Quantum

Marco Lewis

August 11, 2021

Contents

1 CNOT in QHLProver **1**
1.1 Environment Settings 1
1.2 State Vectors and Properties 2
1.3 Precondition and Postcondition 2
1.4 CNOT Gate and Properties 3
1.5 Deduction 5
1.5.1 Sanity Check with Identity 5
1.5.2 CNOT Correctness Deductions 5

1 CNOT in QHLProver

theory *CNOT*
 imports *QHLProver.Gates*
 QHLProver.Partial-State
 QHLProver.Quantum-Hoare
 QHLProver.Quantum-Program
 QHLProver.Grover
 QHLProver.Complex-Matrix
begin

1.1 Environment Settings

locale *cnot-state* = *state-sig* +
 fixes *n* :: *nat*
 assumes *n*: *n* = (*2*::*nat*)
 assumes *dims-def*: *dims* = [*2*,*2*]
begin

definition *N* **where**
 N = (*2*::*nat*) ^ *n*

lemma *N-4*:
 N == *4* **using** *N-def n* **by** *auto*

lemma *d-4*:
  d == 4
  using *d-def dims-def* **by** *auto*

lemma *N-d*:
  N == d
  using *d-4 N-4* **by** *auto*

1.2 State Vectors and Properties

abbreviation *proj* :: *complex vec* ⇒ *complex mat* **where**
  proj v ≡ *outer-prod v v*

definition *ket-10* :: *complex vec* **where**
  ket-10 = *Matrix.vec N* (λ*i* . *if i* = *2 then 1 else 0*)

lemma *ket-10-dim* [*simp*]:
  ket-10 ∈ *carrier-vec N*
  dim-vec ket-10 = *N*
  by (*simp add*: *ket-10-def*)+

lemma *ket-10-eval*:
  i < *N* ⟹ *ket-10* $ *i* = (*if i* = *2 then 1 else 0*)
  by (*simp add*: *ket-10-def*)

definition *ket-11* :: *complex vec* **where**
  ket-11 = *Matrix.vec N* (λ*i*. *if i* = *3 then 1 else 0*)

lemma *ket-11-dim* [*simp*]:
  ket-11 ∈ *carrier-vec N*
  dim-vec ket-11 = *N*
  by (*simp add*: *ket-11-def*)+

lemma *ket-11-eval*:
  i < *N* ⟹ *ket-11* $ *i* = (*if i* = *3 then 1 else 0*)
  by (*simp add*: *ket-11-def*)

1.3 Precondition and Postcondition

definition *proj-10* **where**
  proj-10 = *proj ket-10*

lemma *norm-pre*:
  inner-prod ket-10 ket-10 = *1*
  unfolding *ket-10-def*
  apply (*simp add*: *scalar-prod-def*)
  using *sum-only-one-neq-0*[*of* {*0..*<*N*} *2* λ*i*. (*if i* = *2 then 1 else 0*) ∗ *cnj* (*if i* = *2 then 1 else 0*)] *N-def*
  n **by** *auto*

lemma *qp-pre*:
is-quantum-predicate proj-10
unfolding *is-quantum-predicate-def*
proof (*intro conjI*)
show *proj-10* $\in$ *carrier-mat d d* **using** *proj-10-def ket-10-def N-d* **by** *auto*
show *positive proj-10*
using *positive-same-outer-prod*
unfolding *proj-10-def ket-10-def*
by *auto*
show *proj-10* $\leq_L$ 1_m *d*
unfolding *proj-10-def*
using *norm-pre N-d outer-prod-le-one ket-10-def*
by *auto*
qed

definition *proj-11* **where**
proj-11 = proj ket-11

lemma *norm-post*:
inner-prod ket-11 ket-11 = 1
unfolding *ket-11-def*
apply (*simp add*: *scalar-prod-def*)
using *sum-only-one-neq-0*[*of* {*0..*<*N*} *3* λ*i*. (*if i = 3 then 1 else 0*) ∗ *cnj* (*if i = 3 then 1 else 0*)] *N-def*
n **by** *auto*

lemma *qp-post*:
is-quantum-predicate proj-11
unfolding *is-quantum-predicate-def*
proof (*intro conjI*)
show *proj-11* $\in$ *carrier-mat d d* **using** *proj-11-def ket-11-def N-d* **by** *auto*
show *positive proj-11*
using *positive-same-outer-prod*
unfolding *proj-11-def ket-11-def*
by *auto*
show *proj-11* $\leq_L$ 1_m *d*
unfolding *proj-11-def*
using *norm-post N-d outer-prod-le-one ket-11-def*
by *auto*
qed

1.4 CNOT Gate and Properties

definition *cnot* :: *complex mat* **where**
cnot = mat N N (λ(*i*,*j*).
if i=j then
(*if* (*i=0* ∨ *i=1*) *then 1 else 0*)
else (*if* (*i=2* ∧ *j=3*) ∨ (*i=3* ∧ *j=2*)
then 1 else 0))

```
lemma cnot-dim:
   cnot ∈ carrier-mat N N
  unfolding cnot-def by auto

lemma hermitian-cnot:
  hermitian cnot
  by (auto simp add: hermitian-def cnot-def adjoint-eval)

lemma cnot-cnot-eq-id:
  shows cnot *  cnot = 1m N
  apply(rule eq-matI,simp)
  apply(auto simp add: carrier-matD[OF cnot-dim] scalar-prod-def)
  unfolding cnot-def
  apply(simp-all)
  apply(case-tac j = 3)
  subgoal for j
    unfolding N-4
    by (simp add: sum-only-one-neq-0[of - 2])
  apply (case-tac j = 2)
  subgoal for j
    unfolding N-4
    by (simp add: sum-only-one-neq-0[of - 3])
  apply (case-tac j = 1)
  subgoal for j
    unfolding N-4
    by (simp add: sum-only-one-neq-0[of - 1])
  subgoal for j
    unfolding N-4
    by (simp add: sum-only-one-neq-0[of - 0])
  done

lemma unitary-cnot:
  unitary cnot
proof −
  have cnot ∈ carrier-mat N N
    using cnot-dim by auto
  moreover have cnot * adjoint cnot = cnot * cnot
    using hermitian-cnot unfolding hermitian-def by auto
  moreover have cnot * cnot = 1m N
    using cnot-cnot-eq-id by auto
  ultimately show ?thesis
    unfolding unitary-def inverts-mat-def by auto
qed

definition cnot-circ :: com where
  cnot-circ = Utrans cnot

lemma well-com-cnot:
```

```
  well-com cnot-circ
  unfolding cnot-circ-def
  using unitary-cnot cnot-dim N-4 d-4
  by auto
```

1.5 Deduction

1.5.1 Sanity Check with Identity

```
definition id :: complex mat where
  id = mat N N (λ(i,j). if i=j then 1 else 0)

lemma id-times-11:
  id *_v ket-11 = ket-11
  by (auto simp add: id-def ket-11-def
    scalar-prod-def sum-only-one-neq-0)

thm scalar-prod-def
thm sum-only-one-neq-0[of - 1 λi. i+ 5]
```

1.5.2 CNOT Correctness Deductions

```
lemma cnot-times-11:
  cnot *_v ket-11 = ket-10
  apply (rule eq-vecI, simp)
  apply(auto simp add: carrier-matD[OF cnot-dim] scalar-prod-def)
  unfolding cnot-def ket-11-def ket-10-def N-4 scalar-prod-def
    by (simp add: sum-only-one-neq-0[of - 3])



lemma cnot-times-post-is-pre:
  adjoint cnot * proj-11 * cnot = proj-10
proof −
  let ?m = cnot
  have eq: adjoint ?m = ?m
    using hermitian-def hermitian-cnot by auto
  {
    let ?p = proj-11
    have ?m * ?p * ?m = outer-prod ket-10 ket-10
      unfolding proj-11-def
      using ket-11-def cnot-dim cnot-times-11 eq
      apply (subst outer-prod-left-right-mat[of - N - N  - N - N])
      by auto
  }
  note p = this
  have adjoint cnot * proj-11 * cnot = ?m * proj-11 *?m
    using eq by auto
  also have ... = proj-10
    unfolding proj-10-def
```

using *p*
by *auto*
finally show *?thesis* **by** *auto*
qed

lemma *adjoint-deduction*:
$\vdash_p$
$\{$*adjoint cnot* $*$ *proj-11* $*$ *cnot*$\}$
cnot-circ
$\{$*proj-11*$\}$
unfolding *cnot-circ-def* **using**
hoare-partial.intros(2)
unitary-cnot qp-post
by *auto*

lemma *prog-partial-deduct*:
$\vdash_p$
$\{$*proj-10*$\}$
cnot-circ
$\{$*proj-11*$\}$
using *cnot-times-post-is-pre adjoint-deduction* **by** *auto*

theorem *prog-partial-correct*:
$\models_p$
$\{$*proj-10*$\}$
cnot-circ
$\{$*proj-11*$\}$
using
prog-partial-deduct
well-com-cnot qp-pre qp-post
hoare-partial-sound **by** *auto*

end

end