15/2/19

Reversible Computation.

Part 1, reversibe turing machines.

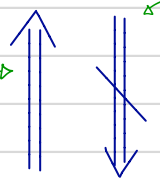
In Deutsch's 84 paper "Quantum Theory, the Church Turing Principle and the Universal Quantum Computer", it is explained how a computational step in his model of quantum computation is performed by applying some unitary operator to a vector in some Hilbert space. Since this operator is unitary, it is in particular invertible. Thus, any quantum computation must at least be reversible. Reversible computation has also been studied on its own for its various applications to the connections between thermal dynamics and computation, see "Notes on the History of Reversible Computation", by Bennett for a review.

But what does it mean for a computation to be reversible?

I thought of two different possible definitions,

There exists a TM M' such that for all inputs w on which M halts, $M'M(w) = w = MM'(w)$.

Not quite as simple as taking the unique inverse of each image to construct M'. Need to consider polite turing machines, (Can't move and then read + write)



M may have states which exhibit non-injective behavior, but which are never reached by any input.

The transition function is injective.

What we really want is:

Def^: A Turing machine M is reversible if it can be "run backwards".

So the 1st definition is close, but the behavior and construction of M' matters.

Bennett essentially asks for the second definition, and then uses injectivity to construct M' as in the first definition. The proof of theorem 1 (later) shows this.

The question now, is what computational limits are imposed, when reversibility is demanded at each stage, ie,

> "Are some logically irreversible operations necessary for nontrivial computation"?

The amazing answer to this question is "no"! In fact there is no loss of computational power in demanding reversibility. Before going any further though, these claims ought to be stated more precisely.

Part of the definition of a turing machine is a transition function $\delta: Q \times \Sigma \longrightarrow Q \times \Sigma \times \{\text{LEFT}, \text{RIGHT}, \text{STAY}\}$, where Q and Σ are finite sets. Thus, δ is given by a finite set of quintuples (satisfying some conditions). The notation $(q, w) \longmapsto (q', w', \sigma)$ will be used for the quintuple (q, w, q', w', σ). An equivalent (in some sense) variant of a turing machine is that of polite turing machines, where the transitions which read/write the tape are seperate to those which move the tape head.

Def^n: A polite turing machine is a turing machine which moves the tape head iff it leaves the symbol just read unchanged, ie, the transition function is such that

$$\forall (q, w,) \longmapsto (q', w', \sigma) \in \delta, \quad \text{if} \quad \sigma \in \{\text{LEFT}, \text{RIGHT}\}, \text{ then } w = w'.$$

The reason why these are considered is because "the reverse" of a TM is not, technically speaking, a TM, as a TM is meant to rewrite, then move, not move and then rewrite.

For what follows, it will be convenient to talk about reversible multitape turing machines, and multitape polite turing machines, with evident definitions.

The formal version of the statement "any turing machine can be made to be reversible" is as follows,

Thm 1: Let M be any turing machine. Then there excises a reversible, 3-tape, polite turing machine S such that,
- if M halts on input w, then S halts on input $(w, \sqcup, \sqcup)$, and
- if $M(w) = w'$, then $S(w, \sqcup, \sqcup) = (w, \sqcup, w')$.

The role of the second tape is to record a history of the steps taken by M. However, the theorem is quite unsubstantial unless it is required that S erases this information. The importance of the theorem is the fact that this erasure can be done reversibly.

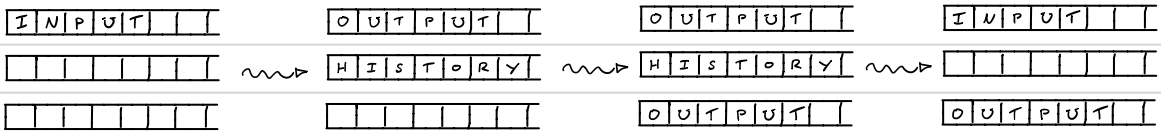
Before presenting the proof of the theorem, the following theorem is mentioned,

Thm 2: Given a reversible multitape turing machine, the corresponding single tape turing machine is also reversible.

The proof is a matter of delicately choosing the states used in the corresponding single tape TM. James' construction from last week works with only a few alterations needed. The details are tedious, but a rigorous proof can be found on page 148 of Kenichi Morita's "Theory of Reversible Computing", (2017).

Proof sketch of thm 1:

Take an arb TM ~~construct a~~> polite (single tape) TM ~~add 2 blank tapes~~> 3-tape, split TM which executes 3 stages.



Proof of thm 1:

Let M be an arbitrary turing machine, with transition function δ (thought of as a set of quintuples). Label the elements of δ with integers 1 through to n in any way, where $|\delta| = n$.

For each quintuple, define a pair of quadruples as follows,

$$\left(k^{th} \text{ quintuple } (q,w) \longmapsto (q',w',\sigma)\right) \longmapsto \left(\begin{array}{l}(q,w) \longmapsto (q'_k, w', \text{STAY}) \\ (q'_k, w') \longmapsto (q', w', \sigma)\end{array}\right)$$

Now add the history and output tapes, and define the transition function to record the history so that the k^{th} pair of 3-tape quadruples is,

$$\begin{array}{l}(q, (w, \sqcup, \sqcup)) \longmapsto (q'_k, (w', \sqcup, \sqcup), (\text{STAY}, \text{RIGHT}, \text{STAY})) \\ (q'_k, (w', \sqcup, \sqcup)) \longmapsto (q', (w', k, \sqcup), (\sigma, \text{STAY}, \text{STAY}))\end{array}$$

This is actually the key step to the proof. The important subtlety is that the 3-tape TM doesn't write the history of the step at the same time as writing image of that step on the first tape, instead, it writes the history as the first tape's head is making a movement. **This ensures that the number k appears in exactly 2 quadruples.** This guarentees injectivity, as the k in the image of any quadruple reduces the size of the preimage to 2, and the presence or absence of the k in the subscript of the state then reduces the preimage to have a unique element.

Lastly, copying can clearly be done reversibly, and erasing the history tape while returning the output on tape 1 to the input is just the reverse of the first step which

has just been seen to be reversible. □

A technical point:
This proof doesn't -really- describe a single reversible TM, but rather 3 back-to-back ones, but these can be made into a single TM by giving unique state names to each of the stages.

The next part of the talk is on reversible logic gates. Turing machines are a theoretical model of computation, and logic gates model the actual engineering behind computers.

Part 2, reversible logic gates.

Clearly, not all logic gates are reversible, for example, if an or gate is emitting an electric signal, it cannot be uniquely determined what configuration the input wires were in without more information.

An easy example of a reversible logic gate is the "not" gate,

Notation:

Truth table:

INPUT	OUTPUT
0	1
1	0

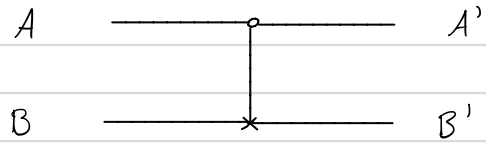
A similar question to that of the first part of this talk arises, "is there a complete set of reversible logic gates"?

Answer: Yes! In fact, there is a complete set of reversible logic gates **consisting of a single element!**

This element is the CCNOT gate. First, the CNOT gate will be defined.

Def^ the CNOT gate (controlled NOT gate), is given by,

Notation:



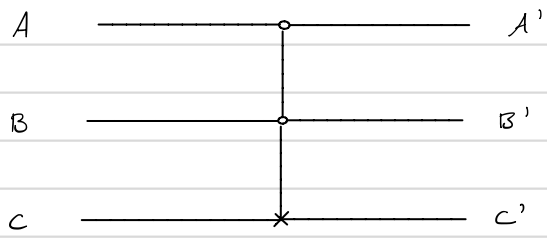
Truth table:

A	B	A'	B'	
0	0	0	0	Identity gate, as A=0,
0	1	0	1	
1	0	1	1	NOT gate, as A=1.
1	1	1	0	

The idea is that a CNOT gate is a not gate which is "controlled" (ie, decided whether it acts like a NOT gate or an identity gate) by the first input.

So the CCNOT gate is then a "controlled CNOT gate", and is given by,

Notation:



Truth table:

A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Reversible by inspection.

Th^{m}: The set containing only the CCNOT gate is complete.

Proof:

An XOR gate can be simulated by a CNOT gate by interpreting output B' as XOR(A,B),

A	B	B'
0	0	0
0	1	1
1	0	1
1	1	0

and CNOT gates can clearly be simulated on a CCNOT gate.

Moreover, the output c' of a CCNOT gate is AND(A,B),

A	B	C	C'
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1

□