```
////////////////////////////////////////////////////////////////
// iteration over Sp(3,R) -> U(3) subspaces & states
////////////////////////////////////////////////////////////////

// Traversal schemes
//
// To traverse Sp(3,R) -> U(3) subspaces:
//
//   for each LGI (in lgi_vector)
//     follow reference to Sp3RSpace
//     for each U3Subspace in Sp3RSpace
//
// To traverse Sp(3,R) -> U(3) states:
//
//   for each LGI (in lgi_vector)
//     follow reference to Sp3RSpace
//     for each U3Subspace in Sp3RSpace
//       for each state index
//
// To traverse Sp(3,R) -> U(3) -> (L,S) states:
//
//   A triangularity constraint may be placed on L to restrict only
//   to those L contributing to a desired final J.
//
//   for each LGI (in lgi_vector)
//     follow reference to Sp3RSpace
//     for each U3Subspace in Sp3RSpace
//       for each state index
//         for each L  (optionally subject to triangularity constraint for J)
//
// To traverse Sp(3,R) -> U(3) -> (L,S) -> J states:
//
//   A triangularity constraint may be placed on L to restrict only
//   to those L contributing to a desired final J.
//
//   for each LGI (in lgi_vector)
//     follow reference to Sp3RSpace
//     for each U3Subspace in Sp3RSpace
//       for each state index
//         for each L
//           for each J

// total dimension counting functions

int TotalU3Subspaces(const LGIVectorType& lgi_vector);
// Compute total number of U(3) subspaces from given LGI set.
//
// That is, we are counting the number of Sp(3,R) -> U(3) subspaces
// in the branching, as defined under "Traversal schemes" above, not
// just the number of distinct final U(3) labels.

int TotalDimensionU3(const LGIVectorType& lgi_vector);
// Compute total number of U(3)xS reduced states from the given LGI
// set.

int TotalDimensionU3LS(const LGIVectorType& lgi_vector);
// Compute total number of LxS branched states from the given LGI
// set.

int TotalDimensionU3LSJConstrained(const LGIVectorType& lgi_vector, const HalfInt& J);
// Compute total number of LxS->J branched states of a given J, from
// the given LGI set.
//
// From the point of view of counting and enumerating, this is more
// simply the number of LxS branched states contributing to the
// given given J, from the given LGI set.

int TotalDimensionU3LSJAll(const LGIVectorType& lgi_vector);
// Compute total number of LxS->J branched states, over all J, from
// the given LGI set.
//
// This is equivalent to the M-space dimension for the universal
// donor value of M (M=0 or 1/2).




int TotalDimensionU3LS(const LGIVectorType& lgi_vector)
{
  int dimension = 0;
  for (auto it=lgi_vector.begin(); it !=lgi_vector.end(); ++it)
    {
      const spncci::LGI& lgi = *it;
      const sp3r::Sp3RSpace& irrep = lgi.Sp3RSpace();
      for (int i_subspace=0; i_subspace < irrep.size(); ++i_subspace)
        {
          const sp3r::U3Subspace& u3_subspace = irrep.GetSubspace(i_subspace);
```

```
          // L branching of each irrep in subspace
          //
          // branching_vector is list of L values tagged with their
          // multiplicity.
          u3::U3 omega = u3_subspace.U3();
          MultiplicityTagged<int>::vector branching_vector = BranchingSO3(omega.SU3());
          int L_dimension = 0;
          for (int i_L=0; i_L<branching_vector.size(); ++i_L)
            L_dimension += branching_vector[i_L].tag;

          // dimension contribution of subspace
          //
          // At LS-coupled level, the dimension contribution is the
          // product of the number of U(3) reduced states and their
          // L substates.
          dimension += u3_subspace.size()*L_dimension;
        }
    }

  return dimension;
}
```