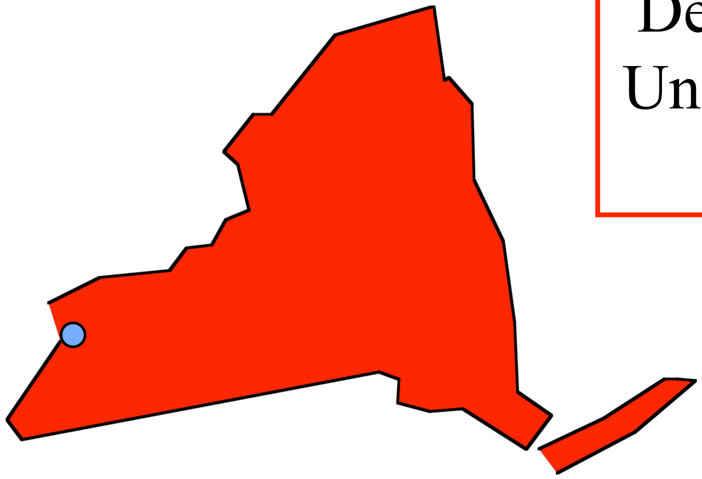
All About Species, Atoms, and Iterators

David A. Kofke

Department of Chemical Engineering
University at Buffalo, State University of New York

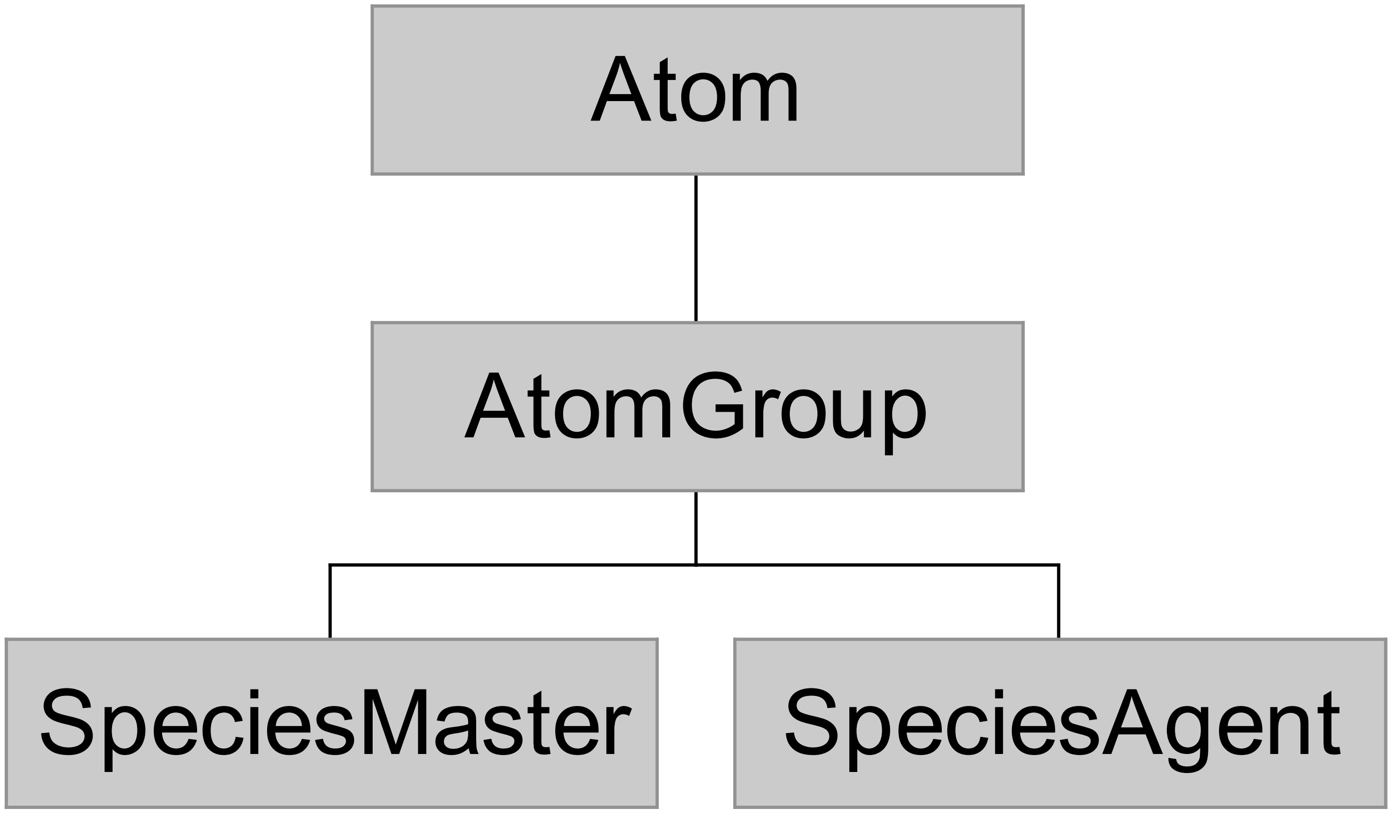


UNIVERSITY AT BUFFALO
State University of New York

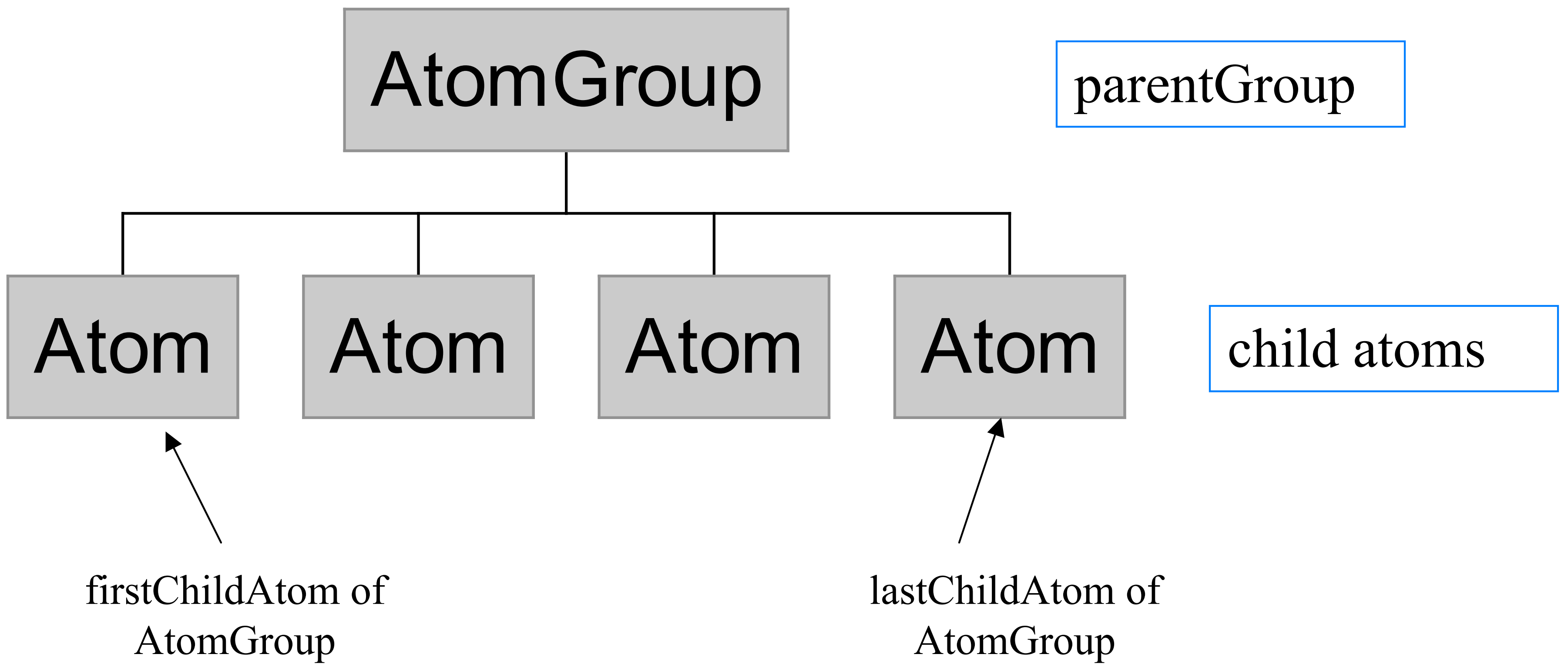
Species

- Top-level simulation element
  - Species, Potential, Integrator, Controller, Meter, Device, Display
- Defines methods and holds fields to specify, build, and arrange molecules/atoms
- Employs several lower-level classes
  - SpeciesMaster, SpeciesAgent
  - Atom, AtomGroup
  - AtomFactory
  - AtomType, Space.Coordinate, Parameter Bond, BondInitializer
  - Configuration

Inheritance Hierarchy

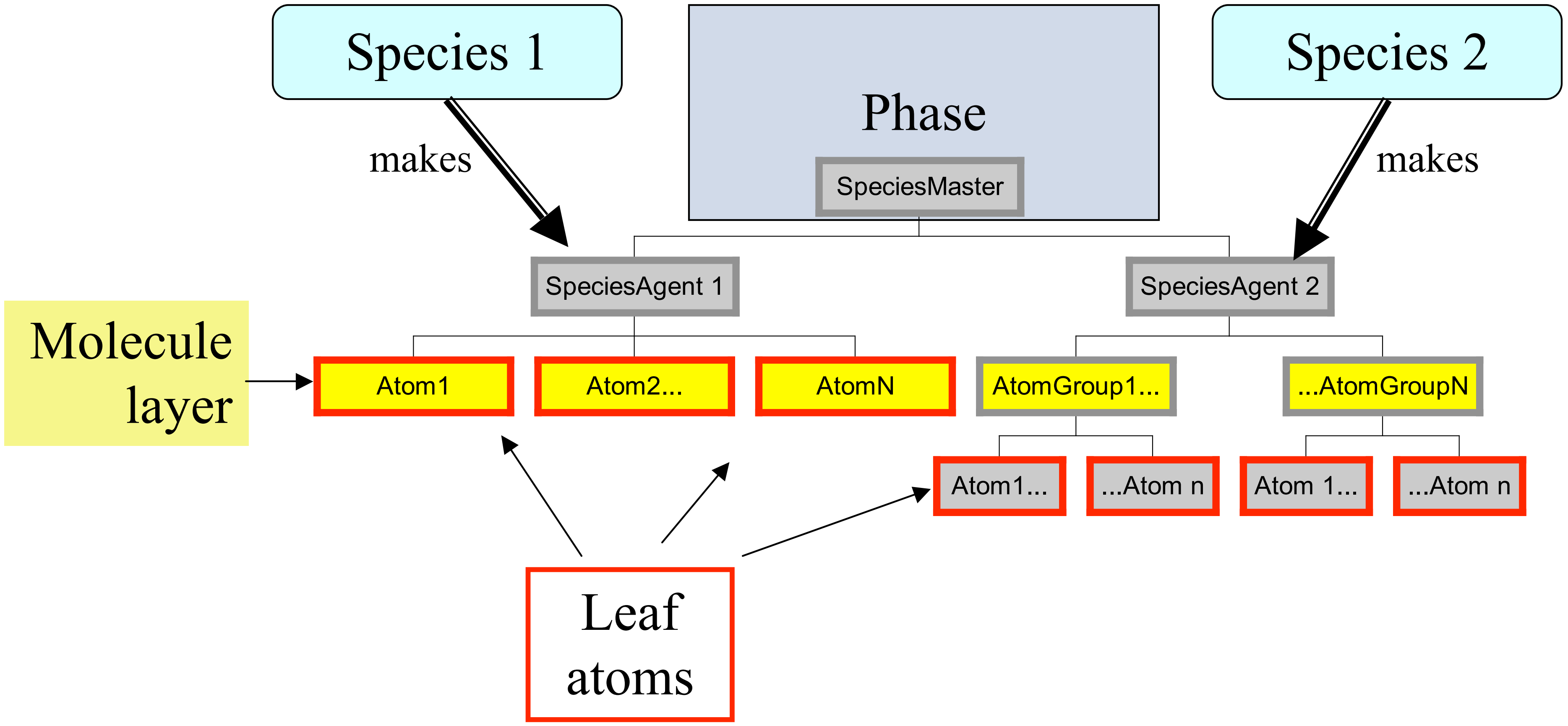


Containment

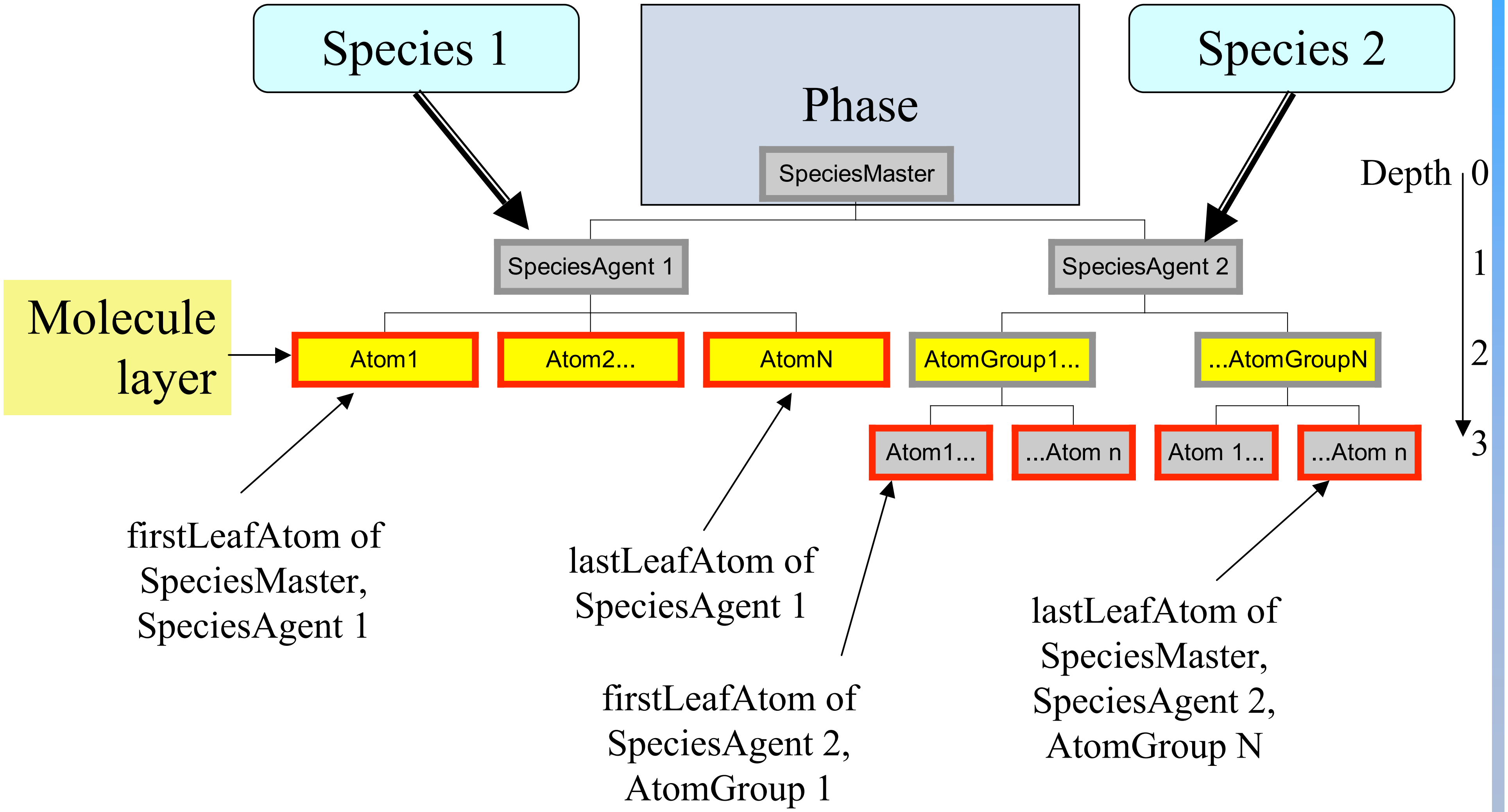


UNIVERSITY AT BUFFALO
State University of New York

Containment Hierarchy



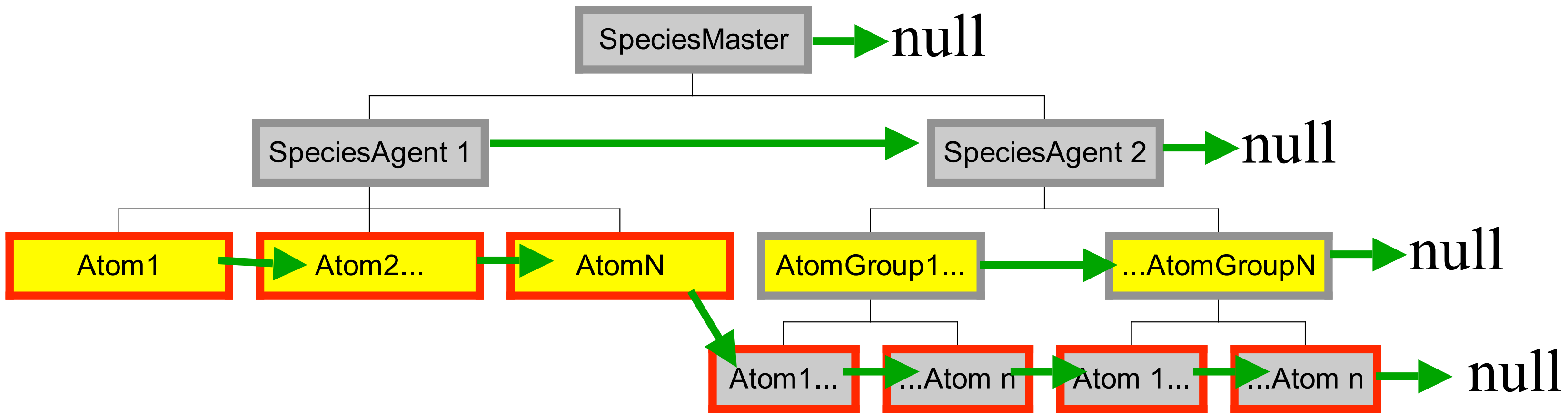
Containment Hierarchy



Species 1
Phase
SpeciesMaster
Species 2
Depth
0
1
2
3
SpeciesAgent 1
SpeciesAgent 2
Molecule layer
Atom1
Atom2...
AtomN
AtomGroup1...
...AtomGroupN
Atom1...
...Atom n
Atom 1...
...Atom n
firstLeafAtom of SpeciesMaster, SpeciesAgent 1
lastLeafAtom of SpeciesAgent 1
firstLeafAtom of SpeciesAgent 2, AtomGroup 1
lastLeafAtom of SpeciesMaster, SpeciesAgent 2, AtomGroup N

Sequencing

- Atoms are sequenced to facilitate looping through them
  - All leaf atoms in phase are sequenced together
  - Otherwise, only siblings are sequenced

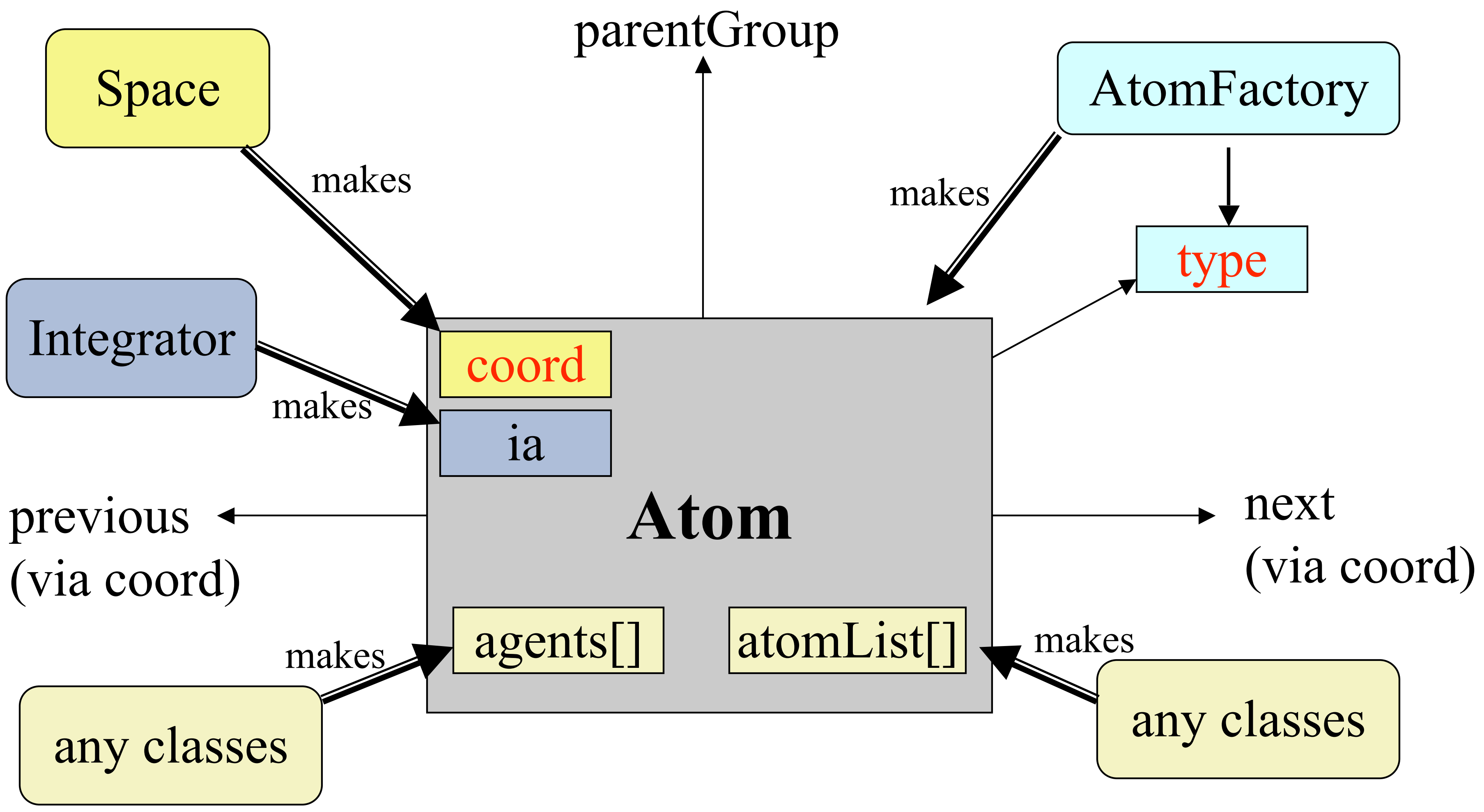


UNIVERSITY AT BUFFALO
State University of New York

Atom

- Corresponds to a physical atom (in simplest case)
- Holds data relating to
  - Spatial coordinate (position and/or momentum)
  - Atom type
  - Position in atom hierarchy
  - Position in atom sequence
- Can also hold auxiliary data
  - Agents from other classes
  - Parameters specified by other classes (in type field)
  - Lists of other atoms for use by other classes

Atom Fields


parentGroup
Space
makes
AtomFactory
makes
type
Integrator
coord
ia
makes
previous
(via coord)
Atom
next
(via coord)
agents[]
atomList[]
makes
makes
any classes
any classes

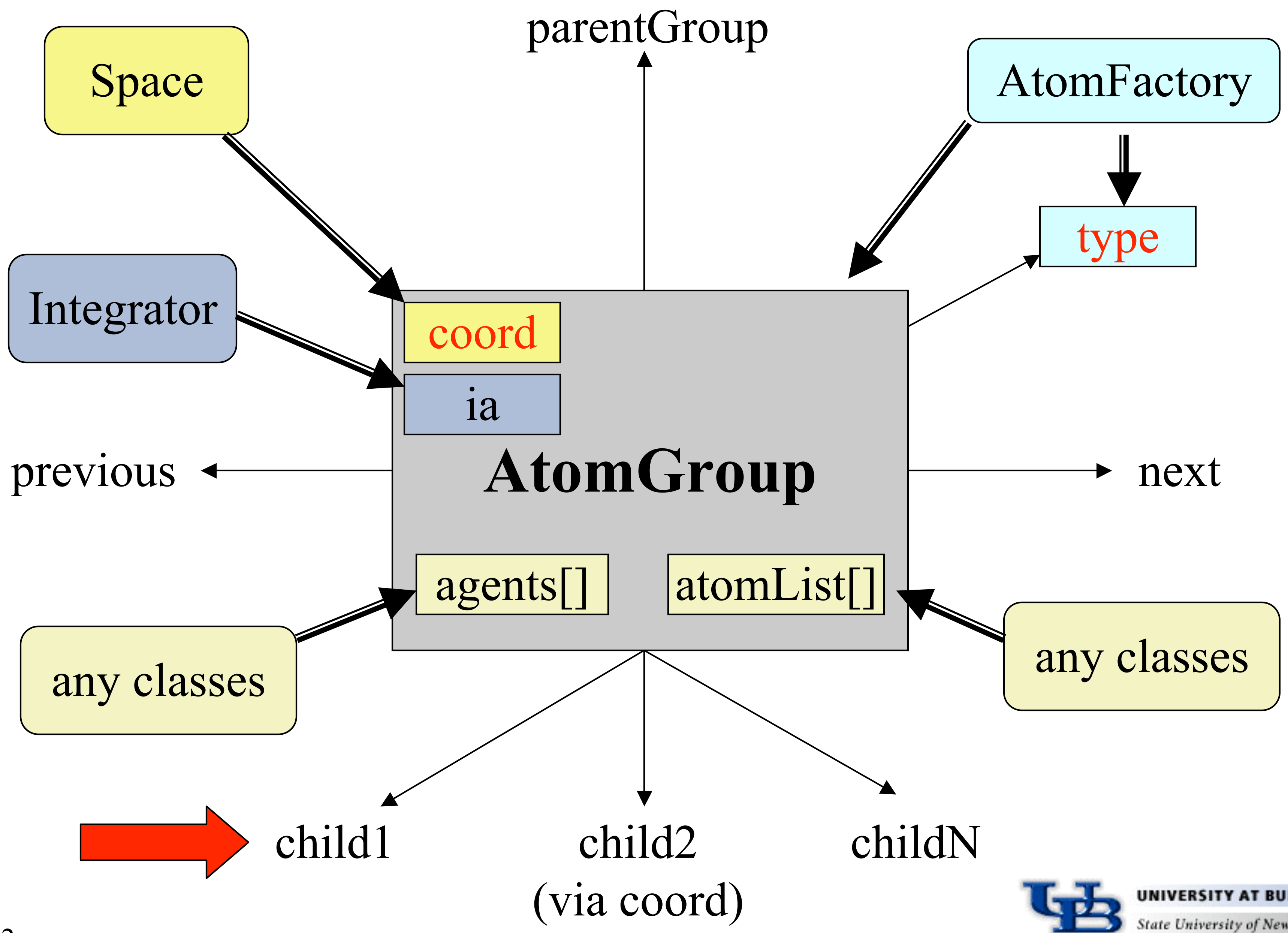
Atom Methods

- Sequence
  - nextAtom; previousAtom
  - preceeds; index
  - clearPreviousAtom, setNextAtom
- Hierarchy
  - parentGroup, parentMolecule, parentSpeciesAgent; isDescendedFrom
  - firstChildAtom; lastChildAtom; firstLeafAtom; lastLeafAtom
    - All return *this* if not an AtomGroup
  - depth, leafAtomCount; signature
  - setParentGroup
- Type
  - type, creator
- Simulation
  - parentPhase; parentSimulation; parentSpecies; requestAtomListIndex

AtomGroup

- Extends Atom
  - Inherits all fields given to an atom
  - Uses CoordinateGroup for Coordinate
    - e.g. translateTo method relays command to all child atoms' coordinates
- Defines a collection of atoms or other atom groups
  - Molecule
  - Atom subgroup, *e.g.*, CH_2, amino acid, polymer segment, *etc.*
- Has methods to add, remove, count contained atoms
  - addAtom; removeAtom; removeAll
  - addAtomNotify; removeAtomNotify
  - childAtomArray; getAtom; randomAtom
  - firstChildAtom; lastChildAtom
  - firstLeafAtom; lastLeafAtom
  - childAtomCount; leafAtomCount

Fields in an AtomGroup



UNIVERSITY AT BUFFALO
State University of New York

AtomFactory

- Constructs an Atom (or AtomGroup)
  - Refers to other atom factories to build group from subgroups
- Uses a BondInitializer to attach “bonds” to atom pairs
  - Bonds may be used by potentials or displays
- Uses a Configuration to put the atoms in some arrangement
- Reservoir can accept atoms and hold them for reuse to avoid repeated construction
  - Grand-canonical simulations
- Important methods
  - makeAtom() takes atom from reservoir, or builds new one if empty
  - build() defines how to make a new atom
  - set/get methods for bondInitializer and configuration
  - requestAgentIndex() gives integer for index of placement of an agent

Basic AtomFactory Classes

- AtomFactoryMono
  - Produces a single atom
  - setType method specifies AtomType instance to be referenced by all Atoms made by factory
- AtomFactoryHomo
  - Produces AtomGroup composed of arbitrary number of identical atoms
  - Constructor:
    - AtomFactoryHomo(Space, AtomFactory, int *(nAtoms)*, BondInitializer,Configuration)
- AtomFactoryHetero
  - Produces AtomGroup composed of non-identical atoms
    - AtomFactoryHetero(Space, AtomFactory[ ], Configuration)

UNIVERSITY AT BUFFALO
State University of New York

More Complex Atom Factories

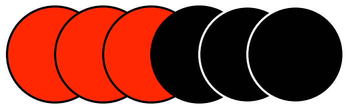
- Other atom factories can be built up from the basic ones

```
    //red atoms
AtomFactoryMono atomFactoryMono0 = new AtomFactoryMono(space);
AtomType type0 = new AtomType.Sphere(atomFactoryMono0, Default.ATOM_MASS, Default.ATOM_SIZE);
atomFactoryMono0.setType(type0);

    //black atoms
AtomFactoryMono atomFactoryMono1 = new AtomFactoryMono(space);
AtomType type1 = new AtomType.Sphere(atomFactoryMono1, Default.ATOM_MASS, Default.ATOM_SIZE);
atomFactoryMono1.setType(type1);

    //red segment
atomFactoryHomo0 = new AtomFactoryHomo(space,atomFactoryMono0,3,new BondInitializerChain(),
                                        new ConfigurationLinear(space));
    //black segment
atomFactoryHomo1 = new AtomFactoryHomo(space,atomFactoryMono1,3,new BondInitializerChain(),
                                        new ConfigurationLinear(space));

    //molecule
AtomFactoryHetero atomFactoryHetero = new AtomFactoryHetero(space,
                              new AtomFactory[] {atomFactoryHomo0,atomFactoryHomo1},
                              new MyBondInitializer(),
                              new ConfigurationLinear(space,1.25*Default.ATOM_SIZE));
speciesSpheres2 = new Species(this,atomFactoryHetero);
```



UNIVERSITY AT BUFFALO
State University of New York

More Complex Atom Factories

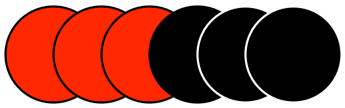
- Other atom factories can be built up from the basic ones

```
    //red atoms
AtomFactoryMono atomFactoryMono0 = new AtomFactoryMono(space);
AtomType type0 = new AtomType.Sphere(atomFactoryMono0, Default.ATOM_MASS, Default.ATOM_SIZE);
atomFactoryMono0.setType(type0);

    //black atoms
AtomFactoryMono atomFactoryMono1 = new AtomFactoryMono(space);
AtomType type1 = new AtomType.Sphere(atomFactoryMono1, Default.ATOM_MASS, Default.ATOM_SIZE);
atomFactoryMono1.setType(type1);

    //red segment
atomFactoryHomo0 = new AtomFactoryHomo(space,atomFactoryMono0,3,new BondInitializerChain(),
                                        new ConfigurationLinear(space));
    //black segment
atomFactoryHomo1 = new AtomFactoryHomo(space,atomFactoryMono1,3,new BondInitializerChain(),
                                        new ConfigurationLinear(space));

    //molecule
AtomFactoryHetero atomFactoryHetero = new AtomFactoryHetero(space,
                              new AtomFactory[] {atomFactoryHomo0,atomFactoryHomo1},
                              new MyBondInitializer(),
                              new ConfigurationLinear(space,1.25*Default.ATOM_SIZE));
speciesSpheres2 = new Species(this,atomFactoryHetero);
   //another approach
AtomFactoryHetero atomFactoryMolecule = new AtomFactoryHetero(space,
          new AtomFactory[] {atomFactoryMono0, atomFactoryMono0, atomFactoryMono0,
                              atomFactoryMono1, atomFactoryMono1, atomFactoryMono1} ...);
```

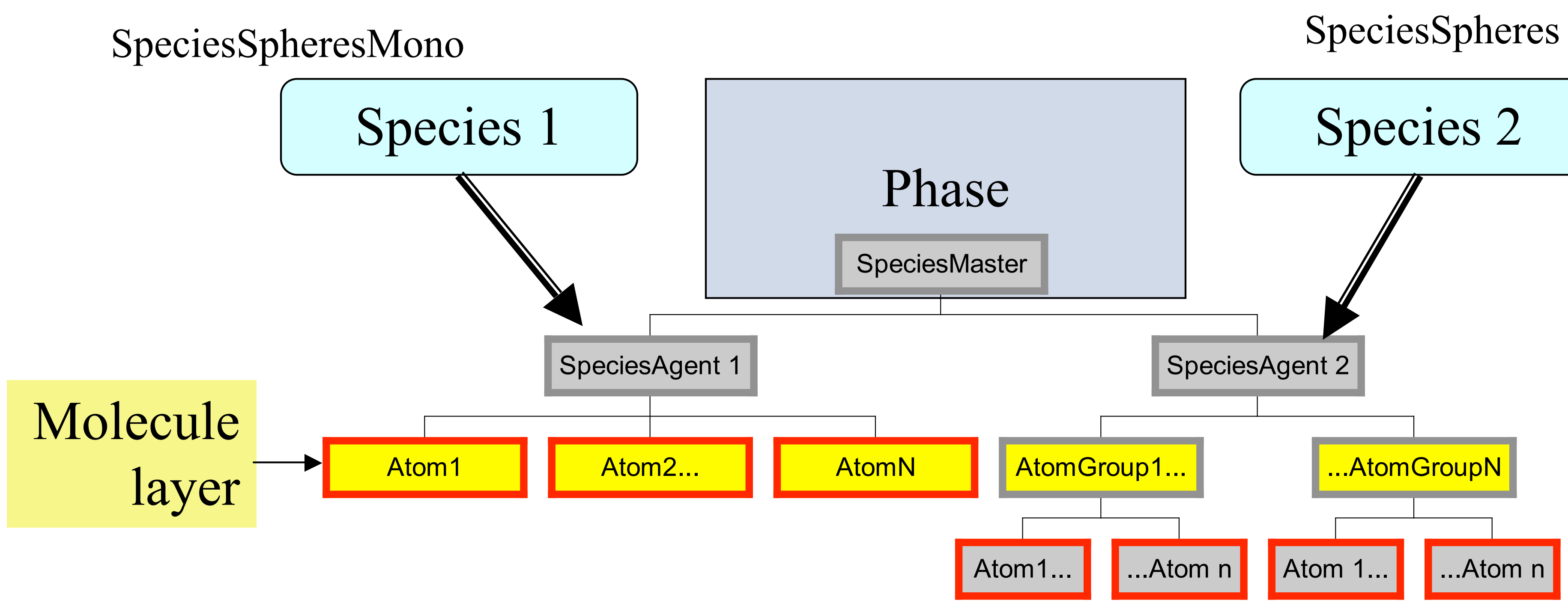


UNIVERSITY AT BUFFALO
State University of New York

Species Definition

- Species differ in the AtomFactory they use
  - Species constructor (the only one)
    - Species(Space, AtomFactory)
- A few species are pre-defined
  - SpeciesSpheres
    - Multiatomic molecules
    - SpeciesSpheres(Simulation, int nM, int nA, BondInitializer, Configuration)
  - SpeciesSpheresMono
    - Molecules are monatomic (no AtomGroup layer)
    - SpeciesSpheresMono(Simulation, int nM)
  - SpeciesWalls, SpeciesSphereWells, SpeciesSpheresRotating, etc.
- Extensions should define particular chemical compounds via AtomFactory
  - *new Species(space, atomFactoryWater)* instead of *new SpeciesWater*

Spheres vs. SpheresMono



UNIVERSITY AT BUFFALO
State University of New York

Species Methods

- public void allAgents(AtomAction action);
  - Performs given action on all agents of this species in all phases
- public void allAtoms(AtomAction action);
  - Performs given action on all leaf atoms of this species in all phases
- public void allMolecules(AtomAction action);
  - Performs given action on all molecules of this species in all phases
- public SpeciesAgent getAgent(Phase p);
  - Returns agent of this species in the given phase
- public void setNMolecules(int nM);
  - Performs setNMolecules on all agents of this species in all phases
- makeAgent(SpeciesMaster); makeMolecule();

UNIVERSITY AT BUFFALO
State University of New York

Iteration

- Common simulation activities involve looping through atoms and atom pairs, calculating properties based on their positions and separations
- Iterators are used to facilitate the looping

UNIVERSITY AT BUFFALO
State University of New York

AtomIterator Interface

- public boolean hasNext();
  - true if iterator has another atom to give
- public boolean contains(Atom atom);
  - true if given atom or its parents would among this iterator's iterates
- public Atom next();
  - returns the next iterate
- public Atom reset();
  - resets iterator to be ready to loop over all its iterates
- public Atom reset(IteratorDirective id);
  - resets iterator to be ready to loop over iterates consistent with the directive
- public void setBasis(Atom atom);
  - defines the basic set of atoms given by this iterate
- public int size();
  - number of iterates given by this iterator
- public void setAsNeighbor(boolean b);
  - more on this later

Using an Atom Iterator

- Perform actions on iterates (pass atoms to action)

```
iterator.reset();
while(iterator.hasNext()) {
    Atom atom = iterator.next();
    //do something with atom
}
```

- Also could pass action to iterator, but not fully implemented

```
AtomAction action = ...//define action
iterator.allAtoms(action);
     //performs action on all atoms
```

Iterator Directive

- IteratorDirective passed to *reset* instructs how to iterate
- Key attributes
  - direction
    - UP, DOWN, BOTH, NEITHER
  - atom
    - iteration performed in reference to atom, if not null
- Directive is conditioned before passing to iterator
  - set()
    - sets atom to *null*
    - direction unchanged
  - set(Atom a)
    - sets atom to *a*
    - direction unchanged
  - set(Direction d)
    - atom unchanged
    - sets direction to *d*

UNIVERSITY AT BUFFALO
State University of New York

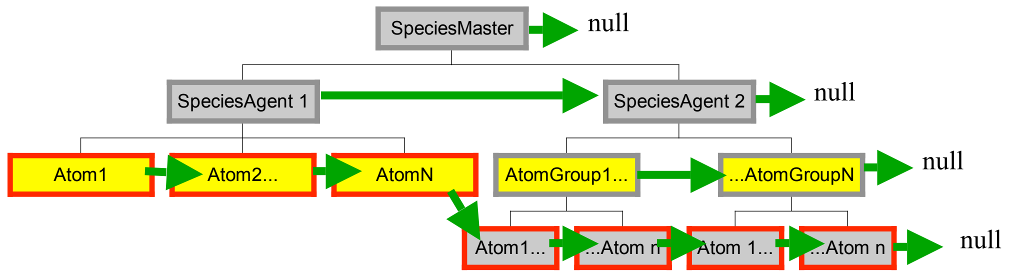
Setting the Basis

- Basis defines the source of atoms for the iterator
  - setBasis(Atom);
- Within a basis, iterator may be set to loop over atoms in different ways
  - Depends on iteratorDirective
  - Depends on design of iterator
- Setting basis does not reset the iterator
  - Need to call reset() before beginning iteration

UNIVERSITY AT BUFFALO
State University of New York

AtomIteratorSequential 1.

- Most commonly used iterator
- Presents atoms in sequence defined with heirarchy

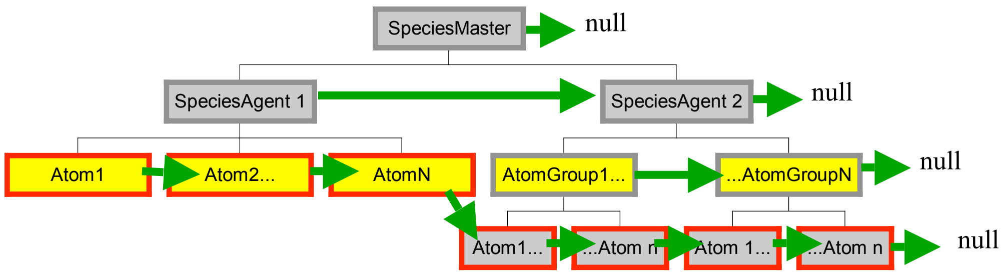


- setBasis indicates the parent atom of the iterates
  - isLeafIterator flag indicates if children of basis are iterated, or if all leaf atoms below basis are iterated

UNIVERSITY AT BUFFALO
State University of New York

AtomIteratorSequential 2.

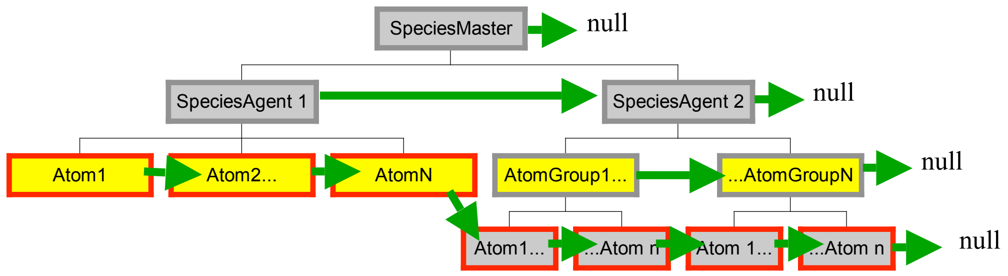
- Directives
  - atom == null indicates to do all atoms
  - atom != null indicates to begin iteration with *atom*
  - direction indicates which way to go in sequence
    - UP: natural order, beginning with specified atom
    - DOWN: opposite direction, beginning with specified atom
    - BOTH: *up* beginning with atom, then *down* beginning after it
    - NEITHER: do just specified atom



UNIVERSITY AT BUFFALO
State University of New York

AtomIteratorSequential 3.

- Examples (set as leaf iterator)
  - setBasis(speciesAgent1);
  - iterator.reset(iteratorDirective.set().set(UP))
    - Loop through all molecules of species 1, from first to last
  - iterator.reset(iteratorDirective.set(atom2).set(UP));
    - Loop from atom2 up through last
  - setBasis(speciesMaster);
  - iterator.reset(iteratorDirective.set(atomN).set(BOTH));
    - Loop from atomN up through last atom in phase, then from N-1 down to first atom in phase



UNIVERSITY AT BUFFALO
State University of New York

Other Atom Iterators

- AtomIteratorCompound
  - Forms a single iterator from a collection of atom iterators
  - Can also form from an atom group
- AtomIteratorBonds
  - Loops over atoms Bonded to Basis atom
- AtomIteratorNonbonded
  - Loops over atoms in molecule not bound to given atom
- AtomIteratorList
  - Loops over atoms in an AtomList
- AtomIteratorSinglet
  - Gives one atom and then expires
- AtomIteratorNeighbor
  - Loops over predefined “neighbors” of a given atom

AtomPairIterator

- Iterates over pairs of atoms, returning each as an AtomPair
- Formed from two AtomIterator instances
- IteratorDirective interpretation is slightly different
  - Details for another discussion!