1 The formula

1. $W$ is the network topology and node times. It can also be seen as a multiply labelled tree.
2. $\theta$ is the population parameters. Exactly how many of these there are, and where they are in $W$ is not settled, but one scheme would be: one value at each tip, one at the rootward end of each branch, and one just after hybridization events.
3. $\lambda$ is the parameters for $W$, that is, the topology and node times. $\lambda$ could consist of a speciation rate, an extinction rate, and a hybridization rate.
4. $\eta$ is the population mean, appearing in a hyperprior for $\theta$
5. $n$ is the number of gene trees.
6. $\tau_i$ is the i’th gene tree topology and node times.
7. $\alpha_i$ is all the other parameters belonging to the i’th gene tree: parameters for site rate heterogeneity, substitution model, branch rate model, root model.
8. $g_i$ is $(\tau_i, \alpha_i)$, that is, all the parameters for the i’th gene tree.
9. $\gamma_i$ is the permutations of sequences within individuals for the i’th gene.
10. $d_i$ is the sequence data for the i’th gene.

$\tau = (\tau_1, ..., \tau_n)$, and similarly for $\alpha, g, \gamma, d$.

$$
Pr(W, \theta, g, \gamma | d) \propto Pr(W|\lambda) \cdot Pr(\lambda) \times \\
Pr(\theta|\eta) \cdot Pr(\eta) \times \\
Pr(\gamma) \times \\
\prod_i Pr(\tau_i|W, \theta, \gamma_i) \times \\
\prod_i Pr(d_i|g_i)
$$

1. $Pr(W|\lambda) \cdot Pr(\lambda)$ is the network prior: the probability of $W$ before seeing any molecular data.
2. $Pr(\theta|\eta) \cdot Pr(\eta)$ is the population prior.
3. $Pr(\gamma)$ is the permutation prior. Quite likely uniform, so can be ommitted.
4. \(\Pr(\tau_i|W, \theta, \gamma_i)\) is the probability of \(\tau_i\), when permuted by \(\gamma_i\), fitting into the network \(W\) with populations determined by \(\theta\). Note that this probability does nor depend on \(\alpha_i\). See below for what permuting \(\tau_i\) by \(\gamma_i\) means.

5. \(\Pr(d_i|g_i) = \Pr(d_i|\tau_i, \alpha_i)\) is the ‘Felsenstein likelihood’ of the data for the i’th gene given the i’th gene tree.

I previously thought of

\[
\Pr(d_i|g_i, \gamma_i) \Pr(g_i|W, \theta)
\]

as

\[
\Pr(\gamma_i(d_i)|g_i) \Pr(g_i|W)
\]

so that the \(\gamma_i\) are thought of as permuting the sequence data \(d_i\). This doesn’t work well in implementation in BEAST: lots of existing code does not expect sequences to be swapped around. It would be possible to regard the \(\gamma_i\) as specifying a topological change in a gene tree (eg two terminal branches would be swapped). However, it seems simplest to regard \(\gamma_i\) as permuting the tip labels of the gene tree. The sequences attached to a tip don’t change, nor does the gene tree topology. Instead, the way in which the sequences are assigned tips in the multiply labelled tree \(W\) is changed. Thus one can write:

\[
\Pr(\tau_i|W, \theta, \gamma_i) = \Pr(\gamma_i(\tau_i)|W, \theta)
\]

2 Classes

2.1 List of classes I have added

2.1.1 Parsers in dr.evomodelxml.operators

- AlloppNetworkNodeSlideParser
- AlloppSequenceReassignmentParser
- AlloppMoveLegsParser

2.1.2 Parsers in dr.evomodelxml.speciation

- AlloppMSCoalescentParser
- AlloppNetworkPriorModelParser
- AlloppNetworkPriorParser
- AlloppSpeciesBindingsApSpInfoParser
- AlloppSpeciesBindingsIndividualParser
- AlloppSpeciesBindingsParser
- AlloppSpeciesNetworkModelParser

2.1.3 Parsers in dr.inferencexml.loggers

- AlloppDEBUGTUNELoggerParser

2.1.4 Classes in dr.evomodel.operators

- AlloppNetworkNodeSlide. An operator which changes node heights and tree topology within a homoploid tree and changes hybridization times and split height in no-diploid case.
- AlloppSequenceReassignment An operator which changes assignments of sequences within an individual
- AlloppMoveLegs An operator which changes the way a tetraploid subtree joins the diploid tree

2.1.5 Classes in dr.evomodel.speciation

- AlloppLeggedTree is a homoploid ‘tree with legs’.
- AlloppMSCoalescent computes coalescent log-likelihood of a set of gene trees embedded inside a allopolyploid species network. It is an instance of Likelihood.
- AlloppNetworkPrior computes log-likelihood of prior for the network. It is an instance of Likelihood.
- AlloppNetworkPriorModel stores parameters for network prior (eg rates). It is an instance of Model.
- AlloppSpeciesBindings knows how species are made of individuals and individuals are made of taxa (= diploid genomes within individuals). It is an instance of Model.
- AlloppSpeciesNetworkModel implements the species network as a collection of ‘trees with legs’, and converts this representation into a multiply labelled binary tree. It is an instance of Model.

2.1.6 Classes in dr.inference.loggers

- AlloppDEBUGTUNELogger is for testing.

2.1.7 Classes in test.dr.evomodel.speciation

- AlloppSpeciesNetworkModelTEST is for testing.
2.1.8 Classes in dr.util

- AlloppMisc is odds and ends, mainly for testing.

3 Jargon

I have started using the words leg, foot, and union to mean special things.

The species network is composed of trees, which are joined together. A higher ploidy tree is joined to a lower ploidy tree with one or two legs. The point where a leg meets the branch of the lower ploidy tree is a foot.

A union is a set of indices where each index represents a species, or more often, a species and a sequence. For example, if a, b, c, ... are species, 1, 2, 3, ... are individuals and A, B, C, ... label sequences, so that a single sequence (taxon) in an alignment might be c2B, then a union can represent a set of elements like {aA, bA, bC}. The identity of individuals 1, 2, 3, ... is ignored. At a tip in a gene tree, or a tip in the multiply labelled tree, there will be a single species and a single sequence, like aA or bC. At internal nodes, there are sets of them, each one being a union of its children’s sets. The union at a node is unique to that node in the multiply labelled tree (with one exception near the root in the no-diploids case). Unions can therefore be used to identify nodes. In the species network, the identity of the sequence is lost, but a similar mechanism is used.


4 XML Parsing

I have made the AlloppJun29.xml file which contains a specification for a complete analysis. It is simple and has odd data, and does not log everything it should. The parsers can parse everything in it.

I have added my parsers to development parsers file.
beast16/bin/dr/app/beast/development.parsers.properties

I have added getSyntaxRules() implementations, but it is hard to tell if I have done as much as I should or could.

5 Tests

I have written two JUnit tests in AlloppSpeciesNetworkModelTEST. I have decided to create a new (inner) class for each test, which is passed to constructors and other functions. This distinguishes test code from normal code, and sometimes the class is used to supply extra information needed for the test. In other cases it looks messy.
5.1 Test 1: network to MUL-tree

I have code for translating a network representation into a mullab representation, which is tested by some small cases in a test unit AlloppSpeciesNetworkModelTest. That does 3 tetra species, and 2 diploid species, in various arrangements. This isn’t very relevant to the tetra only case, but it will be needed.

5.2 Test 2: $\text{Pr}(g_i|W)$

The second test is a likelihood calculation for $\text{Pr}(g_i|W)$ for a case with two tetraploids and five individuals. It is compared to the result calculated in R.

6 Main things done since 21 June

Implementations of operate(), accept(), reject() for my MCMCOperators, and getModel(), getLogLikelihood(), storeState(), restoreState() for AlloppSpeciesNetworkModel, and AlloppNetworkPriorModel().

Hooked up with the gene tree part of MCMC. Any change in any gene tree requires a recalculation of the $\text{Pr}(g_i|S)$. Implemented an equivalent of NodeReheight for the species network.

Implemented sequence assignment operator.

Implemented some instrumentation code.

Some testing and debugging with simple cases. It appears to work on a couple of (similar) XML files. Two species, one gene, total of three individuals and therefore six sequences.

7 AlloppNetworkNodeSlide

Extension of TreeNodeSlide to deal with allopolyploid networks. TreeNodeSlide is an operator written by Joseph Heled for a species tree based on the ideas of Mau et al.

7.1 Methods

Implements standard simple methods for operator. The key one is doOperation() which calls operateOneNodeInNet().

private:

randomnode(). Chooses a random node in network, which can mean a node in a tree, a hybridization event, the foot of a leg, or the split time of a joined leg.

operateOneNodeInNet(). Calls randomnode() then one of operateOneNodeInLeggedTree() or operateHybridHeightInLeggedTree() or operateLegsInLeggedTree().
operateHybridHeightInLeggedTree(). Moves the hybridization height to somewhere between root of tree and split height or most recent foot height.

operateLegsInLeggedTree(). Only written for one-tetratree case. Moves split height to somewhere between split height and value returned by diploidSplitUpperBound().

operateOneNodeInLeggedTree(). This is based on TreeNodeSlide. It avoids moving the root before the hybridization height. It calls mauCanonical() and mauReconstruct().

mauCanonical(). As TreeNodeSlide except operates on a AlloppLeggedTree.

mauReconstruct(). As TreeNodeSlide except operates on a AlloppLeggedTree.

mauCanonicalSub. As TreeNodeSlide. Recursive.

mauReconstructSub. As TreeNodeSlide. Recursive.

8 AlloppSequenceReassignment

Operator which changes the assignment of sequences belonging to a randomly chosen individual in a randomly chosen species.

8.1 Methods

Implements standard simple methods for operator. The key one is doOperation() which calls permuteOneGeneOneSpeciesOneIndiv() on the AlloppSpeciesBindings.

9 AlloppLeggedTree

This is a ‘tree with legs’, which is used for a homoploid species tree which is attached to a tree of lower ploidy via its legs. It is also used for the diploid tree; in this case there are no legs.

The tree is a SimpleTree. Its nodes contain taxa at tips, and heights. There are several arangements of legs.

NONE: for the diploid tree.

TWOBRANCH: Two legs attached to different branches in a lower ploidy tree.

ONEBRANCH: Two legs attached to the same branch in a lower ploidy tree at different times.

JOINED: One leg, representing two species which (going back in time) join before their ancestor is attached to a branch in a lower ploidy tree.

NODIPLOIDS: similar to JOINED, but a single diploid ‘trunk’ is assumed which has no tips.

The tree has a hybridization height before the root, and in cases JOINED and NODIPLOIDS, a split height before hybridization.
9.1 Inner classes

Leg - defines attachment to a lower ploidy tree. Uses a union to specify the branch.

9.2 Methods in small inner classes

Leg.Leg(). Clone constructor for store and restore methods.

9.3 Methods

AlloppLeggedTree(). Constructor. Passed an array of Taxons, plus a leg type. Makes a random Yule-type tree and fills in random times for hybridheight and legs(s) and split time.

AlloppLeggedTree(AlloppLeggedTree). Clone constructor for store and restore methods.

There are also two constructor for testing, which make small nonrandom trees.

scaleAllHeights(). Passed a scaling factor this multiplies all heights in the tree and legs: node heights, hybrid heights, leg heights, split height.

gInternals( . Returns array of speciation heights, for use by AlloppNetworkPrior.

gRootHeight(). Returns what it says.

getSplitHeight(). Returns what it says.

getHybridHeight(). Returns what it says.

getMaxFootHeight(). Returns maximum height of all (0, 1 or 2) legs.

getMaxHeight(). Returns maximum of all heights. Can be the root height, a foot height or the split time, depending on leg type.

setFootUnion(). Sets the ‘foot union’ of a specified leg. The foot union defines a node within a species tree by specifying the (species index, sequence index) to which the leg is attached.

getFootUnion(). Passed a leg index, it returns what it says.

getNumberOfLegs(). Passed a leg index, it returns what it says.

getFootHeight(). Passed a leg index, it returns what it says.

setHybridHeight(). Modifies what it says. Called by AlloppNetworkNodeSlide.

setSplitHeight(). Modifies what it says. Called by AlloppNetworkNodeSlide.

private:

randomnodeheight(). Used by constructor.

randomsplitheight(). Used by constructor.
9.3.1 Lots of delegations

9.3.2 Tree

gRoot()
gNodeCount()
gNode(int i)
gInternalNode(int i)
gExternalNode(int i)
gExternalNodeCount()
gInternalNodeCount()
gNodeTaxon(NodeRef node)
hasNodeHeights()
setNodeHeight(NodeRef node)
hasBranchLengths()
geBranchLength(NodeRef node)
geNodeRate(NodeRef node)
geNodeAttribute(NodeRef node, String name)
geNodeAttributeNames(NodeRef node)
isExternal(NodeRef node)
isRoot(NodeRef node)
geChildCount(NodeRef node)
geChild(NodeRef node, int j)
geParent(NodeRef node)
geCopy()
geTaxonCount()
geTaxon(int taxonIndex)
geTaxonId(int taxonIndex)
geTaxonIndex(String id)
geTaxonIndex(Taxon taxon)
asList()
geTaxonAttribute(int taxonIndex, String name)
iterator()
getUnits()
setUnits(Type units)
setAttribute(String name, Object value)
getAttribute(String name)
Iterator<String> getAttributeNames()
addTaxon(Taxon taxon)
removeTaxon(Taxon taxon)
setTaxonId(int taxonIndex, String id)
setTaxonAttribute(int taxonIndex, String name, Object value)
addMutableTaxonListListener(MutableTaxonListListener listener)

9.3.3 MutableTree which extends Tree, MutableTaxonList

beginTreeEdit()
endTreeEdit()
addChild(NodeRef parent, NodeRef child)
removeChild(NodeRef parent, NodeRef child)
replaceChild(NodeRef node, NodeRef child, NodeRef newChild)
setRoot(NodeRef root)
setNodeHeight(NodeRef node, double height)
setNodeRate(NodeRef node, double rate)
setBranchLength(NodeRef node, double length)
setNodeAttribute(NodeRef node, String name, Object value)
addMutableTreeListener(MutableTreeListener listener)

9.3.4 TreeLogger.LogUpon

logNow(int state). Not yet implemented.

10 AlloppMSCoalescent

Computes coalescent log-likelihood of a set of gene trees embedded inside a allopolyploid species network.
10.1 Methods

AlloppMSCoalescent(). Constructor. Stores AlloppSpeciesBindings and AlloppSpeciesNetworkModel and adds itself to their model-listeners.

There is also a constructor for testing.

Implements standard simple methods for likelihood. Key ones follow.

calculateLogLikelihood(). Calls geneTreeFitsInNetwork and geneTreeLogLikelihood in AlloppSpeciesBindings for each gene.

getLikelihoodKnown(). Returns false. Always recalculate from scratch.

11 AlloppNetworkPrior

Computes prior log-likelihood of allopolyploid species network.

11.1 Methods

Implements standard simple methods for likelihood. Key ones follow.

calculateLogLikelihood(). Uses getSpeciationHeights() and rate parameter from AlloppNetworkPriorModel. Simple implementation.

getLikelihoodKnown(). Returns false. Always recalculate from scratch.

12 AlloppNetworkPriorModel

Very simple. Basically just a Parameter wrapped up as a model.

13 AlloppSpeciesBindings

AlloppSpeciesBindings knows how species are made of individuals and individuals are made of taxa (= diploid genomes within individuals).

It also contains the list of gene trees - tree topologies and node times, plus popfactors. Given a AlloppSpeciesNetworkModel it can say if a gene tree is compatible, and calculate the loglikelihood of the gene tree ‘fitting’ into the network.

It is here that assignments of sequence copies within individuals get permuted. See GeneTreeInfo.AlignmentRowInfo below.
13.1 Inner classes

Individual - Simple helper class for one individual containing one or more sequences.
SpeciesIndivPair - Simple helper class used by permuteSetOfIndivs() which is part of sequence reassignment operator.
ApSpInfo - Simple helper class for one (allopolyploid) species, containing one or more individuals
GeneTreeInfo - A gene tree as used by BEAST, plus popfactor, plus indices for each individual to map sequences to indices (0 or 1 for tetraploids) which identify the legs of the tetraploid subtree.
GeneTreeInfo.SequenceAssignment - where the indices just mentioned are stored.
GeneTreeInfo.GeneUnionNode - for GeneUnionTree.
GeneTreeInfo.GeneUnionTree - serves same function as JH's CoalInfo, storing the set (=union) of species-sequence pairs belonging to a node in the gene tree. I copy gene tree topology and times into a GeneUnionTree to do calculations.

13.2 Methods

There are a lot of mapping of one kind of index or indices to others, one to get list of species at a given ploidy level, etc. fitsInNetwork() and geneTreeLogLikelihood() are key methods.

13.3 Methods for small inner classes

ApSpInfo.taxonFromIndSeq() Passed indices for individual, sequence, returns taxon.
GeneTreeInfo.SequenceAssignment.toString(). For header in log file.
GeneTreeInfo.GeneUnionNode.asText(). For testing.

13.4 GeneTreeInfo.GeneUnionTree methods

GeneUnionTree(). Constructor. Calls genetree2geneuniontree() to build the GeneUnionTree.
private to GeneUnionTree:
subtreeFitsInNetwork() Recursive. Calls coalescenceIsCompatible() in network.
subtreeRecordCoalescence() Recursive. Calls recordCoalescence() in network.
genetree2geneuniontree() Recursive. Copies topology, fills in union fields.
asText(). For testing. Makes textual rep of GeneUnionTree.
subtreeAsText(). For asText().
13.5 GeneTreeInfo methods

GeneTreeInfo(). Constructor. Fills array of AlignmentRowInfo s. Fills int array of lineage counts at tips.

seqassignsAsText(). For testing.
genetreeAsText(). For testing.
fitsInNetwork(). Calls subtreeFitsInNetwork() in GeneUnionTree.
GeneTreeInfo.treeLogLikelihood(). Calls clearCoalescences() in network, makes new GeneUnionTree, calls subtreeRecordCoalescence() in GeneUnionTree, then sortCoalescences() in network, then recordLineageCounts() in network, and finally geneTreeInNetworkLogLikelihood() in network.
storeGeneTreeState(). Stores sequence assignments.
restoreGeneTreeState(). Restores stored sequence assignments.
speciationUpperBound(). Passed two sets of species, it finds the most recent coalescence for this gene such that the children of this gene in the gene tree contain at least one species from each set. Used by AlloppNetworkNodeSlide.
diploidSplitUpperBound(). For one tetra tree case, this finds the most recent coalescence for this gene such that the children of this gene in the gene tree contain at least one from each sequence assignment. Used by AlloppNetworkNodeSlide.
permuteOneSpeciesOneIndiv(). Chooses a random species and a random individual, and calls permuteOneAssignment().
permuteSetOfIndivs(). Chooses a set of (species, individual) pairs, and calls permuteOneAssignment(). 2011-08-12: How it chooses the set is odd. Probably need revisiting when more testing of MCMC efficiency done.
getSeqassigns(). Passed taxon index, it returns (reference to) a SequenceAssignment. Used by logger.
wasChanged(). Does nothing. (Might set dirty flag one day.)

private:
collectIndivsOfNode(). Used by permuteSetOfIndivs().
subtreeSpeciationUpperBound(). Called by speciationUpperBound() to do the real work.
subtreeDiploidSplitUpperBound(). Called by diploidSplitUpperBound() to do the real work.
permuteOneAssignment(). Passed indices for a species and an individual, to do one ‘flip’ of a sequence assignment.

13.6 AlloppSpeciesBindings methods

AlloppSpeciesBindings(). Constructor. Made from array of ApSpInfos, array of TreeModels, array of popFactors, and minheight. Makes 'flattened' arrays of species, individuals, taxa, sets up
maps of indices. Makes array of GeneTreeInfos from TreeModels and popFactors, and then fixes
the node heights to at least minheight.

There are also two constructors for testing.

initialMinGeneNodeHeight(). Returns what it says. Used for starting state of network.

spsqunion2spunion(). Converts a set (a union) containing (species index, sequence index) pairs
into a set containing just species indices.

numberOfGeneTrees(). Returns what it says.

maxGeneTreeHeight(). Returns what it says. Used for Pr(gi|S) calculation in root.

geneTreeFitsInNetwork(). Passed index of a gene tree, it calls fitsInNetwork() in a
GeneTreeInfo.

geneTreeLogLikelihood(). Passed index of a gene tree, it calls treeLogLikelihood() in a
GeneTreeInfo.

numberOfSpecies(). Returns what it says.

apspeciesName(). Passed index of a species, returns what it says.

SpeciesWithinPloidyLevel(). Passed a ploid level, it returns an array of Taxons for species.

spseq2spseqindex(). Converts a (species index, sequence index) pair into a single index.

spseqindex2sp(), spseqindex2seq(). Inverse of above, they convert a single index into a
(species index, sequence index) pair. They call spseqindex2spandseq().

apspeciesId2index(). Species name to index.

apspeciesId2speciesindiv(). Converts a species id (name like 03_d_B) to a pair of indices
(species,indiv). (Loosely, 03_d_B → (d,03)). used by permuteSetOfIndivs().

numberOfSpSeqs(). Returns number of (species index, sequence index) pairs.

nLineages() Passed index of a species, returns lineage count at tip.

taxonFromSpIndSeq(). Passed three indices, for species, individual, sequence, returns a Taxon.

speciationUpperBound(). Calls method of same name (which see) in each GeneTreeInfo to find
minimum.

diploidSplitUpperBound(). Calls method of same name (which see) in each GeneTreeInfo to
find minimum.

permuteOneGeneOneSpeciesOneIndiv(). Chooses a random gene and calls
permuteOneSpeciesOneIndiv() on it.

permuteSetOfIndivsForOneGene(). Chooses a random gene and calls permuteSetOfIndivs() on
it.

seqassignsAsText(). Calls method of same name (which see) in one GeneTreeInfo.

genetreeAsText(). Calls method of same name (which see) in one GeneTreeInfo.

handleModelChangedEvent(Model model, Object object, int index). Calls wasChanged() on
each gene tree.
handleVariableChangedEvent(Variable variable, int index, ChangeType type). Never called. (asserts false.)

storeState(). Calls storeGeneTreeState() on each GeneTreeInfo.

restoreState(). Calls restoreGeneTreeState() on each GeneTreeInfo.

acceptState(). Does nothing.

getColumns(). Returns array of LogColumn's for logger, for logging sequence assignments. One column for each (gene,taxon) pair.

private:

spseqindex2spandseq(). Used by spseqindex2sp(), spseqindex2seq().

14 AlloppSpeciesNetworkModel

This contains two representations of the network. It implements the species network as a collection of 'trees with legs' and converts this representation into a multiply labelled binary tree. The general idea is that the network is easiest to change (eg detach and re-attach tetraploid subtrees) while likelihood calculations are easiest to do in the multiply labelled tree.

The individual 'trees with legs' are implemented by AlloppLeggedTree's.

14.1 Inner classes

MulLabNode - for MulLabTree. Contains information about populations size, coalescent times, and a union field which is a set of (species index, sequence index) pairs which identifies the node.

MulLabTree - represents the species network as single binary tree with tips that can be multiply labelled with species.

MulLabTree.LegLink, MulLabTree.FootLinks - for gathering and organising the links between trees of different ploidy, so that the rootward-pointing legs can become tipward-pointing branches.

MulLabTree.SpSqUnion - low level class used for mapping population values to nodes in MulLabTree.

MulLabTree.PopulationAndLineages - records the information (times, populations, number of lineages) needed to calculate the probability of coalescences in a single branch of the MulLabTree.

14.2 Methods

Lots of key methods here.
14.3 Methods in small inner classes

MulLabNode.asText(). For testing
MulLabTree.PopulationAndLineages.populationAt(). For calculating the probability of coalescences in a single branch of the multiply labelled tree.

14.4 MulLabTree methods

MulLabTree(). Constructor. Makes a single multiply labelled tree from the set of homoploid SimpleTrees. It counts tips, makes array of MulLabNodes. Then it copies the homoploid trees into array using simpletree2mullabtree(), with eg two copies for tetraploid subtrees, collecting leg info in LegLinks while copying. Then it re-organises the root-pointing LegLinks into tip-pointing FootLinks, and adds nodes to lower-level ploidy subtrees appropriately. Finally it calls fillinUnionsInSubtree().

mullabTreeAsNewick(). Converts the multiply labelled tree to a Newick string, currently just for testing.

asText(). For testing.
clearCoalescences(). Removes coalescent information from nodes. Calls clearSubtreeCoalescences(). The method for recording coalescences ‘accumulates’ them as they are found in gene trees, so need to remove them all first.

recordLineageCounts(). Fills in counts of lineages at nodes. Calls recordSubtreeLineageCounts().
geneTreeInMLTreeLogLikelihood(). Calculates the log-likelihood for a gene tree in the multiply labelled tree. Calls fillinpopvals() and geneTreeInMLSubtreeLogLikelihood().

private:

subtreeAsText(). Recursive. For asText().
simpletree2mullabtree(). Recursive. Makes copy of a SimpleTree, as used by AlloppLeggedTree in array of MulLabNodes. It fills in union fields as it copies.

fillinUnionsInSubtree() fills in union fields after most of MulLabTree is constructed.

nodeOfUnion(). Passed FixedBitSet x, it returns the most tipward node whose union contains x. If x is known to be a union of one of the nodes, it finds that node, so acts as a map from union to node. Calls nodeOfUnionInSubtree().

nodeOfUnionInSubtree(). Recursive, for nodeOfUnion().
mullabSubtreeAsNewick(). Recursive, for mullabTreeAsNewick().
clearSubtreeCoalescences(). Recursive, for clearCoalescences().
recordSubtreeLineageCounts(). Recursive, for recordLineageCounts().

fillinpopvals(). This copies population values in the Parameter popvalues to nodes in the MulLabTree. The population values are per-species-clade (per-branch in network), but of course more than one node in MulLabTree may correspond to the same species. The other complications
are that tips are different from internal nodes, and that nodes which roots of tetratrees or just below, as well as the root are special cases. It collects unions (which represent sets whose elements identify a species and a sequence) from the nodes and then sorts them so that sets of nodes with same species clade are grouped together. This mainly does what is required, since nodes with the same species clade are treated the same. Calls \texttt{fillinpopvalsforspunionNoDiploids()} and \texttt{fillinpopvalsforspunionTwoDiploids()} to deal with a set of nodes with same species clade.

\texttt{fillinpopvalsforspunion()}. For \texttt{fillinpopvals()}. Deals with a one union of species indices, that is, for all nodes in the multiply labelled tree which contain a particular group of species in their clade.

\texttt{fillinpopvalsforspunionTwoDiploids()}. This copies 0 to 3 population values into a set of nodes with same species clade. Quite complex. In two diploids case, get one of: (1) one diploid node (tip or root); (2) one foot node, in case where feet meet different diploid branches; (3) two nodes from different tetratrees; (4) two nodes which are two feet of tettree meeting one diploid branch; (5) three nodes which are two tetros and a leg-join. I have written the code (2011-08-12) in the hope it works for more general cases (bigger diploid tree, more than one tetraploid tree) but that is untested.

\texttt{fillinpopvalsforspunionNoDiploids()}. Similar to but simpler than previous method. In no diploids case, either get two nodes from different tetratrees or two tetros and the root.

\texttt{geneTreeInMLSubtreeLogLikelihood()}. Recursive, for \texttt{geneTreeInMLTreeLogLikelihood()}. Calls \texttt{branchLLInMULtreeNoDiploids()} or \texttt{branchLLInMULtreeTwoDiploids()}.\n
\texttt{branchLLInMULtreeTwoDiploids()}. Likelihood calculation of gene tree in multiply labelled tree for one branch in the case of no diploids. This\n
\texttt{branchLLInMULtreeNoDiploids()}. Likelihood calculation of gene tree in multiply labelled tree for one branch in the case of no diploids. It collects information (coalescent heights, start and end populations, number of lineages) and calls \texttt{limbLogLike()}. For some nodes it involve before and after hybridization within branch.

\texttt{limbLogLike()}. A 'limb' is part or all of a branch in which the population varies linearly (no hybridization or other jumps). This is used by \texttt{geneTreeInMLSubtreeLogLikelihood()}.\n
\texttt{limbLinPopIntegral()}. For \texttt{limbLogLike()}. Comparators:

\texttt{FOOTHEIGHT\_ORDER.compare()}. For \texttt{MulLabTree()}.\n
\texttt{SPUNION\_ORDER.compare()}. For \texttt{fillinpopvals()}. This is quite a complex sort, see \texttt{fillinpopvals()}. Its main task is to sort the nodes (defined by clades of (species, sequence) pairs) so that nodes with the same species clades are grouped together. But it also sorts the groups so that tips occur first, and it sorts nodes within groups in a well-defined way that \texttt{fillinpopvalsforspunionNoDiploids()} and \texttt{fillinpopvalsforspunionTwoDiploids()} rely on. More detail follows.

The comparator sorts the unions of (species, sequence) pairs (SpSqUnions) so that all unions containing the same set of species (ignoring sequence) are grouped together. Call the sets of SpSqUnions for the same species a group. There can be 1, 2 or 3 SpSqUnions in a group. The groups are sorted in order of increasing number of species (clade size). All groups for a single
species (a tip in the network) come first, then those groups for two species, and so on to the root for all species. For groups that have equal numbers of species, a lexicographical ordering using species indices is used.

Within each group, species and sequence information is used to sort the 1 to 3 SpSqUnions. The size of the ‘clade’ of (species, sequence) pairs is used first in the comparison, which ensures that the three nodes with the same species - corresponding to two roots of tetratrees in the multiply-labelled tree plus a leg-join - are ordered so that the two roots come first.

### 14.5 AlloppSpeciesNetworkModel methods

**AlloppSpeciesNetworkModel()**. Constructor. Made from an AlloppSpeciesBindings, a popvalue, and a flag for single hybridization (2011-08-12 this must be true). 2011-08-12, it calls `makeInitialOneTetraTreeNetwork()` or `makeInitialOneTetraTreeTwoDiploidsNetwork()` to make a random Yule-type tree with a leg to diploid history, then scales it to be shorter than `aspp.initialMinGeneNodeHeight()`. It makes a population Parameter of right size with values all equal to popvalue. Then it converts this to a multiply labelled tree, that is, a MulLabTree.

There are also two constructors for testing.

- **scaleAllHeights()**. Scales all heights by calling method of same name in each AlloppLeggedTree. Used by constructor and by operator.
- **coalescenceIsCompatible()**. Passed a gene coalescence height and union. Called from AlloppSpeciesBindings to check if a node in a gene tree is compatible with the network.
- **clearCoalescences()**. Called from AlloppSpeciesBindings to remove coalescent information from branches of nullabtree.
- **recordCoalescence()**. Called from AlloppSpeciesBindings to add a node from a gene tree to its branch in nullabtree.
- **sortCoalescences()**. Sorts coalescences within each branch of the multiply labelled tree.
- **recordLineageCounts()**. Records the number of gene lineages at nodes of the multiply labelled tree.
- **geneTreeInNetworkLogLikelihood()**. Calculates the log-likelihood for a single gene tree in the network. Requires that clearCoalescences(), recordCoalescence(), recordLineageCounts() called to fill nullabtree with information about a particular gene tree’s coalescences first.
- **getTipCount()**. Returns total number of tips of all AlloppLeggedTree’s. Used by AlloppNetworkPrior.
- **getEventHeights()**. Returns array of ‘event’ heights in the one-tetra-tree case. Used by AlloppNetworkPrior. An ‘event’ is not yet clearly defined. It should include speciations, but not sure what else.
- **getNumberOfDiTrees()**. Returns number of diploid trees.
- **getNumberOfTetraTrees()**. Returns number of tetraploid trees.
- **getNumberOfNodeHeightsInTree()**. Passed ploidy and index, returns the number of node heights in a AlloppLeggedTree in the network.
getHomoploidTree(). Passed ploidy and index, returns a tree in the network.

mulabTreeAsText(). For testing. Calls asText() in MulLabTree.

beginNetworkEdit(). For operators.

endNetworkEdit(). For operators. Remakes multiply labelled tree and calls fireModelChanged().

public String getName(). Returns model name.

public int scale(double scaleFactor, int nDims). Operator. Scales all heights, or scales
one node height.

handleModelChangedEvent(). Calls fireModelChanged().

handleVariableChangedEvent(). Does nothing.

protected void storeState(). Makes copies of all trees in network.

protected void restoreState() Restores copies of all trees in network, and remakes multiply
labelled tree.

protected void acceptState(). Does nothing.

public Type getUnits().?

public void setUnits(Type units).?

Private:

makeInitialOneTetraTreeNetwork(). For simple case of one tetraploid tree and no diploids.
Assumes a history before root of a diploid speciating, the two diploids (or two descendants)
forming a hybrid, which speciates at the root of the tetraploid tree.

There is also a version of makeInitialOneTetraTreeNetwork() for testing.

makeInitialOneTetraTreeTwoDiploidsNetwork(). For case of one tetraploid tree and two
diploids.

numberOfPopParameters(). Calculates the number of pop parameters, 2011-08-12 only for
no-diploid and two-diploid cases.

union2spseqindex(union). Passed union for a tip, hence only containing one (species index,
sequence index) pair. It returns the index of that.

Testing:

testExampleNetworkToMulLabTree(). Builds arrangements of trees with legs to test conversion to
the multiply labelled tree.