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, $A \vee \neg A$ 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. $\mathrm{inl} : A \Rightarrow A \vee B$
  - 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) | length v [=] [1] }`
- `Definition gate (n:nat) :=`
  `{ m:matrix (2^n) (2^n) | unitary m }`
- `Function apply (n:nat):`
  `(qubit n) -> (gate n) -> (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

Inductive `Pchoice: Set := N: Pchoice | X: Pchoice.`
Inductive `Qchoice: Set := Pch: Pchoice -> Qchoice`
`| H: Qchoice.`
Inductive `game: Set := Game:`
`Qchoice -> Pchoice -> Qchoice -> game.`
Function `play: game -> qubit 1.`
Definition `Qwins (g: game) :=`
`play g {=} (base_q 1).`
Theorem `winning:` **exists** `q q': Qchoice,`
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?)

Entanglement

```
Definition entangled_tp {n} (q: qubit n) :=
 ~exists m (q1: qubit m) (q2: qubit (n-m)),
  out_matrix q1 {o} out_matrix q2 {==} out_matrix
q.

Definition entangled_p {n} (q: qubit n)
 (p1: nat | p1 < n) (p2: nat | p2 < n) :=
  forall v, exists pr, exists res,
   List.In (pr, res)
    (outcome_evaluation [1] q (measure p1 empty
empty))
   ∧ probability q p2 v [~=] probability res p2
v.

Definition entangled {n} (q: qubit n) :=
 exists p1 p2, (`p1) <> (`p2) ∧ entangled_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 not necessarily true

Measurement

- 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: nat | i < n) (q: qubit n),
  fst (sum_pair (`i) q) [=] [0] or
  [0] [<] fst (sum_pair (`i) q).
```

Measurement part II

```
Program Definition measure {n} (i: nat | 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.
```

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 | p1<3) (p2: nat | 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
:=
 match x, y with
 | false,false => apply psix
  (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_not_gate {o} identity` and `hadamard {o} identity {o} 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

(x, y)	Bob′s action		restored
$(0,0)$	$I(a\,|0\rangle + b\,|1\rangle)$	$=$	$a\,|0\rangle + b\,|1\rangle$
$(0,1)$	$X(a\,|1\rangle + b\,|0\rangle)$	$=$	$a\,|0\rangle + b\,|1\rangle$
$(1,0)$	$Z(a\,|0\rangle - b\,|1\rangle)$	$=$	$a\,|0\rangle + b\,|1\rangle$
$(1,1)$	$Y(a\,|1\rangle - b\,|0\rangle)$	$=$	$a\,|0\rangle + b\,|1\rangle$

- 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:`

```
 forall phi: qubit 1, exists z: qubit 2,
  Bob (Alice phi) {=} (z {o} phi).
```