



Quantum Genetic Programming

(What about quantum software engineering?)

John A Clark

Senior Lecturer in Critical Systems

Dept. of Computer Science

University of York, UK

jac@cs.york.ac.uk

www.cs.york.ac.uk/~jac

research interests: software testing, cryptography/security,
quantum information processing

(Acknowledgements: Ideas with Prof Susan Stepney)



Outline of the talk

- Why bother?
- Outline of some work using evolutionary techniques to generate quantum algorithms.
- Speculate about the way forward.



Quantum Information Processing

- Quantum Information Processing is one of the most important topics to emerge in recent years. Why?
- Because it looks extremely useful practically as well as theoretically
 - Quantum cryptography 'works' (OK - there are some key distribution protocols around with successful experimental prototypes), it may even get highly commercial soon.
 - New computations seem possible
 - Shor's prime factorization (Quantum Discrete Fourier Transform). We do not actually know for sure what the limits of quantum computation are.
 - Grover's search on unstructured databases has complexity $O(\sqrt{N})$

where N is the number of states.



But.....

- There is no tidal wave of innovation in algorithms...

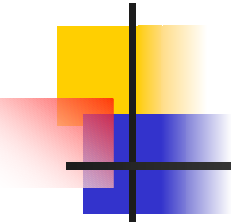
Of course computer scientists would like to develop a repertoire of quantum algorithms that can, in principle, solve significant computational problems faster than any classical algorithm. Unfortunately the discovery of Shor's algorithm for factoring large composite integers was not followed by a wave of new quantum algorithms. To date there are only about seven quantum algorithms known. **Williams and Clearwater**

- Why?



Comments

- Our intuition is rooted too firmly in the classical computational model.
 - We have yet to fully exploit even conventional non-standard architectures, e.g. Field Programmable Gate Arrays (FPGAs) - see comments on non-standard computation at www.cs.york.ac.uk/~susan
- We seem happier thinking in terms of circuits and gates rather than algorithms.
- Basically, we lack the models, metaphors, guidelines etc. that software engineering has evolved over the past 40 years or so.
- We don't have the abstractions we need and we don't have a means of getting from real-world problems to excellent implementation solutions.
- The standard refinement technique is genius, or inspiration (for those who don't live in the modern PR world)



Where have all the Computer Scientists Gone?

- If quantum software engineering is ever to emerge we will need to build it up - and this WILL require computer scientists to get seriously involved.
- Deriving algorithms is obviously hard, what can we do?
- Same as we usually do when things get THAT hard.
 - Reach for techniques inspired by nature, genetic algorithms, genetic programming, simulated annealing, ants, swarms etc.



Genetic Programming Example

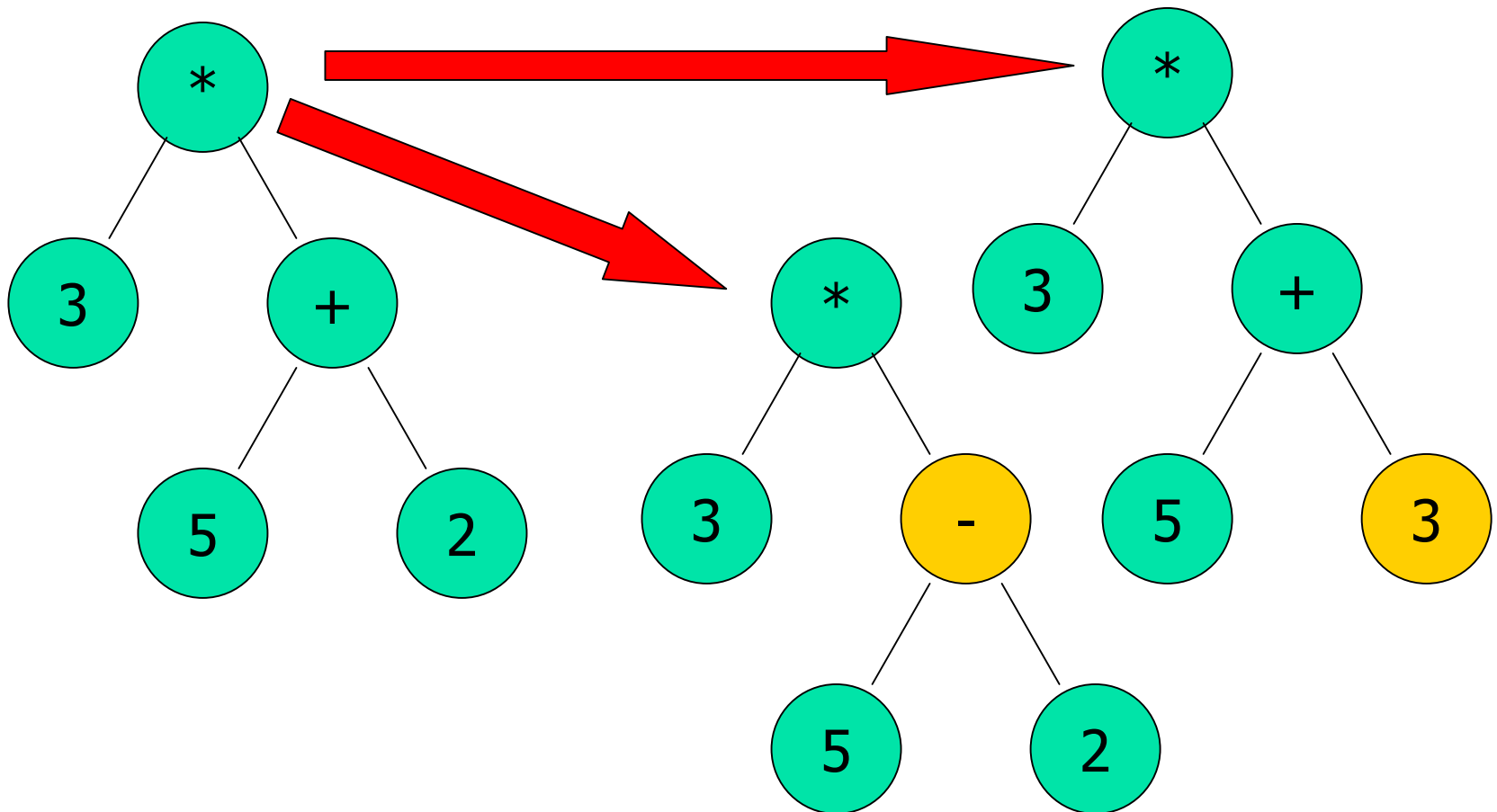


Genetic programming

- Based on populations of programs.
- Programs are mated and mutated.
- Some programs are better than others. Fitter ones survive and breed between generations.

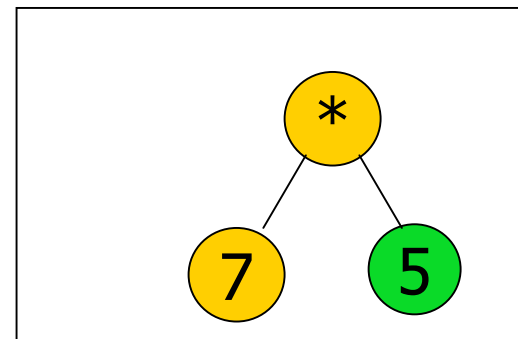
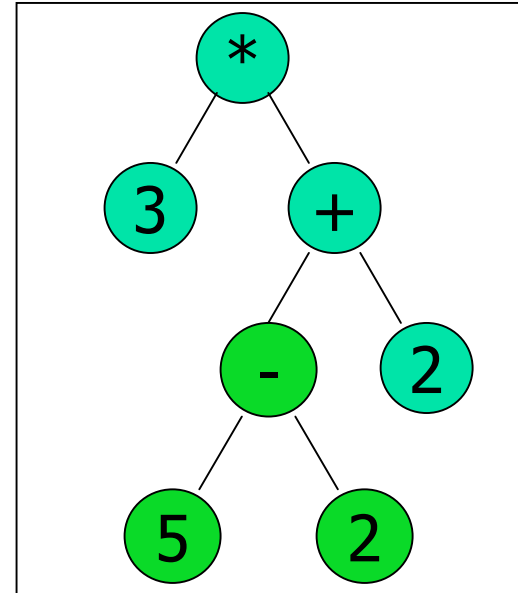
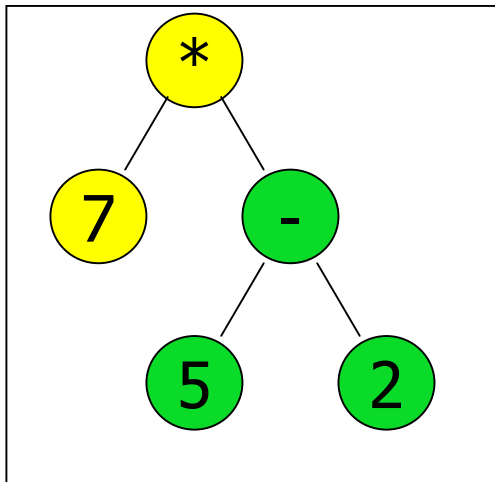
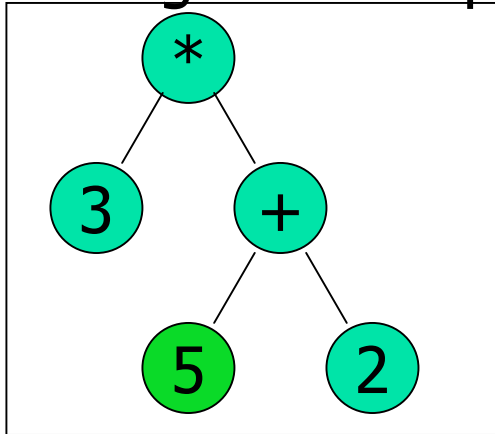
Mutation

- Some part of the program is changed.



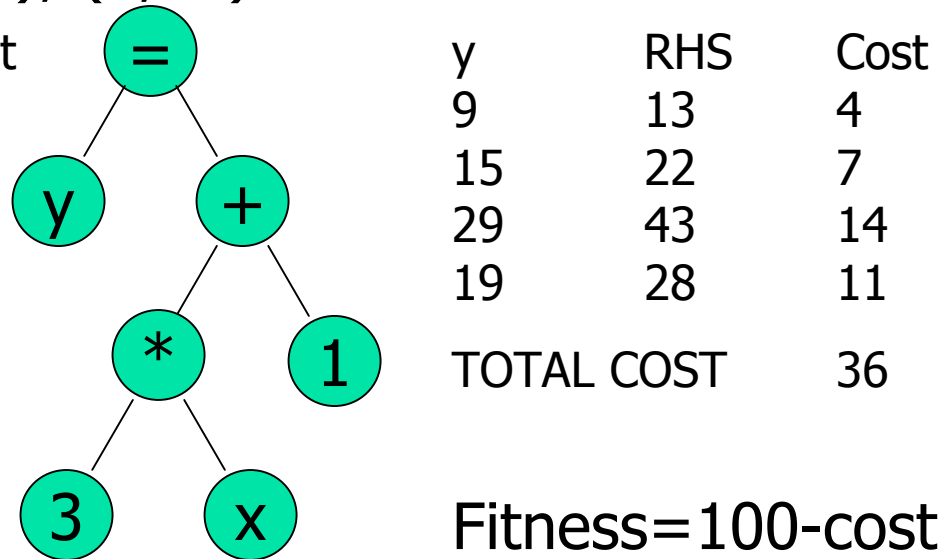
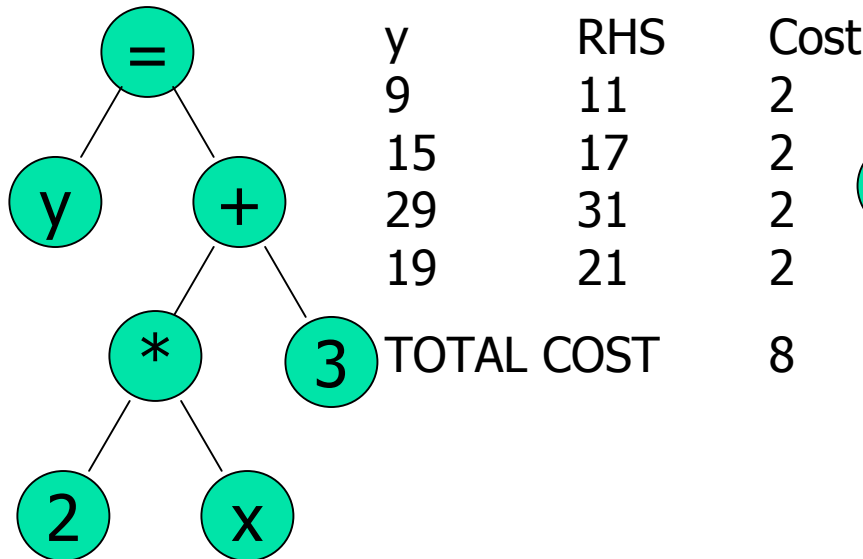
Crossover-Mating Programs

- Programs swap subelements.



Fitness - some are fitter than others

- Each program is assigned a fitness indicator. This measures how 'good' the program is.
- Problem - evolve a symbolic expression to describe the following (x,y) data elements:
 - (4,9), (7,15), (14, 29), (9,19)



Fitness = 100 - cost
or similar



Selection

- Each program is assigned a fitness indicator. This measures how 'good' the program is.
- Given a current population individual programs are selected for the next population according to fitness.
- Better programs have greater tendency to survive this process.
- Survivors are then mated and offspring mutated. These offspring then form the next generation.
- Variations on this theme exist. Some portion of the population is replaced rather than all. Best ever program is guaranteed survival (elitism) etc.



Language

- Of course you need a language in which programs can be expressed.
- In examples we have assume that this was the language of arithmetic expressions.
- Need a basic function set: $+, -, *,$ etc.



Spector et al.

- Attempted to evolve a faster than classical algorithm for the Early Promise Algorithm. You are guaranteed that a Boolean function f on n -binary inputs is either **balanced** (equal number of 0s and 1s) or **uniform** (all 1s or all 0s)

$$\begin{array}{ll} f_1 : f(0) = f(1) = 0 & \leftarrow \text{Uniform} \\ f_2 : f(0) = f(1) = 1 & \leftarrow \text{Uniform} \\ f_3 : f(0) = 0, f(1) = 1 & \leftarrow \text{Balanced} \\ f_4 : f(0) = 1, f(1) = 0 & \leftarrow \text{Balanced} \end{array}$$

How many function evaluations does this require?



Spector et al.

- Attempted to evolve algorithm for two input bits. There are 8 permissible functions represented by the rows below:

	$f(00)$	$f(01)$	$f(10)$	$f(11)$
	0	0	0	0
	1	1	1	1
	1	1	0	0
	1	0	1	0
	1	0	0	1
	0	1	1	0
	0	1	0	1
	0	0	1	1



Spector et al.

- Actual means was rather interesting. Use GP to evolve a program that when executed generated a quantum circuit that can be applied to examples. The function set included:
 - H-Gate(qubit no). Hadamard gate with one parameter coerced to be 0-2.
 - U-Theta-Gate(QubitNumber, RotationInRadians)
 - CNOT-GATE(ControlQubit,TargetCubit)
 - Some iteration control structures
 - Various bits of arithmetical elements (+, -, 1-, *2 etc.)
- Terminal set included `"*NUM-QUBITS*"`, `"*NUM INPUT QUBITS*"`, π , $i = \text{ROOT}(-1)$ and a few more



Quantum Program

(IQ

(NAND-GATE

(+ (* (1- 0) (ITERATE PI PI))

(U-THETA-GATE -1 (*2 * NUM-INPUT-QUBITS*)))

(%2

(+ H-GATE IQ (IQ (1- PI))))

(ITERATE

(1- (SQRT (CNOT-GATE (U-THETA-GATE 1 (IVAR *NUM-QUBITS*))

(IVAR (ITERATE PI *NUM-OUTPUT-QUBITS*))))

(1/x

(NAND-GATE

(* (SQRT -1) (-(%P (IVAR 0) *NUM-QUBITS*) * NUM-INPUT-QUBITS*))

(1/x * NUM-INPUT-QUBITS*) PI))))

(NAND-GATE (IQ (1- (IQ (%2 (%2 (IQ (*2 * NUM-INPUT-QUBITS*))))))))

(IQ (IVAR PI))

(SQRT (%2 (1- (ORACLE-GATE))))))



Interpreting functions

- As well as generating a gate for the circuit, the function will return some value
- $H(x)$ returns x as a numeric value. Etc.



Spector et al.

(UN-THETA 2 4)

(Hadamard 0)

(UN-THETA 1 1)

(Oracle)

(NAND 2 1 0)

(UN-THETA 2 4)

(Hadamard 0)

(UN-THETA 1 2)

(CNOT 1 2)

qubit 2 $\theta=4$ radians

Hadamard on qubit 0

qubit 1 $\theta=1$ radian

acts on qubit 2

inputs are 2 & 1, output is 0

qubit 2 $\theta=4$ radians

Hadamard on qubit 0

qubit 2 $\theta=2$ radians

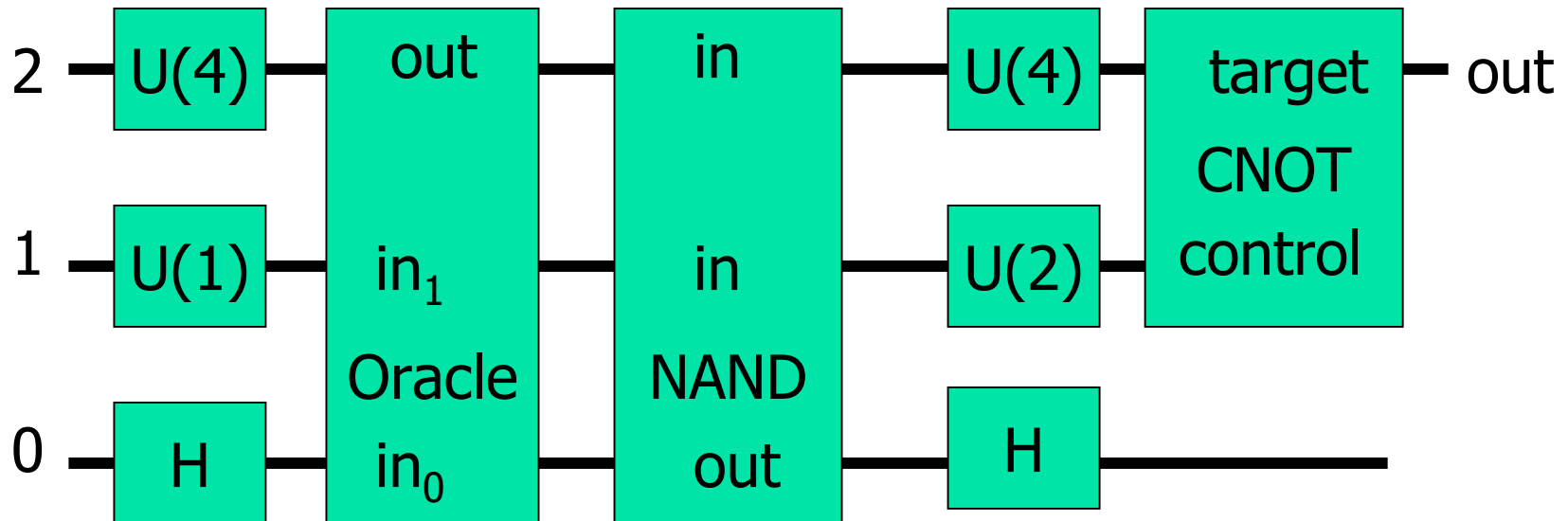


Spector et al.

- Max number of generations 1001
- Size of population 10000
- Selection method tournament (pick five and the fittest wins and survives)

Spector et al.

- Generates the circuit





Spector et al.

- Aim to read off answer in qubit 2 with a single measurement, e.g. 0 for balanced, 1 for uniform.
- Cost function of the form...

$$\frac{\sum_{i=1}^{numCases} \max(0, error - 0.48)}{\max(hits, 1)}$$

- A hit is when the program produces an error less than 0.48 for a test case.

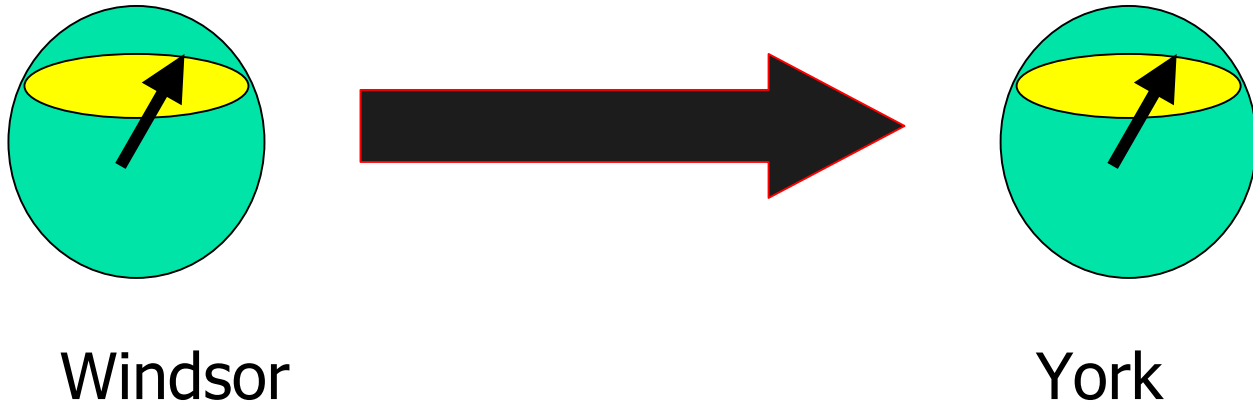


Spector et al.

- Ideas could also be used for scalable algorithms:
- importance of “Num-Qubits” rather than just ‘3’ etc.
 - Not yet obviously exploited - deserves more attention
 - Some preliminary but simple results indicated.
 - Bumping up the abstraction is crucial.
- A GP approach is not essential. Others, e.g. Massey (with Clark, Stepney) have used genetic algorithms to evolve simple circuits.

Teleportation

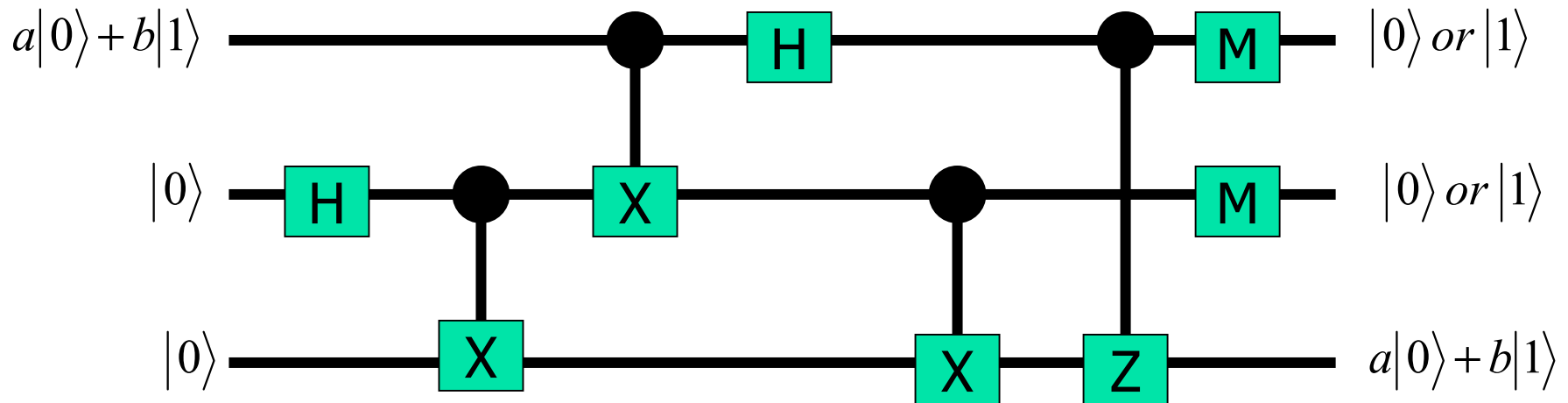
- We can use entanglement to teleport a quantum system.



- But the copy in Windsor gets destroyed. No cloning applies (one photon has been teleported so far)

Can teleportation be evolved?

- The answer may be yes.
- Won't give details but you easily imagine possible cost functions.
- Three qubits, the first is the source qubit, the other two are for the teleportation engine. If qubit 3 is the destination then we can check how close the final qubit 3 superposition is to that of qubit 1.





Where Next?



You must ignore the physics

- We are still at a very low level of abstraction.
- We need to find new specifications means, new building blocks, new language primitives etc.
 - Has anyone seen teleportation as a quantum language level primitive:-)))) This is arguably still very low level.
 - We need useful specification languages too.



Why Are Computer Scientists So Servile?

- We tend to put up with the physics (gates etc.) we are given.
- We should be going to the physicists with an idea of what we want in terms of abstract gates etc and asking:
 - 'Can you build these?' (You know you want to).



You must not ignore the physics

- Functionality is important, but so are non-functional properties.
 - Practical limitations on the number of qubits that will be manageable - and early exploitation of what becomes available may be key.
 - Reliability and efficiency
- How about equivalent subnetwork substitution (metaheuristic search has been used for traditional program optimisation).
- What about multi-criteria approaches (MOGAs etc)?
- Why limit ourselves to (binary) qubits
 - physicists already working with d-valued quantum registers - qudits.



Software Engineering Inspiration

- What is the equivalent of program specialisation?
 - Do things like program slicing or partial evaluation have quantum analogues?



Quantum Protocols

- Teleportation is a protocol but quantum mechanics permits a wide variety of uses.
- Traditional protocols design has its own correctness calculi and refinement guidelines/rules.
 - Where are the analogues for quantum protocols?
- We have evolved security protocols in BAN logic. Let's do it for quantum.



Visualise (Perceptualise)

- The search spaces may be very unusual.
- Our intuition on QIP seems poor.
- We need help simply to build up intuition about the subject matter.



Summary

- We need new abstractions and new ideas.
- We need new building blocks (but we don't know what they are).
- We may be faced with limited physical resources where rapid exploitation may lead to commercial success.
- We need to develop quantum software and systems engineering.