Algorithm 1 AddAtomAndConstructChildOctants

Require: maxDepth, the octree's maximum allowable depth
Require: currentTreeLevel, the octant's current level in the octree
Require: minBoxCoord, the octant's minimum box coordinate
Require: maxBoxCoord, the octant's maximum box coordinate
Require: atomCoord, the coordinates of the atom's centre
Require: atom, the atom object to add
Ensure: filledOctree

```
1: if isOctantNew = true then
2:    octantMinBoxCoord ← minBoxCoord
3:    octantMaxBoxCoord ← maxBoxCoord
4:    octantCentroid ← (minBoxCoord+maxBoxCoord)*0.5
5:    octantRadius ← (minBoxCoord-maxBoxCoord)*0.5
6:    isOctantNew ← false
7: end if
8: childrenTreeLevel ← currentTreeLevel+1
9: // reached a tree leaf node, thus assign to this octant the given atom
10: if childrenTreeLevel > maxDepth then
11:    octantsAtomList.AddEnd(atom)
12:    isOctantNew ← true
13:    return atom added
14: else
15:    for RP = 1 to 8 do
16:       childMinBoxCoord ← get the min box coordinates for child octant
          childOctant<RP>
17:       childMaxBoxCoord ← get the max box coordinates for child octant
          childOctant<RP>
18:       if atomCoord intersect/being enclosed by childMinBoxCoord and child-
          MaxBoxCoord then
19:          if childOctant<RP> has not been created then
20:             childOctant<RP> ← create new Octant Node
21:             isOctantALeaf ← false
22:             // add this child at the end of the octant's children list
23:             octantsChildrenList.AddEnd(childOctant<RP>)
24:          end if
25:          // forward/add the atom to childOctant<RP> octants recursively
26:          return childOctant<RP>.AddAtomAndConstructChildOctants(maxDepth,
             childrenTreeLevel, childMinBoxCoord, childMaxBoxCoord, atom-
             Coord, atom)
27:       end if
28:    end for
29: end if
30: end
```

Algorithm 2 DeriveInteractingAtomPairsSet

Require: T_{New}, the combined viewing transformation matrix
Require: octree1Octant, an octant from the first octree structure
Require: octree2Octant, an octant from the second octree structure
Require: cutoff, the cut-off distance
Ensure: S_{Pairs}

```
1: retValue ← false
2: if both octree1Octant AND octree2Octant are leaf-octants then
3:    for all atoms a_r in octree1Octant and all atoms a_l in octree2Octant do
4:       d_atoms ← compute inter-atomic distance between a_r and a_l
5:       if d_atoms ≤ cutoff then
6:          save pair (a_r, a_l) in S_Pairs
7:          retValue ← true
8:       end if
9:    end for
10: else
11:    if octree1Octant OR octree2Octant is a leaf-octant then
12:       // set non-leaf octant to tmpNLOctant and leaf octant to tmpOctant
13:       if octree1Octant is a leaf-octant then
14:          tmpNLOctant ← octree2Octant
15:          tmpOctant ← octree1Octant
16:       else
17:          tmpNLOctant ← octree1Octant
18:          tmpOctant ← octree2Octant
19:       end if
20:       for all child octants oct_c in tmpNLOctant do
21:          d_Net ← compute net distance between oct_c and tmpOctant
22:          if d_Net ≤ (cutoff+e_s) then
23:             if DeriveInteractingAtomPairsSet(T_New, oct_c, tmpOctant, cutoff)
                then
24:                retValue ← true
25:             end if
26:          end if
27:       end for
28:    else
29:       for all child octants oct_r in octree1Octant and all child octants oct_l in
          octree2Octant do
30:          update necessary octant coords. (i.e. the shortest octree) with T_New
31:          d_Net ← compute net distance between oct_r and oct_l
32:          if d_Net ≤ (cutoff+e_s) then
33:             if DeriveInteractingAtomPairsSet(T_New, oct_r, oct_l, cutoff) then
34:                retValue ← true
35:             end if
36:          end if
37:       end for
38:    end if
39: end if
40: return retValue
41: end
```