Royal Skousen, QAM

Reversible Operators

1. *Reversibility and irreversibility*

In designing a quantum computational system for analogical modeling, every operator meets the following two requirements:

(1) *simultaneity:* each operator must be defined so that it can apply simultaneously to each of the 2^n supracontexts;

(2) *reversibility:* each operator must be reversible.

The first requirement allows us to take advantage of the simultaneity of quantum computing. The second requirement basically means that no erasure of data is permitted prior to observation of the system (that is, prior to observation of the superpositioned supracontexts). Each data occurrence, after being read, must be kept or, if altered, must remain recoverable. Any computational result must be recoverable, and by keeping all the input data, we insure recoverability.

Let us consider what we mean by a reversible operator. The basic idea is that after an operator has been applied, we are able to determine from the final (or output) state what initial (or input) state it came from. This requirement of recoverability basically means that there is a one-to-one connection between inputs and outputs, that no mergers or splits occur, only a shifting (or renaming, so to speak) of representations.

One clear example of a reversible operator is *negation.* A *not* gate, or N gate (where the N stands for negation), is reversible because we simply switch or flip the polarity of a state a (true to false and false to true). In the following listing, a_i represents the initial state of a, while a_f represents the final state of a:

N gate

a_i	a_f
0	1
1	0

So given a final state a_f of 0 (false), we know that a_i was 1 (true); similarly, $a_f = 1$ implies that $a_i = 0$.

On the other hand, the *and* operator is not reversible. With an *and* gate, the final state c_f is true (or 1) only if a_i and b_i were both true (or 1). If the final state is false (or 0), then there are three possible sets of initial states (00, 01, or 10), and we do not know which set of initial states produced the false output:

and gate

a_i	b_i	c_f
0	0	0
0	1	0
1	0	0
1	1	1

2. *Reversibility of* and / nand

In quantum computing, however, we can construct a reversible gate that can be used as an *and* gate. We do this by constructing what is called a *control-control-not* gate (or CCN gate, for short). In this system, we switch the polarity of an initial state c_i only if two other initial states a_i and b_i are each true. The initial states a_i and b_i act as control states and c_i acts as a *not* state (thus, *control-control-not*). We get the following input-output relationships for the CCN gate:

CCN gate

a_i	b_i	c_i	a_f	b_f	c_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

For this reversible gate, there are eight possible sets of initial states and eight possible sets of final states. For the first six cases, the set of final states is identical to the set of input states (thus 000 → 000, 001 → 001, 010 → 010, 011 → 011, 100 → 100, 101 → 101). For the last two cases, we simply switch the polarity of the *c* state (thus 110 → 111 and 111 → 110). This results in a unique one-to-one function between all the sets of states. No information is lost, and from every set of output states we can determine the unique set of input states from which it was derived. We also emphasize here that with a CCN gate the two control states *a* and *b* make no change whatsoever. In a sense, these two states represent labels.

Now from this CCN gate we can define a reversible *and* operator by considering only those cases where the initial state c_i equals zero. Given the entire CCN gate, we mark these four cases with a check mark:

	a_i	b_i	c_i	a_f	b_f	c_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

If we isolate these four cases where $c_i = 0$, we can see that we have the equivalent of an *and* gate:

and (a CCN gate with $c_i = 0$)

	a_i	b_i	c_i	a_f	b_f	c_f
✓	0	0	0	0	0	0
✓	0	1	0	0	1	0
✓	1	0	0	1	0	0
✓	1	1	0	1	1	1

The basic difference between a nonreversible *and* gate and a reversible CCN gate acting as an *and* gate is that in the reversible gate the input states *a* and *b* are carried over identically as output states. In other words, the initial information about the states *a* and *b* is kept intact in the reversible gate.

Reversibility essentially requires that we have to keep track of the input. Richard Feynman, one of the first who proposed applying quantum mechanics to computing, realized that reversibility meant that the input could be recovered along with the output at the end of the computation:

> But note that input data must typically be carried forward to the output to allow for reversibility. Feynman showed that in general the amount of extra information that must be carried forward is just the input itself.
>
> Richard Hughes, "Quantum Computation", in Anthony J. G. Hey (editor), *Feynman and Computation: Exploring the Limits of Computers,* page 196 (Perseus Books: Reading, Massachusetts, 1999).

This result is of great significance for analogical modeling and, in fact, for all exemplar-based systems – namely, reversibility means that we maintain the exemplars, either directly or in some recoverable form. If some form of quantum computing is used for language prediction, then all the exemplars used in a computation must be recoverable (at least up until observation). Quantum computation of any language-based system will therefore be an exemplar-based one, even if the system ends up acting as a neural net or as a set of rules.

Returning now to our example of the reversible CCN gate, we may ask what non-reversible gate do we get when we consider the other four cases – namely, the four unchecked cases. It turns out that when $c_i = 1$, we end up with the equivalent of the *nand* gate:

nand (a CCN gate with $c_i = 1$)

a_i	b_i	c_i	a_f	b_f	c_f
0	0	1	0	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

By re-arranging the order of the 0/1 assignments so that all the checked cases occur together, we can directly see the simultaneous *and* / *nand* distribution for the CCN gate:

	a_i	b_i	c_i	a_f	b_f	c_f	
✓	0	0	0	0	0	0	
✓	1	0	0	1	0	0	*and* (0001)
✓	0	1	0	0	1	0	
✓	1	1	0	1	1	1	
	0	0	1	0	0	1	
	1	0	1	1	0	1	*nand* (1110)
	0	1	1	0	1	1	
	1	1	1	1	1	0	

In this CCN gate, the *and* and *nand* operators can be said to form a *conjugate pair* of operators since 1110 (*nand*) is the complement of 0001 (*and*).

In general, the operation of an N gate on a qubit *a* can be represented as N(a) or more simply as Na. Similarly, the operation of a CCN gate on qubits *a, b,* and *c* can be represented as CCN (a,b,c) or again more simply as CCNabc.

3. *Reversibility of* non-implication / implication

Now we use the reversible negation operator with CCNabc in order to produce other logical configurations involving either a single 0 or a single 1. For instance, if we first negate the *b* qubit, then apply CCNabc, and finally re-negate the *b* qubit, we get the conjugate pair *non-implication / implication* ; that is, *non-implication* for initial $c = 0$ and *implication* for initial $c = 1$. In each case, the desired answer is found in the final state of the *c* qubit.

	Nb			→	CCNabc			→	Nb		
	a_1	b_1	c_1		a_2	b_2	c_2		a_3	b_3	c_3		a_4	b_4	c_4
✓	0	0	0		0	1	0		0	1	0		0	0	0
✓	1	0	0		1	1	0		1	1	1		1	0	1
✓	0	1	0		0	0	0		0	0	0		0	1	0
✓	1	1	0		1	0	0		1	0	0		1	1	0
	0	0	1		0	1	1		0	1	1		0	0	1
	1	0	1		1	1	1		1	1	0		1	0	0
	0	1	1		0	0	1		0	0	1		0	1	1
	1	1	1		1	0	1		1	0	1		1	1	1

non-implication ($c_1 = 0$)

	a_1	b_1	c_1		a_4	b_4	c_4
✓	0	0	0		0	0	0
✓	1	0	0		1	0	1
✓	0	1	0		0	1	0
✓	1	1	0		1	1	0

implication ($c_1 = 1$)

a_1	b_1	c_1	a_4	b_4	c_4
0	0	1	0	0	1
1	0	1	1	0	0
0	1	1	0	1	1
1	1	1	1	1	1

Note the effect of the reversible operators (Nb,CCNabc,Nb) on the 8 possible cases. Each application merely shifts the positioning of the 8 possible cases, but never merges them or eliminates any of the possibilities.

	Nb	→	CCNabc	→	Nb

a_1	b_1	c_1	a_2	b_2	c_2	a_3	b_3	c_3	a_4	b_4	c_4
0	0	0	0	1	0	0	1	0	0	0	0
1	0	0	1	1	0	1	1	1	1	0	1
0	1	0	0	0	0	0	0	0	0	1	0
1	1	0	1	0	0	1	0	0	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1
1	0	1	1	1	1	1	1	0	1	0	0
0	1	1	0	0	1	0	0	1	0	1	1
1	1	1	1	0	1	1	0	1	1	1	1

It should also be noted that no new possibility can be created since the 8 possibilities already cover every 0/1 possibility for each of the 3 given qubits, thus $2^3 = 8$. (More generally, given n qubits there are 2^n possibilities.)

For any of these basic reversible operators, we swap the value of only one qubit at a time, thus we can immediately re-apply that operator and return to the original state. Similarly, for any given sequence of such basic reversible operators, we can re-apply the operators in their reverse order and return to the original state as well. Thus in the case of the conjugate pair *implication* and *non-implication,* having applied (Nb,CCNabc,Nb), we can apply (Nb,CCNabc,Nb) and end up with what we started with:

			Nb			→		CCNabc			→		Nb		
	a_4	b_4	c_4		a_3	b_3	c_3		a_2	b_2	c_2		a_1	b_1	c_1
	0	0	0		0	1	0		0	1	0		0	0	0
	1	0	1		1	1	1		1	1	0		1	0	0
	0	1	0		0	0	0		0	0	0		0	1	0
	1	1	0		1	0	0		1	0	0		1	1	0
	0	0	1		0	1	1		0	1	1		0	0	1
	1	0	0		1	1	0		1	1	1		1	0	1
	0	1	1		0	0	1		0	0	1		0	1	1
	1	1	1		1	0	1		1	0	1		1	1	1

Thus reversibility is maintained through sequences of operators.

In this example, the negating of qubit *b* allows us to shift, so to speak, the distribution of the 0's and 1's. With no negation, we had *and / nand* (with its distributions of 0001/1110). By negating the qubit *b,* we now end up with *non-implication / implication* (and its distributions of 0100/1011).

4. *Reversibility of* non-consequence / consequence

Similarly, by negating qubit *a* rather than qubit *b,* we can get the conjugate pair *consequence* and *non-consequence* and its distribution 0010/1101:

				Na			→		CCNabc			→		Na		
		a_1	b_1	c_1		a_2	b_2	c_2		a_3	b_3	c_3		a_4	b_4	c_4
✓		0	0	0		1	0	0		1	0	0		0	0	0
✓		1	0	0		0	0	0		0	0	0		1	0	0
✓		0	1	0		1	1	0		1	1	1		0	1	1
✓		1	1	0		0	1	0		0	1	0		1	1	0
		0	0	1		1	0	1		1	0	1		0	0	1
		1	0	1		0	0	1		0	0	1		1	0	1
		0	1	1		1	1	1		1	1	0		0	1	0
		1	1	1		0	1	1		0	1	1		1	1	1

non-consequence ($c_1 = 0$)

	a_1	b_1	c_1	a_4	b_4	c_4
✓	0	0	0	0	0	0
✓	1	0	0	1	0	0
✓	0	1	0	0	1	1
✓	1	1	0	1	1	0

consequence ($c_1 = 1$)

a_1	b_1	c_1	a_4	b_4	c_4
0	0	1	0	0	1
1	0	1	1	0	1
0	1	1	0	1	0
1	1	1	1	1	1

5. *Reversibility of* nor / or

Finally, by negating both qubits *a* and *b*, we get the conjugate pair *nor / or* (and its distributions 1000/0111). Since the order of the negation makes no difference, we will show only the combined result of the negations:

				Na,Nb			→	CCNabc			→	Nb,Na		
	a_1	b_1	c_1	a_3	b_3	c_3		a_4	b_4	c_4		a_6	b_6	c_6
✓	0	0	0	1	1	0		1	1	1		0	0	1
✓	1	0	0	0	1	0		0	1	0		1	0	0
✓	0	1	0	1	0	0		1	0	0		0	1	0
✓	1	1	0	0	0	0		0	0	0		1	1	0
	0	0	1	1	1	1		1	1	0		0	0	0
	1	0	1	0	1	1		0	1	1		1	0	1
	0	1	1	1	0	1		1	0	1		0	1	1
	1	1	1	0	0	1		0	0	1		1	1	1

nor ($c_1 = 0$)

	a_1	b_1	c_1	a_6	b_6	c_6
✓	0	0	0	0	0	1
✓	1	0	0	1	0	0
✓	0	1	0	0	1	0
✓	1	1	0	1	1	0

or ($c_1 = 1$)

a_1	b_1	c_1	a_6	b_6	c_6
0	0	1	0	0	0
1	0	1	1	0	1
0	1	1	0	1	1
1	1	1	1	1	1

These results show that negating a qubit x is a kind of renaming, replacing x with not-x, while the single CCN operator is responsible for flip-flopping a single 0 with a 1. The negation operator acts on a single qubit and thus leaves other qubits unaffected. But the CCN operator conditionally negates a qubit (depending on the states of two other qubits), thus setting up dependencies (or entanglements) between the qubits.

6. *Reversibility of* copy a / copy not-a

Applying the CCN twice can be used to further increase the dependencies (or entanglements) between the qubits, but only when at least one of the qubits has been negated. If no qubit is negated, then we just end up with the same states as in the beginning, without any entanglement; that is, (CCNabc,CCNabc) is equivalent to the identity operator since CCNabc is its own inverse. But let us suppose that we first apply CCNabc, then negate the *b* qubit, apply CCNabc once more, and then re-negate the *b* qubit. This sequence of operators (CCNabc,Nb,CCNabc,Nb) will produce the equivalent of *copy a* or *copy not-a,* depending on whether the *c* qubit is initially set to 0 to 1:

	CCNabc	→	Nb	→	CCNabc	→	Nb

	a_1	b_1	c_1	a_2	b_2	c_2	a_3	b_3	c_3	a_4	b_4	c_4	a_5	b_5	c_5
✓	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
✓	1	0	0	1	0	0	1	1	0	1	1	1	1	0	1
✓	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0
✓	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1
	0	0	1	0	0	1	0	1	1	0	1	1	0	0	1
	1	0	1	1	0	1	1	1	1	1	1	0	1	0	0
	0	1	1	0	1	1	0	0	1	0	0	1	0	1	1
	1	1	1	1	1	0	1	0	0	1	0	0	1	1	0

copy a ($c_1 = 0$)

	a_1	b_1	c_1	a_5	b_5	c_5
✓	0	0	0	0	0	0
✓	1	0	0	1	0	1
✓	0	1	0	0	1	0
✓	1	1	0	1	1	1

copy not-a ($c_1 = 1$)

a_1	b_1	c_1	a_5	b_5	c_5
0	0	1	0	0	1
1	0	1	1	0	0
0	1	1	0	1	1
1	1	1	1	1	0

For this conjugate pair, we get the distribution 0101/1010, identical to the distribution for the *a* qubit when $c_1 = 0$ and its negation when $c_1 = 1$. Of course, this negation is not the same as *not a* (that is, Na), which would just reverse the states of the *a* qubit and not affect the *c* qubit at all. Rather, this operation makes the *c* qubit equal to the reverse of the *a* qubit when $c_1 = 1$ (but identical to the *a* qubit when $c_1 = 0$).

The copying aspect also needs some explanation. A qubit can be copied in analogical modeling because its state is either 0 and 1. More generally, in quantum mechanics, a qubit has the general state of $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. Under such general conditions, the state of the qubit cannot be copied identically unless either $\alpha = 1$ or $\beta = 1$, which means that the qubit can only take on the orthogonal states 0 and 1. For discussion and proof of this result, referred to as the no-cloning theorem, see Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information,* pages 24-25, 532 (Cambridge University Press: Cambridge, England, 2000).

7. *Reversibility of* copy b / copy not-b

Now suppose we wish to copy the *b* qubit rather than the *a* qubit. To do this, we use Na rather than Nb in our series of operators – that is, we start with the same initial operator CCNabc, but now we follow it by applying (Na,CCNabc,Na):

	CCNabc →			Na →			CCNabc →			Na		
	a_1	b_1	c_1	a_2	b_2	c_2	a_3	b_3	c_3	a_4	b_4	c_4	a_5	b_5	c_5
✓	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
✓	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0
✓	0	1	0	0	1	0	1	1	0	1	1	1	0	1	1
✓	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1
	0	0	1	0	0	1	1	0	1	1	0	1	0	0	1
	1	0	1	1	0	1	0	0	1	0	0	1	1	0	1
	0	1	1	0	1	1	1	1	1	1	1	0	0	1	0
	1	1	1	1	1	0	0	1	0	0	1	0	1	1	0

copy b ($c_1 = 0$)

	a_1	b_1	c_1	a_5	b_5	c_5
✓	0	0	0	0	0	0
✓	1	0	0	1	0	0
✓	0	1	0	0	1	1
✓	1	1	0	1	1	1

copy not-b ($c_1 = 1$)

a_1	b_1	c_1	a_5	b_5	c_5
0	0	1	0	0	1
1	0	1	1	0	1
0	1	1	0	1	0
1	1	1	1	1	0

For this conjugate pair we get the distribution 0011/1100, the same one as the *b* qubit. Replacing Nb with Na in order to copy qubit *b* rather than qubit *a* reminds us that the use of Na and Nb is equivalent to renaming the qubits. If Nb is necessary in copying *a* and its negation, then Na will be necessary in copying *b* and its negation.

We should also note that in copying qubit *a* (or qubit *b*) the application of the two Nb's (or equivalently the two Na's) must be properly sequenced if we wish to obtain the desired results. Because of reversibility, we do not allow CCNabc to be directly followed by another CCNabc since this ends up doing nothing. Similarly, Nb cannot be directly followed by another Nb (nor can Na by another Na). Keeping these restrictions in mind, we end up with the following possibilities:

copy *a* and *not-a*

CCNabc → Nb → CCNabc → Nb
Nb → CCNabc → Nb → CCNabc

copy *b* and *not-b*

CCNabc → Na → CCNabc → Na
Na → CCNabc → Na → CCNabc

8. *Reversibility of* exclusive or / equivalence

Thus far we have negated only one of the qubits. When we negate both, qubit *c* ends up telling us whether qubits *a* and *b* are different or the same; that is, we get the conjugate pair *exclusive or* (abbreviated as XOR) and *equivalence:*

Nb → CCNabc → Nb,Na → CCNabc → Na

	a_1	b_1	c_1	a_2	b_2	c_2	a_3	b_3	c_3	a_5	b_5	c_5	a_6	b_6	c_6	a_7	b_7	c_7
✓	0	0	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0
✓	1	0	0	1	1	0	1	1	1	0	0	1	0	0	1	1	0	1
✓	0	1	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	1
✓	1	1	0	1	0	0	1	0	0	0	1	0	0	1	0	1	1	0
	0	0	1	0	1	1	0	1	1	1	0	1	1	0	1	0	0	1
	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0
	0	1	1	0	0	1	0	0	1	1	1	1	1	1	0	0	1	0
	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	1

exclusive or ($c_1 = 0$)

	a_1	b_1	c_1	a_7	b_7	c_7
✓	0	0	0	0	0	0
✓	1	0	0	1	0	1
✓	0	1	0	0	1	1
✓	1	1	0	1	1	0

equivalence ($c_1 = 1$)

a_1	b_1	c_1	a_7	b_7	c_7
0	0	1	0	0	1
1	0	1	1	0	0
0	1	1	0	1	0
1	1	1	1	1	1

The resulting distribution of 0110/1001 means that in the end the *c* qubit will tell us whether or not qubits *a* and *b* have the same state or not. If the *c* qubit is initially set at 0, then in its final state it will give us XOR(*a*,*b*), thus telling us if the states of qubits *a* and *b* differ. Or if $c_1 = 1$, then c_7 will tell us if *a*'s state is logically equivalent to *b*'s.

As we might suspect, there are a number of different possible sequences of applying the Na's and Nb's that will result in obtaining the conjugate pair *exclusive-or* / *equivalence*. We have two major cases: (1) there is a single negative

operator (either Na or Nb) between the two occurrences of CCNabc; or (2) both Na and Nb occur between the two occurrences of CCNabc. If there were no negations, then the two CCNabc's would cancel each other out. Similarly, if Na occurs twice without any intervening CCNabc, then those two Na's would also cancel themselves out. The same holds for Nb. Finally, we should note that whenever Na and Nb occur in a sequence, there is no need to specify the order. Negations of single qubits act independently of each other. Thus in the following list of the possibilities, a specification like "Na,Nb" means that we could have either Na → Nb or Nb → Na:

(1) one negation between the two occurrences of CCNabc:

Nb	→	CCNabc	→	Na	→	CCNabc	→	Na,Nb
Na	→	CCNabc	→	Nb	→	CCNabc	→	Na,Nb
Na,Nb	→	CCNabc	→	Na	→	CCNabc	→	Nb
Na,Nb	→	CCNabc	→	Nb	→	CCNabc	→	Na

(2) two negations between the two occurrences of CCNabc:

		CCNabc	→	Na,Nb	→	CCNabc	→	Na,Nb
Nb	→	CCNabc	→	Na,Nb	→	CCNabc	→	Na
Na	→	CCNabc	→	Na,Nb	→	CCNabc	→	Nb
Na,Nb	→	CCNabc	→	Na,Nb	→	CCNabc		

Some of these sequences can be simplified in terms of other sequences. For instance, the last one is equivalent to *nor / or* immediately followed by CCNabc. We recall that *nor / or* was defined as Na,Nb → CCNabc → Na,Nb. Having applied that sequence, we only need apply CCNabc to effect *exclusive-or / equivalence:*

nor / or → CCNabc

a_1	b_1	c_1	a_6	b_6	c_6	a_7	b_7	c_7
0	0	0	0	0	1	0	0	1
1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0
1	1	0	1	1	0	1	1	1
0	0	1	0	0	0	0	0	0
1	0	1	1	0	1	1	0	1
0	1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1	0

As shown by the arrows, the *nor / or* operator uses the two negations Na and Nb to force CCNabc to switch the states for the *c* qubit when both *a* and *b* are 0. Then *exclusive-or / equivalence* is derived by switching the states for the *c* qubit for the case when both *a* and *b* are 1. This, of course, is precisely the difference between *inclusive-or* and *exclusive-or*. The latter does not allow *a* and *b* to both equal 1.

9. *Reversibility of* contradiction / tautology

We now have one final case to list – namely, the logical result when no CCNabc operation (or negative operation either) is applied. Under these conditions, the final state of the *c* qubit is the initial state, so when we set either $c_1 = 0$ or $c_1 = 1$, we get the original distribution 0000/1111. This conjugate pair can be logically referred to as *contradiction / tautology* (or FALSE / TRUE):

	a_1	b_1	c_1
✓	0	0	0
✓	1	0	0
✓	0	1	0
✓	1	1	0
	0	0	1
	1	0	1
	0	1	1
	1	1	1

contradiction ($c_1 = 0$)

	a_1	b_1	c_1
✓	0	0	0
✓	1	0	0
✓	0	1	0
✓	1	1	0

tautology ($c_1 = 1$)

a_1	b_1	c_1
0	0	1
1	0	1
0	1	1
1	1	1

10. *A summary*

Finally, we can summarize all 8 possible conjugate pairs of binary boolean operations in terms of 8 reversible sequences of CCN and N gates:

CCNs	Ns	name	distribution
---	---	contradiction / tautology (FALSE / TRUE)	0000 / 1111
one	---	and / nand	0001 / 1110
	Na	non-consequence / consequence	0010 / 1101
	Nb	non-implication / implication	0100 / 1011
	Na,Nb	nor / or	1000 / 0111
two	Na	copy *b* / copy not-*b*	0011 / 1100
	Nb	copy *a* / copy not-*a*	0101 / 1010
	Na,Nb	exclusive-or / equivalence	0110 / 1001

We will frequently have recourse to the using the following boolean symbols in representing these conjugate pairs:

name	symbolic	distribution
contradiction / tautology (FALSE / TRUE)	F / T	0000 / 1111
and / nand	a·b / a↑b	0001 / 1110
non-consequence / consequence	a↚b / a←b	0010 / 1101
non-implication / implication	a↛b / a→b	0100 / 1011
nor / or	a↓b / a+b	1000 / 0111
copy *b* / copy not-*b*	b / b′	0011 / 1100
copy *a* / copy not-*a*	a / a′	0101 / 1010
exclusive-or / equivalence	a⊕b / a≡b	0110 / 1001

For additional discussion of the relationship between the 16 boolean operators, see A. K. Dewdney, *The (New) Turing Omnibus: 66 Excursions in Computer Science,* 14-21 (W. H. Freeman: New York, New York, 1993); Robert R. Korfhage, *Discrete Computational Structures*, 2nd edition, 292-295, 304-308 (Academic Press: Orlando, Florida, 1984); David Gries and Fred B. Schneider, *A Logical Approach to Discrete Math,* 26-27 (Springer-Verlag: New York, 1993); Lennart Råde and Bertil Westergren, *Mathematics Handbook for Science and Engineering*, 4th edition, 32 (Springer-Verlag: Berlin, 1999).

11. *Simplification using* control-not

Besides the reversible operators of *not* (N) and *control-control-not* (CCN), another helpful one is *control-not* (abbreviated as CN). In this instance, only one qubit is controlled by another. For instance, using our array of possibilities as before, we see that CN(a,c) – or CNac – depends only on qubit *a* to switch the 0/1 of qubit *c*. Any other qubit, such as qubit *b,* can be ignored:

CNac

a_i	c_i	a_f	c_f
0	0	0	0
1	0	1	1
0	1	0	1
1	1	1	0

We can see from the resulting c_f that it is equivalent to the *exclusive or* of a_i and c_i:

$$c_f = \text{XOR}\ (a_i\ ,\ c_i)$$

If we set $c_i = 0$, we see that $c_f = a_i$, while setting $c_i = 1$ we get $c_f = \textit{not}\ a_i$. In other words, we can use CNac to do the equivalent of *copy a / copy not-a:*

CNac

	a_1	c_1	a_2	c_2
✓	0	0	0	0
✓	1	0	1	1
	0	1	0	1
	1	1	1	0

copy a ($c_1 = 0$)

	a_1	c_1	a_2	c_2
✓	0	0	0	0
✓	1	0	1	1

copy not-a ($c_1 = 1$)

a_1	c_1	a_2	c_2
0	1	0	1
1	1	1	0

Similarly, CNbc will give us the equivalent of *copy b / copy not-b.*

One important difference between CN and CCN in doing copying is that with CN there is no need to first negate some other qubit. To get *copy a / copy not-a,* CCN first required us to negate an auxiliary *b* qubit. With CN the copying is direct.

Now consider what happens when we apply both CNac and CNbc. In this situation, qubit *c* will end up storing the result of *exclusive-or / equivalence* for qubits *a* and *b:*

			CNac →			CNbc			
a_1	b_1	c_1		a_2	b_2	c_2	a_3	b_3	c_3	
0	0	0		0	0	0	0	0	0	
1	0	0		1	0	1	1	0	1	$c_3 = a \oplus b$ $(c_1 = 0)$
0	1	0		0	1	0	0	1	1	
1	1	0		1	1	1	1	1	0	
0	0	1		0	0	1	0	0	1	
1	0	1		1	0	0	1	0	0	$c_3 = a \equiv b$ $(c_1 = 1)$
0	1	1		0	1	1	0	1	0	
1	1	1		1	1	0	1	1	1	

All of these examples using CN produce the three conjugate pairs of boolean operators that result from two applications of CCN: (1) *copy a / copy not-a,* (2) *copy b / copy not-b,* and (3) *exclusive-or / equivalence*. CN definitely makes the task easier for producing these three cases. With CCN we must apply the negative operator (Na to copy qubit *b,* Nb to copy qubit *a,* and both Na and Nb to get *exclusive-or / equivalence*).

We further note that one application of CCN switches only a single 0/1 pair. For instance, in producing the conjugate pair *and / nand,* the initial distribution for qubit *c* goes from 0000/1111 to 0001/1110. On the other hand, one application of CN always switches two 0/1 pairs. Thus in producing *copy b / copy not-b,* CN takes the initial distribution for qubit *c* from 0000/1111 directly to 0011/1100, a switching of two 0/1 pairs. Since CN always involves such a double switching, it can never be used to produce a case involving only a single switching. This means that CN (even when combined with N) cannot do the four conjugate pairs that involve only a single CCN: (1) *and / nand,* (2) *non-consequence / consequence,* (3) *non-implication / implication,* and (4) *nor / or*. From this fact we deduce that CCN and N form a complete base of reversible boolean operators, but CN and N do not – that is, all reversible boolean operators on three qubits can be written as a product of CCN and N.

We can go further and show that CCN can be used in place of CN and N. Suppose we wish to replace CNac with a CCN. We do this by using an auxiliary qubit z set to 1 and do CCNzac:

CCNzac

z_1	a_1	c_1	z_2	a_2	c_2
0	0	0	0	0	0
0	1	0	0	1	0
0	0	1	0	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	0	1	1	1
1	0	1	1	0	1
1	1	1	1	1	0

CNac = CCNzac when $z = 1$

In the case of N, we use two auxiliary qubits x and y and set both to 1. In the following, we see how to negate qubit c by means of CCNxyc:

CCNxyc

x_1	y_1	c_1	x_2	y_2	c_2
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

Nc = CCNxyc when $x = 1, y = 1$

The importance of this result is that the CCN reversible operator alone is a universal operator – that is, CCN can be used by itself to form a complete base,

which is not the case with CN and N. Thus any of the 16 standard boolean operators can be derived by a sequence of CCNs. We only need replace each CN and N operator with its equivalent CCN operator, providing of course we set the appropriate 0/1 value for some auxiliary qubits.

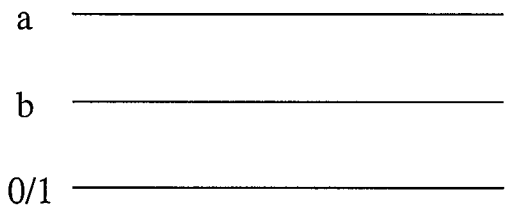
For simplicity's sake, we will use all three reversible operators to describe the algorithm for analogical modeling. CCN and N will be used for the four cases involving a single switching of a 0/1 pair. CN will be used for the cases involving a double switching, *copy / copy not* and *exclusive-or / equivalence.* Our original summary of all 8 possible conjugate pairs of binary boolean operators is now revised to use the CN operator in those cases where it simplifies the application sequence of reversible operators:

Ns	CCNs	Ns	CNs	symbolic	distribution
---	---	---	---	F / T	0000 / 1111
---	CCNabc	---	---	a·b / a↑b	0001 / 1110
Na	CCNabc	Na	---	a↚b / a←b	0010 / 1101
Nb	CCNabc	Nb	---	a↛b / a→b	0100 / 1011
Na,Nb	CCNabc	Na,Nb	---	a↓b / a+b	1000 / 0111
---	---	---	CNbc	b / b′	0011 / 1100
---	---	---	CNac	a / a′	0101 / 1010
---	---	—	CNac,CNbc	a⊕b / a≡b	0110 / 1001

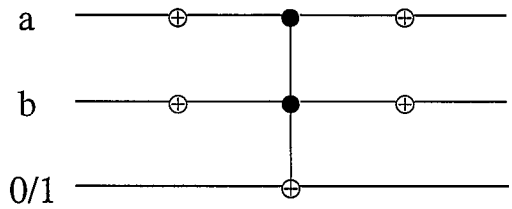
12. *Circuit representations of reversible operators*

Very often in quantum computing a circuit is used to show the application of the reversible operators. We construct these circuits in terms of N, CN, and CCN. A solid dot ● stands for a controlling qubit, and the symbol ⊕ stands for the negation of a qubit. In order to get the equivalence of the boolean operators, we must set the *c* qubit to either 0 or 1. In the following, we list each of the 8 conjugate pairs, with the *a* and *b* qubits serving as the controlling qubits. The *c* qubit gives the result. For the origin of these symbols, see pages 49-52 in Jozef Gruska, *Quantum Computing* (McGraw-Hill: London, 1999).

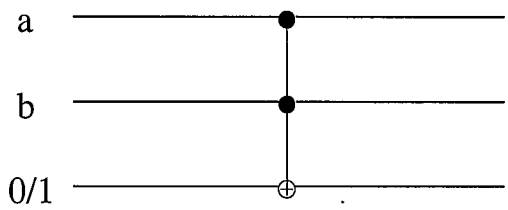
F/T



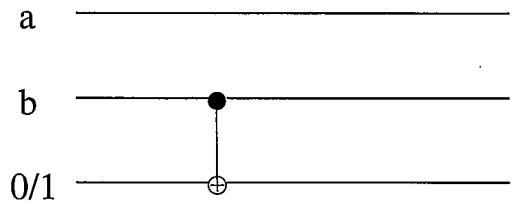
a↓b / a+b



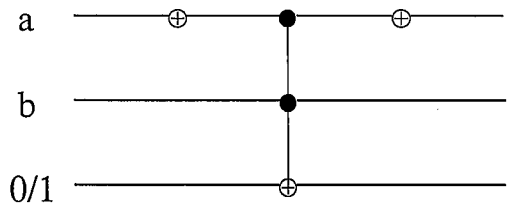
a·b / a↑b



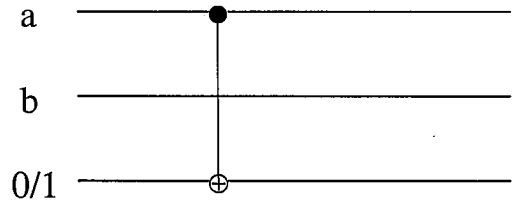
b / b′



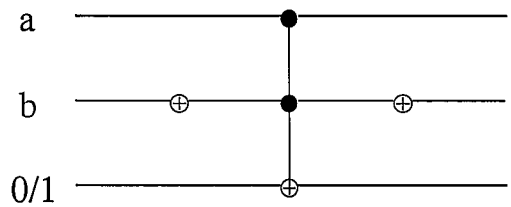
a↚b / a←b



a / a′



a↛b / a→b



a⊕b / a≡b