Use ACL2 to Prove the Correctness of Clebsch-Gordan Coefficient

Xiaohui Chen
04/20/2015

Timeline

- 04/13/2015-04/19/2015
- Finish coding and finish verify the theorem
- 04/20/2015-04/26/2015
- Finish proving the main theorem
- Finish project report (first draft) and presentation slides
- 04/26/2015-05/03/2015
- Finish presentation and modify project report
- Finish final draft of project report by 05/06/2015

My model

- ((A . (j . m_j)) .
- ((B . ((l . m_l) . (s . m_s)))
- (C. ((l . m_l) . (s . m_s)))
- ….))

- We have to verify that after a series of operations on the above pair, A=B+C+D+...

Progress

- Finished proving the lemmas. Beginning proving the main theorem
- Finished optimizing some of the lemmas
- Some of them still have a lot of splitting, but way better
- The proving time is endurable

```
1411 (defthm append-valid
1412         (implies (and (true-coupled-state x)
1413                       (true-coupled-state y))
1414                  (true-coupled-state (append-and-merge-states x y))))
1415 :hints (("Goal" :in-theory (disable same-denominator-add
1416                                     remove-strict-inequalities
1417                                     remove-weak-inequalities
1418                                     default-less-than-1
1419                                     default-less-than-2
1420                                     default-plus-1
1421                                     default-rational-denominator
1422                                     |(+ x (if a b c))|
1423                                     |(- (if a b c))|
1424                                     |(< (if a b c) x)|
1425                                     |(< x (if a b c))|
1426                                     DEFAULT-MINUS
1427                                     DEFAULT-PLUS-2
1428                                   calculate-merge-coefficient
1429                                   REDUCE-MULTIPLICATIVE-CONSTANT-EQUAL))))
```

This was the theorem which took more than 80,000,000 steps and 15 minutes

- (defthm append-valid
- (implies (and (true-coupled-state x)
- (true-coupled-state y))
- (true-coupled-state (append-and-merge-states x y)))
- :hints (("Goal" :in-theory (disable same-denominator-add
- remove-strict-inequalities
- remove-weak-inequalities
- merge-same-property-4))))
- This theorem takes more than 84,000,000 steps, 983 seconds

NOW

```
Warnings:  Non-rec
Time:  0.77 seconds (prove: 0.75, print: 0.01, other: 0.01)
Prover steps counted:  39227
 APPEND-VALID
ACL2 !>> I-AM-HERE
 (:STOP-LD 2)
ACL2 !>
```

- Inspect the if-intro and disable all the splits
- Gradually enable back the if-intros
- Use lemmas that could minimize the splits
- Sometimes proof-checker is more convenient

```
1310
1311 (DEFTHM APPEND-AND-MERGE-PROPERTY-4
1312         (IMPLIES (AND (CONSP X)
1313                       (TRUE-COUPLED-LIST X)
1314                       (CONSP Y)
1315                       (TRUE-COUPLED-LIST Y))
1316                  (TRUE-COUPLED-LIST (APPEND-AND-MERGE-STATES-HELPER X Y)))
1317         :INSTRUCTIONS ((:INDUCT (APPEND-AND-MERGE-STATES-HELPER X Y))
1318                        :PROMOTE :SPLIT (:DV 1)
1319                        :EXPAND :S-PROP :TOP
1320                        (:REWRITE APPEND-AND-MERGE-PROPERTY-2)
1321                        (:USE APPEND-AND-MERGE-PROPERTY-1)
1322                        :PROMOTE (:FORWARDCHAIN 1)
1323                        (:DV 1)
1324                        :EXPAND :S-PROP :EXPAND
1325                        :S-PROP :TOP :SPLIT :EXPAND :S-PROP
1326                        (:REWRITE DELETE-SAME-PROPERTY-1)
1327                        (:REWRITE MERGE-SAME-PROPERTY)
1328                        (:REWRITE APPEND-AND-MERGE-PROPERTY-1)
1329                        (:DV 1)
1330                        :EXPAND :S-PROP :TOP
1331                        (:REWRITE APPEND-AND-MERGE-PROPERTY-2)
1332                        (:REWRITE MERGE-SAME-PROPERTY)
1333                        (:REWRITE APPEND-AND-MERGE-PROPERTY-1)
1334                        (:DEMOTE 4)
1335                        (:DV 1)
1336                        :EXPAND :S-PROP :TOP :S-PROP (:DV 1)
1337                        :EXPAND :S-PROP (:DV 2 2)
1338                        (:REWRITE DELETE-SAME-PROPERTY-2)
1339                        :UP :EXPAND :TOP :SPLIT
1340                        (:REWRITE APPEND-AND-MERGE-PROPERTY-2)
1341                        :EXPAND
1342                        :S-PROP (:REWRITE MERGE-SAME-PROPERTY)
1343                        (:REWRITE APPEND-AND-MERGE-PROPERTY-1)
1344                        (:REWRITE APPEND-AND-MERGE-PROPERTY-2)
1345                        (:REWRITE APPEND-AND-MERGE-PROPERTY-3)
1346                        (:REWRITE MERGE-SAME-PROPERTY)
1347                        (:REWRITE APPEND-AND-MERGE-PROPERTY-1)
1348                        :EXPAND :S-PROP (:DV 1)
1349                        :EXPAND :S-PROP (:DV 2)
```

```
1350                          :EXPAND :S-PROP :TOP :SPLIT
1351                          (:REWRITE APPEND-AND-MERGE-PROPERTY-2)
1352                          :EXPAND
1353                          :S-PROP (:REWRITE MERGE-SAME-PROPERTY)
1354                          (:REWRITE APPEND-AND-MERGE-PROPERTY-1)
1355                          (:REWRITE APPEND-AND-MERGE-PROPERTY-2)
1356                          :EXPAND
1357                          :S-PROP (:REWRITE MERGE-SAME-PROPERTY)
1358                          (:REWRITE APPEND-AND-MERGE-PROPERTY-1)))
```

Result

```
ACL2 !>(quantum-operator '((1 . (3/2 . 3/2)) . ((1 . ((1 . 1) . (1/2 .
1/2))))))
((3 3/2 . 1/2)
 (2 (1 . 0) 1/2 . 1/2)
 (1 (1 . 1) 1/2 . -1/2))
ACL2 !>(quantum-operator (quantum-operator '((1 . (3/2 . 3/2)) . ((1 .
((1 . 1) . (1/2 . 1/2)))))))
((4 3/2 . -1/2)
 (4/3 (1 . -1) 1/2 . 1/2)
 (8/3 (1 . 0) 1/2 . -1/2)
 . 0)
ACL2 !>(quantum-operator (quantum-operator (quantum-operator '((1 . (3/
2 . 3/2)) . ((1 . ((1 . 1) . (1/2 . 1/2)))))))
((3 3/2 . -3/2)
 (3 (1 . -1) 1/2 . -1/2)
 . 0)
ACL2 !>(quantum-operator (quantum-operator (quantum-operator (quantum-o
perator '((1 . (3/2 . 3/2)) . ((1 . ((1 . 1) . (1/2 . 1/2)))))))))
(0 . 0)
ACL2 !>
```

Prove

Given an initial state, all the outputs subsequent operations
either have A=B+C+D... or
equal to (0 . 0) for the operations on the lowest states

This ought to be done around Wednesday and Thursday!!!!