Formalization of quantum protocols using Coq

Jaap Boender    Rajagopal Nagarajan    Florian Kammüller

July 17, 2015
Motivation for Interactive Theorem Proving (ITP)

- Automated Reasoning
- High Expressivity
- Trade off to automation
- Good applications: complex models, tedious reasoning, and high risk of faults (and impact of failures) in details
Coq

- Constructive type theory as logical basis:
  - For example, $A \lor \neg A$ is not a theorem!
  - Proof is construction: executable code (OCAML) can be extracted
  - Higher level of expressivity: dependent types
- Code-Extraction interesting for prototypes
Classical Reasoning and Curry Howard Paradigm

- Curry Howard paradigm in Coq
  - Proofs as *terms* and propositions as *types*
  - E.g. \( \lambda x.x : P \Rightarrow P \)
  - E.g. \( \text{inl} : A \Rightarrow A \vee B \)
  - Proof checking \( \equiv \) type checking
  - Automated proof \( \equiv \) type inference
Formalisation in Coq

- Needs complex numbers and matrices
- When we started, no library provided both
Formalisation in Coq

- Needs complex numbers and matrices
- When we started, no library provided both
- Selected CoRN
  - Complex numbers
  - Fast arithmetic
  - Matrices implementable with typeclasses
Formalisation in Coq, part II

- CoRN not the ideal solution
  - No real development recently
  - Little documentation
  - Constructive
Formalisation in Coq, part II

- CoRN not the ideal solution
  - No real development recently
  - Little documentation
  - Constructive
- Now switching to Ssreflect
Qubits and Gates in Coq

- **Definition qubit (n:nat) :=**
  
  \{ v:vector (2^n) \mid \text{length } v \;[=] \;[1] \} 

- **Definition gate (n:nat) :=**
  
  \{ m:\text{matrix} \; (2^n) \; (2^n) \mid \text{unitary } m \} 

- **Function apply (n:nat):**
  
  (qubit n) \rightarrow (gate n) \rightarrow (qubit n)

- **Apply needs to construct proof that resulting qubit is a qubit**
The coin flipping game

The normal version:

- One coin (initially heads), two players
- Three turns (Q, then P, then Q)
- Heads: P wins, tails: Q wins
- Each player can either flip the coin or not
- No one can see the coin
- Therefore, no winning strategy
The QUANTUM coin flipping game

The QUANTUM version:

- One QUANTUM coin (initially $|1\rangle$), two players
- Three turns (Q, then P, then Q)
- $|0\rangle$: P wins, $|1\rangle$: Q wins
- Each player can either flip the coin or not
- Q can additionally apply the Hadamard gate
- No one can see the QUANTUM coin
- Now, Q has a winning strategy
Protocol example: coin flipping

\textbf{Inductive} \texttt{Pchoice}: \texttt{Set := N: Pchoice} | \texttt{X: Pchoice}.
\textbf{Inductive} \texttt{Qchoice}: \texttt{Set := Pch: Pchoice \rightarrow Qchoice}
| \texttt{H: Qchoice}.
\textbf{Inductive} \texttt{game}: \texttt{Set := Game:}
\texttt{Qchoice \rightarrow Pchoice \rightarrow Qchoice \rightarrow game}.
\textbf{Function} \texttt{play}: \texttt{game \rightarrow qubit 1}.
\textbf{Definition} \texttt{Qwins (g: game) :=}
\texttt{play g} \{=\} (\texttt{base_q 1}).
\textbf{Theorem} \texttt{winning: exists q q': Qchoice,}
\texttt{forall p: Pchoice, Qwins (Game q p q')}.
Entanglement in Coq

- Definition: state cannot be expressed as tensor product of smaller states
- Proving non-existence of something constructively is hard!
- Alternative definition by probabilities
- Qubit is entangled if measuring one bit affects probabilities of other bits
- Prove equivalence of two notions (hard?)
Definition \texttt{entangled}\_\texttt{tp} \{n\} (q: \texttt{qubit n}) :=
\sim \exists m (q1: \texttt{qubit m}) (q2: \texttt{qubit (n-m)}),
\texttt{out\_matrix} q1 \{o\} \texttt{out\_matrix} q2 \{==\} \texttt{out\_matrix} q.

Definition \texttt{entangled}\_\texttt{p} \{n\} (q: \texttt{qubit n})
(p1: \texttt{nat | p1 < n}) (p2: \texttt{nat | p2 < n}) :=
\forall v, \exists pr, \exists res,
\texttt{List.In} (pr, res)
(outcome\_evaluation [1] q (measure p1 empty empty))
\land \texttt{probability} q p2 v [\sim=] \texttt{probability} res p2 v.

Definition \texttt{entangled} \{n\} (q: \texttt{qubit n}) :=
\exists p1 p2, (\langle p1 \rangle \neq \langle p2 \rangle) \land \texttt{entangled}\_\texttt{p} q p1 p2.
Measurement

- Here we run into a problem with CoRN
- Measuring uses division
- Constructive: need to prove that we’re not dividing by zero
- This is not necessarily true
• Here we run into a problem with CoRN
• Measuring uses division
• Constructive: need to prove that we’re not dividing by zero
• This not necessarily true

Axiom sum_pair1:
\[
\forall \{n\} (i : \text{nat} \mid i < n) (q : \text{qubit} \ n),
\]
\[
\text{fst} (\text{sum\_pair} ('i) q) \equiv [0] \text{ or } [0] < \text{fst} (\text{sum\_pair} ('i) q).
\]
Program Definition measure \{n\} (i: nat \mid i < n) 
(q: qubit n): list (IR * qubit n) := 
match sum_pair1 i q with 
| inl _ => (* zero *) \[[[1], q]\]%list 
| inr sum0_gt => 
  match sum_pair2 i q with 
  | inl _ => (* zero *) \[[[1], q]\]%list 
  | inr sum1_gt => \[(fst (sum_pair i q), existT _ (nqv (‘i) negb (fst (sum_pair i q)) sum0_gt q)) _); 
    (snd (sum_pair i q), existT _ (nqv (‘i) (fun x => x) (snd (sum_pair i q)) sum1_gt q)) _)\]%list 
  end 
end 
end.
Quantum teleportation: Alice

Definition firstgate: (gate 3) :=
c_not_gate \{o\} identity.
Definition sndgate: (gate 3) :=
hadamard \{o\} identity \{o\} identity.
Definition Alice_spoor: (spoor 3) :=
transform firstgate (transform sndgate empty).
Definition Alice (p1: nat \mid p1<3) (p2: nat \mid p2<3)
(phi: qubit 1): list (IR * qubit 3) :=
outcome_evaluation [1] (comp3 phi)
(measure p1 (measure p2 Alice_spoor empty) empty)
Quantum teleportation: Bob

**Definition** Bob (psix: qubit 3) (x y: bool): qubit 2 :=

```plaintext
match x, y with
| false,false => apply psix
  (identity {o} identity {o} identity {o} identity)
| false,true => apply psix
  (identity {o} identity {o} x_gate)
| true,false => apply psix
  (identity {o} identity {o} z_gate)
| true,true => apply psix
  (identity {o} identity {o} y_gate)
end.
```
Future work

- Convert development to Ssreflect
- Think about representing processes
- Properly do quantum teleportation
Quantum teleportation: protocol

- **Coq function Alice**
  - joins input qubit \( \phi \) with entangled pair \( \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \)
  - applies to resulting qubit triplet two gates \( c_{\text{not}} \) _and_ hadamard _identity_ _identity_
  - Sends classical bits 00, 01, 10, or 11 depending on results of measuring first two qubits
- Depending on received pair of classical bits, Coq function Bob applies I, X, Z, or Y

<table>
<thead>
<tr>
<th>((x, y))</th>
<th>Bob’s action</th>
<th>restored</th>
</tr>
</thead>
<tbody>
<tr>
<td>((0, 0))</td>
<td>I((a</td>
<td>0\rangle + b</td>
</tr>
<tr>
<td>((0, 1))</td>
<td>X((a</td>
<td>1\rangle + b</td>
</tr>
<tr>
<td>((1, 0))</td>
<td>Z((a</td>
<td>0\rangle - b</td>
</tr>
<tr>
<td>((1, 1))</td>
<td>Y((a</td>
<td>1\rangle - b</td>
</tr>
</tbody>
</table>

- We can prove that Bob’s after Alice’s function preserve \( \phi \) – i.e., “teleport” it from first two third position in the triple

**Theorem** teleportation:

\[
\text{forall } \phi: \text{qubit } 1, \text{ exists } z: \text{qubit } 2, \quad \text{Bob}(\text{Alice}(\phi)) = (z \text{ } \text{identity} \text{ } \phi).
\]