QMC: A Model Checker For Quantum Systems

Nick Papanikolaou
`nikos@dcs.warwick.ac.uk`

Department of Computer Science
University of Warwick

Joint work with **Rajagopal Nagarajan** (Warwick)
and **Simon Gay** (Glasgow).

Tuesday 3rd July 2007
Warwick Postgraduate Colloquium in Computer Science
2-3 July 2007, Department of Computer Science, University of Warwick

Outline

Outline

1. Introduction
2. Methodology
3. The Stabiliser Formalism
4. The QMC Tool
5. Directions for Future Work and Review

Context

- Quantum communication and quantum cryptographic protocols are among the greatest successes of QIP research
  - QI protocols combine quantum and classical phenomena in a practical way
  - QI protocols do not require very sophisticated physical resources
  - QI protocols are implementable **today**
  - QC systems are already available
- Some considerations:
  - Quantum phenomena enable protocols with advantages over classical counterparts (e.g. unconditional security for QKD) and also protocols with no classical equivalent (e.g. teleportation)
  - Protocols tend to combine classical computations with quantum transmissions (e.g. BB84 + secret-key reconciliation, privacy amplification) and include quantum computations conditioned on classical measurements

Motivation

Key Point Design of classical communication and cryptographic protocols is a notoriously difficult task with known (and unknown) pitfalls.

- Analysis and verification of **classical protocols** and **systems** is an active and fruitful research area with important benefits
  - Discovery of flaw in Needham–Schröder Public Key Protocol (Lowe, 1996)
  - Pentium V, ARIANE, ...
- Increasing need for **design**, **simulation**, **analysis tools** for quantum communication and cryptographic protocols

Intended Contribution

- No dedicated tool currently exists for automated verification of *quantum* protocols and communication systems
- (Joint) research programme:
    - To develop a **verification framework** for analysing quantum protocols, esp. for reasoning about **quantum state**, **time**, and **knowledge**.
    - Approach: **Model–checking** (Clarke and Emerson, 1981; Quielle and Sifakis, 1981)

Raja

Simon

Nick

Paulo^{++}

History

- Application of verification techniques to quantum protocols initiated by Nagarajan and Gay **(2002)**
  - Modelled **BB84 protocol** for quantum cryptography in **CCS** and verified simple property using CWB tool.
- Extension of CCS model, first attempt at **PRISM** model by Papanikolaou **(2002-3)**
- Verification of core BB84 protocol using PRISM by Papanikolaou **(2004)**
- Development of CQP specification formalism by Gay, Nagarajan **(2004-5)**
- Verification of simple quantum protocols using PRISM by Gay, Nagarajan, Papanikolaou **(2005)**
- Development of QMC tool and extensions by Gay, Papanikolaou, Nagarajan, Mateus, Baltazar **(2005-present)**

Related Work

- Quantum Programming Languages
  - QCL (Ömer, 1998), QPL (Selinger 2003), ...
  - Quantum process algebras: QPA (Jorrand and Lalire, 2004), **CQP** (Gay and Nagarajan, 2004)
- Quantum Simulators
  - `QCL`, `jaQuzzi`, `QCSim`, `QuIDD`, ...
  - `CHP` (Aaronson and Gottesman, 2005)
- Logics for Quantum Information
  - Abramsky and Duncan, 2004
  - Baltag and Smets, 2004
  - Mateus and Sernadas, 2005+
  - Van Der Meyden and Patra, 2004

Outline

Formal Methods

Formal Methods is a branch of TCS which deals with the mathematical description (**specification**) of complex computing systems and comprises techniques for automated analysis and testing (**verification** or **validation**) of such systems.

Specification	is important for eliminating ambiguities from an informal system description; specification formalisms are designed so as to have well-defined semantics.
Verification	involves the use of specialised algorithms for checking whether a system specification satisfies any number of given properties, usually expressed in some formal logic (e.g. propositional logic, predicate logic, temporal logic, logic of knowledge, . . .)

A verification framework comprises a **modelling language** (for describing systems), a **property specification language or logic**, and an **algorithmic method** for comparing the two.

Automated Verification Techniques

Model–checking A system is first described using a **modelling language;** the variables in the model are used to describe important system states. **Properties** are expressed using some logic ranged over those variables. A **model-checking algorithm** checks whether the properties are satisfied in all the various states of the system. Model–checking tends to involve an **exhaustive search** over all possible system behaviours. Tools include `SPIN`, `SMV`, `FDR`, . . .

Automated Theorem Proving A system and its properties are described using a **formal logic** (typically predicate logic); the **inference rules** of the logic are built into **theorem-proving software**, which may be used to prove results about the system. The `HOL` theorem-prover is widely used.

Towards Verification of Quantum Protocols

For a verification technique to be developed, one must have an **adequate model** of the types of system to be analysed. For quantum protocols, an adequate model should account for:

- Quantum states*
- Unitary operators
- Measurements
- Classical bits and operations

Model We will model a QI protocol as a **finite, ordered set** of operators applied to a **finite, closed set** of pure quantum states.

Properties We will use the logic **EQPL** (Mateus and Sernadas, 2005) to express properties of quantum states arising in protocols.

Quantum States* We will restrict ourselves to protocols involving quantum states within the **stabiliser formalism** (Gottesman, 1997).

Outline

1. Introduction
2. Methodology
3. The Stabiliser Formalism
4. The QMC Tool
5. Directions for Future Work and Review

The Stabiliser Formalism (Gottesman, 1997)

- The operators in the Clifford group are those which arise in most simple quantum protocols.
- The **stabiliser formalism** allows us to capture the effect of these operators and of standard qubit measurement without looking at the actual quantum states.
- Circuits involving only stabiliser operations can be efficiently simulated on a classical computer (**Gottesman–Knill Theorem**).
- We have implemented a **polynomial-time algorithm** for simulating stabiliser circuits (Aaronson and Gottesman, 2004).
- These operators are **not universal**, not even for classical computing: the problem of simulating stabiliser circuits is **complete for the classical complexity class $\oplus L$ (parity-L).**

Outline

A Model Checking Tool for Quantum Protocols

- We have built a **dedicated model–checking tool**, QMC, for protocols which can be modelled within the stabiliser formalism.
- QMC has a high–level modelling language related to **CQP** (Gay and Nagarajan, 2005) and **LanQ** (Mlnarík, 2006).
- It allows model–checking of EQPL state formulas over stabiliser states.
- Stabiliser states are represented internally using a binary check matrix, denoting the generators of the corresponding stabiliser group.

Key Point QMC allows the user to simulate a stabiliser circuit. At each step of the simulation, a state formula can be checked.

Properties in QMC: EQPL formulae

Core Syntax for Classical Formulae:

$$\phi := \mathbf{q_k} \mid (\neg\phi) \mid (\phi \rightarrow \phi)$$

Core Syntax for Quantum Formulae:

$$\gamma := \phi \mid (t \leq t) \mid [S] \mid (\boxminus\gamma) \mid (\gamma \sqsupset \gamma)$$

Core Syntax for Terms:

$$\begin{aligned} t &:= r \mid (\textstyle\int \alpha) \mid (t + t) \mid (t \cdot t) \mid Re(u) \mid Im(u) \mid \ldots \\ u &:= z \mid |\top\rangle_{FA} \mid (t + it) \mid te^{it} \mid \ldots \end{aligned}$$

where t is a term, S a list of qubit constants. Note $[S]$ is true if the qubits in S are disentangled from the rest of the system.

Interpretation of EQPL Over Stabiliser Generators

Example

Consider quantum state $|\psi\rangle = \frac{1}{\sqrt{2}}(|001\rangle + |101\rangle)$. These formulae are true:

$$(\mathbf{q_0} \vee \mathbf{q_3}), \qquad (\textstyle\int(\mathbf{q_0}) \leq \frac{1}{2}), \qquad [\mathbf{q_0}]$$

- EQPL is defined over arbitrary pure states in $\mathcal{H}^{2^n}$.
- We have restricted our implementation of EQPL to stabiliser states.
- Formulae must be checked efficiently, without computing state vector representation if possible.
  - This computation has worst-case complexity $O(2^n)$
- Most formulae seem to require this computation (!) but some optimisations are possible.

Model–checking algorithms

QMC has two main modes of operation:

Simulation mode EQPL formulae are checked on an individual quantum state arising during simulation of a quantum protocol.

Model–checking mode A protocol is simulated several times, each time with a different measurement outcome. QMC automatically computes all possible measurement outcomes, producing a different protocol run in each case. An EQPL formula is checked on the final quantum state **for all runs**.

Simulation of protocols is efficient: QMC implements a polynomial time algorithm for simulation of stabiliser circuits due to Aaronson and Gottesman (2005).

Implementation of temporal EQPL will involve developing extensions of classical CTL model–checking algorithms.

Outline

Goals for Future Work

1. to overcome **efficiency limitations** within current approach
2. to implement **temporal extension of EQPL**!
   - need to consider mixed states - redefinition of EQPL in terms of **density operators**
3. to formalise semantics of the modelling language; also to consider concurrency
4. to consider going **outside stabiliser formalism**
5. Proof system for the logic
6. SAT algorithm and complexity analysis for the logic

Collaboration

We have started a joint Warwick–Glasgow–Lisbon collaboration working towards these goals. (P. Baltazar, S. Gay, P. Mateus, R. Nagarajan, N. Papanikolaou, A. Sernadas)

Review and Conclusion

- We have presented an overview of the QMC model-checking tool for quantum protocols.
- The background and motivation for our automated verification techniques have been discussed.
- The use of the quantum stabiliser formalism for representing and simulating a selected class of protocols has been detailed.
- We have also covered the EQPL logic and aspects of its implementation.

Thanks for listening!