Model–Checking Quantum Protocols

Simon J. Gay[1]*, Rajagopal Nagarajan[2]**, and Nikolaos Papanikolaou[2]**

[1] Department of Computing Science, University of Glasgow
simon@dcs.gla.ac.uk
[2] Department of Computer Science, University of Warwick
{biju,nikos}@dcs.warwick.ac.uk

Abstract. We introduce model-checking techniques for the automated analysis of quantum information protocols. These protocols take advantage of features of quantum theory such as entanglement and quantum measurement and some implementations already exist. Our techniques enable us to model a class of quantum protocols (those expressible within the stabilizer formalism) which are simulable in polynomial time. We discuss the QMC model checker, which verifies models of systems that combine both quantum and classical computations. The modelling language of the tool is presented, along with the temporal quantum propositional logic (QCTL) that is used for specifying properties. We discuss model-checking procedures and efficiency issues. We apply QMC to a case study.

1 Introduction and Background

The novel field of quantum computation and quantum information has been growing at a rapid rate; the study of quantum information in particular has led to the emergence of communication and cryptographic protocols with no direct analogues in classical computing. Quantum information protocols have interesting properties which are not exhibited by their classical counterparts, but they are most distinguished for their applications in cryptography. Notable results include the unconditional security proof [1] of quantum key distribution [2, 3] and the impossibility proof of unconditionally secure quantum bit commitment [4]. The former of these results in particular is one of the reasons for the widespread interest in this field, and it demonstrates an achievement not possible in classical cryptographic systems.

The benefits of automated verification techniques are well known for classical communication protocols, especially in the cryptographic setting. *Model-checking* has been used to uncover subtle flaws in protocols and system designs [5, 6]. Our research programme is to apply similar techniques to quantum protocols with the

* Partially supported by the EPSRC grant EP/E00623X/1 *(Semantics of Quantum Computation)* and EP/F004184/1 *(Quantum Computation: Foundations, Security, Cryptography and Group Theory)*.

** Partially supported by the EU Sixth Framework Programme (Project SecoQC: *Development of a Global Network for Secure Communication based on Quantum Cryptography*) and the EPSRC grant EP/E00623X/1 *(Semantics of Quantum Computation)*.

expectation of gaining corresponding benefits. Today, while simulation tools for quantum information systems abound (see [7] for a list), to our knowledge no other authors have developed a tool aimed at verification.

In this paper we describe just such a tool and apply it to a case study; this tool is based on our earlier work [8, 9], and is named QMC (Quantum Model Checker); it allows for automated verification of properties of quantum systems using model-checking techniques. Properties to be verified are expressed using QCTL [10], a CTL-based temporal logic designed specifically for quantum information. QMC analyses systems which can be expressed within the *stabilizer formalism*, which is known to be simulable in polynomial time [11, 12, 13]. This is significant, as any kind of model-checking necessarily involves simulation and, in general, quantum systems cannot be simulated efficiently on classical hardware. The systems expressible in this formalism are restricted, in the sense that the set of operations which they can perform is not universal for quantum computation. Nevertheless, stabilizer circuits are sufficient to describe a number of systems of practical interest.

Existing simulation tools for quantum systems [7] are designed to help the user understand the function of a given quantum circuit; some tools have a graphical user interface, and many allow the simulation of circuits with arbitrary quantum gates, even if there is a substantial computational cost due to the limited power of the classical machine running the simulation. Simulators which allow only stabilizer circuits include CHP [12] and GraphSim [13]; the algorithms in QMC are based on those used in the former of these two, as well as on particular algorithms developed in [14]. We do not know of any previous tool which provides automated checking of a circuit specification. Another distinctive characteristic of QMC is the automatic exploration of all possibilities generated by quantum measurements, which are probabilistic by nature [15].

Acknowledgements. We would like to acknowledge the help of Paulo Mateus and Pedro Baltazar (IST Lisbon) regarding the semantics of the logics EQPL and QCTL.

Background. We do not assume a familiarity with the nuts and bolts of quantum computing; instead, we review the basic concepts here and refer the reader to standard texts, e.g. [15], for details. In this paper we are interested in communication systems which may involve both classical computing devices and quantum components. In our models we assume the availability of quantum systems consisting of a finite number of particles, distributed between some users; the users may perform certain specific types of quantum operations on these particles, as well as standard-basis measurements. We confine ourselves to the states and operations which arise in the so-called stabilizer formalism; according to the Gottesman-Knill theorem [11], quantum circuits in this formalism are simulable in polynomial time on a classical computer. These terms will now be explained in more detail.

According to the mathematical formalism of quantum mechanics, the state of a quantum system is represented as a vector in a Hilbert space. The most basic system of interest is the *qubit,* or quantum bit, which corresponds to a particle

with two degrees of freedom (such as a polarised photon, or a spin-$\frac{1}{2}$ particle). The general state of a qubit is given by the vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$, and the vectors $\{|0\rangle, |1\rangle\}$ form a basis of the two-dimensional Hilbert space $\mathcal{H}_2$. The quantum systems we are interested in consist of multiple qubits; a 2-qubit system will generally be in a state belonging to the four-dimensional space $\mathcal{H}_4$, and an n-qubit system has a state space which is the tensor product of n copies of $\mathcal{H}_2$, and is spanned by 2^n basis vectors.

Computations on systems of qubits are expressed by linear operators that are unitary. An operator U is *unitary* if it satisfies $UU^\dagger = U^\dagger U = I$ (the operator I is the identity matrix of suitable dimension). Examples of operators include the Pauli gates X, Y, Z, the phase gate P, the Hadamard operator H and the controlled-not gate C. The action of the Pauli gates and the phase gate on the state of a single qubit are shown below.

$$\begin{aligned}
X(\alpha|0\rangle + \beta|1\rangle) &= \alpha|1\rangle + \beta|0\rangle \quad \text{(bit flip)} \\
Y(\alpha|0\rangle + \beta|1\rangle) &= \alpha|1\rangle - \beta|0\rangle \quad \text{(bit and sign flip)} \\
Z(\alpha|0\rangle + \beta|1\rangle) &= \alpha|0\rangle - \beta|1\rangle \quad \text{(sign flip)} \\
P(\alpha|0\rangle + \beta|1\rangle) &= \alpha|0\rangle + i\beta|1\rangle \quad \text{(phase change)}
\end{aligned}$$

The Hadamard operation creates superpositions (linear combinations of the basis states) from a single basis state (see [15] for the formal definition and relevant notation). For example, applying the Hadamard gate to the first qubit of a two–qubit system, $|\varphi\rangle = |1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$, will put the system in the state

$$\begin{aligned}
(H \otimes I)|\varphi\rangle &= H|1\rangle \otimes \frac{1}{\sqrt{2}}(I|0\rangle + I|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
&= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)
\end{aligned}$$

Note that the sum of the amplitudes of the coefficients should be identically one for the state vector to be normalised. The controlled-not gate acts on two qubits, the *control* and *target;* if the control qubit is in state $|1\rangle$, then the target qubit is flipped:

$$C_{1,2}|\varphi\rangle = \frac{1}{\sqrt{2}}(C_{1,2}|00\rangle + C_{1,2}|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

The actual state of a quantum system is unknown until it is measured; however, the act of measurement is destructive and collapses the system probabilistically to a basis state. Measuring qubit $|\psi\rangle$ above with respect to the so-called standard basis $\{|0\rangle, |1\rangle\}$ will collapse the qubit into either state $|0\rangle$ (with probability $|\alpha|^2$) or state $|1\rangle$ (with probability $|\beta|^2$).

The stabilizer formalism (originally due to Gottesman [11]) is concerned with the quantum states that arise in computations involving only the operations mentioned above. In order to perform arbitrary quantum computations, a universal set of gates would be required; however, the Clifford gates $\{H, C, P\}$ do not constitute a universal set. What is distinctive about this set of gates is that the

operations may be simulated efficiently on a classical computer (this is known as the Gottesman–Knill theorem; see [11, 15]).

2 Modelling Quantum Protocols

A *quantum protocol* is loosely defined as a set of operations and measurements on a global quantum state, which may be distributed amongst a number of users; measurement outcomes may be communicated classically between users. Usually quantum protocols are expressed schematically, using the quantum circuit model [15]. Of course, this model only allows one to describe the computational parts of a protocol. For the sake of illustration, we consider the quantum teleportation protocol, shown using standard quantum circuit notation (see [15] for details) in Figure 1.

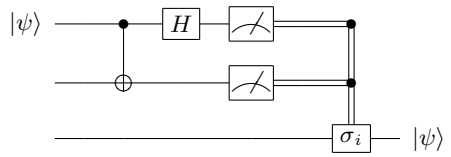

Fig. 1. Quantum circuit diagram for the teleportation protocol.

Teleportation is a process which allows two users who share an entangled pair of particles, to exchange an unknown qubit state by communicating only two classical values, namely, the outcomes of two measurements. The protocol is described below.

Alice wishes to send Bob the qubit state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$; we shall call this particle '1'. To begin with, a pair of particles (labelled '2' and '3') is placed in the quantum state

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Particle 2 is given to Alice and particle 3 is given to Bob. Since $|\Psi\rangle$ is an *entangled state,* any measurement performed on the one particle will affect the state of the other particle irreversibly. (For instance, if Alice were to measure particle 2 with respect to the standard basis, she would collapse the state of particles 2 and 3 to either $|00\rangle$ or $|11\rangle$ at random with equal probability $\left(\frac{1}{\sqrt{2}}\right)^2 = 0.5$.) So we have a system of three particles described by the overall quantum state $|\psi\rangle \otimes |\Psi\rangle$.

Alice starts by applying the gate $C_{1,2}$ and then the Hadamard H_1. She subsequently measures qubits 1 and 2 with respect to the standard basis, and records the outcomes of her measurements M_1 and M_2. She sends these two classical values to Bob (this is represented by the double line in the figure), who then applies

the operator $X^{M_1} \cdot Z^{M_2}$. At the end of the protocol, particle 3 will be in the state $|\psi\rangle$, thus achieving Alice's goal of sending the original state of particle 1.

This example shows some typical characteristics of a quantum protocol. This particular protocol is quite simple, as it involves only two users and just a handful of operations and measurements. As such, it is amenable to analysis by hand and can easily be shown to be correct. However, large systems (such as quantum key distribution networks [16]) would typically include the teleportation protocol as a primitive and would combine it with other computations and transmissions, both quantum and classical.

Note the style in which the protocol is presented. Presentations of quantum protocols in the physics literature tend to be rather informal, and often mix low-level implementation details (e.g. how to prepare a particular state in a physical device) with structural aspects (the ordering of operations, measurements and transmissions). Larger systems tend to combine various quantum computations with classical procedures for processing and transmitting bit values arising from measurements.

Various programming and specification formalisms have been proposed in order to address the shortcomings of the quantum circuit model when describing quantum protocols. These include quantum programming languages and quantum process algebras (see [17] for a survey). We have built an imperative-style concurrent specification language for the needs of the quantum model-checking tool QMC. For the purpose of this paper, we will demonstrate the syntax of this language by example. In this language the teleportation protocol (assuming we are trying to teleport the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$) may be expressed as follows:

```
program Teleport;
var e1,e2:qubit; ch:channel of integer;
process Alice;
var q:qubit; a,b:integer;
begin
 q := newqubit; had q;
 e1 := newqubit; e2 := newqubit;
 had e1; cnot e1 e2;
 cnot q e1; had q;
 a := meas q;
 b := meas e1;
 ch!a; ch!b;
end;
process Bob;
var c,d: integer;
begin
 ch?c; ch?d;
 if
 :: ((c=1) and (d=0)) -> X q; break;
 :: ((c=0) and (d=1)) -> Z q; break;
 :: ((c=1) and (d=1)) -> X q; Z q; break;
```

```
 :: ((c=0) and (d=0)) -> break;
 fi
end;
endprogram.
```

In our setting, we allow for global variables (such as `e1`, `e2`), typed communication channels (such as `ch`) which are always global, and local (private) variables for each process (such as `a`,`b`,`c`,`d`,`q`). Communication is asynchronous, with executability rules restricting the way in which the interleaving of process is performed. For instance, the process `Bob` cannot start unless channel `ch` is filled with a value.

The semantics of a protocol model is given by a tree structure, whose nodes each consist of a tuple $(\rho, |\psi\rangle, \mu, \nu)$, where ρ is a classical store that maps variable names to values, $|\psi\rangle$ is a global quantum state given by the tensor product of the states of all qubits declared in the model, μ is a mapping from variable names to process names (so as to keep track of local variables), and ν is a mapping from variable names to variable types (variable types are: integers, booleans, floating point numbers/reals, qubit indices). Note that a qubit variable is a pointer to a qubit in the overall quantum state; its value is just an integer, namely the index of the corresponding qubit. Branching in the tree structure occurs whenever there is non-deterministic choice, looping (which contains non-deterministic choice), or an indeterminate measurement (i.e. a measurement of the quantum state which could produce a random result). A formal definition of the operational semantics of protocol models for QMC is work in progress.

3 Specifying Properties

The properties of quantum protocols which we are interested in reasoning about are properties of quantum state (e.g. which qubits are 'active' in a given state, which qubits are entangled with the rest of the system) over time. We are also interested in outcomes of different measurements, and the way in which the values of classical variables evolve. We have elected to use quantum computational temporal logic (QCTL [10]) for this purpose.

QCTL adds the usual temporal connectives (AX, EF, EU) of computational tree logic [18] to the propositional logic EQPL [19].The meaning of formulae in EQPL is expressed in terms of valuations, which are truth-value assignments for the symbols $\mathsf{qb}_0, \mathsf{qb}_1, \ldots, \mathsf{qb}_n$ corresponding to each qubit in the system. For instance, the quantum state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is understood as a pair of valuations (v_1, v_2) for a 2-qubit system such that $v_1(\mathsf{qb}_0) = 0$, $v_1(\mathsf{qb}_1) = 0$, $v_2(\mathsf{qb}_0) = 1$, $v_2(\mathsf{qb}_1) = 1$.

The formulae accepted by the QMC tool for verification allow the user to reason about the state of individual qubits, and involve usual logical connectives such as negation and implication. There are two levels of formulae: classical formulae, which hold only if all valuations in a state satisfy them, and quantum formulae, which are essentially logical combinations of classical formulae. For instance, the quantum conjunction in the formula $\phi_1 \curlywedge \phi_2$ is only satisfied if both the classical formulae ϕ_1 and ϕ_2 are satisfied in the current state. A particularly distinctive

type of quantum formula is of the form $[Q]$, where Q is a list of qubit variables $\mathsf{qb}_i, \mathsf{qb}_j, \ldots$; this type of formula is satisfied only if the qubits listed are disentangled from all other qubits in the system. The syntax of QCTL is given below (from [10]):

Classical formulae:	$\alpha = \bot \mid \mathsf{qb} \mid \alpha \Rightarrow \alpha \mid \alpha \vee \alpha \mid \alpha \wedge \alpha$
Terms:	$t = x \mid (t + t) \mid (tt) \mid \mathrm{Re}(\lvert\top\rangle_A) \mid \mathrm{Im}(\lvert\top\rangle_A) \mid \int \phi$
Quantum formulae:	$\gamma = (t \leq t) \mid \bot \mid (\alpha \sqsupset \alpha) \mid (\alpha \curlyvee \alpha) \mid (\alpha \curlywedge \alpha) \mid [\mathsf{qb}_i, \mathsf{qb}_j, \ldots]$
Temporal formulae:	$\theta = \gamma \mid \theta \sqsupset \theta \mid (\mathsf{EX}\theta) \mid ([\theta\ \mathsf{EU}\ \theta]) \mid (\mathsf{AF}\theta)$

3.1 Example

The requirement for the teleportation protocol described in section 2 is that, at the end of the protocol, no matter the measurement outcomes, the third qubit will be in the same state as the first qubit was to begin with, and this qubit will be disentangled from the rest of the system. We can express this requirement, for the case where the input is the quantum state $|0\rangle$, in the input language of QMC using the specification

```
finalstateproperty ([q2]) #/\ (!q2);
```

which corresponds to the EQPL formula $[q_2] \curlywedge (\neg q_2)$. The first part of the formula asserts that the last qubit (`q2`) is disentangled from the rest of the system, while the second part asserts that the current valuation assigns to this qubit a value of 0. The entire formula is true if both parts are true, indicated by the connective of quantum conjunction (we represent $\curlywedge$ in ASCII form by `#/\`). We can also use a temporal formula:

```
property true EU (([q2]) #/\ (!q2));                    (1)
```

4 The QMC Model Checker

We have developed a software tool known as QMC, which automatically explores all possible behaviours arising from a protocol description (in the format shown in Section 2), and enables QCTL properties to be checked over the resulting structure.

A protocol model will always consist of definitions of one or more processes; the commands performed by each of these processes must be interleaved (so as to emulate concurrent execution), and non–determinism (which occurs explicitly in selection structures (`if :: a -> ... :: b -> ... fi`) and implicitly when measurements with indeterminate outcomes are performed) must be resolved, producing an execution tree for the modelled system. A concrete example of this execution tree will be given in Section 5, where we will use QMC to analyse a particular protocol.

The QMC tool can be seen as comprising three main components: (1) a process scheduler, (2) a language interpreter, and (3) a verifier. The role of component (1) is

essentially to perform the tasks described in the previous paragraph. The language interpreter handles the execution of individual commands and keeps track of the overall classical and quantum state at each step. Finally, the verifier is responsible for evaluating QCTL formulae over the structure generated by (1) and (2).

It is important to explain the way in which global state is represented for QMC models. As described in Section 2, each node in the execution tree for a given model contains a tuple $(\rho, |\psi\rangle, \mu, \nu)$ where $|\psi\rangle$ in particular is the overall quantum state at that point in the simulation. The quantum state $|\psi\rangle$ is represented internally in an implicit way: rather than storing the so-called *state vector representation* of $|\psi\rangle$ (which grows exponentially in length as a function of the total number of qubits in $|\psi\rangle$), we use the *stabilizer array representation,* which is a binary representation of the set of Pauli operators that fix (or stabilize) $|\psi\rangle$. Using the stabilizer array representation, we gain significant computational benefits in terms of both space and time when simulating a given protocol, given that simulation of stabilizer circuits is performed using a polynomial time algorithm [12], and the representation of the state grows polynomially with the number of total qubits. There is scope for optimising the simulation further, notably by using low-level binary operations and by packing the the stabilizer array representation into 32-bit registers, as Aaronson and Gottesman chose to do in the final implementation of the CHP simulator (see [12] for details).

4.1 Verification Algorithms and Complexity

We turn now to the algorithms which QMC uses for the *verification* of QCTL formulae over protocol models.

Firstly, one should note that the logic QCTL comprises a purely propositional fragment, namely the exogenous quantum propositional logic (EQPL) proposed by Mateus and Sernadas [19]. This fragment may be interpreted, without much loss of generality, over a single quantum state $|\psi\rangle$. The general definition of the semantics of EQPL has been given [19] in terms of a so-called quantum interpretation structure, which includes not only a quantum state $|\psi\rangle$, but also a classical state ρ and a means of specifying entanglement partitions of $|\psi\rangle$. Note that in our setting we also have a global classical state, which takes the place of ρ.

Evaluating EQPL formulae over any state $|\psi\rangle$ arising from the simulation of a protocol model requires being able to determine all the valuations in that state, so that the truth value of any propositional constant (e.g.: qb_i where $0 \leq i \leq N$ for an N-qubit system - this constant corresponds to the state of the ith qubit in the quantum state) can be computed. What this means in more practical terms is that, in order to determine whether a given qubit has valuation *true* (1) or *false* (0) in the current state, it is necessary to extract all the basis vectors which are present in the state vector expansion of $|\psi\rangle$. The process of extracting all the basis vectors requires converting from the space–efficient stabilizer array representation to the state vector corresponding to $|\psi\rangle$, and this conversion can take up to a maximum of 2^N steps if all the 2^N basis vectors appear in $|\psi\rangle$. Even when $|\psi\rangle$ is a stabilizer state, it may contain all of the basis vectors with non-zero coefficients. Therefore in general, evaluating a classical formula requires solving a SAT problem,

and of course this is **NP**-complete. This observation seems rather discouraging given that the process of verifying a state formula requires us to lose the efficient state representation which is used during simulation.

However, there are cases for which we can avoid the conversion from stabilizer array to state vector; for certain classes of formula we can extract the necessary valuation information by processing the stabilizer array directly. We have observed that certain classical EQPL formulae, which do not involve the conjunction operator $\wedge$, may be checkable on a given state $|\psi\rangle$ by just examining the contents of those columns in the stabilizer array corresponding to the qubits in the formula. We are still investigating optimisations and heuristics such as this, bearing in mind that the most general EQPL formulae still require performing a state vector conversion. In future, we should probably investigate using an off-the-shelf SAT solver.

The most interesting class of quantum state formulae for which we were able to obtain efficiency improvements (esp. by avoiding the state vector conversion) were those of the form $[\mathsf{qb}_i, \mathsf{qb}_j, \ldots]$, i.e. *entanglement partition formulae*. Such a formula is true with respect to a given quantum state $|\psi\rangle$ if all the qubits indexed within the square brackets may be separated from all the other qubits in the state (that is to say, if qubits $\mathsf{qb}_i, \mathsf{qb}_j, \ldots$ are *disentangled* from the rest). We refer to a list of qubits $\mathsf{qb}_i, \mathsf{qb}_j, \ldots$ as a partition of the state $|\psi\rangle$ if the formula $[\mathsf{qb}_i, \mathsf{qb}_j, \ldots]$ is satisfied by $|\psi\rangle$.

Checking whether a given set of qubits constitutes a partition of a *stabilizer state* $|\psi\rangle$ is possible using the polynomial time algorithm of Audenaert and Plenio [14] for the so-called "bipartite Clifford Normal Form (CNFP)." We have implemented this algorithm in QMC. It is worth noting that the algorithm involves a special sequence of row and column operations on the stabilizer array and allows us to extract entanglement information from this representation directly.

Note that the implementation of the temporal connectives of QCTL follows directly from the definitions in [10]. The usual marking algorithms for computational tree logic are applied to the tree structure generated by QMC. The state formula contained in a temporal formula is evaluated for all states in the tree, producing the initial marking.

5 Case Study

QMC is ideally designed to allow for the analysis of systems comprising protocols with both classical and quantum computations (subject to the restriction that those quantum computations are expressible within the stabilizer formalism). In this section we consider a model of a system combining part of the quantum key distribution protocol BB84 [2] with the quantum error correcting code described in [20] (the quantum bit flip code). We will discuss these protocols and proceed to explain the structure of a QMC model for this system. Then we will turn to the verification of this model using the tool.

Quantum key distribution enables two users, Alice and Bob, to establish a common cryptographic key which is secure in the presence of eavesdropping. The

two users are assumed to be linked by a quantum channel (such as an optical fibre, through which qubits can be transmitted in the form of polarised light bursts), and they also can communicate over a public, authenticated, classical channel. The attacker has full access to the quantum channel, but can only monitor the classical channel passively. Quantum key distribution has been proven unconditionally secure against all attacks permitted by the laws of quantum mechanics [1] and implementations of this technology already exist, both in academic [21] and commercial settings (see e.g. `http://www.idquantique.com`). We consider a simplified version of the BB84 protocol for quantum key distribution here, omitting some classical post-processing steps (esp. secret-key reconciliation and privacy amplification, see [15] for details) that the users must perform after the quantum transmission is complete.

A single round of the BB84 protocol consists of the following steps:

1. Alice prepares a new qubit in one of the four states: $|0\rangle$, $|1\rangle$, $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The first two form the so-called *rectilinear basis,* the other two constitute the *diagonal basis*. Each of these states represents a classical bit with respect to the chosen basis (e.g. a 0 is represented by $|0\rangle$ in the rectilinear basis and by $H|0\rangle$ in the diagonal basis). She sends the qubit to Bob over the classical channel, and keeps record of her choice of basis and bit.
2. Bob receives the qubit; not knowing which basis Alice used to prepare the state, he chooses one of the two bases and measures it. If his choice of measurement is compatible with Alice's (i.e. if he has chosen the same basis that was used to prepare the state) then the laws of quantum mechanics guarantee that his measurement result will match Alice's original bit; otherwise his measurement result will be either 0 or 1 with equal probability, and the state of the qubit will become one of the two states of the basis used for measurement.
3. After Alice reveals to Bob which basis she used to prepare the qubit; if his choice of basis is correct, then both users keep the corresponding bit value and use it as part of the final key. Otherwise the corresponding bit is discarded by both users.

The protocol must be repeated several times in order for a key to be established; a common key bit is generated every time Bob makes a compatible basis choice for his measurement. We model one run of the protocol here.

A simple attack that can be made on this protocol is the following:

– The attacker, Eve, intercepts the qubit that Alice has sent (note that it is impossible to copy the qubit due to the no-cloning theorem of quantum mechanics [15]), chooses a measurement basis of her own, and measures the qubit to obtain a (possibly correct) bit value. She then sends the measured qubit to Bob.

It is not difficult to see that an incorrect choice of measurement basis by Eve would disturb the qubit which Alice originally sent, placing it in a different basis

state. It is possible for Alice and Bob to detect this disturbance by exchanging a few of their bits, but we will not include this step in our model.

We can model Alice's behaviour as the following QMC process (we assume that the qubit `q` is a global variable, and similarly for `ch`):

```
process Alice;
var bit,basis:integer;
begin
q := newqubit;
    q1:= newqubit; q2:= newqubit; /*for redundant encoding of q*/
    q3:= newqubit; q4:= newqubit; /*will be used for error recovery*/
if
:: basis:=0; bit:=0;
:: basis:=0; bit:=1; X q;
:: basis:=1; bit:=0; had q;
:: basis:=1; bit:=1; X q; had q;
fi
cnot q q1; cnot q q2;
ch!q;
end;
```

Notice how Alice uses the X and H gates to prepare the different qubit states in each case. Applying the H gate essentially transforms a rectilinear basis state into a diagonal basis state.

The processes describing the actions of Eve and Bob are quite similar, since both involve a basis choice and a measurement. Here is the definition of Eve's process in QMC's input language.

```
process Eve;
var ebit,ebasis:integer;
begin
ch?q;
if
:: ebasis:=0;
:: ebasis:=1; had q;
fi
bbit := meas q;
ch!q;
end;
```

So far we have assumed that transmissions of qubits are noise-free, i.e. the communication channels are perfect. Now we will revise our model to describe the case where a quantum error-correcting code is used to recover from a single bit flip caused by a noisy channel. The code requires each individual qubit prepared by Alice to be redundantly encoded into a 3-qubit system, so that the state $|0\rangle$ is transmitted as $|000\rangle$, and $|1\rangle$ is transmitted as $|111\rangle$. We assume that the quantum channel may induce a bit flip error on any one of the three qubits that are used in

this code; for instance, the channel might transform the state

$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \quad \text{into} \quad \frac{1}{\sqrt{2}}(|010\rangle + |101\rangle)$$

In this case, the second qubit has been disturbed by the channel. In order to detect such an error, two additional qubits are used; they are known as *ancillas*. By applying a sequence of operations and measurements to the ancillas, the so–called *error syndrome* is obtained, which determines the location of the error. Then, the X operator is applied to the erroneous qubit, thus restoring the initial quantum state of the 3–qubit system (i.e. $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ in the above example). The quantum circuit for the bit–flip code is given in Figure 2.

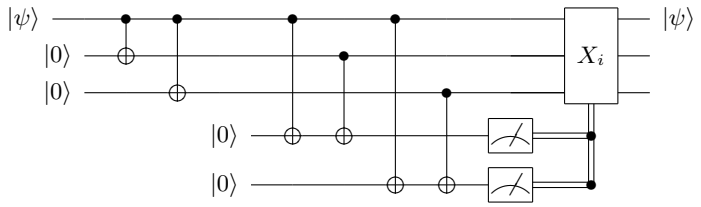

Fig. 2. Quantum circuit diagram for the qubit bit–flip code.

In order to account for the error correction in the model of the protocol we have been describing, we need: (1) to introduce two qubits `q1` and `q2` which are used to encode the original state as a three-qubit state, (2) to introduce two qubits `q3` and `q4` corresponding to the ancillas which are used to detect the location of the error in the three-qubit state. The QMC process for Alice needs to include the additional commands `cnot q q1; cnot q q2;` just before the transmission `ch!q`, thus encoding the state of qubit `q` across the three qubits.

We model the channel that causes the disturbance as a separate process `Disturb`, defined below:

```
process Disturb;
begin
ch?q;
if
:: X q;
:: X q1;
:: X q2;
fi
ch!q;
end;
```

The final part of the model consists of the process `Correct`, which is responsible for applying the error correction procedure suggested directly by the circuit in Figure 2. The process uses a variable `count` to enforce the order in which the actions stated are executed.

```
process Correct;
var a,b: integer;
begin
ch?q;
cnot q q3; cnot q1 q3;
cnot q q4; cnot q2 q4;
a:=meas q3;
b:=meas q4;
if
:: ((a=1) and (b=1)); -> X q;
:: ((a=1) and (b=0)); -> X q1;
:: ((a=0) and (b=1)); -> X q2;
fi
ch!q;
end;
```

5.1 Properties for Verification

The model of a quantum key distribution system with an error correcting component described in the previous section is quite realistic, given that we are taking into account the possibility of a direct attack on the protocol as well as the presence of a noisy quantum channel. A larger system for quantum key distribution might also involve quantum teleportation, so that a qubit is transferred not via a direct quantum channel but through the use of entangled quantum states. We discussed the quantum teleportation protocol in Section 2 separately.

There are number of combinatorial possibilities which arise during quantum key distribution. Depending on the choices of basis made by Alice, Bob and Eve, it may or may not be possible to detect Eve's presence. For instance, a compatible choice of basis by all three users implies that Eve's measurement of the transmitted qubit does not cause a disturbance to its state. There will be cases when the outcome of Bob's measurement matches the choice of bit originally made by Alice, cases in which Eve's measurement is correct, and so on. QMC explores all possibilities automatically and constructs the correct tree structure for the model described. We can verify different QCTL formulae expressing success or failure of the protocol. For instance, the BB84 protocol normally succeeds to produce a key bit if Alice and Bob use compatible basis choices, i.e. when the following state formula is satisfied (`bbasis,bbit` are the variables corresponding to Bob's chosen basis and bit value):

```
property ((basis=bbasis) => (bit=bbit))
```

This property applies to the entire protocol including the quantum error correction procedure.

6 Summary and Conclusions

We have described QMC, which is to the best of our knowledge, the first automated verification tool targeted specifically at protocols for quantum communica-

tion. A case study for QMC of a system combining quantum key distribution and quantum error correction was also presented. The modelling language of the tool was demonstrated by means of an example, and we discussed the logic QCTL, originally due to Baltazar, Chadha and Mateus [10], which is used in QMC for specifying properties of systems. Although the examples presented in this paper are quite simple, they constitute the basic building blocks for practical long-distance quantum communication and cryptographic systems.

An important aspect of QMC is that it allows the user to model those protocols involving operations within the quantum stabilizer formalism [11]; while this set of operations is not universal for quantum computation, it allows us to express a large class of interesting protocols and has the benefit of efficient simulation on a classical computer. We discussed the computational cost of verifying EQPL state formulae on stabilizer states and the possibility of various efficiency improvements, while noting that the evaluation of such formulae on stabilizer states is an instance of the SAT problem, which is known to be complete for the complexity class **NP**.

Our long-term objective is to build a model checker that would be able to model and analyse *any* protocol with quantum and classical components. However, it is worth noting that, in contrast to classical systems, the protocols that are currently in prevalence in the quantum setting are few in number. We envisage that our tool, at least in the near term, would be used to analyse existing protocols and systems based on them. This means we can focus on developing techniques to handle these existing protocols, and we feel that the **NP**-completeness of the model-checking procedure is not a major obstacle at this point. For the same reason, working within the stabilizer formalism is not also a significant limitation, as we propose to extend our techniques to verify nearly all protocols of current interest. For example, we should be able to model-check protocols which fall outside the stabilizer formalism efficiently, as long as the number of non-Clifford gates used in the particular protocol is small, and they are only applied to a limited number of qubits [12].

We intend to extend QMC in several ways. For instance, we intend to use a more expressive modelling language, such as the quantum process calculus CQP [22]. Computing the probability that a formula is satisfied, for all runs of a protocol, is a feature to be added. Also we are working on formalising the semantics of the input language and introducing several efficiency improvements and heuristics in the verification algorithms (as explained in Section 4.1). It will be useful to experiment with models of larger systems, combining many sub-protocols with both classical and quantum data. We also plan to develop metrics to characterise the performance of our tool on various case studies. We hope that QMC will develop into a useful and essential tool for designers of quantum communication and quantum cryptographic systems.

References

[1] Mayers, D.: Unconditional security in quantum cryptography. Journal of the ACM **48**(3) (2001) 351—406

[2] Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: Proceedings of International Conference on Computers, Systems and Signal Processing. (1984)
[3] Ekert, A.: Quantum cryptography based on Bell's theorem. Physical Review Letters **67**(6) (1991) 661—663
[4] Mayers, D.: Unconditionally secure quantum bit commitment is impossible. In: Fourth Workshop on Physics and Computation — PhysComp '96, Springer–Verlag (1996)
[5] Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, B.: Modelling and Analysis of Security Protocols. Pearson Education (2001)
[6] Holzmann, G.: The SPIN Model Checker: Primer and Reference Manual. Pearson Education (2003)
[7] Glendinning, I.: Links on simulation, modelling, and error prevention for quantum computers (2006) http://www.vcpc.univie.ac.at/ ian/hotlist/qc/simulation.shtml.
[8] Nagarajan, R., Gay, S.J.: Formal verification of quantum protocols. arXiv:quant-ph/0203086 (2002)
[9] Gay, S.J., Nagarajan, R., Papanikolaou, N.: Probabilistic model–checking of quantum protocols. DCM 2006: Proceedings of the 2nd International Workshop on Developments in Computational Models (2006) arXiv:quant-ph/0504007.
[10] Baltazar, P., Chadha, R., Mateus, P.: Quantum computation tree logic – model checking and complete calculus. International Journal of Quantum Information **6**(2) (2008) In press.
[11] Gottesman, D.: The Heisenberg representation of quantum computers. In Corney, S., Delbourgo, R., Jarvis, P., eds.: Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, International Press (1999)
[12] Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. Physical Review A **70**(52328) (2004)
[13] Anders, S., Briegel, H.: Fast simulation of stabilizer circuits using a graph state representation. Physical Review A **73**(22334) (2006)
[14] Audenaert, K., Plenio, M.: Entanglement on mixed stabiliser states: Normal forms and reduction procedures. New Journal of Physics **7**(170) (2005)
[15] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2000)
[16] Elliot, C.: Quantum cryptography. IEEE Security & Privacy Magazine **2**(4) (2004) 57—61
[17] Gay, S.J.: Quantum programming languages: Survey and bibliography. Mathematical Structures in Computer Science **16**(4) (2006)
[18] Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics. MIT Press (1990) 995–1072
[19] Mateus, P., Sernadas, A.: Weakly complete axiomatization of exogenous quantum propositional logic. Information and Computation **204**(5) (2006) 771—794
[20] Steane, A.M.: Quantum computing and error correction. In Gonis, A., Turchi, P., eds.: Proceedings of the NATO Advanced Research Workshop, Mykonos, IOS Press (2000) 284—298
[21] Bouwmeester, D., Ekert, A., Zeilinger, A., eds.: The Physics of Quantum Information. Springer–Verlag (2000)
[22] Gay, S.J., Nagarajan, R.: Communicating quantum processes. In: POPL '05: Proceedings of the 32nd ACM Symposium on Principles of Programming Languages, Long Beach, California. (2005)