CMPE598 Lecture notes

Mert Çıkla

02/04/2018

1 Two-way quantum finite automata

As opposed to one-way automata, being two-way allows the machine to move on its input tape which is marked with a ¢ at its start and \$ at the end. A superoperator(transition matrix) is applied to the quantum part. The moves consist of two parts.

1. Evolution of the quantum state according to the superoperator associated with the current configuration.The superoperator is applied giving a new superposition and coin outcome with an associated probability.

2. The classical state, the coin outcome and the scanned symbol determine the next classical state and tape head direction.

$A = \{a^n b^n | n \in \mathbb{N}\}$ is a non-regular language that cannot be recognized by two-way classical DFAs. Given below is a bounded error algorithm that recognizes this language in polynomial expected time for two way automaton augmented with a few quantum bits as described above. k is the error adjustment parameter to be increased if a lower error bound is needed.

Algorithm 1: Bounded error, polynomial algorithm for the $a^n b^n$

```
1  Check if the input is of the form a^+ b^+, otherwise REJECT
2  Move the head to the first input symbol and set the quantum state to |q_0⟩
3  LOOP While the currently scanned symbol is not $
4      if current symbol is a then
5          Rotate the qubit with angle √2π
6          Move the head to the right
7      if current symbol is b then
8          Rotate the qubit with angle −√2π
9          Move the head to the right
10 Measure the qubit. If the result is |q_1⟩, REJECT
11 DO two times
12     Move the tape head to the first input symbol
13     LOOP While the currently scanned symbol is not $ or ¢
14         Simulate a classical coin flip. If the result is heads, move right.
           Otherwise, move left.
15 If the both times the loop ended at $, simulate k coin flips.
16 If all coin flip results are heads, ACCEPT.
```

If the input is of the form $a^n b^{n'}$ and $n \neq n'$, this machine rejects after the first loop at line **10** with probability at least $\frac{1}{2(n-n')^2}$.

Proof. The quantum state gets rotated by a total of $\sqrt{2}(n-n')\pi$ radians. The probability of observing $|q_1\rangle$ is

$$Pr(q_1) = \sin^2\left(\sqrt{2}(n-n')\pi\right)$$

We bound $Pr(q_1)$ as follows.Let k be the closest integer to $\sqrt{2}(n-n')$. Assume that $k < \sqrt{2(n-n')^2-1}$ (the other case is symmetric). Then k $\leq \sqrt{2(n-n')^2-1}$ because k^2 is an integer and $2(n-n')^2-1$ is the largest integer that is less than $(\sqrt{2}(n-n')^2)$

$$(\sqrt{2}(n-n') - \sqrt{2(n-n')^2-1})(\sqrt{2}(n-n') + \sqrt{2(n-n')^2-1})$$

$$= 2(n-n')^2 - 2(n-n')^2 + 1 = 1$$

$$\sqrt{2}(n-n') - k \geq \sqrt{2}(n-n') - \sqrt{2(n-n')^2-1}$$

$$= \frac{1}{\sqrt{2}(n-n') + \sqrt{2(n-n')^2-1}} > \frac{1}{2\sqrt{2}(n-n')}$$

We have

$$0 < \sqrt{2(n-n') - k} < \frac{1}{2}$$

(because k is the closest integer) $\forall x \in \left[0, \frac{1}{2}\right]$,

$$\sin(x\pi) \geq 2x$$

So,

$$\sin^2(\sqrt{2}(n-n')\pi) = \sin^2\left((\sqrt{2}(n-n') - k)\pi\right)$$

$$\geq 4\left(\sqrt{2}(n-n') - k\right)^2 \geq 4\left(\frac{1}{2\sqrt{2}(n-n')}\right)^2 = \frac{1}{2(n-n')^2} > \frac{1}{(n+n'+1)^2}$$

□

Each random walk takes expected $(n+n'+1)^2$ time. The random walk's probability of reaching the $(n+n'+1)^{st}$ cell is exactly

$$\frac{1}{n+n'+1}$$

So, repeating it twice and flipping k coins after that gives a probability of acceptance

$$p_{acc} = \frac{1}{2^k(n+n'+1)^2}$$

Machine rejects at line 10 with probability

$$p_{rej} > \frac{1}{2(n-n')^2}$$

Overall, if $n = n'$ the machine accepts with 0 error. If $n \neq n'$

$$\frac{p_{acc}}{p_{rej}} = \frac{1}{2^k}$$

Loop at line 3 takes $\mathcal{O}(n+n')$ time and each random walk takes $\mathcal{O}\left((n+n')^2\right)$ time. Hence, the expected running time of the machine is at most $\mathcal{O}\left((n+n')^4\right)$

2 Algorithms for non-constant space machines

Usually our algorithms have flexible memory and we are not limited by a space restriction. The amount of memory used by algorithm in question is measure by space complexity of the algorithm and the runtime is measured by time complexity. It is generally expected to have polynomial size memory and that harder problems require more memory. Finite state machines were not able to have access to a flexible memory and had a very limited amount of it. Turing machine model is not restricted in this sense with access to an unlimited tape that can be written onto. This lets the Turing machine to solve any problem a computer is able to do.

There are two main models of computation widely used in the literature quantum computation.

1. Quantum Turing Machine Model
2. Circuit Model

Quantum Memory is a number of qubits/registers. A 3 qubit operation is an 8x8 matrix. Hadamard(H) and Toffoli(T) gates form a universal gate set for quantum computation. Meaning any other quantum operation can be approximated to a required degree of precision with enough Hadamard and Toffoli gates.

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \qquad \text{—}\boxed{H}\text{—}$$

Figure 1: Hadamard transform in its matrix and gate forms

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

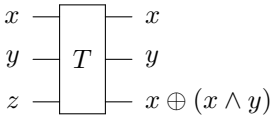

Figure 2: Toffoli gate in its matrix and circuit gate forms

We can always assume that a computation that takes $t(n)$ steps in a classical computer to compute $f(x) = y$ where x is a string of n bits and y is a single bit saying yes/no can be computed on a quantum turing machine which realizes the function

$$f(|x\rangle\,|c\rangle) = (|x\rangle\,|c \oplus y\rangle)$$

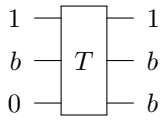

Figure 3: Fanout gate with a Toffoli gate where input b is copied

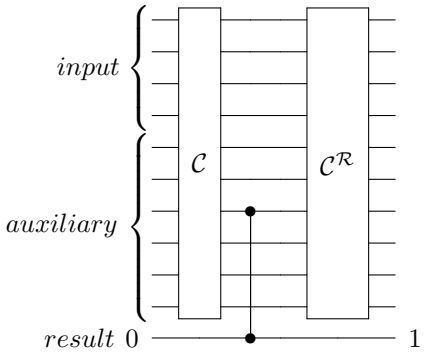

Figure 4: Reversing a computation

where we do not show the auxiliary bits which are unchanged at the end in $2t(n) + 1$ steps. Figure 4 depicts a reversible computation using gate C and its reverse C^R which takes the input and auxiliary bits and reverts them back to their original states. The result of the computation is extracted before reversal occurs.