_{01} \\ \mathbf{0} & \mathbf{0} & U_{10} & U_{11} \end{pmatrix}$$

Daisy-chaining control gates

Controlled gates, can themselves be controlled, etc.

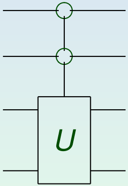


- The first register does not change.
- For the second register:
  - If the first register is $|1010\rangle$, **apply U**.
  - Otherwise, **do nothing.**

Matrices for multiply-controlled gates

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \quad \text{Controlled on } 00$$

The circuit:



The matrix:

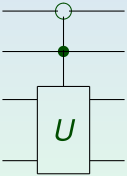
$$\begin{pmatrix}
U_{00} & U_{01} & 0 & 0 & 0 & 0 & 0 & 0 \\
U_{10} & U_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

Matrices for multiply-controlled gates

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \quad \text{Controlled on } \; 01$$

The circuit:



The matrix:

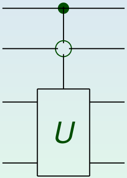
$$\begin{pmatrix}
I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & U_{00} & U_{01} & 0 & 0 & 0 & 0 \\
0 & 0 & U_{10} & U_{11} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I
\end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

Matrices for multiply-controlled gates

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \quad \text{Controlled on } 10$$

The circuit:



The matrix:

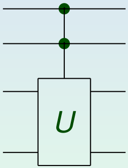
$$\begin{pmatrix}
\mathbf{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{I} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & U_{00} & U_{01} & 0 & 0 \\
0 & 0 & 0 & 0 & U_{10} & U_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{I} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{I}
\end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

Matrices for multiply-controlled gates

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \quad \text{Controlled on } 11$$

The circuit:



The matrix:

$$\begin{pmatrix}
\mathbf{I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{I} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{I} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{I} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \mathbf{I} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & U_{00} & U_{01} \\
0 & 0 & 0 & 0 & 0 & 0 & U_{10} & U_{11}
\end{pmatrix}$$

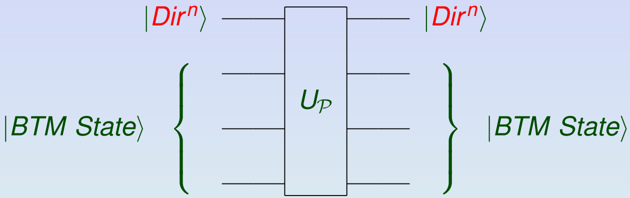
Peter.Hines@cs.york.ac.uk

THE CONSTRUCTION

Peter.Hines@cs.york.ac.uk

A preliminary ...

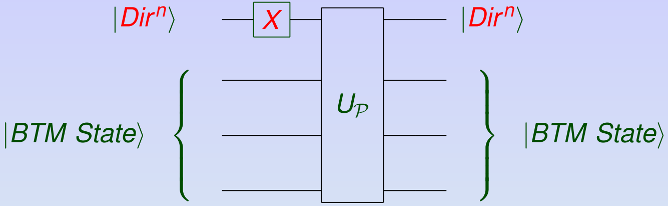
We modify the (oracle for the) *Primitive Evolution:*



Before each application:

The "forward / reverse gear" qubit is flipped.

Peter.Hines@cs.york.ac.uk



Call this operation ${}^X U_{\mathcal{P}}$

As matrices:

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \qquad {}^X U_{\mathcal{P}} = \begin{pmatrix} B & A \\ D & C \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

The input ...

- An ancilla of $log_2(T+1)$ qubits, all in state $|0\rangle$.
- A starting state $|S\rangle$, for our Bounded Turing machine.

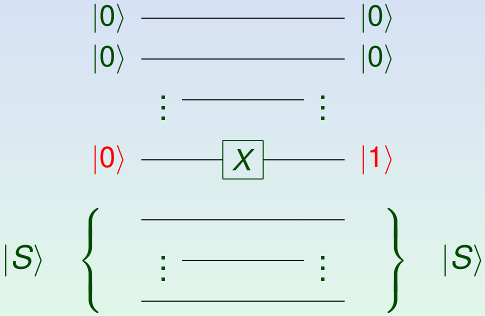
The action:

$$|00\ldots00\rangle\,|S\rangle \longrightarrow |00\ldots01\rangle\,|S\rangle$$

Peter.Hines@cs.york.ac.uk

The input ...

- An ancilla of $log_2(T+1)$ qubits, all in state $|0\rangle$.
- A starting state $|S\rangle$, for our Bounded Turing machine.



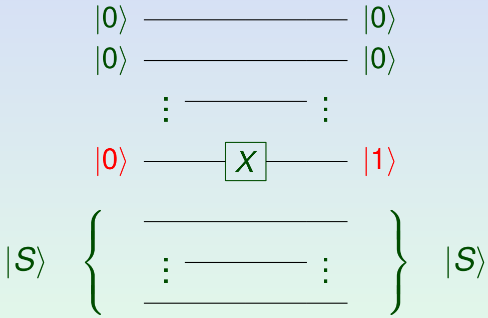
The action:

$$|00\ldots00\rangle\,|S\rangle$$
$$\downarrow$$
$$|00\ldots01\rangle\,|S\rangle$$

Peter.Hines@cs.york.ac.uk

The input ...

- An ancilla of $log_2(T+1)$ qubits, all in state $|0\rangle$.
- A starting state $|S\rangle$, for our Bounded Turing machine.



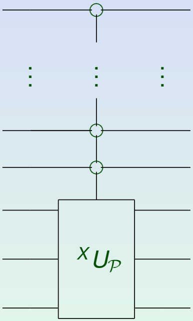
The action:

$$|00\dots00\rangle\,|S\rangle$$
$$\downarrow$$
$$|00\dots01\rangle\,|S\rangle$$

Peter.Hines@cs.york.ac.uk

We then apply a series of maps:

The map $^X U_{\mathcal{P}}$, controlled on $|0\ldots 00\rangle$



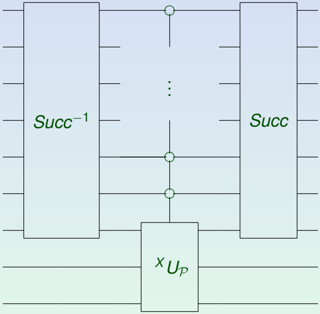
This has matrix:

$$\begin{pmatrix} B & A & 0 & 0 & 0 & 0 & \cdots \\ D & C & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \mathbf{I} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \mathbf{I} & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{I} & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & \mathbf{I} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

Followed by:

The same thing, with the ancilla conjugated by the *Succ*



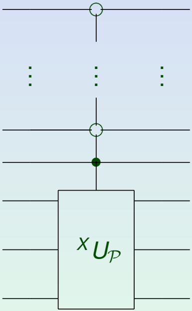
This has matrix:

$$\begin{pmatrix} I & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & B & A & 0 & 0 & 0 & \cdots \\ 0 & D & C & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & I & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & I & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & I & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

Followed by:

The map ${}^X U_{\mathcal{P}}$, controlled on $|0 \ldots 01\rangle$



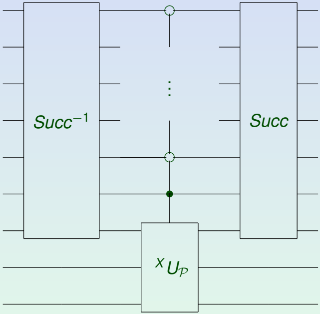
This has matrix:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & B & A & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & D & C & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

Followed by:

The same thing, with the ancilla conjugated by the *Succ*

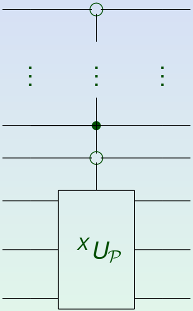


This has matrix:

$$\begin{pmatrix}
I & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & I & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & I & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & B & A & 0 & \cdots \\
0 & 0 & 0 & D & C & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & I & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

Followed by:

The map ${}^{X}U_{\mathcal{P}}$, controlled on $|0\ldots10\rangle$



This has matrix:

$$\begin{pmatrix} I & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & I & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & I & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & I & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & B & A & \cdots \\ 0 & 0 & 0 & 0 & D & C & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

This process continues for T steps.

Peter.Hines@cs.york.ac.uk

What is the overall effect?

This is given by (very tedious) matrix multiplications:

$$\begin{pmatrix}
B & A & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD & BC & A & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD^2 & BDC & BC & A & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD^3 & BD^2C & BDC & BC & A & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD^4 & BD^3C & BD^2C & BDC & BC & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\
BD^{T-1} & BD^{T-2}C & BD^{T-3}C & BD^{T-4}C & BD^{T-5}C & \ldots & A & \mathbf{0} & \mathbf{0} & \ldots \\
D^T & D^{T-1}C & D^{T-2}C & D^{T-3}C & D^{T-4}C & \ldots & C & \mathbf{0} & \mathbf{0} & \ldots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{I} & \mathbf{0} & \ldots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{I} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & & \ddots
\end{pmatrix}$$

Note the individual terms of the Resolution, in each column.

Peter.Hines@cs.york.ac.uk

This is given by (very tedious) matrix multiplications:

$$
\begin{pmatrix}
B & A & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\
BD & BC & A & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\
BD^2 & BDC & BC & A & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\
BD^3 & BD^2C & BDC & BC & A & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\
BD^4 & BD^3C & BD^2C & BDC & BC & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\
BD^{T-1} & BD^{T-2}C & BD^{T-3}C & BD^{T-4}C & BD^{T-5}C & \dots & A & \mathbf{0} & \mathbf{0} & \dots \\
D^T & D^{T-1}C & D^{T-2}C & D^{T-3}C & D^{T-4}C & \dots & C & \mathbf{0} & \mathbf{0} & \dots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{I} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & & \ddots
\end{pmatrix}
$$

Note the individual terms of the Resolution, in each column.

Peter.Hines@cs.york.ac.uk

What is the overall effect?

This is given by (very tedious) matrix multiplications:

$$\begin{pmatrix}
B & A & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD & BC & A & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD^2 & BDC & BC & A & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD^3 & BD^2C & BDC & BC & A & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
BD^4 & BD^3C & BD^2C & BDC & BC & \ldots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\
BD^{T-1} & BD^{T-2}C & BD^{T-3}C & BD^{T-4}C & BD^{T-5}C & \ldots & A & \mathbf{0} & \mathbf{0} & \ldots \\
D^T & D^{T-1}C & D^{T-2}C & D^{T-3}C & D^{T-4}C & \ldots & C & \mathbf{0} & \mathbf{0} & \ldots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{I} & \mathbf{0} & \ldots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \mathbf{I} & \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & & \ddots
\end{pmatrix}$$

Note the individual terms of the Resolution, in each column.

The effect of this matrix:

$$|1\rangle\,|S\rangle \xrightarrow{\quad\mathrm{M}\quad} |t_S - 1\rangle\,|H\rangle$$

Where:

- S is a **starting state.**
- H is the Resolution, applied to S

  i.e. the corresponding **halting state.**
- t_S is the number of steps between S and H.

Peter.Hines@cs.york.ac.uk

This is not the quantum state we are looking for ...

Consider two bounded Turing machine computations:

$$S \xrightarrow{t_S \ \textit{steps}} H$$

$$S' \xrightarrow{t_{S'} \ \textit{steps}} H'$$

When given the *superposition* $|S\rangle + |S'\rangle$, this procedure produces $|t_S\rangle\,|H\rangle + |t_{S'}\rangle\,|H'\rangle$.

Unless $t_S = t_{S'}$

We have *entanglement* between the **ancilla**, and the **result**.

This is not the quantum state we are looking for ...

Consider two bounded Turing machine computations:

$$S \xrightarrow{t_S \ \textit{steps}} H$$

$$S' \xrightarrow{t_{S'} \ \textit{steps}} H'$$

When given the *superposition* $|S\rangle + |S'\rangle$, this procedure

produces $|t_S\rangle\,|H\rangle + |t_{S'}\rangle\,|H'\rangle$.

Unless $t_S = t_{S'}$

We have *entanglement* between the **ancilla**, and the **result**.

This is not the quantum state we are looking for ...

Consider two bounded Turing machine computations:

$$S \xrightarrow{t_S \ \ steps} H$$

$$S' \xrightarrow{t_{S'} \ \ steps} H'$$

When given the *superposition* $|S\rangle + |S'\rangle$, this procedure

produces $|t_S\rangle \, |H\rangle + |t_{S'}\rangle \, |H'\rangle$.

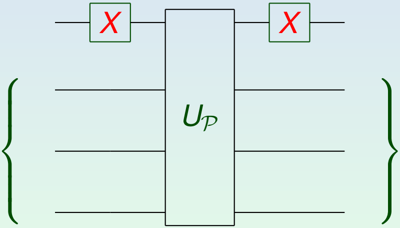
Unless $t_S = t_{S'}$

We have *entanglement* between the **ancilla**, and the **result.**

Repeat the same procedure ...

... using the *inverse* of $U_{\mathcal{P}}$, instead of $U_{\mathcal{P}}$ itself.

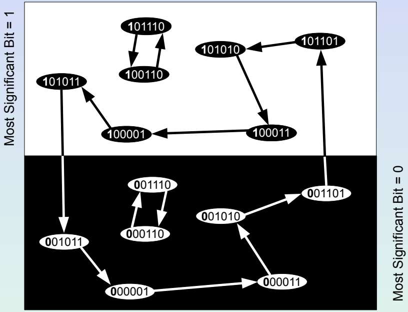
This has a convenient circuit form:



Peter.Hines@cs.york.ac.uk

A reminder:

The "Primitive Evolution", $\mathcal{P}$



We compute $\mathcal{P}^{-1}$ by

1. Flipping the M.S.B.
2. Applying $\mathcal{P}$
3. Flipping the M.S.B. again.

Peter.Hines@cs.york.ac.uk

This time, we continue
for $2T$ steps.

Peter.Hines@cs.york.ac.uk

This then ‘uncomputes’ the computation just performed.

$$S \xrightarrow{B.T.M.} H \xrightarrow{Reversed\ B.T.M} S$$

The ancilla, counting the steps taken, is still increasing

$$|1\rangle\,|S\rangle \xrightarrow{Comp.} |t_S - 1\rangle\,|H\rangle \xrightarrow{Uncomp.} |2t_s - 2\rangle\,|S\rangle$$

What is the effect of this ?

This then ‘uncomputes’ the computation just performed.

$$S \xrightarrow{B.T.M.} H \xrightarrow{Reversed\ B.T.M} S$$

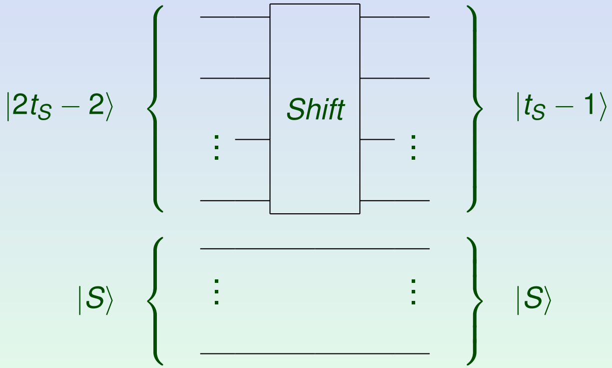
The ancilla, counting the steps taken, is still <u>increasing</u>

$$|1\rangle\,|S\rangle \xrightarrow{Comp.} |t_S - 1\rangle\,|H\rangle \xrightarrow{Uncomp.} |2t_s - 2\rangle\,|S\rangle$$

Peter.Hines@cs.york.ac.uk

The natural next step:

We use the *cyclic shift* to divide the ancilla by 2.



Peter.Hines@cs.york.ac.uk

The final section

For the final section:

Repeat the first section,
with the operations in the reverse order.

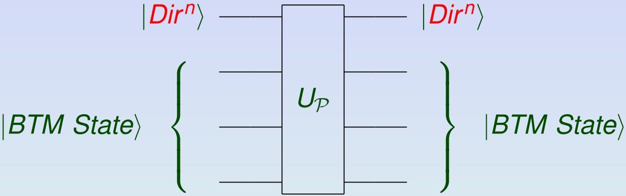
The intention:

Re-do the computation, with the ancilla *decreasing*.

Peter.Hines@cs.york.ac.uk

We reverse *everything* ...

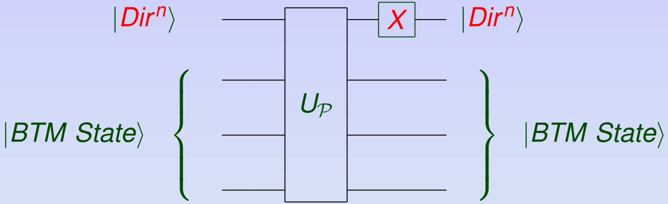
The (oracle for the) *Primitive Evolution is modified:*



After each application:

The “forward / reverse gear” qubit is flipped.

Peter.Hines@cs.york.ac.uk



Call this operation $U_{\mathcal{P}}{}^X$

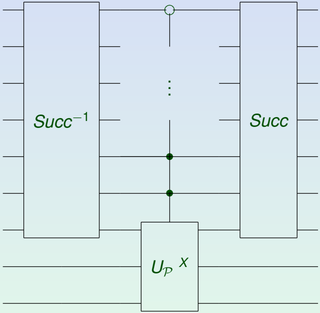
As matrices:

$$U_{\mathcal{P}} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \qquad U_{\mathcal{P}}{}^X = \begin{pmatrix} C & D \\ A & B \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

We apply a series of maps:

The map $U_{\mathcal{P}}{}^{X}$, controlled on $|01\ldots11\rangle$
with the ancilla conjugated by the successor.



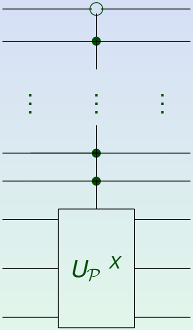
This has matrix:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & C & D & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & A & B & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

We apply a series of maps:

The same thing
without the conjugation.

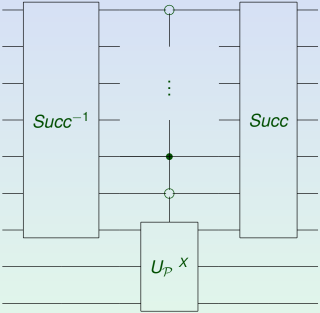


This has matrix:

$$\begin{pmatrix} \mathbf{I} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \mathbf{I} & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ 0 & 0 & \cdots & \mathbf{I} & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 0 & \mathbf{I} & 0 & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 0 & 0 & C & D & 0 & \cdots \\ 0 & 0 & \cdots & 0 & 0 & A & B & 0 & \cdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \mathbf{I} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

We apply a series of maps:

The map $U_{\mathcal{P}}{}^{X}$, controlled on $|01\ldots10\rangle$
with the ancilla conjugated by the successor.

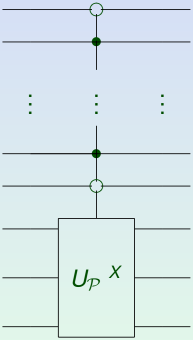


This has matrix:

$$\begin{pmatrix}
\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & C & D & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A & B & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

We apply a series of maps:

The same thing
without the conjugation.



This has matrix:

$$\begin{pmatrix}
\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & C & D & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & A & B & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

We continue this for T steps.

Peter.Hines@cs.york.ac.uk

What is the overall operation?

This is given by (tedious) matrix multiplications:

$$\begin{pmatrix} C & DC & \cdots & D^{T-4}C & D^{T-3}C & D^{T-2}C & D^{T-1}C & D^T & \mathbf{0} & \cdots \\ A & BC & \cdots & BD^{T-5}C & BD^{T-4}C & BD^{T-3}C & BD^{T-2}C & BD^{T-1} & \mathbf{0} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \mathbf{0} & \mathbf{0} & \cdots & A & BC & BDC & BD^2C & BD^3 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A & BC & BDC & BD^2 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & A & BC & BD & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & A & B & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

What is the overall operation?

This is given by (tedious) matrix multiplications:

$$\begin{pmatrix} C & DC & \cdots & D^{T-4}C & D^{T-3}C & D^{T-2}C & D^{T-1}C & D^T & \mathbf{0} & \cdots \\ A & BC & \cdots & BD^{T-5}C & BD^{T-4}C & BD^{T-3}C & BD^{T-2}C & BD^{T-1} & \mathbf{0} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \mathbf{0} & \mathbf{0} & \cdots & A & BC & BDC & BD^2C & BD^3 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & A & BC & BDC & BD^2 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & A & BC & BD & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & A & B & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Peter.Hines@cs.york.ac.uk

The effect of this matrix:

$$|t_{S-1}\rangle\,|S\rangle \longrightarrow |1\rangle\,|H\rangle$$

Where:

- S is our **starting state.**
- H is the Resolution, applied to S

  i.e. the corresponding **halting state.**
- t_S is the number of steps between S and H.

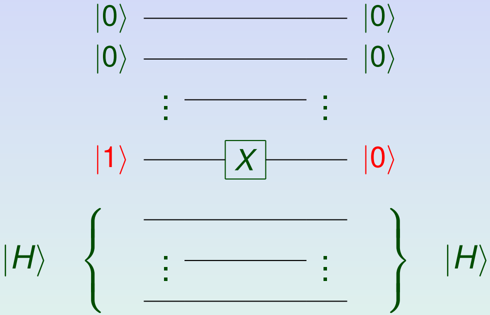
Peter.Hines@cs.york.ac.uk

As a very last step:

The action:

$$|00\ldots01\rangle\,|H\rangle \longrightarrow |00\ldots00\rangle\,|H\rangle$$

As a very last step:

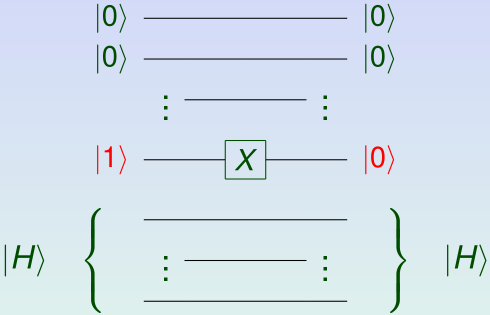


The action:

$$|00\ldots01\rangle\,|H\rangle$$
$$\downarrow$$
$$|00\ldots00\rangle\,|H\rangle$$

Peter.Hines@cs.york.ac.uk

As a very last step:



The action:

$$|00\ldots01\rangle\,|H\rangle$$
$$\downarrow$$
$$|00\ldots00\rangle\,|H\rangle$$

Peter.Hines@cs.york.ac.uk

An overview of the whole process

S is a starting state for the bounded Turing machine.

$$\begin{tikzcd}
\vert 0\rangle\, \vert S\rangle \arrow[r] \arrow[dd, dotted] & \vert 1\rangle\, \vert S\rangle \arrow[r] & \vert t_S - 1\rangle\, \vert H\rangle \arrow[d] \\
 & & \vert 2t_S - 2\rangle\, \vert S\rangle \arrow[d] \\
\vert 0\rangle\, \vert H\rangle & \vert 1\rangle\, \vert H\rangle \arrow[l] & \vert t_S - 1\rangle\, \vert S\rangle \arrow[l]
\end{tikzcd}$$

H is the corresponding halting state.

Peter.Hines@cs.york.ac.uk

What about superpositions ?

Consider a series of BTM computations:

$$\begin{array}{cccc} S_1 & S_2 & \cdots & S_N \\ \downarrow & \downarrow & & \downarrow \\ H_1 & H_2 & \cdots & H_N \end{array}$$

We run this procedure, starting with a superposition:

$$\alpha_1|S_1\rangle + \alpha_2|S_2\rangle + \ldots + \alpha_N|S_N\rangle$$

The action on superpositions

As the ancilla *starts* and *finishes* in a constant state $|0\rangle$,

It is not entangled with the result of the computation.

$$|0\rangle \otimes (\alpha_1 |S_1\rangle + \alpha_2 |S_2\rangle + \ldots + \alpha_N |S_N\rangle)$$

$$\downarrow$$

$$|0\rangle \otimes (\alpha_1 |H_1\rangle + \alpha_2 |H_2\rangle + \ldots + \alpha_N |H_N\rangle)$$

We have a genuine **oracle** for a bounded Turing machine.

A Coda:

"After my lecture, no one raised any objections, or asked any embarrassing questions. I must say this very fact proved a terrible disappointment to me."

— Niels Bohr (1952).

Peter.Hines@cs.york.ac.uk