From Turing Machines to Quantum Computers

Wolfram Schroers
FB 8, University of Wuppertal

Wolfram.Schroers@Feldtheorie.de

October 2000

Overview

- What are "computable problems"?
- Turing machines and Computational Complexity
- Boolean Algebra and Reversibility
- Prospects and Problems of Quantum Computers
- Summary

What are "computable problems"?

Alan Turing 1936: Definition of "Turing machine" (deterministic/non-deterministic).

Church-Turing Thesis: A Turing machine can compute any function computable by a reasonable physical device.

$\Rightarrow$ Cannot be proven, only falsified!

Set of Turing machines is countable, the set of *families of functions* is not!

$\Rightarrow$ Most functions *cannot* be computed!

Probabilistic Turing machine:

Example: Database of N bits either all equal ("constant") or half 0 and half 1 ("balanced"). Distinguish between the two cases.

Deterministic TM requires $O(N/2+1)$ operations.

Probabilistic TM decreases error exponentially, e.g. $k = 100 \Rightarrow$ error probability $< 10^{-10}$.

Turing machines

DEFINTION 1 *A Turing machine is a triplet* $M = (\Sigma, K, \delta)$, *where*

- Σ *is a finite set of symbols called* $alphabet$ *containing the blank "␣",*
- K *is a finite set of machine states, containing*
  - *unique* $s_0 \in K$, *called* $initial\ state$,
  - *set* $\{e\} \subset K$ *called* $final\ states$,
- $\delta : K \otimes \Sigma \mapsto K \otimes \Sigma \otimes \{-1, 0, 1\}$ *is the* $transition\ function$.

Observations:

- Set of Turing machines is countable.
- Set of input languages is countable (cf. grammars and languages).

Accepted languages:

DEFINTION 2 *A finite sequence $x \in L \subseteq \Sigma^*$ is $accepted$ by a TM M iff M is started in state s_0 on the left end of the tape and reaches an accepting state $e_i \in K$ in a finite number of steps. We write $L(M)$ for the set of sequences accepted by M.*

Turing machines are computable:

THEOREM 1 *There exists a Turing machine U which can simulate any other Turing machine M with input x. We write $U(M, x) = M(x)$. This U is called the $universal\ Turing\ machine$.*

Halting problem: Decide if TM M halts on arbitrary input x.

THEOREM 2 *There is no Turing machine that solves the $halting\ problem$ on all inputs (M, x). Thus this problem is not computable.*

Non-deterministic TMs

Nondeterministic TM: Transition function δ maps into a subset of $K \otimes \Sigma \otimes \{-1, 0, 1\}$ rather than a single element. This machine accepts input iff there is a sequence of choices which lead from the starting configuration to an accepting state.

Remember: Deterministic universal TM can emulate NDTM!

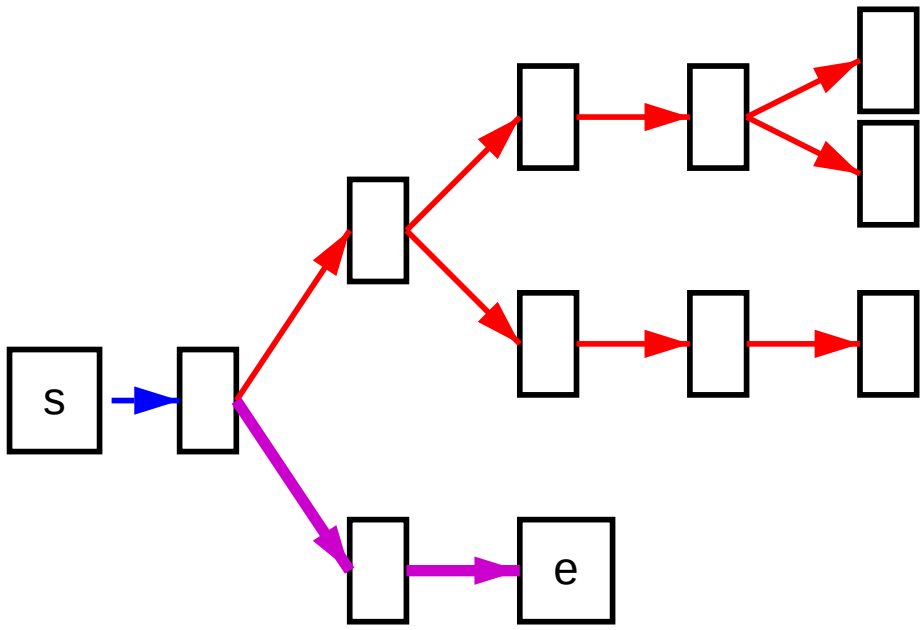


Figure 1: Example of the tree traversed by the machine states of a non-deterministic Turing machine (NDTM).

Computational Complexity

Measure of Complexities: Time and Space.

T, S: functions over the natural numbers,

Σ: a fixed alphabet,

Σ^*: the set of all finite sequences over Σ,

$L \subseteq \Sigma^*$: A language built from Σ.

For $x \in L$, $|x|$ is the number of symbols in x.

DEFINTION 3 *A TM M runs in time (space) $T(n)$ ($S(n)$) iff $\forall x \in L(M) : M(x)$ halts in at most $T(|x|)$ steps (uses no more than $S(|x|)$ tape cells).*

Most important classes

Polynomial time computations:

DEFINTION 4 *For the problem class given by the language* $L(D_i) \subseteq \Sigma^*$ *let* $D_1, D_2, \ldots$ *be a numeration for deterministic TMs with clocks stopping in polynomial time. This class of languages define the set of* $polynomial\ time\ computations$*:*

$$\boxed{P = \{L(D_i) \,|i \geq 1\}}.$$

Non-deterministic polynomial time computations:

DEFINTION 5 *For the problem class given by the language* $L(N_i) \subseteq \Sigma^*$ *let* $N_1, N_2, \ldots$ *be a numeration for non-deterministic TMs with clocks stopping in polynomial time. This class of languages define the set of* $non\text{-}deterministic\ polynomial\ time\ computations$*:*

$$\boxed{NP = \{L(N_i) \,|i \geq 1\}}.$$

Polynomial space bounded:

DEFINTION 6 *For the problem class given by the language* $L(D_i) \subseteq \Sigma^*$ *let* $D_1, D_2, \ldots$ *be a numeration for deterministic TMs with polynomially bounded space consumption. This class of languages define the set of* $polynomial\ space\ bounded\ computations$*:*

$$PSPACE = \{L(D_i) \,|\, i \geq 1\}.$$

Non-deterministic polynomial space bounded:

DEFINTION 7 *For the problem class given by the language* $L(D_i) \subseteq \Sigma^*$ *let* $N_1, N_2, \ldots$ *be a numeration for non-deterministic TMs with polynomially bounded space consumption. This class of languages define the set of* $non\text{-}deterministic\ polynomial\ space\ bounded\ computations$*:*

$$NPSPACE = \{L(D_i) \,|\, i \geq 1\}.$$

Relations between classes

THEOREM 3 *The two problem classes $PSPACE$ and $NPSPACE$ are equivalent (Savitch 1970):*

$$\boxed{PSPACE = NPSPACE}.$$

THEOREM 4 *Among the problem classes P, NP, $PSPACE$, $NPSPACE$ the following implications hold:*

$$\boxed{P \subseteq NP \subseteq PSPACE = NPSPACE}.$$

Open question:

$$\boxed{P = NP \ ?}$$

NP-completeness

DEFINTION 8 *1. Language* $A \subseteq \Sigma^*$ *is* $many\text{-}one$ $polynomial\ time\ reducible$ *to* $B \subseteq \Gamma^*$ *iff* $\exists$ *a function* $f \in P, f : \Sigma^* \to \Gamma^*$ *such that*

$$x \in A \Leftrightarrow f(x) \in B.$$

We write $A \leq_m^p B$.

2. A *is* NP $hard$ *iff:*

$$\forall B : B \in NP \Rightarrow B \leq_m^p A.$$

3. *A language* A *is* NP*-"complete" iff* A *is in* NP *and* A *is* NP *hard.*

$\Rightarrow$ All NP-complete problems are of equivalent difficulty!

Open problems

Decision problems:

DEFINTION 9 *A $decision\ problem$ is a problem which can be formulated such that the result is a yes or no answer.*

We can subdivide NP into two subsets for *decision problems*:

Class NP: The class NP is defined by the polynomial time non-deterministic problems to checking for a YES solution.

Class $CoNP$: If the problem is to check for a NO solution, it is said to belong to the class $CoNP$.

$\Rightarrow$ The problems in $CoNP$ are complement to those in NP.

If $NP \neq CoNP$ we can define the non-empty class of "open" problems:

DEFINTION 10 *A language A is $open$ if $A \in NP \cap CoNP$.*

Relations of classes

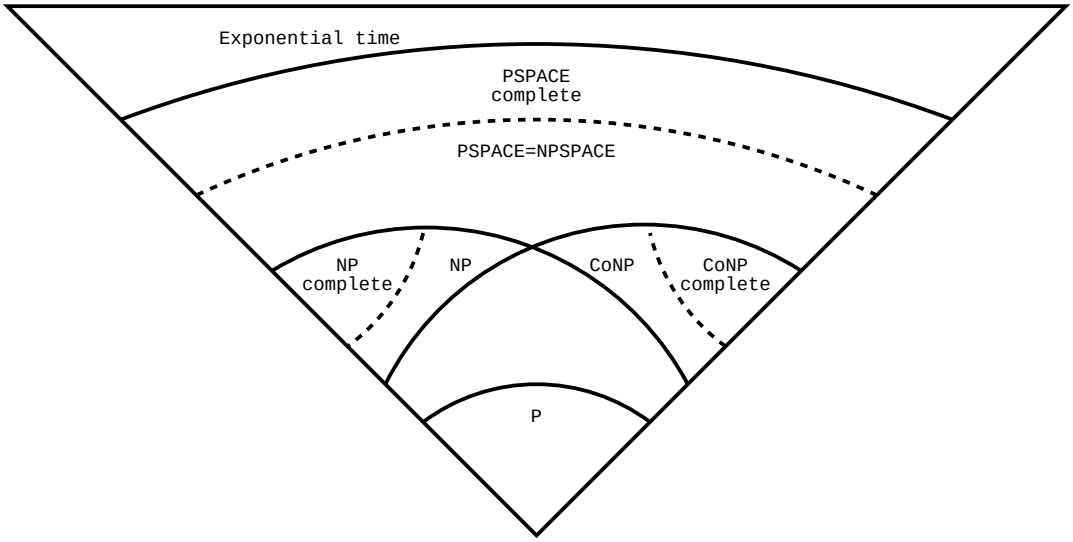


Figure 2: The sets of computable problems

Boolean Circuit Model

DEFINTION 11 *A $Boolean\ circuit$ is a directed acyclic graph with nodes which are associated with Boolean functions. These nodes are called $logical\ gates$. A node with n input and m output lines is associated with a function*

$$f : \{0,1\}^n \mapsto \{0,1\}^m.$$

A	B	AND	OR	XOR	NOT B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	1
1	1	1	1	0	0

Table 1: Truth table for Boolean operations.

Additional operations: FANOUT and ERASE.

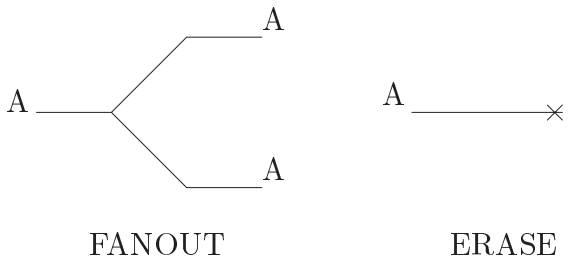


Figure 3: FANOUT and ERASE operations.

Boolean circuits and Turing machines

DEFINTION 12 *The $uniform$ Boolean circuit model can be designed with polynomial cost by a Turing machine.*

THEOREM 5 *The models of uniform Boolean circuits and Turing machines are polynomially equivalent.*

Reversible computation

Biggest difficulty with classical computing: Heat dissipation!

Landauer 1961: All but one operation can be performed in logical (and physical) reversible manner.

$\Rightarrow$ No heat is dissipated for most operations!

Primitive ERASE: Erasing the last copy of a bit requires the dissipation of at least $k_B T \ln 2$ heat ($Landauer´s\ principle$).

Making computation reversible: Compute a function $f : a \mapsto f(a)$:

$$f : a \mapsto (a, j(a), f(a)) \,,$$

with junk bits $j(a)$.

Uncomputing is straightforward, but keeps extra junk unless ERASE is used:

$$f^\dagger : (a, j(a), f(a)) \mapsto (a, f(a)) \,.$$

Classical Toffoli gate

Classical Toffoli gate: A reversible, classical gate based on boolean operations.

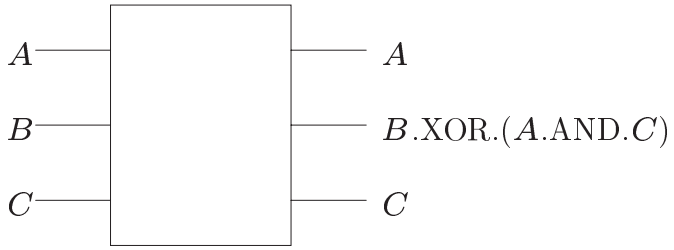


Figure 4: The Classical Toffoli gate.

$$TG\,(A,B,C) = \begin{cases} A.\text{AND}.C, & \text{for } B = 0 \\ A.\text{XOR}.B, & \text{for } C = 1 \\ \bar{A}, & \text{for } B = C = 1 \\ A, & \text{for } B \neq C = 1 \end{cases}$$

Depending on inputs, it works as AND, XOR, NOT or FANOUT.

Quantum Boolean Algebra

Qubit: Quantum mechanical variant of classical *bit* (binary digit).

DEFINTION 13 *A quantum bit ("qubit") is described by the Hilbert space of a two-level spin-$\frac{1}{2}$ particle. Its basis states are $|0\rangle$ and $|1\rangle$.*

k qubits span a Hilbert space of dimension 2^k denoted by $\{|00\cdots00\rangle, |00\cdots01\rangle \ldots |11\cdots11\rangle\}$.

DEFINTION 14 *A quantum gate is a unitary operator acting on a finite Hilbert space spanned by k qubits.*

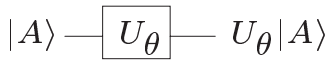
THEOREM 6 *The most general 1-qubit quantum gate is given by*

$$U_\theta = \begin{pmatrix} e^{i(\delta+\sigma/2+\tau/2)}\cos(\theta/2) & e^{i(\delta+\sigma/2-\tau/2)}\sin(\theta/2) \\ -e^{i(\delta-\sigma/2+\tau/2)}\sin(\theta/2) & e^{i(\delta-\sigma/2-\tau/2)}\cos(\theta/2) \end{pmatrix}.$$

We usually take $\delta = \sigma = \tau = 0$.

Important Quantum Gates

Single quantum bit gate: $f : |A\rangle \mapsto U_\theta |A\rangle$



THEOREM 7 *The quantum NOT operation is performed by the unitary operator*

$$U_{\theta=\pi} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

THEOREM 8 *The quantum XOR operation for 2 qubits using the following set of basis states*

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

is given by the unitary matrix

$$U_{XOR} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Multi-Qubit Quantum Gates

The *quantum XOR* gate is already sufficient to build a *qubit swapping gate*; together with a single qubit $U_{\pi/4}$-gate it allows construction of a *quantum Toffoli gate*!

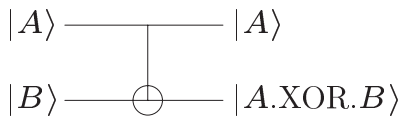


Figure 5: Quantum XOR gate.

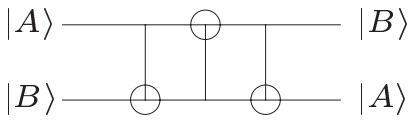


Figure 6: Swapping of 2 qubits.

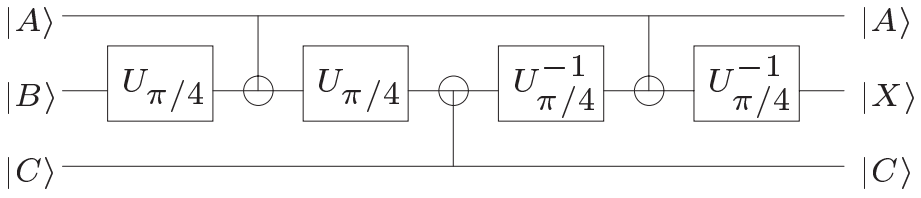


Figure 7: Quantum variant of Toffoli gate with $|X\rangle = |B.\mathrm{XOR}.\,(A.\mathrm{AND}.C)\rangle$.

Quantum Turing machines

Unlike classical TM, there is no QTM definition which is likely to be used as a model for implementation.

DEFINTION 15 *A $Quantum\ Turing\ machine$ is a triplet $Q = (\Sigma, Q, \delta)$, where*

- Σ *is the alphabet containing the blank "$\sqcup$",*
- $Q = \{q_0, \ldots, q_s\}$ *is a finite set of machine states containing*
  - $s \in Q$ *the $initial\ state$,*
  - $e \in Q$ *the $final\ state$,*
- $\delta : Q \otimes \Sigma \otimes Q \otimes \Sigma \otimes \{-1, 0, 1\} \mapsto \mathcal{C}$.

Quantum Complexity Classes

The quantum complexity classes QP, BQP are defined analoguesly to their classical counterparts. However, there are no non-deterministic QTMs!

The following implications can be proven:

THEOREM 9 *Among the classical complexity classes P, NP, BPP, $PSPACE$ and the quantum classes QP, BQP the following implications hold:*

1. $\boxed{P \subseteq QP \subseteq BQP}$,

2. $\boxed{BPP \subseteq BQP \subseteq PSPACE}$.

Furthermore, Bennett, Bernstein, Brassard, Vasirani 1997: Special case of "Oracle QTM" cannot solve NP-problems in polynomial time!

$\Rightarrow$ Possibly

$$\boxed{NP \not\subset BQP\ ?}$$

Shor´s algorithm

Important problem: Factoring numbers (e.g. for Cryptography!)

$\Rightarrow$ This problem is NP-complete!

Classical computers: Best algorithm on classical Turing machines runs in $O\left(\exp\left[(64/9)^{1/3}(\ln N)^{1/3}(\ln\ln N)^{2/3}\right]\right)$ steps (Odlyzko 1995).

Quantum computers: Best algorithms runs in $O\left((\ln N)^3\right)$ steps (Shor 1994)!

$\Rightarrow$ Hope to gain exponential factor for other algorithms, too!

But: This is the *only* algorithm known so far which shows exponential speedup over classical machines!

Sources of Errors

Assumption: Unitary evolution operator U acting on the QC state $|A\rangle$.

$\Rightarrow$ $|A\rangle$ will always be a pure state!

Problems:

- Interaction with environment will evolve QC state (e.g. $\alpha|0\rangle + \beta|1\rangle$) and environment state $|e\rangle$ into a linear superposition of both ($\rightarrow$ $\alpha\left(|e_{00}\rangle|0\rangle + |e_{01}\rangle|1\rangle\right) + \beta\left(|e_{11}\rangle|1\rangle + |e_{10}\rangle|0\rangle\right)$!

  $\Rightarrow$ From the p.o.v. of the QC alone the system is now in a **mixed state**, thus information is destroyed.

- Decoherence of entangled states will destroy computation.

Solution: Must find a way to implement quantum error correction, i.e. the quantum algorithm must implement extra information to correct deviations from desired evolution!

Physical implementations

Problem: Decorrelations are modeled by

$$1 - e^{-t/\tau_{\text{dec}}}.$$

$\Rightarrow$ Probability for getting the correct result is

$$\boxed{P \approx P_0 e^{-\text{is}(n)t(n)/\tau_{\text{dec}}}},$$

where P_0 is probability for result with no errors, $s(n)$ is total number of qubits and $t(n)$ is the time needed for the computation.

Proposed quantum system with decorrelation times and number of possible operations:

Quantum system	t_s/s	τ_{dec}/s	**Comp. steps**
Mössbauer nucles	10^{-9}	10^{-10}	10^{9}
GaAs electrons	10^{-13}	10^{-10}	10^{3}
Au electrons	10^{-14}	10^{-8}	10^{6}
Trapped indium ions	10^{-14}	10^{-1}	10^{13}
Optical microcavity	10^{-14}	10^{-5}	10^{9}
Electron spin	10^{-7}	10^{-3}	10^{4}
Electron quantum dot	10^{-6}	10^{-3}	10^{3}
Nuclear spin	10^{-3}	10^{-4}	10^{7}

Conclusions and Outlook

- Quantumcomputation is a young subject attracting huge interest
- QMs are conceptually different from classical computers and require new algorithms
- Quantum Computers allow reversible computations
- It is unclear if larger machines can actually be built and perform long runs
- Up to today only one algorithm has been discovered which shows an exponential speedup over classical algorithms
- However, much progress has been achieved in the past 6 years both theoretical as well as experimental

References

[1] J. Hartmanis, Computational Complexity Theory, Proceedings of Symposia in Applied Mathematics, Vol. 38, AMS Providence, Rhode Island 1988

[2] L. Kronsjö, Computational Complexity of Sequential and Parallel Algorithms, John Wiley & Sons Ltd. 1986

[3] R. Herschel, Einführung in die Theorie der Automaten, Sprachen und Algorithmen, R. Oldenbourgh Verlag, München 1974

[4] S. L. Braunstein, Quantum Computation in: S. Braunstein (ed.) Quantum Computing – Where do we want to go tomorrow?, Wiley-VCH 1999

[5] J. Gruska, Quantum Computing, McGraw-Hill Publishing Company 1999

[6] D. Aharonov, Quantum Computation, quant-ph/9812037

[7] M. Ohya, I. V. Volovich, Quantum Computing, NP-complete Problems and Chaotic Dynamics, quant-ph/9912100

[8] E. Rieffel, An Introduction to Quantum Computing for Non-Physicists, quant-ph/9809016v2

[9] Seminar Quantencomputer, SS98, MPI, http://www.mpi-sb.mpg.de/ roehrig/courses/qc98/

[10] Frank Schön, http://www.mpi-sb.mpg.de/ roehrig/courses/qc98/kompl.html

[11] E. Knill, R. Laflamme, G. Milburn, Thresholds for Linear Optics Quantum Computation, quant-ph/0006120

[12] B. E. Kane, Silicon-based Quantum Computation, quant-ph/0003031

[13] A. M. Steane, D. M. Lucas, Quantum Computing with trapped ions, atoms and light, quant-ph/0004053

[14] V. Giovannetti, D. Vitali, P. Tombesi, A. Ekert, Towards scalable quantum computation with cavity QED systems, quant-ph/0004107

[15] D. V. Averin, Quantum computing and quantum measurement with mesoscopic Josephson junction, quant-ph/0008114

[16] J. A. Jones, NMR Quantum Computation, quant-ph/0009002

[17] H. De Raedt, A. Hams, K. Michielsen, On the Problem of Programming Quantum Computers, quant-ph/0008015