Towards a More Efficient Representation of Functions in Quantum and Reversible Computing

Oscar Galindo[a], Laxman Bokati[b], and
Vladik Kreinovich[a,b]
[a]Department of Computer Science
[b]Computational Science Program
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
ogalindomo@miners.utep.edu, lbokati@miners.utep.edu,
vladik@utep.edu

1. Need for Faster Computing

- While computers are very fast, in many practical problems, we need even faster computations.
- For example, we can, in principle, with high accuracy predict in which direction a deadly tornado will turn in the next 15 minutes.
- However, this computation requires hours even on the most efficient high performance computers.
- This is too late for the resulting prediction to be of any use.
- Faster computations means faster processing within each cell and faster communication between cells.
- At present, communication between cells is fast, but, potentially, it can be further increased.
- However, there is a limit to this increase: according to modern physics, all processes cannot move faster than the speed of light.

2. Desirability of Quantum Computing

- For a chip of size ≈ 3 cm, it takes at least 0.1 nanosecond (10^{-10} sec) for a signal to move from one side of the chip to the other.
- This is close to the time ≈ 0.25 nanoseconds during this time a usual 4 GHz laptop performs an elementary operation.
- Thus, to further speed up computations, it is desirable to further decrease the size of the chip.
- Thus, we need to further decrease the size of its memory units and processing units.
- Already the size of a memory cell in a computer is compatible with the size of a molecule.
- If we decrease the computer cells even more, they will consist of a few dozen molecules.
- Thus, to describe the behavior of such cells, we will need to take into account the physical laws that describe such micro-objects.
- These are the laws of quantum physics.

3. Quantum Computing Means Reversible Computing

- For macro-objects, we can observe irreversible processes.
- If we drop a china cup on a hard floor, it will break into pieces.
- No physical process can combine these pieces back into the original whole cup.
- However, on the micro-level, all the equations are reversible.
- Reversibility holds for Newton's equations that describe the non-quantum motion of particles and bodies.
- Reversibility holds for Schroedinger's equation that takes into account quantum effects that describes this notion.
- Thus, in quantum computing, all elementary operations must be reversible.

4. Reversible Computing Beyond Quantum

- Reversible computing is also needed for a different reason.
- Even at the present level of micro-miniaturization, theoretically, we could place more memory cells and processing cells into the same small volume.
- E.g., if, instead of the current 2-D stacking of these cells into a planar chip, we could stack them in 3-D.
- For example, if we have a terabyte of memory, i.e., 10^{12} cells in a 2-D arrangement, this means $10^6 \times 10^6$.
- If we could get a third dimension, we would be able to place $10^6 \times 10^6 \times 10^6 = 10^{18}$ cells in the same volume – million times more than now.
- The reason why we cannot do it is that already modern computers emit a large amount of heat.
- Even with an intensive inside-computer cooling, a working laptop warms up so much that it is possible to be burned if you keep it in your lap.
- If we have several 2-D layers forming a 3-D structure, the amount of heat will increase so much that the computer will simply melt.

5. Reversible Computing Beyond Quantum (cont-d)

- What causes this heat? One of the reasons may be design imperfections.
- Some part of this heat may be decreased by an appropriate engineering design.
- However, there is also a fundamental reason for this heat: Second Law of Thermodynamics.
- According to this law, every time we have an irreversible process, heat is radiated, in the amount $T \cdot S$, where S is the entropy.
- In our case, S is the number of bits in information loss.
- Basic logic operations (that underlie all computations) are irreversible.
- For example, when $a \,\&\, b$ is false, it could be that both a and b were false, it could be that one of them was false.
- Thus, the usual "and"-operation $(a, b) \to a \,\&\, b$ is not reversible.
- So, to decrease the amount of heat, a natural idea is to use only reversible operations.

6. How Operations Are Made Reversible Now?

- At present, in quantum (and reversible) computing:
  - a bit-valued function $x_1, \ldots, x_n \to f(x_1, \ldots, x_n)$ of n bit-valued variables x_i
  - is transformed into the following reversible operation:

$$T_f : (x_1, \ldots, x_n, x_0) \to (x_1, \ldots, x_n, x_0 \oplus f(x_1, \ldots, x_n)).$$

- Here x_0 is an auxiliary bit-valued variable, and $\oplus$ denotes "exclusive or", i.e., addition modulo 2.
- It is easy to see that the above operation is indeed reversible.
- Indeed, if we apply it twice, we get the same input back:

$$T_f(T_f(x_1, \ldots, x_n, x_0)) = T_f(x_1, \ldots, x_n, x_0 \oplus f(x_1, \ldots, x_n)) =$$
$$(x_1, \ldots, x_n, x_0 \oplus f(x_1, \ldots, x_n) \oplus f(x_1, \ldots, x_n)).$$

- For addition modulo 2, $a \oplus a = 0$ for all a, so indeed

$$x_0 \oplus f(x_1, \ldots, x_n) \oplus f(x_1, \ldots, x_n) = x_0 \oplus (f(x_1, \ldots, x_n) \oplus f(x_1, \ldots, x_n)) =$$
$$T_f(T_f(x_1, \ldots, x_n, x_0)) = (x_1, \ldots, x_n, x_0).$$

7. Limitations of the Current Representation of Functions

- We rarely write algorithms "from scratch", we usually use existing algorithms as building blocks.
- For example, when we write a program for performing operations involving sines and cosines (e.g., a program for Fourier Transform):
  - we do not write a new code for sines and cosines from scratch,
  - we use standard algorithms for computing these trigonometric functions – algorithms contained in the corresponding compiler.
- From this viewpoint:
  - if we want to make a complex algorithm – that consists of several moduli – reversible,
  - it is desirable to be able to transform the reversible versions of these moduli into a reversible version of the whole algorithm.
- It is desirable to generate a reversible version of each function so that composition would be transformed into composition.

8. Limitations of the Current Representation (cont-d)

- Unfortunately, this is not the case with the existing scheme described above.
- Indeed, even in the simple case when we consider the composition $x_1 \to f(f(x_1))$ of the same function f of one variable:
  - by applying the above transformation twice, we get – as we have shown – the same input x_1 back,
  - and *not* the desired value $f(f(x_1))$.
- Thus, if we use the currently used methodology to design a reversible version of a modularized algorithm, we cannot use the modular structure.
- We have, in effect, to rewrite the algorithm from scratch.
- We propose a solution to this problem.

9. Case of Fixed-Point Real Numbers

- Let us start with the simplest case of numerical algorithms $y = f(x)$, with a single real-valued input x and a single real-valued output y.
- Of course, in a computer, we do not process actual real numbers (which form an infinite set).
- We process computer-representable real numbers –
- In general, the transformation $y = f(x)$ is not reversible.
- So, to make it reversible, we need to consider an auxiliary input variable u – and, correspondingly, an auxiliary output variable v.
- The resulting transformation $(x, u) \to (f(x), v_f(x, u))$ should be reversible.
- In the actual computations, we will use a specific value u_0 of the auxiliary variable u.

10. How to Make Sure That Composition Is Transformed into Composition

- If we first apply the reversible analogue of the function f, then each original state (x, u_0) is transformed into $(y, v) \stackrel{\text{def}}{=} (f(x), v_f(x, u_0))$.
- To this new state, we would like to apply the reversible analogue of the second function g.
- We have agreed that the reversible analogue of applying a function should start with a state of the type (y, u_0).
- Thus, it is reasonable to require that $v_f(x, u_0) = u_0$ for all x.
- Then the state emerging after applying the reversible analogue of f will be $(y, u_0) = (f(x), u_0)$.
- If we apply the reversible analogue of g, we get $(g(y), u_0) = (g(f(x)), u_0)$.
- This is exactly what we would have got if we applied the reversible version of composition $x \to g(f(x))$.
- One can easily see that without losing generality, we can assume, e.g., that $u_0 = 0$.

11. What Does "Reversible" Mean Here?

- As we have mentioned earlier, in the computer, real numbers are represented with some accuracy.
- Because of this, there are finitely many possible computer representations of real numbers.
- In this section, we consider the case of fixed-point real numbers, when all the computer representations have the same accuracy ε.
- Reversibility means that inputs and outputs are in 1-1 correspondence.
- Thus, for each 2-D region r, its image after the transformation $(x, u) \to (y, v)$ should contain exactly as many pairs as the original region r.
- Each pair (x, u) of computer-representable real numbers takes the area of ε^2 in the (x, u)-plane.
- In each region of this plane, the number of possible computer-representable numbers is therefore proportional of the area of this region.
- Thus, reversibility implies that the transformation $(x, u) \to (f(x), v_f(x, u))$ should preserve the area.

12. What Does "Reversible" Mean Here (cont-d)

- Vice versa, let us assume that the transformation is area-preserving.
- Let us show that this implies reversibility.
- Indeed, let $\delta > 0$ be the accuracy beyond which the different between inputs and/or outputs makes no physical difference.
- So, all the inputs within an area of size $\delta \times \delta$, are, in effect, practically equivalent.
- Since the area is preserved, the set of all corresponding outputs has the exact same area.
- Since these outputs correspond to practically equivalent inputs, they are also practically equivalent.
- Each of the two regions is formed by small $\varepsilon \times \varepsilon$ squares that correspond to machine accuracy.
- Since the original area and its image have the same area, this means that they consist of the same number of such small squares.

13. What Does "Reversible" Mean Here (cont-d)

- So, we can put these squares in 1-1 correspondence with each other – and thus, make the transformation reversible.
- The same argument can be applied to transformations $\mathbb{R}^n \to \mathbb{R}^n$ for any n.
- So, in this general case too, reversibility is equivalent to preserving the area.
- Let us consider a general transformation

$$(x_1, \ldots, x_n) \to (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n)).$$

- It is known that under this transformation, the n-dimensional volume is multiplied by the determinant of the matrix with elements $\dfrac{\partial f_i}{\partial x_j}(x_1, \ldots, x_n)$.
- Thus, reversibility means that this determinant should be equal to 1.

14. Let Us Go Back to Our Simple Case

- For $(x, u) \to (f(x), v_f(x, u))$, the derivatives matrix is:

$$\begin{pmatrix} f'(x) & 0 \\ \dfrac{\partial v_f}{\partial x}(x, u) & \dfrac{\partial v_f}{\partial u}(x, u) \end{pmatrix}.$$

- Thus, equating the determinant of this matrix to 1 leads to

$$f'(x) \cdot \frac{\partial v_f}{\partial u}(x, u) = 1, \text{ hence } \frac{\partial v_f}{\partial u}(x, u) = \frac{1}{f'(x)} \text{ and}$$

$$v_f(x, U) = v_f(x, 0) + \int_0^U \frac{\partial v_f}{\partial u}(x, u)\, du = v_f(x, 0) + \int_0^U \frac{1}{f'(x)}\, du =$$

$$v_f(x, 0) + \frac{U}{f'(x)}.$$

- We know that $v_f(x, 0) = 0$, thus $v_f(x, u)(x, u) = \dfrac{u}{f'(x)}$, and the transformation takes the form $(x, u) \to \left(f(x), \dfrac{u}{f'(x)}\right)$.

15. Examples and Comment

- For $f(x) = \exp(x)$, we have $f'(x) = \exp(x)$ and thus, the reversible analogue is $(x, u) \to (\exp(x), u \cdot \exp(-x))$.
- For $f(x) = \ln(x)$, we have $f'(x) = 1/x$ and thus, the reversible analogue is $(x, u) \to (x, u \cdot x)$.
- *Comment:*
  - Our formulas cannot be applied when $f'(x) = 0$.
  - However, we consider all the numbers modulo the "machine zero" ε – the smallest positive number representable in a computer.
  - So, we can replace $f'(x)$ with the machine zero.

16. General Case

- In the general case, we add an auxiliary variable u:

$$(x_1, \ldots, x_n, u) \to (f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n), v_f(x_1, \ldots, x_n, u)).$$

- To preserve composition, we take $v_f(x_1, \ldots, x_n, 0) = 0$.
- Thus, from the requirement that the volume is preserved, we get

$$v_f(x_1, \ldots, x_n, u) = \frac{u}{\det \left\| \dfrac{\partial f_i}{\partial x_j}(x_1, \ldots, x_n) \right\|}.$$

- So, we should consider the following mapping:

$$(x_1, \ldots, x_n, u) \to$$

$$\left(f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n), \frac{u}{\det \left\| \dfrac{\partial f_i}{\partial x_j} \right\|} \right).$$

17. Case of Floating-Point Numbers

- For floating-point numbers, we, crudely speaking, represent the logarithms.
- For example, in the decimal case, 1 000 000 000 is represented as 10^9, where 9 is the decimal logarithm of the original number.
- In this case, we represent all these logarithms with the same accuracy ε.
- So, the volume should be preserved for the transformation of logarithms $\ln(x_i)$ into logarithms $\ln(f_j)$, for which $\dfrac{\partial \ln(f_i)}{\partial \ln(x_j)} = \dfrac{x_j}{f_i} \cdot \dfrac{\partial f_i}{\partial x_j}$, so we get:

$$(x_1, \ldots, x_n, u) \to$$

$$\left(f_1(x_1, \ldots, x_n), \ldots, f_n(x_1, \ldots, x_n), \frac{u}{\det \left\| \dfrac{x_j}{f_i} \cdot \dfrac{\partial f_i}{\partial x_j} \right\|} \right).$$

- In some cases (e.g., for $\exp(x)$), the input is fixed-point, the output is floating-point number; then, we consider the dependence of $\ln(f)$ of x.

18. Case of Functions of Two Variables

- In this case, it makes senseto add an extra output:

$$(x_1, x_2) \to (f(x_1, x_2), g(x_1, x_2)), \text{ for some } g.$$

- The condition that the volume is preserved is $\dfrac{\partial f}{\partial x_1} \cdot \dfrac{\partial g}{\partial x_2} - \dfrac{\partial f}{\partial x_2} \cdot \dfrac{\partial g}{\partial x_1} = 1$.
- E.g., for $f(x_1, x_2) = x_1 + x_2$, we get $\dfrac{\partial g}{\partial x_2} - \dfrac{\partial g}{\partial x_1} = 1$.
- This expression can be simplified if we use new variables $u_1 = x_1 - x_2$ and $u_2 = x_1 + x_2$ for which $x_1 = \dfrac{u_1 + u_2}{2}$ and $x_2 = \dfrac{u_2 - u_1}{2}$.
- In terms of the new variables, the original function g takes the form

$$g(x_1, x_2) = G(u_1, u_2) \stackrel{\text{def}}{=} g\left(\frac{u_1 + u_2}{2}, \frac{u_2 - u_1}{2}\right).$$

- For G, we have $\dfrac{\partial G}{\partial u_1} = \dfrac{1}{2} \cdot \dfrac{\partial g}{\partial x_1} - \dfrac{1}{2} \cdot \dfrac{\partial g}{\partial x_2} = -\dfrac{1}{2}$, thus, $G(u_1, u_2) = -\dfrac{1}{2} \cdot u_1 + C(u_2)$ for some function C.

19. Functions of Two Variables (cont-d)

- Substituting the expressions for u_i in terms of x_1 and x_2, we get

$$g(x_1, x_2) = \frac{x_2 - x_1}{2} + C(x_1 + x_2).$$

- So, to make addition reversible, we may want to have subtraction – the operation inverse to addition; this makes intuitive sense.
- Similarly, for $f(x_1, x_2) = x_1 \cdot x_2$, we get $x_2 \cdot \dfrac{\partial g}{\partial x_2} - x_1 \cdot \dfrac{\partial g}{\partial x_1} = 1$.
- For $X_i = \ln(x_i)$, we get $\dfrac{\partial g}{\partial X_2} - \dfrac{\partial g}{\partial X_1} = 1$, so $g(X_1, X_2) = \dfrac{X_2 - X_1}{2} + C(X_1 + X_2)$
- Here, $X_2 = X_1 = \ln(x_2) - \ln(x_1) = \ln\left(\dfrac{x_2}{x_1}\right)$, so

$$f(x_1, x_2) = \frac{1}{2} \cdot \ln\left(\frac{x_2}{x_1}\right) + C(x_1 \cdot x_2).$$

- So, to make multiplication reversible, we need to add a (function of) division – the operation inverse to multiplication; this makes sense.

20. Acknowledgments

This work was partially supported by the US National Science Foundation via grant HRD-1242122 (Cyber-ShARE Center of Excellence).