Universality in Quantum Computation

Salvador E. Venegas-Andraca

Facultad de Ciencias, UNAM
svenegas@ciencias.unam.mx and salvador.venegas-andraca@keble.oxon.org
https://www.linkedin.com/in/venegasandraca/
https://unconventionalcomputing.org/
https://www.venegas-andraca.org/

October 2024

Fields of the Theory of Computation

Automata theory. Definitions and properties of abstract models of computation.

Computability theory. What problems can(not) be solved by computers?

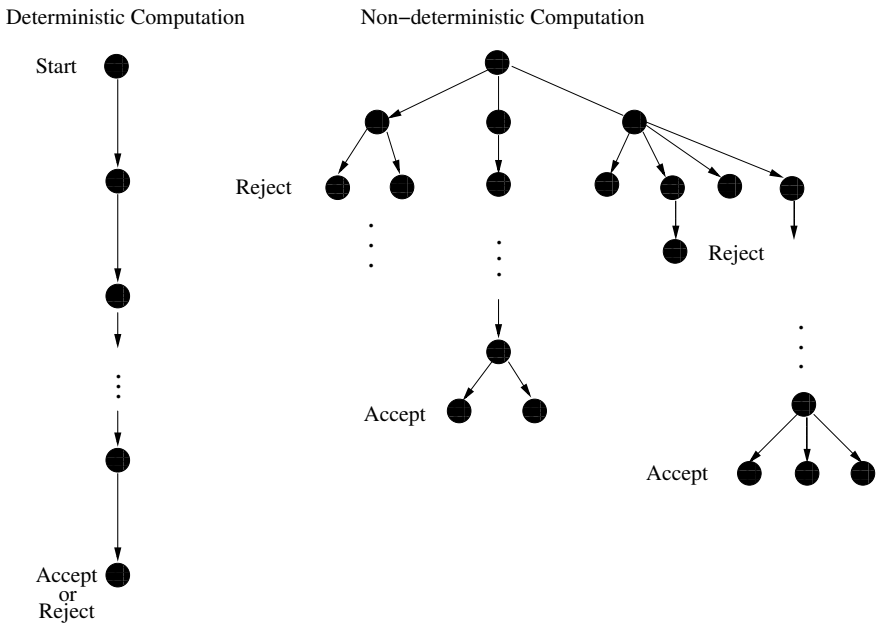
Complexity theory. What makes some problems computationally hard and other easy?

Automata

Automata are mathematical models of physical devices used to compute.

The definition of any automaton requires three components: states, language and transition rules.

(Non)Deterministic Computation



Deterministic automata follow a simple rule: given an input datum, every state of the computation is followed by only one state.

Alan Turing and his machines [dixit R.P. Feynman] (1/3)

Alan M. Turing published a most influential paper in 1936[1] in which he:

- Defined a systematic method, our modern definition of an **algorithm**.
- Provided a rigorous definition of a Turing machine, i.e. a powerful model of computation.
- Proved that it was possible to build a particularly powerful machine called **Universal Turing Machine (UTM)** that could simulate any other Turing machine in reasonable time.

A.M. Turing, "On Computable Numbers, with an application to the Entscheidung problem", Proc. London Math. Soc., **42**, 230-265 (1936-37).

Alan Turing and his machines [dixit R.P. Feynman] (2/3)

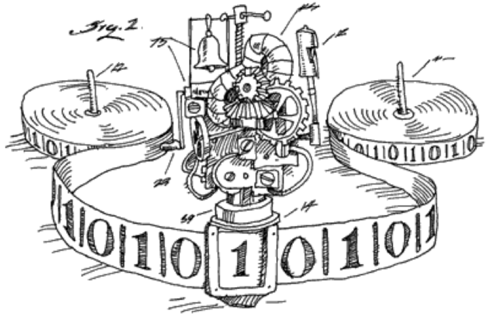
- Conjectured the **Church-Turing Thesis**, in which he established an equivalence correspondence between the existence of Turing machines and that of systematic methods. If the Church-Turing thesis is correct, then the existence or non-existence of systematic methods can be replaced *throughout mathematics* by the existence or non-existence of Turing machines.

Alan Turing and his machines [dixit R.P. Feynman] (3/3)

- Explained the fundamental principle of the modern computer, the idea of controlling the machine's operation by means of a program of coded instructions stored in the computer's memory.
- Proved that not all precisely stated mathematical problems can be solved by computing machines (examples: the decision and the halting problems.)

DTM (1/2)

A Deterministic Turing Machine (DTM) is an accurate model of a general purpose computer.



Turing Machines (drawing by Tom Dunne, American Scientist, March-April 2002)

DTM (2/2)

Definition of a Deterministic Turing Machine. A Deterministic Turing Machine (DTM) is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states
2. Σ is the input alphabet not containing the blank symbol $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{accept}} \neq q_{\text{reject}}$.

DTM Example (1/4)

The program specified by $\Gamma = \{0, 1, \sqcup\}$ ($\sqcup$ is the blank symbol), $Q = \{q_0, q_1, q_2, q_3, q_{\text{accept}}, q_{\text{reject}}\}$ and δ (Table 1), is used in a Deterministic Turing machine M to determine whether a number can be divided by 4 or, equivalently, whether the last two digits of a binary number, reading from left to right, are two consecutive zeros.

Table 1. Example of a deterministic Turing machine.

Q/Γ	0	1	$\sqcup$
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, \sqcup, L)$
q_1	$(q_2, \sqcup, L)$	$(q_3, \sqcup, L)$	$(q_{\text{reject}}, \sqcup, L)$
q_2	$(q_{\text{accept}}, \sqcup, L)$	$(q_{\text{reject}}, \sqcup, L)$	$(q_{\text{reject}}, \sqcup, L)$
q_3	$(q_{\text{reject}}, \sqcup, L)$	$(q_{\text{reject}}, \sqcup, L)$	$(q_{\text{reject}}, \sqcup, L)$

DTM Example (2/4)

For example, let $X = 10100$ be an input binary string (we shall read the input string from left to right). The initial state of M is q_0 and M's tape reads as

$\sqcup$	$\mathbf{1}$	0	1	0	0	$\sqcup$

where our first input symbol (in bold face) is the leftmost $\mathbf{1}$. So,the initial configuration of M is $q_0, \mathbf{1}$.

DTM Example (3/4)

The transition function specifies that for a state q_0 and input symbol 1, M must take q_0 as new state, write the value 1 in the cell where its read-write head is now located ($\mathbf{1}$) and take its read-write head one cell forward, i.e. to the right. So, M is now in the configuration $q_0, \mathbf{0}$ and its tape reads as

$\sqcup$	1	$\mathbf{0}$	1	0	0	$\sqcup$

DTM Example (4/4)

The full run of this program is given in the following sequence:

$$
\begin{array}{lrcl}
\text{Step 1:} & q_0, \sqcup\mathbf{1}0100\sqcup & \rightarrow & q_0, \sqcup 1\mathbf{0}100\sqcup \\
\text{Step 2:} & q_0, \sqcup 1\mathbf{0}100\sqcup & \rightarrow & q_0, \sqcup 10\mathbf{1}00\sqcup \\
\text{Step 3:} & q_0, \sqcup 10\mathbf{1}00\sqcup & \rightarrow & q_0, \sqcup 101\mathbf{0}0\sqcup \\
\text{Step 4:} & q_0, \sqcup 101\mathbf{0}0\sqcup & \rightarrow & q_0, \sqcup 1010\mathbf{0}\sqcup \\
\text{Step 5:} & q_0, \sqcup 1010\mathbf{0}\sqcup & \rightarrow & q_0, \sqcup 10100\sqcup \\
\text{Step 6:} & q_0, \sqcup 10100\sqcup & \rightarrow & q_1, \sqcup 1010\mathbf{0}\sqcup \\
\text{Step 7:} & q_1, \sqcup 1010\mathbf{0}\sqcup & \rightarrow & q_2, \sqcup 101\mathbf{0} \sqcup \sqcup \\
\text{Step 8:} & q_2, \sqcup 101\mathbf{0}b\sqcup & \rightarrow & q_y, \sqcup 10\mathbf{1} \sqcup \sqcup\sqcup
\end{array}
$$

Turing Machines (1/3)

- The Turing Machine is an idealization of the human computer.
- There are Deterministic and Nondeterministic Turing machines.

Both Turing machines models are equally powerful in terms of computability, i.e.

Any computable function f which can be computed by a DTM can also be computed by an NTM, and vice versa.

Turing Machines (2/3)

Four important properties of Turing Machines:

1) **The Universal Turing Machine** (UTM), can simulate the behavior of any other Turing machine.

2) The UTM and the von Neumann computer (our silicon-based computers) are equivalent in terms of computability power, i.e. any problem that can be solved using a von Neumann computer can also be solved using the UTM.

Turing Machines (3/3)

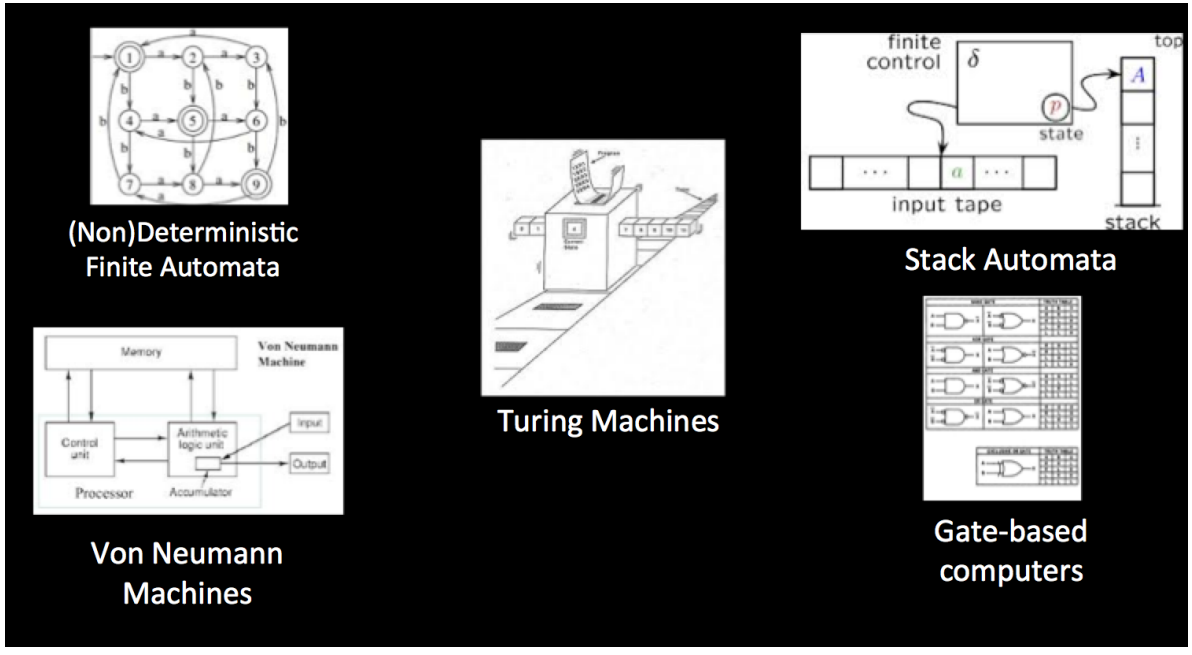
3) The same rationale applies to the UTM and gate-based computers that use sets of OR,NOT gates or AND,NOT gates, those computers are equivalent in terms of computability power. In other words, any problem that can be solved using a von Neumann computer can also be solved using the UTM or gate-based computers that use sets of OR,NOT gates or AND,NOT gates.

4) Although highly nontrivial, an arbitrary program written in any computer language can be translated into a series of elementary steps in a DTM.

Question: can any program running on a Turing machine be translated into a program written in a specific computer language, e.g. Python, C, C++, Basic, Go?

Universal Computation in Classical Computer Science (1/4)

Are these models of computation equally powerful, i.e. are their sets of computable functions identical?



Not necessarily! Different models of computation (may) have different computability power.

Universal Computation in Classical Computer Science (2/4)

Universality in Classical Computer Science is based on the following notion:

Two computer machines are equivalent if they are capable of simulating one another (i.e., of recognizing the same set of languages)

For example, Finite Deterministic Automata and Finite Nondeterministic Automata **are equivalent**, while Finite Deterministic Automata and Turing Machines **are not equivalent**.

Universal Computation in Classical Computer Science (3/4)

The Church-Turing Thesis. Three ways to express the thesis are:

1. The UTM can perform any calculation that any human computer can carry out.
2. Any systematic method can be carried out by the UTM.
3. **Every function which would be naturally regarded as computable can be computed by the Universal Turing Machine.**

Universal Computation in Classical Computer Science (4/4)

So,

A function is computable if and only if it is computable by the Universal Turing Machine.

Which also means,

A function is not computable if and only if it is not computable by the Universal Turing Machine.

Of course, this is highly controversial. However, so far no automata T has been designed so that a function computable in T is not computable by the UTM.

Computable functions and reversible computing (1/4)

Reversible computation is a central concept in quantum computation because Unitary operators are reversible.

Computable functions and reversible computing (2/4)

Let us start with an interesting claim:

Any computable function can be computed using reversible computing machines.

In other words,

For any irreversible function, it is always possible to produce a reversible circuit/algorithm that computes the same function.

Proof. There are several ways to prove this claim, among them the following two:

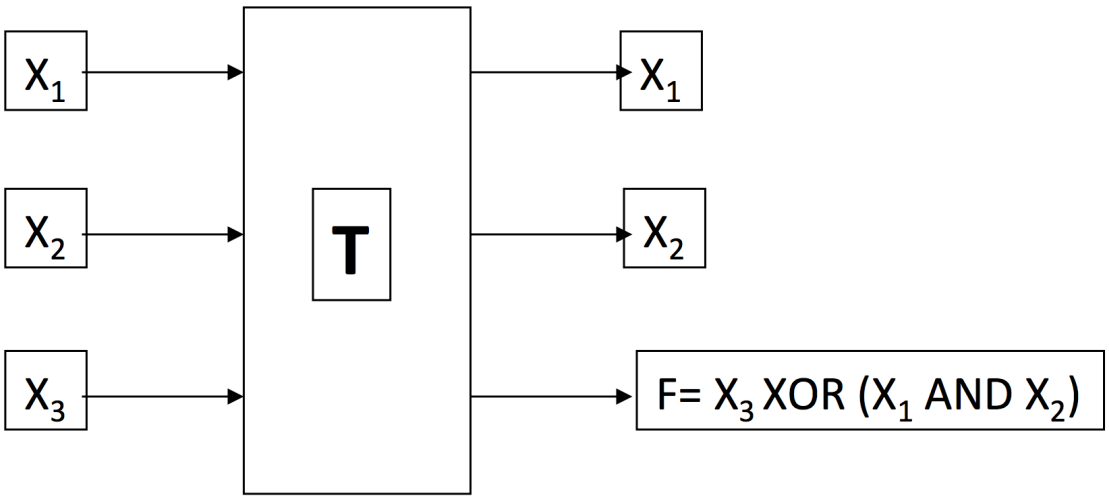
Computable functions and reversible computing (3/4)

1) In [1], C. Bennett showed that any Turing Machine can be simulated by a *Reversible* Turing Machine.

[1] C. Bennett. Logical Reversibility of Computation. IBM J. Res. Develop. 17(6), pp. 525-532 (Nov 1973).

Computable functions and reversible computing (4/4)

2) Universal reversible computation.



Input			Output		
X_1	X_2	X_3	X_1	X_2	F
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**

NOT(X_3) = Toffoli$(1, 1, X_3)$
AND(X_1, X_2) = Toffoli$(X_1, X_2, 0)$
NAND(X_1, X_2) = Toffoli$(X_1, X_2, 1)$

The Church-Turing Principle (1/5)

It is somewhat ironic, at least to me, that the branch of S&T that is perceived by society at large as a very precise and quantitative discipline, rests upon a conjecture!

This is one of the reasons for the tremendous philosophical and scientific relevance of David Deutsch's formulation of the Church-Turing principle:

The Church-Turing Principle (2/5)

The Church-Turing principle. Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.

The Church-Turing Principle (3/5)

In Deutsch's words, the rationale behind the Church-Turing principle was:

"To reinterpret Turing's 'functions which would be naturally regarded as computable' as the functions which may in principle be computed by a real physical system. For it would surely be hard to regard a function 'naturally' as computable if it could not be computed in Nature, and conversely".

The Church-Turing Principle (4/5)

Moreover, Deutsch argues the following.

"The reason why we are confident that the machines we call calculators do indeed compute the arithmetic functions they claim to compute is not that we can 'check ' their answers, for this is ultimately a futile process of comparing one machine with another: *Quis custodiet custodios ipsos?*

The real reason is that we believe the detailed physical theory that was used in their design. That theory, including its assertion that the abstract functions of arithmetic are realized in Nature, is empirical."

The Church-Turing Principle (5/5)

So,

A function is computable if and only if it is possible to build a physical system, like the Universal Quantum Turing Machine, to run it.

Quantum Computational Universality (1/2)

In 1995, Deutsch, Barenco and Ekert proposed the following notion of Universal computer:

A universal set of components is one that is adequate for the building of computers to perform *any physically possible computation*.

A **Universal Computer** is a single machine that can perform any physically possible computation.

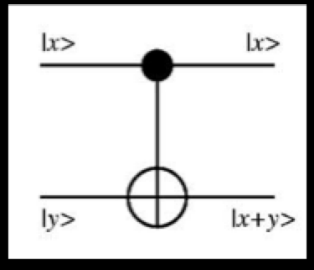
D. Deutsch, A. Barenco, A. Ekert. Universality in Quantum Computation. Proc. R. Soc. Lond. 449, 669-677 (1995)

Quantum Computational Universality (2/2)

In this sense, the computational models known as Quantum Circuit/gate model, Quantum Adiabatic Computation and Quantum Walks constitute Universal Models of Computation.

Quantum Gate Model

So, Universal Quantum Computation can be achieved by any experiment/physical platform/theoretical proposal that succeeds at implementing the following gates:



Controlled-not gate

$$\hat{H} = \frac{1}{\sqrt{2}}(|0\rangle_c\langle 0| + |0\rangle_c\langle 1| + |1\rangle_c\langle 0| - |1\rangle_c\langle 1|)$$

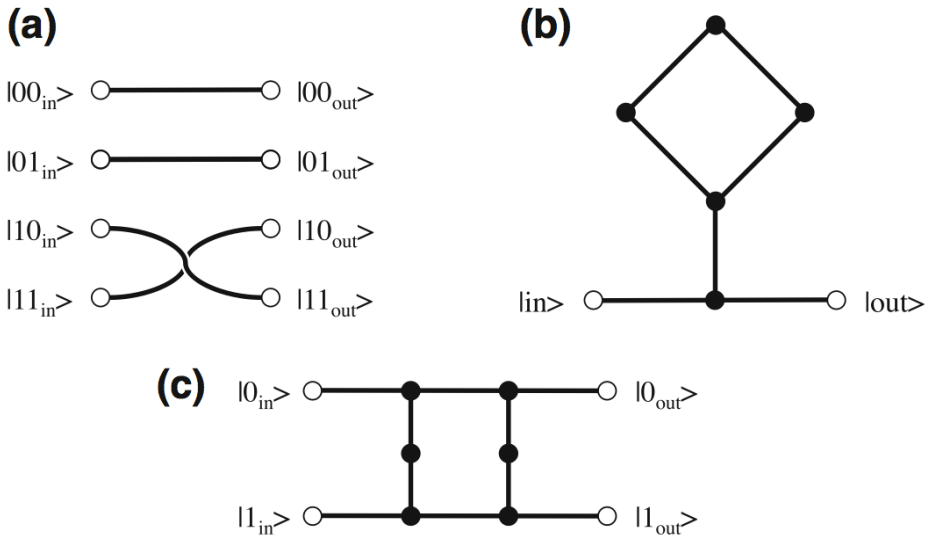
Hadamard gate

$$R(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i\theta} \end{bmatrix}$$

A phase-rotation gate

D. DiVincenzo. Phys. Rev. A. 51(2), 1015-1021 (1995)

Universal Computation by Quantum Walks (1/2)



a Widget for C_{not} gate, **b** widget for phase gate, and **c** widget for the basis-changing gate

A.M. Childs. Universal computation by quantum walk. Phys. Rev. Lett., 102:180501 (2009)

Universal Computation by Quantum Walks (2/2)

Two fundamental contributions:

- First to prove that the continuous quantum walk model is universal for quantum computation
- Scattering algorithms

Correctness of Quantum Algorithms

So,

A quantum algorithm is correct if and only if it finds a solution to the problem for which it was designed, according to a probability distribution computed from quantum amplitudes.