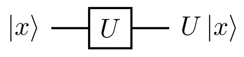
**MAT 4930: Quantum Algorithms and Complexity** **Spring 2021**

# Lecture 7: Reversible Circuits

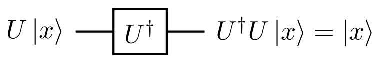
*Lecturer: Jean-François Biasse* *TA: Robert Hart*

## 7.1 Quantum Computations are Reversible

In this course, we demonstrate the superiority of quantum computers over their classical counterpart for certain well identified problems, in particular search problems and the factoring of large intergers. However, we first need to address the following question: *Are quantum computers at least as good as classical ones?* This may seem like a silly question when we seen all the hype surrounding quantum computers, but it is in fact a *very* relevant question in light of a major constraint: quantum computations need to be reversible. Note that here we assume no measurement is performed as part of the computation (this would not alleviate our problem). Assume $U \in \mathbb{C}^{2^n \times 2^n}$ is a unitary matrix representing a circuit on $n$ qubits:

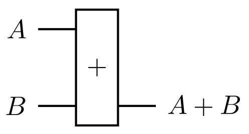


Then we necessarily have an inverse circuit that takes any $U|x\rangle$ back to $|x\rangle$, namely:



This restriction is anything but innocent. Indeed, a large proportion of the algorithms we want to be able to perform are inherently *not reversible*

**Example 1** *For example, the following circuit (which we do not denote with the braket notation to avoid fusion) is not reversible:*



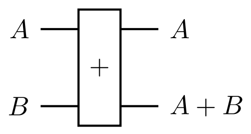
*Indeed, there is no way to retrieve $A$ and $B$ from $A + B$.*

The above restriction raises two questions

- Can we turn non-reversible operations into reversible ones?
- What is the cost of such an operation?

## 7.2 Using Ancillae Qubits

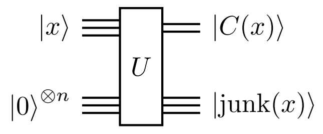
Turning a non-reversible operation into a reversible one is an easy task if we are willing to accept redundancy of information. Indeed, it suffices to return extra information about the input to be sure that the reverse path can be implemented. For example, the addition between two numbers can be made reversible if we return one of the inputs:



Since the question of reversibility is not an issue, we turn ourselves to the second issue: what is the cost of making a computation reversible? This is a non-trivial question, and depending on the particular algorithm we are trying to adapt, there might be optimizations possible, in particular in terms of additional memory required. However, there is also a generic result from Bennett that gives estimates on on the extra resources required.

**Theorem 7.1 (Bennett)** *Let $\varepsilon > 0$. There is a generic technique to convert any algorithm taking time $T$ and space $S$ into a reversible algorithm taking time $T^{1+\varepsilon}$ and space $O(S\log(T))$.*

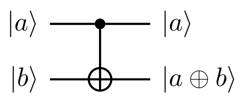
To interpret the above in terms of the cost functions we have defined, remember that the time $T$ of a circuit is its depth, while its width is the space $S$ is the number of (qu)bits it requires. This means that we can always hope to implement a classical function $x \mapsto C(x)$ (not necessarily reversible) by a quantum curcuit of the formal



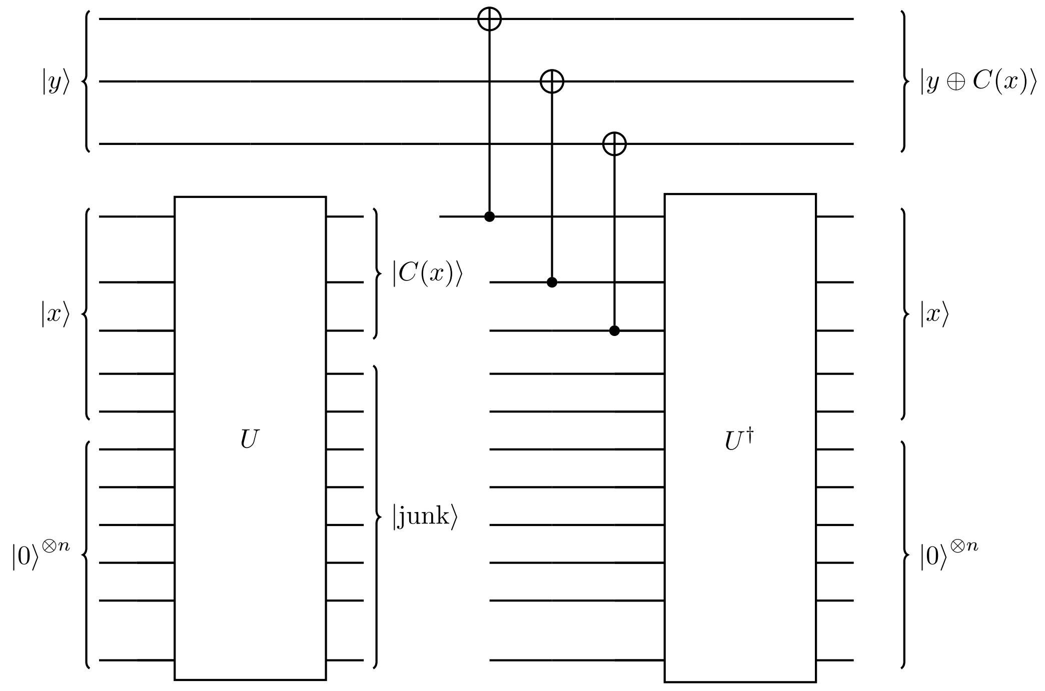
Here $|\text{junk}(x)\rangle$ denotes the extra information that is required in order for the operation to be reversible. The extra qubits necessary that enter the circuit in the state $|0\rangle^{\otimes n}$ and exit in the state $|\text{junk}(x)\rangle$ are called *ancilla* qubits. In general, the junk qubits cannot be forgotten (or measured) because they are potentially entangled with the answer $C(x)$ of the classical calculation. They must be kept in memory until the end of the computation. This restriction is particularly relevant when considering a superposition of $|C(x)\rangle$ for many different $x$. If we somehow learn junk$(x)$, then the state collapses to $|C(x)\rangle$ for the corresponding $x$. It is usually preferable to "uncompute" the junk information in order to perform the following operation:

$$|x\rangle\,|0\rangle^{\otimes n} \xrightarrow{U_C} |x\rangle\,|C(x)\rangle\,|0\rangle^{m},$$

for some $m \leq n$. This is done by using the previously seen CNOT gate, which is represented by the following circuit



The CNOT gate allows us to copy $C(x)$ qubit-by-qubit on a set of ancilla qubits, before the the "uncomputation" with the action of $U^\dagger$.

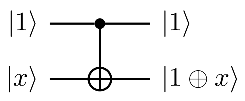


## 7.3 Converting Classical Universal Gate Sets

Having set the principles of ancilla qubit management, and knowing that all classical circuits can be turned into a quantum one, we have one last task: checking that a given classical reversible circuit can be turned into a quantum one. The input classical reversible circuit is given with respect to a universal set of classical gates. There exist many of them, for example

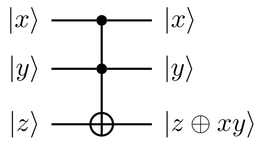
- AND, OR, and NOT.
- AND and NOT.
- OR and NOT.
- NAND.
- NOR.

Then, we can turn a classical logical circuit into a quantum one by finding the quantum circuit that implements each logical gate in the universal set used to write the input classical circuit, and following the ancilla management rules given above. Note that the CNOT quantum gate directly implemented the classical XOR operation. The CNOT gate also does the NOT operation via



Finally, to do an AND operation, we introduce a very important gate: the Toffoli gate.

**Definition 7.2 (Toffoli gate)** *The Toffoli gate is given by* $|x\rangle |y\rangle |z\rangle \mapsto |x\rangle |y\rangle |z \oplus xy\rangle$ *and is represented by*



The Toffoli gate is not a part of of the Clifford + $T$ + CNOT universal quantum gate set predominantly used. However, it can be efficiently generated by Clifford + $T$ + CNOT gates as shown in Figure 7.3.

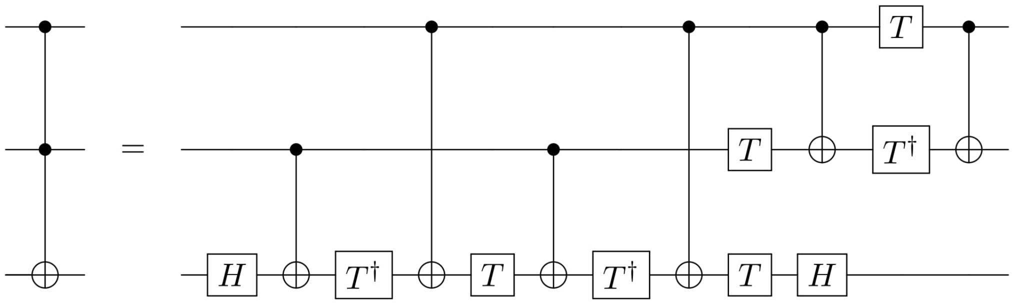


Figure 7.1: Toffoli gate with Clifford + $T$ + CNOT

The Toffoli gate implements the AND classical gate, thus completing the AND, OR, NOT gate set required to synthesize any classical circuit.