HÖGSKOLAN SKÖVDE

Institutionen för kommunikation och information
Examensarbete i datavetenskap 20p (Bioinformatik 20p)
C-nivå
Vårterminen 2005

Probability calculation of orthologous genes

Alice Lagervik Öster

HS-EA-DVA-2005-001

Probability calculation of orthologous genes

Submitted by Alice Lagervik Öster to University of Skövde as a dissertation for the degree of B.Sc., in the School of Humanities and Informatics.

2005-09-12

I certify that all material in this dissertation which is not my own work has been identified and that no material is included for which a degree has previously been conferred on me.

Signed: ____

Probability calculation of orthologous genes

Alice Lagervik Öster

Abstract

The aim of this thesis is to formulate and implement an algorithm that calculates the probability for two genes being orthologs, given a gene tree and a species tree. To do this, reconciliations between the gene tree and the species trees are used. A birth and death process is used to model the evolution, and used to calculate the orthology probability. The birth and death parameters are approximated with a Markov Chain Monte Carlo (MCMC). A MCMC framework for probability calculations of reconciliations written by Arvestad et al. (2003) is used. Rules for orthologous reconciliations are developed and implemented to calculate the probability for the reconciliations that have two genes as orthologs. The rules where integrated with the Arvestad et al. (2003) framework, and the algorithm was then validated and tested.

Keywords: Phylogeny, Reconciliation, Orthology, Birth and Death process, Markov Chain Monte Carlo.

Acknowledgment

I would like to tank my supervisor Lars Arvestad, for all help and support during the way. I would also like to tank my husband, Daniel Öster, for all the help with the implementation and for all the support during the whole work.

Table of Contents

1 Introduction

Today exploding rates of genome sequence data are made available through different gene projects. This creates a demand for computerized analyzing methods for biological problems (Attwood and Parry-Smith, 1999). There are different ways to investigate genes and genomes. Some scientists use functional genomics research methods (*Functional and Comparative Genomics Fact Sheet*, 2004), to understand human and other species on a molecular level. Meanwhile other scientists use comparative genomics to find similarities between different species. When investigating new genes through homology analysis (comparative genomics), the most reliable functional information can be found in orthologs to the gene (Storm and Sonnhammer, 2002). Two genes that are orthologs have their last common ancestor in a speciation, and are paralogs when they have diverged from the last common ancestor through a duplication. One of the many purposes of examining orthologous genes is that they are/have been the same gene in different species. Orthologs therefore often tend to have similar or equal function. This is why orthologs can be used with high reliability to transfer knowledge from a gene whose function has been determined by experimental methods to an unknown or novel gene (Storm and Sonnhammer, 2003; Remm et al., 2001). The ability to annotate gene function in this way is very helpful since for example it is not feasible to do *in vivo* experiments with human genes to get knowledge about the gene function (Remm et al., 2001). The possibility to use orthologs can also make the annotation process faster, and thereby cheaper.

With the purpose of finding orthologs, different algorithms have been developed. These algorithms are based on different methods. One method is to do BLAST (Altschul et al., 1990) searches, which can find homologous genes that may be orthologs. BLAST compares the investigated sequence with sequences in a database, by making pairwise alignments with the sequences in the database. It is assumed that two sequences get a higher alignment score if they are orthologs. The main problem when using BLAST is that it reports local similarities, and orthologous sequences are expected to be similar over the whole length of the sequence (Remm et al., 2001). Because of this BLAST can miss to report orthologous genes. This kind of method, demands a lot of work and interpretation from the user.

There are also algorithms that use phylogenetics trees to find orthologs. Phylogeny is the evolutionary relationship of species, and can be expressed as a phylogenetic tree (Durbin et al., 2002). Phylogenetics, which is the analysis of phylogeny, can be used to investigate gene families through reconciling genes with their corresponding species tree. The evolution of genes is the results of different evolutionary events like speciation, duplication, or lateral gene transfers (Arvestad et al., 2003). Even if the genes have evolved along with the species, it is not always the case that the gene tree looks the same as the corresponding species tree (Nei et al., 1997). In this work, only gene speciation and gene duplication will be considered.

One of the algorithms using phylogenetic trees is the Orthostrapper (Storm and Sonnhammer, 2002) that analyses a set of bootstrap trees and calculates orthology support levels. The bootstrap trees are developed through a method called bootstrapping which is a method that is used to approximate sampling error. The sequence data is used in a random sampling with replacement. The sampling generates pseudoreplicates of the original data that will be used to build evolutionary trees. The two steps are repeated and results in a set of bootstrap trees. From this set of trees, the sampling error of the sample can be extracted (Page and Holmes, 1998). The sampling error is then used to calculate a confidence value of the tree. The confidence value must then be interpreted by the user.

All the algorithms available today give a measurement that the user needs to interpret, instead of probabilities. For example if the confidence value is close to 1.0, are they probable orthologs then?

When investigating relationships of genes, reconciliations can give a good picture of how the genes and the species are related. A reconciliation is when a gene tree evolves inside the corresponding species tree. The gene tree represents the evolution of the genes and the species tree represents the evolution of the species. In a reconciliation the locations of all the duplications and speciations in the species tree becomes visible. When looking at the reconciliation it is possible to identify which of the genes that are related and in which way. How can a reconciliation be constructed when the relationship of the genes is unknown? Arvestad et al. (2003) has developed an algorithm that can calculate the probabilities for all reconciliations. The algorithm can also calculate which of the reconciliations that have the maximum likelihood.

1 Introduction

This is the first successful introduction of its kind of probabilistic methods in phylogenetics (Arvestad et al., 2003).

The algorithm developed in this project will calculate the probability of two genes being orthologs.

The resulting program will be a helpful tool for the evaluation of possible orthologs. By outputting the calculations made, the correctness of the developed algorithm will be verified. The results of the calculations will be compared with the results from an approximating algorithm developed by Arvestad et al. The approximating algorithm uses the same evolutionary model for its approximations as the developed algorithm.

The report is structured as; Chapter 2 explains some of the terms and methods that are used. Chapter 3 presents two related works, the Orthostrapper (Storm and Sonnhammer, 2001) and RIO (Zmasek and Eddy, 2002). The problem definition, the aim and the objectives of the thesis, is outlined in chapter 4, and the method used to reach them can be read about in chapter 5. Chapter 6 contains the results of the work and in chapter 7 the results are discussed. The conclusions of the work are presented in chapter 8 and future work chapter 9.

2 Background

In this chapter, some background information needed to understand the subject in the dissertation is outlined. First in 2.1 the concept of phylogeny is explained, and 2.2 define orthology. In 2.3 describes the trees used and how a reconciliation is constructed. In 2.4, the theory concerning the likelihood calculations is outlined, first in 2.4.1 Bayesian inference in phylogenetics is explained, 2.4.2 declares the birth and death process, and 2.4.3 explains the Markov Chain Monte Carlo method.

2.1 Phylogeny

Phylogeny is a method that can be used when trying to figure out how evolution has created all living things from a single common ancestor (Huelsenbeck and Ronquist 2001). It is an historical estimation of how species relates to each other and can be expressed as a phylogenetic tree (Durbin et al., 2002). Because of the different kinds of biological events occurring there can be differences in the phylogeny of genes and those of species (Page and Charleston, 1996). One consequence of these events is multigene families. Multigene families are genes that through duplication have diverged into genes located in different loci but in the same species. For example, the human hemoglobin consists of two subunits of α globin, which is available in two different types in two different loci. The two subunits of β globin exist in four different types coded by different loci. Because of this evolution of species and gene, it is not easy to figure out the evolutionary paths. When using reconciliations the history of genes become clearer and easier to understand.

2.2 Orthology

In this work, the definition of orthologs made by Fitch (1970) is used. The definition states that two genes are orthologs if they have evolved from a common gene and the last common ancestor diverged because of a speciation. Paralogous genes are gene that have diverged though duplication.

2.3 Trees and reconciliations

A species tree is an evolutionary map over how different species have evolved from a common ancestor. The trees are rooted, meaning that they have one node that represents their last common ancestor, e.g. the species that all the other species in the tree have evolved from. They are also binary, which means that every node can have just two children. The event when a tree node is divided into two nodes is called a bifurcation, or a split, and are expected to occur every α years. Every leaf is labeled with the species for which the tree represents the relationships (Bonizzoni et al., 2003). In Figure 1, the first tree represents a species tree. The root node represents species A, which species B and E have evolved from. Species B have then evolved to species C and D.

The gene trees are built in a similar way as can be seen in the second tree in Figure 1. In this gene tree, gene 1 represents the root from which genes 2 and 5 evolved. Gene 2 then evolves to gene 3 and 4.

Reconciliations can be made between the species tree and the gene tree. These reconciliations show the relations between the two trees (Arvestad et al., 2003). The leaf nodes of the gene tree are located in the corresponding leaf nodes of the species tree.

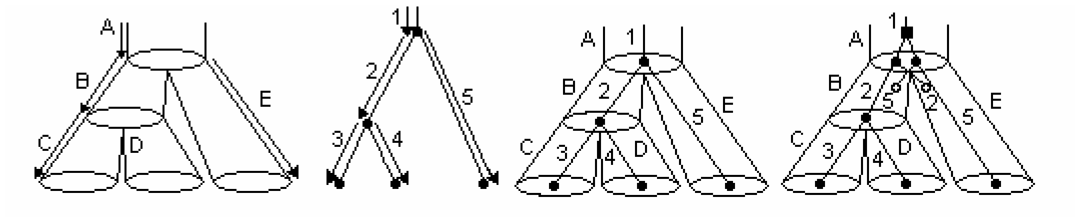

Figure 1: The first tree is the species tree with the nodes marked with the letters A to E. The arrows show the time. The circles represent the split of the species to two other species, except for the circles at the bottom (the ends of species C, D and E) which represent the species as they are today. The second tree is the corresponding gene tree, with gene nodes marked with the numbers 1 to 5. In the first reconciliation, made by the specie tree and gene tree, the gene splits of gene node 1 is located in the species tree split of specie A and the split of gene nodes 2 are located in the species tree splits of species B (speciations in the gene tree are represented by filled circles). This means that the genes 3 and 4 are orthologs, as is 5 with both 3 and 4. In the second reconciliation the gene split is lifted above the first split in the species tree (duplications in the gene tree are represented with a filled square), which means that it is a duplication rather than a speciation. This means that gene 5 is not orthologous with the other genes. Gene 2 is extinct in species E and gene 5 is extinct in species B, this is represented by an unfilled circle. It is assumed that these genes are extinct because they are not represented in the species today.

Then the gene tree nodes are placed inside the species tree. There can be many different reconciliations of the same species and gene tree, by placing the splits in the gene tree in different places in the species tree. The tree where all the gene splits are located as far down in the species tree as possible is called the most parsimonious reconciliation. Two different reconciliations for the same trees can be seen in Figure 1. There are many computer programs that perform reconciliations, for instance those developed by Arvestad et al. (2003) and Zmasek and Eddy (2001).

2.4 Probability calculations

There can be many different reconciliations because of the many different ways the gene tree can be placed inside the species tree, for example see Figure 1. So, how does one know which is the right tree? And how can the probability for two genes being orthologs be calculated? To be able to calculate these probabilities some methods and formulas are used.

2.4.1 Likelihood

A probability for an event E can be expressed like Pr[E]. The conditional probability Pr[E | V], expresses the probability of event E given that event V has occurred. The likelihood calculated here is the conditional probability of the data given the constructed tree Pr[Data | Tree_i]. The data are represented by birth and death parameters (see section 2.4.4) and the species tree (that is considered fixed), and Tree_i is a one constructed gene tree.

2.4.2 Bayesian inference of phylogeny

To calculate the probability for a reconciliation, Bayesian inference can be used. Bayesian inference is an approach that rephrases uncertainty as probability (Radford, 1998). Bayesian inference in phylogeny is based on the posterior probability of a gene tree. The posterior probability is the probability of Tree i given the observed data, and can be formulated as Pr[Tree_i | Data]. This probability can be computed through rewriting the formula using Baye's theorem (Equation 1). The posterior probability can be calculated using the prior probability of a phylogeny, Pr[Tree_i], the probability of the data Pr[Data] which is the birth and death parameters (see section 2.4.4), the species tree, and the likelihood which is the conditional probability, Pr[Data | Tree_i], for each tree.

$$\Pr[Tree_i \mid Data] = \frac{\Pr[Data \mid Tree_i] \times \Pr[Tree_i]}{\Pr[Data]}$$

Equation 1: Baye’s theorem when used for probability calculations of phylogenetic trees after Huelsenbeck et al, 2001.

Since the prior probabilities for the species tree and the birth and death parameters often are hard to estimate, they are considered to have a uniform distribution and thereby not included in the calculations (Arvestad et al., 2003). The posterior probability of a tree can be seen as the probability that the tree is accurate (Huelsenbeck et al, 2001). The likelihood can be obtained by summation of the probabilities for all the reconciliations with the birth and death parameters. The birth and death parameters are estimated by a Markov Chain Monte Carlo (MCMC) method (Gilks et al, 1996), see section 2.4.4.

2.4.3 Birth and Death Process

The birth and death process is a process describing in which ratio the genes have been “born” and have “died”. A gene is born when a duplication or a speciation has occurred and it dies when it get extinct.

In a birth and death process for genes, the birth rate (λ) is the frequency with which genes duplicate. The genes are estimated to be lost with the frequency of the death rate (μ). The birth and death process can be used to obtain the a priori probability of a tree (Arvestad at el., 2003). To get estimated values for λ and μ a MCMC method is used.

2.4.4 Markov Chain Monte Carlo (MCMC)

MCMC is a method that can be used when integrating over high-dimensional probability distributions for deduction of model parameters (Gilks et al, 1996). The Monte Carlo-method samples over a Markov Chain and will converge to a stationary distribution. Here it is used to obtain the posterior distribution $\Pr[\gamma, \lambda, \mu \mid G]$, where γ represents a reconciliation, λ and μ the birth and death rates and G represents the gene tree. The MCMC algorithm used can be explained in five steps which can be seen as pseudo code in Figure 2. Θ is the triplet of the parameters γ, λ and μ.

The starting point of Θ is randomly chosen and because of this, the results from the first iterations should not be considered because of the time it takes for the chain to "forget" the starting state. This first period is called "burn in" (Gilks et al. 1996). The stationary probability for a state is then calculated from the number of times the state has been visited under the simulation, in relation to the total number of visits of all states. The number of iterations preformed by the chain is set before starting the MCMC. For further details about MCMC, see Gilks et al.,1996.

```
1. Randomly chosen starting condition for ϴ
2. Perturb ϴ to create ϴ', by changing elements of ϴ
3. Compute α(ϴ,ϴ') = min $\left\{1, \frac{q[\theta', \theta]p[\theta']}{q[\theta, \theta']p[\theta]}\right\}$
   Where p[ϴ] is the likelihood function
4. Accept the proposed state ϴ' with the probability α
5. Output ϴ, and continue from 2
```

Figure 2: Pseudo code for one iteration of the MCMC algorithm.

3 Related work

In this chapter three related works will be presented. First in section 3.1 the program Orthostrapper (Storm and Sonnhammer, 2001) will be presented, and the program RIO (Zmasek and Eddy, 2002) will be presented in section 3.2. A sampling program (Arvestad et al.) will be presented in 3.3.

3.1 Orthostrapper

Orthostrapper[1] is a web available program for finding orthologous genes between two species or groups of species. As input Orthostrapper takes a multiple alignment in fasta-format, with the phylogenetic relationships between the sequences included. The input sequences should also be assigned to one of four different groups. The two first groups are called the primary groups, and should contain sequences from the two (groups of) species that one wants to find the orthologous relationships between. The third group is called the outgroup and should contain sequences that are from a distant species to the primary groups. The last group is the blank group that will not be included in the analysis. In the blank group should sequences which relationship to the primary group is unclear be placed. These groups of sequences are then used in an algorithm to detect orthologous relationships. Note that no species tree is used in this algorithm.

A phylogenetic tree is built from the input sequences and used to create so called sub-hypothesis for all orthologous sequences in the tree. To find this sub-hypothesis each sequence in the tree is examined as follows; for each leaf node (sequence) in the tree, go up one node in the tree and look at which group/groups the sequences of new branches belongs to. There are three alternatives. One; if all new branch sequences are assigned to the same group as the current sequence, then go further up in the tree. Two; if all new branch sequences belong to the other primary group than the current sequence, then assign orthology between the current sequence and all the new branch sequences, then continue with the next sequence. Three; if some of the new branch sequences belongs to the outgroup or there are sequences from the both primary groups, continue with next sequence.

[1] http://orthostrapper.cgb.ki.se/

A large number of bootstrap trees are then created. A bootstrap tree is created by randomly picking sequences, with replacement, from the original alignment. Trees, called bootstrap trees, are then built from these picked sequences by the neighbor joining algorithm. The bootstrap trees are analyzed for orthologs, in the same way as described above. If an orthologous relationship is found in the bootstrap tree, that sub-hypothesis score is increased with one. All the bootstrap trees add up the scores of the sub-hypotheses. The output is a matrix of bootstrap values for all possible sequence combinations. For more details see Storm and Sonnhammer (2002).

3.2 RIO

The RIO algorithm (Zmasek and Eddy, 2002) has defined three classes of different relationships between sequences that the authors think is important when trying to transfer functional information from a known sequence to an unknown. The three classes are super-orthologs (subtrees that only contain speciations), ultra-paralogs (subtrees that only contain duplications, which also mean that the sequences belong to the same species), and subtree-neighbors (sequences that are descendants for an ancestor on level *k* of the tree, where leafs are on level *k*=0, their parent on level *k*=1, etc.).

The input should be a sequence called Q, with unknown function, and a multiple alignment called A, with sequences from Q's family derived from the pham database. To find out which protein family sequence Q belongs to, hmmpfam from the HMMER package can be used. A profile HMM called H, for Q's family is also required as input.

Sequence Q is aligned with the multiple alignment of its family. Then the 'new' alignment is used to create a number of bootstrap alignments. For each of the bootstrap alignments, maximum likelihood pairwise distances matrices are calculated using amino acid substitution matrices (like BLOSUM or Dayhoff PAM). Then neighbor joining is used to build one phylogenetic tree for each bootstrap alignment. The trees are then rooted with an algorithm, written by the RIO authors (Zmasek and Eddy, 2002).

To be able to assign duplication or speciation to each split in the bootstrap gene tree, it is compared with a species tree.

The bootstrap trees are then analyzed, sequence by sequence. If a sequence have another sequence as super-orthologs, ultra-paralogs, or subtree-neighbors the sequence pair gets plus one in confidence value. Values from all the bootstrap trees are summed separately for the three kinds of relation ships.

3.3 Sampling algorithm

The calculation framework developed by Arvestad et al., calculates all possible reconciliations. To get an approximation of the probability of two genes being orthologs Arvestad et al., developed a sampling algorithm, in this thesis called Beep. This sampling algorithm runs a MCMC over the birth and death parameters and calculates the probability for all reconciliations. Then the algorithm randomly picks 1000 reconciliations (with replacement) from the calculation framework, and counts how many of the reconciliations that have the two genes as orthologs. The number of reconciliations having the genes as orthologs is divided with the number of reconciliations picked (1000). This sampling algorithm does not give exact probabilities, just approximations.

4 Problem Description

In this chapter the aim and objectives of this dissertation will be explained. First in 4.1 the problem is defined, 4.2 describes the aim of the work, and last in 4.3 the objectives set up to reach the aim are outlined.

4.1 Problem Definition

Orthologous genes are genes with the last common ancestor in a speciation. Orthologous genes often have the same function in different species and are therefore useful when annotating novel genes. There are different methods for finding orthologous genes. Many of the methods use similarity scores provided by for example BLAST (Remm et al., 2001). There are many reasons why similarity scores are not always a good way to find orthologs, for example these methods do not use species tree information (Storm and Sonnhammer, 2003). Several of the automated methods have some sort of confidence value (RIO, Zmasek and Eddy, 2002) or support values (Orthostrapper, Storm and Sonnhammer, 2001) for the found orthologs. A confidence value is a value calculated by an algorithm. Confidence value is created by the author of the algorithm. When receiving confidence values as a result, it can be hard to interpret and impossible without the knowledge of the algorithm. If the value is close to say 1, then it is probable that the two genes are orthologs. But what if the value is 0.8? It does not mean that it is 80% probable that the two genes are orthologs. One cannot really be sure of what it means. With a probability value of 0.5 you know that it is a fifty/fifty chance that the two genes are orthologs. Probabilities are well defined which makes them easy to interpret.

Phylogenetics is the analysis of the evolutionary relationship between genes, and can be used for finding orthologs. The method in this thesis is a reconstruction of the gene tree together with the species tree, a node-to-node mapping called reconciliation. When doing the reconciliations the birth and death process is used. Because of the lack of information of the values of the birth and death parameters they have to be estimated. In this project, the birth rate will be represented with the variable λ, and the death rate will be represented with μ.

A problem when searching for orthologs with reconciliations is that there are many possible ways the gene tree could have evolved and we do not know which of them that is the ´real´ reconciliation. When dealing with uncertainties it is always best to use probabilities (Arvestad et al., 2003). To get a good measurement of how likely it is for two genes to be orthologs probabilities will be used. When using probabilities the results are easier to interpret, and the values are calculated in a mathematically sound way (Arvestad et al, 2003).

The desired probability in this project is for two genes being orthologs (gene a and gene b being orthologs is written like a~b), given a gene tree, represented by G, a species tree, represented by S, and the birth and death parameters, λ and μ. This can be expressed like Pr[a~b|G, λ, μ, S]. To calculate this probability, the probability of the gene tree where the two genes are orthologs, Pr[a~b, G|λ, μ, S], has to be divided by the probability of the gene tree, Pr[G |λ, μ, S]. Both of these probabilities are dependent on the species tree and birth and death probabilities, as shown in the formulas. The resulting equation for two genes being orthologs can be seen in Equation 2.

$$\Pr[a \sim b \mid G, \lambda, \mu, S] = \frac{\Pr[a \sim b, G \mid \lambda, \mu, S]}{\Pr[G \mid \lambda, \mu, S]}$$

Equation 2: To get the probability for two genes being orthologs given a gene tree, a species tree, the birth and death probabilities the following equation must be set up.

To calculate the probability of the two genes being orthologs, Pr[a~b, G|λ, μ, S], the sum of the probabilities for all the reconciliations having *a* and *b* as orthologs, should be calculated. To be able to calculate this from all reconciliations, rules have to be set up that will separate the reconciliations which have the genes as orthologs from all the other reconciliations.

4.2 Aim

The aim of this work is to formulate an algorithm that calculates the probability of two genes being orthologs, given a gene tree and a species tree.

4.3 Objectives

In the following sections the objectives to fulfill the presented aim are presented. The objectives are setup stepwise.

4.3.1 Analysis

The probability calculations of the reconciliations are done within a framework developed by Arvestad et al (2003). To be able to understand this framework an analysis of how it operates has to be done. This analysis should give all the necessary knowledge needed to start formulating the rules that will separate the reconciliations that have the two genes as orthologs from the reconciliations that do not.

4.3.2 Formulating rules

The calculation framework developed by Arvestad et al, calculates all possible reconciliations with the given gene and species trees, and also the probability for each of the reconciliations. To gain the probability for the gene tree Pr[G] the probability for all reconciliations are summed. Baye's theorem is used to receive the joint probability for two genes being orthologs Pr[a~b , G], the probability for all reconciliations having the two genes as orthologs Pr[a~b | G] has to be divided with the probability of the gene tree. The algorithm that is to be developed needs to calculate two probabilities hence the probability for all reconciliations having the two genes as orthologs, and the probability for the gene tree. Because the calculation framework already calculates the probability for all reconciliations the additional part should calculate the probability of all reconciliations having the two genes as orthologs. To gain this probability the reconciliations having the two genes as orthologs have to be separated from the other, so that the probabilities for these reconciliations can be summed. This means that the reconciliations that have the two genes as orthologs must be separated from all other reconciliations. To do this rules have to be formulated. When the rules are used, the probability for the reconciliations that has the two genes as orthologs are the only to be calculated.

4.3.3 Implement rules

The formulated rules for the sought reconciliations should then be implemented and integrated into the probability calculation framework. The algorithm should calculate the probability for all reconciliations having the two genes as orthologs. The obtained probability should then be divided with the probability of all possible reconciliations to obtain the wanted probability for the two genes to be orthologs. This implementation would be a good way to evaluate the rules (see 4.3.2) that will be set up.

4.3.4 Testing

To be able to both confirm the rules (see section 4.3.2) and the implementation (see 4.3.3), testing has to be done. First a manual verification that the right probabilities are calculated in the algorithm is conducted. This is done by outputting the probability for each node-to-node pairing in all reconciliation from the program. The probabilities are then manually checked so that the probabilities that should be calculated are calculated and the ones that should have been set to zero really are zero.

Then experiments should be executed with different sizes of trees, and the CPU time of the execution measured. The CPU time should then be compared to the CPU time of execution for the sampling program (see 3.3) developed by Arvestad et al. The probabilities for the found orthologs should also be compared.

5 Method

In this chapter the method that was used when carrying out the objectives will be presented. First in section 5.1 it is explained what the analysis of the calculation framework led to. In 5.2 the method used when formulating the rules is outlined. In section 5.3 it is describe how the implementation was conducted, and in 5.4 the method of testing.

5.1 Analysis

The analysis is divided into three sections; section 5.1.1 present the format of the input of the framework, section 5.1.2 explains the trees used and last section 5.1.3 will give details of the calculation framework.

5.1.1 The input

The input needed to execute the algorithm is one species tree and one gene tree. The algorithm also needs a matching between the two trees. The trees are written in a format that is called Newick format. Newick format is a way to write trees as labels and nested parentheses. The trees in Figure 2 is written as ((A , B) , (C , D)) and ((A:0.2 , B:0.2):0.3 , C:0.5). The length of the branches can also be included in the format, as can be seen in the later format and in the rightmost tree in Figure 2.

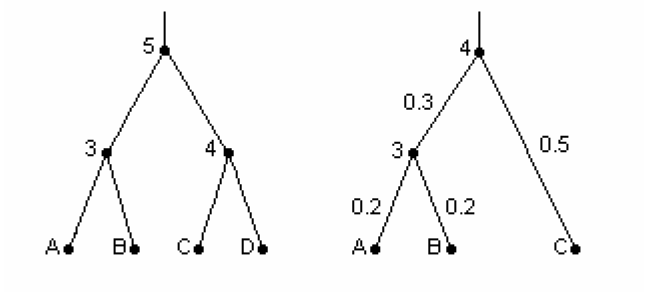

Figure 2: The leftmost tree would be written like ((A , B) , (C , D)). It should be read like; the leaf nodes A and B are siblings and have a common parent (3), so is leaf node C and D also (with common parent 4). The two parents (3 and 4) will have a common parent, shown by the comma in the middle, which is the root node (5). In the rightmost tree the branch lengths are included, and it will be written like ((A:0.2 , B:0.2):0.3 , C:0.5) in Newick format.

The matching between the two trees contains the leaf nodes of both the trees in pairs. The wanted number of iterations in the MCMC can be set by the user otherwise the default value of 10 000 iterations is used. The start rates of the birth and death parameters can also be set.

5.1.2 The trees

The trees that are used in the algorithm are written in Newick format. That means that only the leaf nodes are labeled. The program executing the algorithm will assign unique numbers to each node, in both the trees. In the reconciliation, each node in the species tree will be coupled with one node from the gene tree. There can be several gene nodes coupled to the same species node, and one gene node can be coupled with several species nodes.

There is also a variable *k* in the algorithm that shows if the current gene node has been split in the current species node or not. If there are duplications of the gene node located before the split of the species node then *k* will be equal to the number of gene nodes present in the species node split. When the current gene node itself is placed in the species node split (a speciation), then *k* is one. The way the variable *k* is calculated and why is illustrated in Figure 3.

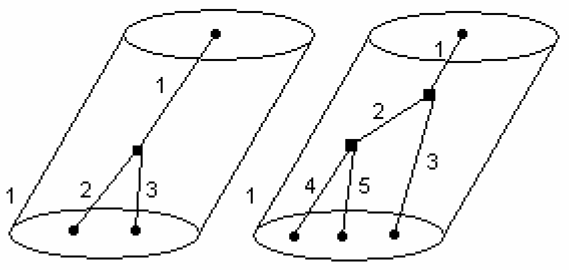

Figure 3: The figure shows the same species node 1, with two different reconciliations. In the leftmost reconciliation when the probability for gene node 1 should be calculated, the variable *k* will be 2. This is illustrating that there has been one duplication, resulting in two gene nodes. *K* will be 1 for both 2 and 3 in the leftmost species node. In the rightmost species node, *k* for gene node 1 will be 3, illustrating that two duplications have occurred.

In the notation used when building reconciliations the number of the species node is written first then the number of the gene node and last the variable *k*. The first reconciliation in Figure 3 is written like; (1, 1, 2): species node 1, gene node 1, one duplication resulting in two gene nodes in the species split of 1. (1, 2, 1) and (1, 3, 1):

species node 1, gene node 2 respectively gene node 3, no duplication have occurred on these gene nodes before the split of this species node.

5.1.3 The calculation framework

The calculating framework is divided into several different components. One component calculates the probability for the reconciliation, another handles the MCMC iterations. The different states in the MCMC chain will give different values for the birth and death rated used to calculate the probability for the gene tree thus all possible reconciliations. Just one of the parameters will be changed in each iteration. The original framework calculates the probabilities for all reconciliations of the gene tree and the species tree. There are different functions in the framework calculating different parts of the reconciliation. There are functions for calculating simple slice likelihood, slice likelihood, duplication likelihood and extinction likelihood.

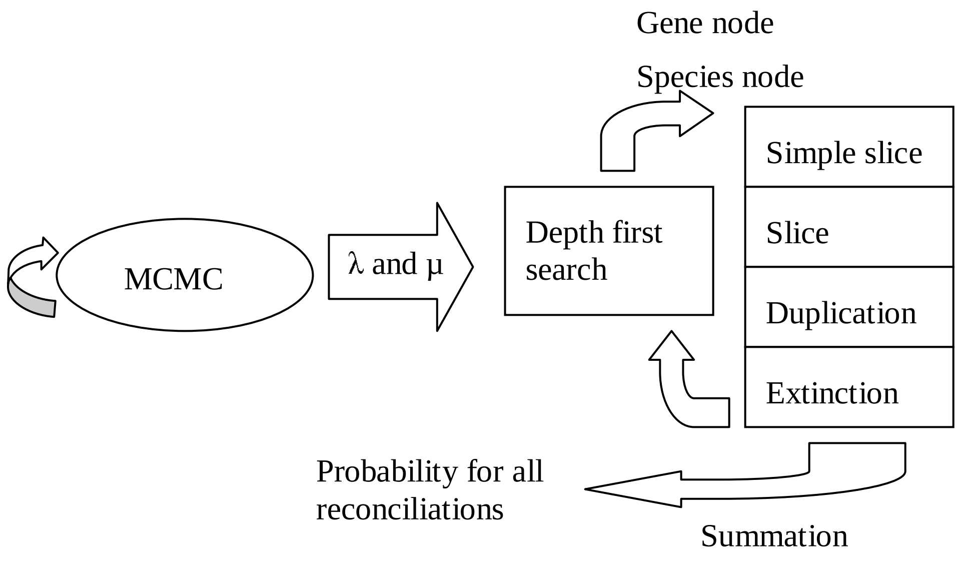

Figure 4: An overview of the calculation framework. For each of the values from the MCMC a depth first search in the species tree is done, and then the different calculation functions is called with one gene node and one species node in the right order. All the probabilities are then summed, to the probability for all reconciliations.

The simple slice function calculated the probability for that a gene leaf node has evolved from the root of the species tree. This is done by successively moving the gene node from the species leaf node to the species parent all the way to the species tree root node. The procedures of the simple slice calculations are showed in Figure 5, where the probability is calculated separate for each tree.

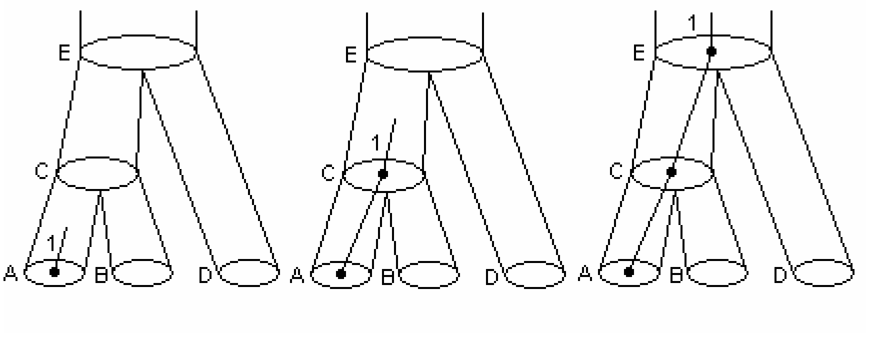

Figure 5: Simple slice calculations. Through the mapping of the leaf nodes between the species tree and the gene tree it is known that gene node 1 belong to species node A (first reconciliation). It is assumed that the gene node could have evolved along with species A and was present in species A’s parent in the tree, which is species C (second reconciliation). It is also assumed that the gene node 1 could have evolved along with species C which means that gene node 1 have been present in species E (third reconciliation).

The slice function calculates the same probabilities as the simple slice function, but for gene nodes other than the gene leaf nodes. Hence slice calculates the probability of the gene node in a position in the species tree and successively up to the root of the species tree. The calculations are illustrated in Figure 6.

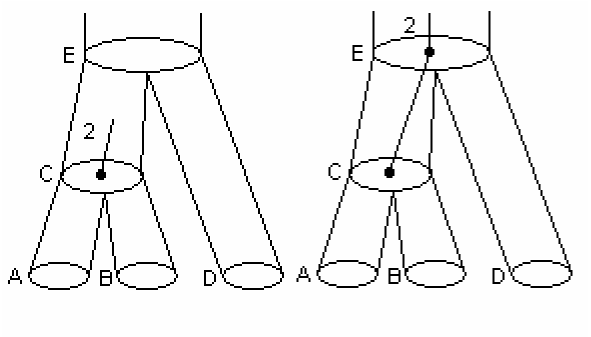

Figure 6: Slice calculations. Calculates the probability for gene node 2 to be in species node C (first reconciliation), then the probability that gene node 2 has evolved from species E (second reconciliation).

Extinction likelihood is calculated of all gene nodes that are lifted above their most parsimonious speciation, and then it is assumed that it has gone extinct in the species sibling node. In Figure 6, second reconciliation when gene node 2 is lifted up to species node E, then it is assumed that gene node 2 should have died in species node D. This is not showed in Figure 6 because of clarification of the slice calculation, it is instead showed in Figure 7.

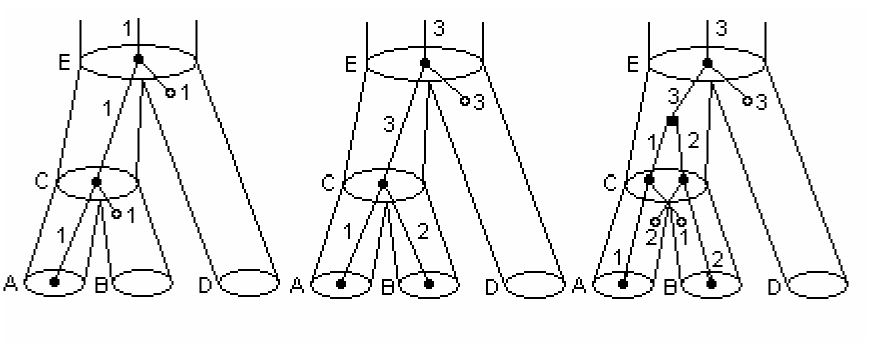

Figure 7: Extinction function. In the first reconciliation gene node 1 has evolved from the root of the species tree (species E), through species C and down to leaf species A. This means that gene node have gone extinct in species D and B, as can be seen by the unfilled circles. In the second reconciliation gene node 3 has evolved from the root of the species tree to species node C. Gene node 3 is then split into gene nodes 1 and 2 in the species split of species C to species A and B. This means that gene node 3 will be extinct in species D. No genes will be extinct in species A and B because the split of gene 3 is in the species split. In the third reconciliation the gene split of gene node 3 is lifted up from the species split and becomes a duplication. When the species split of C occurs, both gene nodes 1 and 2 will be copied into both species A and B. But they are not represented in species A and B as leafs, they are assumed to have gone extinct.

Duplication function calculates the probability for a gene node to be duplicated in a species node. All the splits in the gene tree below the current gene node is drawn successively up to occur (as duplications) in the current species node. An example is show in Figure 8.

The calculation framework starts with a depth first search starting in the root of the species tree, the calculation functions are then called in the right order with one species node and one gene node at a time. All the calculated probabilities are then summed to gain the probability for the gene tree.

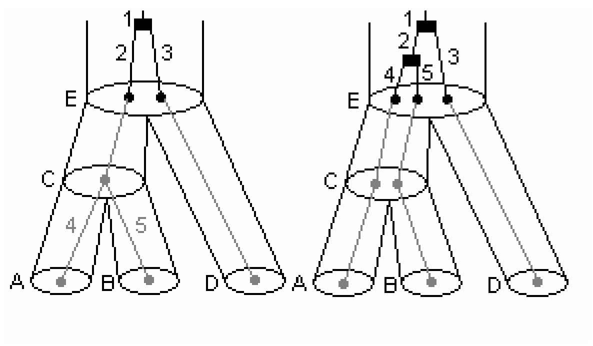

Figure 8: Duplication function. The first reconciliation show when the gene node split of gene node 1 (to gene nodes 2 and 3) are placed in species E. This is the first probability calculated. Then all the gene splits below gene node 1 in the gene tree is pulled up in the species tree successively. The second reconciliation show when the split of gene node 2 (to gene nodes 4 and 5) has been placed in species E. This is the second probability calculated. (The extinct nodes have been left out in both reconciliations for the sake of clarification.)

To obtain the probability just for the reconciliations having two gene's as orthologs the calculations of the reconciliations probability have to be altered, so that the last common ancestor of the two genes always are in the right speciation. The calculations should just calculate and sum the wanted probabilities. The rules should fixate this last common ancestor gene node in the split of the last common species node.

The output from the original framework is the probability of all the reconciliations. Now the probability of the orthologous reconciliations should be divided with that original probability, to gain the wanted probability.

5.2 The rules

When calculating the probability for the given gene tree, the probability for genes and gene splits located in different positions in the species tree are calculated. To get the probability for the gene tree, the probabilities for all reconciliations are summed. When calculating the probability for the gene tree where two genes are orthologs, only the probability for the reconciliations that have the genes most recent common ancestor reconciled as a speciation should be calculated. To be able to know which reconciliations to calculate, rules must be set up. The rules will then separate the reconciliations where genes *a* and *b* are orthologs from the other reconciliations. Two different reconciliations of the same species tree and gene tree can be seen in Figure 9.

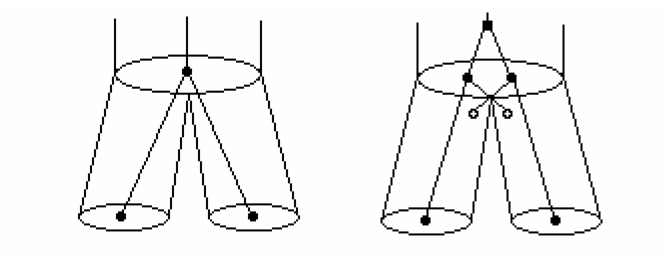

Figure 9: The first reconciliation has the two genes as orthologs because the last common ancestor is located in a species split in the species tree, and is thereby a speciation. The second reconciliation has the genes last common ancestor located above the first speciation in the species tree, which makes it a duplication, and thereby the two genes will also be extinct in the species nodes sibling (represented by empty circles). In the reconciliation to the right the two genes can not be orthologs, which means that the probability for the first reconciliation should be the only allowed in "orthologous" calculation.

The method used when formulating the rules for the "orthologous reconciliations" was to analyze the differences of a reconciliation, having the pair as orthologs and a reconciliation that did not. When doing a reconciliation it is assumed that all the gene duplications could have occurred above the root of the species tree. Every split of a gene node can be a duplication and lifted further up in the tree. This is not the case when two genes are supposed to be orthologs. The last common ancestor of the two genes then has to be placed in the species split of the last common ancestor species of the two genes. The last common ancestor of these orthologous genes will be called LCA, and the species node split LCA has to be located in to make the two genes orthologs will be called S_LCA. When LCA is placed in S_LCA, none of the descendants of LCA in the gene tree can be above S_LCA the species tree. The probability for the gene nodes below LCA to be over S_LCA must be set to zero. The gene nodes below LCA can not be extinct above S_LCA in the species tree either. The rules should check so just the probability for the reconciliations where the LCA gene node is located in the S_LCA species node is calculated. The probability for all the other reconciliations should be set to zero.

5.3 Implementation

When the rules have been formulated they should be implemented and integrated into the calculation framework. The implementation is done in C/C++. The original calculation framework calculates the probabilities of all the reconciliations between a gene tree and a species tree, which represents the probability for the gene tree. When the developed rules in this project are implemented they will calculate the probability for all reconciliations having the two genes as orthologs. The two calculated probabilities, of the gene tree and of all the reconciliations having the two genes as orthologs, are both used. The calculated probabilities for all orthologous reconciliations are divided with the calculated probability for all possible reconciliations to gain the probability for the two genes to be orthologs. An overview of the calculation framework after the rules have been implemented and integrated into it can be seen in Figure 10.

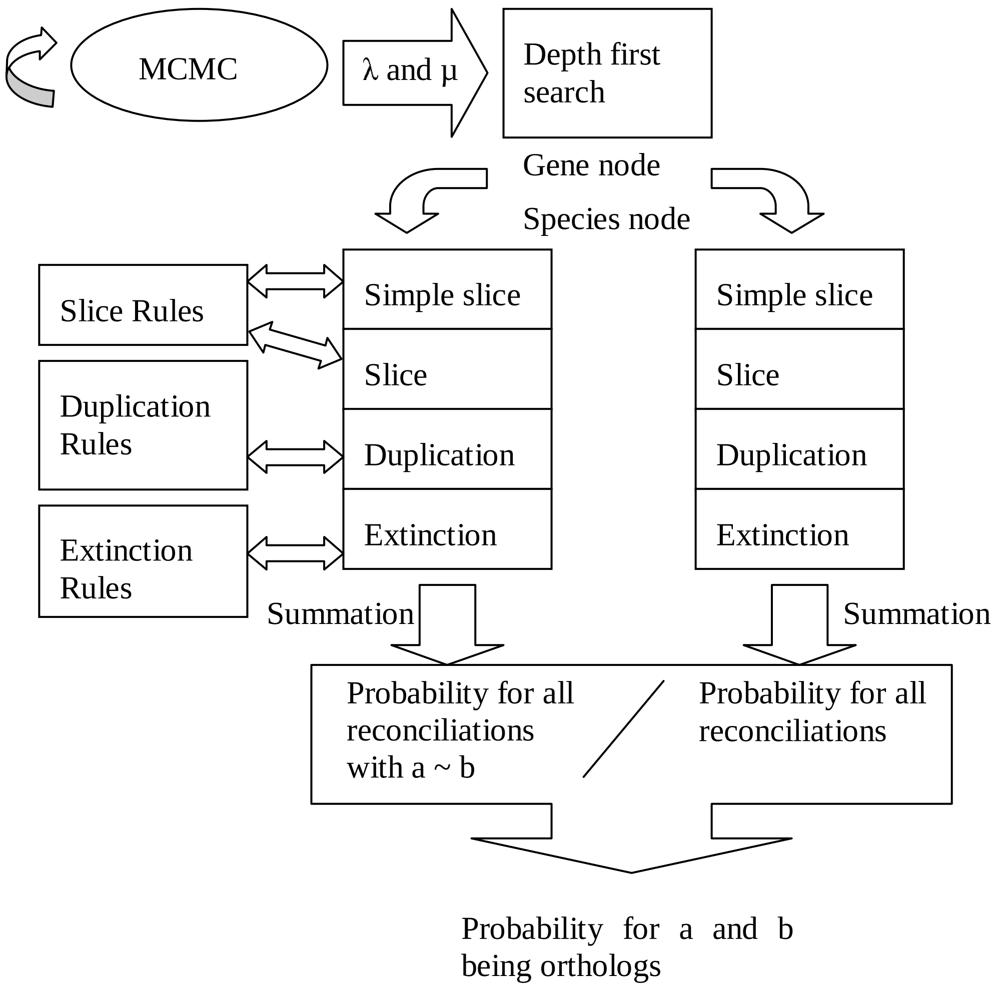

Figure 10: An overview of the calculation framework after the rules have been implemented and integrated.

5.4 Testing

The results from the implementation can not be compared directly with the results given from other ortholog finding algorithms as the Orthostrapper (Storm and Sonnhammer, 2001) and RIO (Zmasek and Eddy, 2001). This is partly because these methods take sequence data as input and the developed algorithm takes one gene tree and one species tree as input, and partly because the differences between confidence values and probabilities. Instead, the results will be compared with a sampling program written by Arvestad et al. (see 3.3). The sampling algorithm and the algorithm developed in this project both use the same model and therefore their results should be similar, but the execution time for the algorithms should differ. Both the algorithms will be executed with the same input and the CPU times will be compared. Different sizes of trees will be used to see if the difference in time will grow with the size of the trees. To know if it is the number of gene leaf nodes or the number of possible orthologs the time is dependent on the following trees sizes where chosen for the test.

Trees	# gene nodes	# species node	# pairs of orthologs
1	3	3	3
2	6	2	5
3	18	4	20
4	12	3	32

Table 1. Trees used for testing the differences in CPU time between the developed algorithm and the sampling algorithm. Their number of nodes and number of ortholog pairs.

Verification of the algorithmic results will also be done. Debug outputs from the probability calculations will be studied, to manually confirm that the right probabilities are set to zero, and the right ones are calculated. It will primary be done on small trees because of the extensive data output.

6 Results

This project has resulted in two results first the formulated rules second the received probabilities. The resulting rules will be presented in section 6.1, and the implementation results will be presented in section 6.2.

6.1 The rules

The analysis of the gene trees in different reconciliations and of how the probability calculations is done in the calculation framework led to the formation of three sets of rules containing a number of rules. The rules were to set all probabilities that did not belong to an orthologous reconciliation, to zero. The rules are divided into sets where one set regulates one part of the calculation framework. The rules and the probability calculations handle one gene node in one species node and at a time. In the duplications calculations and rules the variable *k* will be used as well as the gene and species nodes. The three rule set are explained in different chapters. The rule set for the duplication calculations are explained in 6.1.1, rules for the simple slice and slice calculations are presented in 6.1.2, and the last rule set which is for the extinction calculations is outlined in 6.1.3. All of the sub chapters start with the purpose of the rule set, and then the pseudo code for the rules are shown and then explained one at a time. In all of the sub chapters, the two genes that the orthology probability is to be calculated for are considered fixed. This means that LCA and S_LCA also are fixed.

6.1.1 Rule set for duplication calculations

The purpose of the duplications rule set is to check if a duplication probability should be calculated or set to zero. This rule set consists of three rules that will determine if this probability should be calculated. This rule set the probability to zero when the location of the gene tree node will prevent LCA to be in S_LCA. The rules of the rule set as pseudo code can be seen in Figure 11.

```
if (LCA != NULL AND S_LCA != NULL)
    if (gene != LCA)
        if (k <= U)
            then calculate probability
```

Figure 11: Showing the pseudo code for the duplication rule set. The symbol != means not equal to, and <= means less than or equal to.

First rule checks if it is possible for LCA to be in S_LCA, hence a S_LCA exists. This will depend on the construction of the gene tree and the species tree. To control if S_LCA exists, the reconciliation with the minimum number of duplication is used. This reconciliation is called the most parsimonious reconciliation. If LCA is not placed in S_LCA in the most parsimonious reconciliation then the two genes can not be orthologs, and none orthologous reconciliation can be formed. The most parsimonious reconciliation of one species tree and one gene tree is showed in Figure 12. In this reconciliation gene 1 can not be orthologous with gene 4. This is because of the split of gene 3, which is below gene 5 in the gene tree, is placed in S_LCA which hinder the split of gene 5 to be located there.

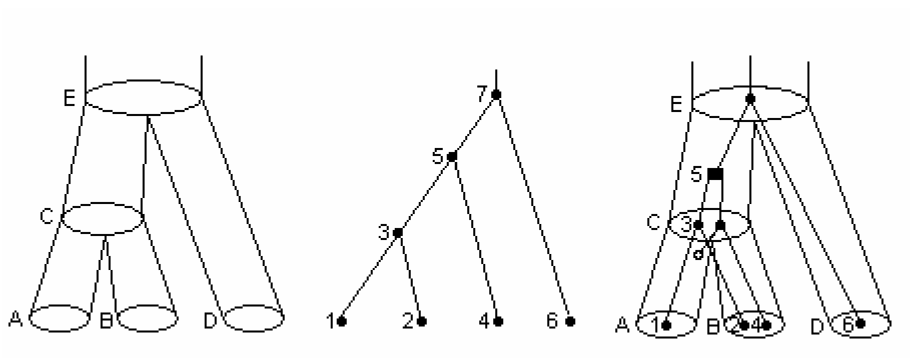

Figure 12: The first tree is the species tree second tree the gene tree and the reconciliation is the most parsimonious. The most parsimonious reconciliation can be used to see which gene pairs that can be orthologous. In this reconciliation it is showed that gene pairs 1 and 2, 1 and 6, 2 and 6, 4 and 6 can be orthologs. Gene 1 can not be orthologous with gene 4 because the gene split of gene 5 (which is the genes last common ancestor) is not places in a species split of species C (which is the species last common ancestor).

If LCA is in S_LCA as a speciation in the most parsimonious reconciliation, then the genes can be orthologs, and probabilities will be calculated.

The second rule of the duplication rule set handles the case when the gene tree node is LCA. Because it is the duplication probability that is calculated here, the probability should be set to zero. LCA can not be a duplication, because then the genes *a* and *b* will not be orthologs. Two examples of reconciliations where LCA (gene node 3) is placed as a duplication can be seen in Figure 13. The first reconciliation show when LCA is located in S_LCA. The duplication probability for LCA in species node C (second reconciliation in Figure 13) and in species node E (third reconciliation in Figure 13) will both be set to zero due to the second rule in the duplication rule set.

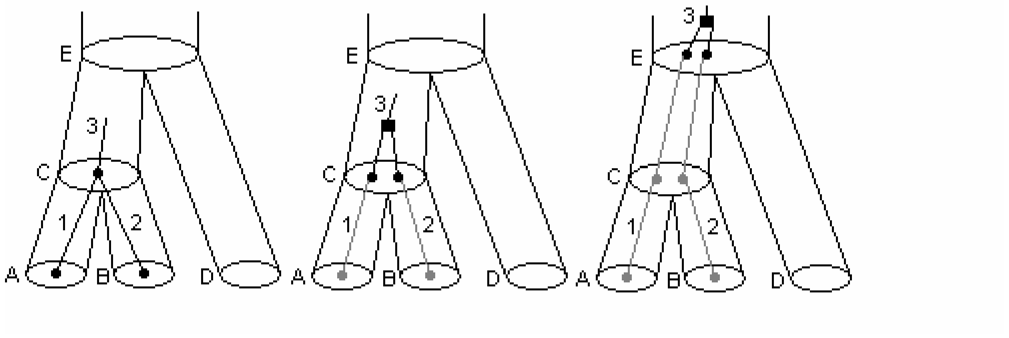

Figure 13: The first reconciliation shows when the two genes 1 and 2 are orthologs. Gene node 3 is the last common ancestor of the two genes and is hence LCA, and consequently species node C is S_LCA. The second reconciliation shows the case when gene node 3 is a duplication in species node C (variable *k* will be 2 here). When the probability for this duplication is to be calculated the second rule will check if the gene node is equal to LCA. This check will give a positive result for gene node 3 and the probability will be set to zero. The third reconciliation also shows gene node 3 as a duplication but in species node E. The calculation of the probability for this duplication should also be stopped by the second rule in the duplication rule set. All extinct gene nodes have been excluded from the reconciliations for clarification.

The third rule for the duplications; the calculations should be done if *k* is equal or less than U which represents the upper bound of possible duplications still allowing LCA to be in S_LCA. U is calculated through counting the number of the descendant leaf nodes the current gene nodes have. The LCA gene node will count as a leaf node. The calculation of U is illustrated in Figure 14.

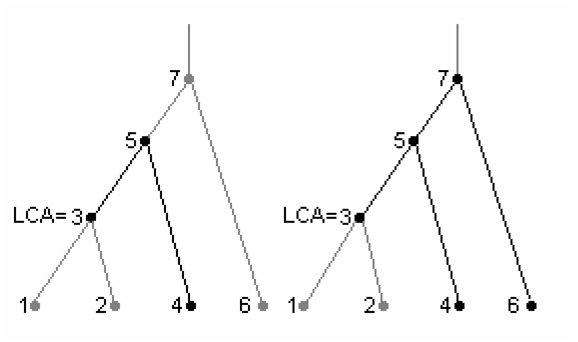

Figure 14: In the above gene trees gene node 3 is LCA. U for gene node 5 will be 2, the two counted nodes are black in the first tree. U for gene node 7 is 3 as can be seen as the black nodes in the second tree.

This rule will regulate so that LCA and the gene splits below LCA will not be lifted up above S_LCA in the species tree. Two reconciliations can be seen in Figure 15, the first having gene 1 and 2 as orthologs, the other without orthologs.

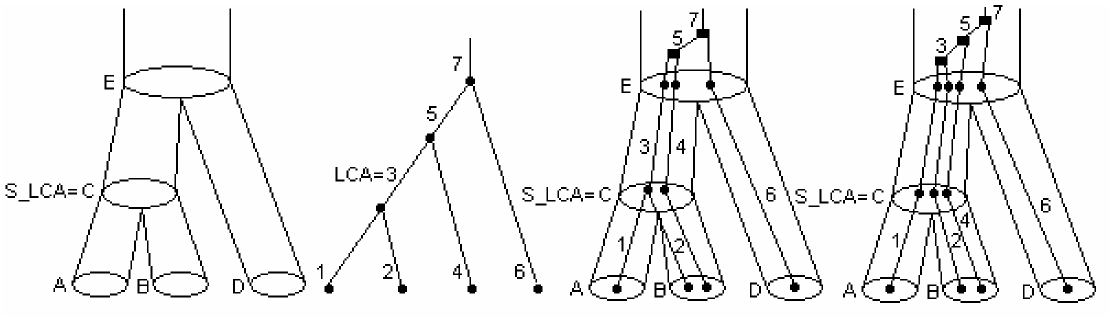

Figure 15: If genes 1 and 2 should be orthologs gene node 3 is LCA and species node C is S_LCA. In the first reconciliation *k* for gene node 5 in species E will be 2 and *k* will be 3 for gene node 7 in species node E. As can be seen in Figure 14 U is 2 for gene node 5, and U is 3 for gene node 7. The third rule in the duplication rule set says that U should be equal or less than *k*, which it is in both the duplications in the first reconciliation. The two duplications in species node E will not hinder LCA from being in S_LCA. In the second reconciliation *k* will be 3 for gene node 5 and 4 for gene node 7. This means that none of the probabilities for these duplications will be calculated, though the *k* value is larger than the U value in both cases. These duplications probabilities are set to zero because they would hinder LCA to be in S_LCA which would mean that gene 1 and 2 could not be orthologs.

The probability for the all duplications in the second reconciliation in Figure 15 will be set to zero. The duplication of gene node 3 will be set to zero by the second rule, because it is equal to LCA. The duplication of gene node 5 and 7 will be set to zero due to the third rule in the duplication rule set, because *k* will be larger than U in both cases.

6.1.2 Rule set for slice calculations

The second rule set will determine if a slice will be calculated in the simple slice and slice functions. A slice represents the probability that a gene node has evolved first in the current species node, then in the species nodes parent, and so on to the root of the species tree. The rule set that will determine if a slice probability will be calculated consists of a few rules. The pseudo code for the rules in the slice rule set can be seen in Figure 16.

```
if (LCA != NULL AND S_LCA != NULL)
      if(LCA descendant to gene AND S_LCA descendant to species)
            then calculate probability
      if(gene descendant to LCA AND species descendant to S_LCA)
            then calculate probability
      if(!(gene descendant to LCA) AND !(LCA descendant to gene))
            then calculate probability
```

Figure 16: The rules of the slice rule set. The symbol ! means not.

The first rule is the same as the first in the first rule set; if LCA can not be in S_LCA then all probabilities should be set to zero. One example where LCA can not be in S_LCA, is shown in Figure 12.

The second rule will check if the current gene node is above LCA in the gene tree, if so the current species node has to be higher in the species tree than S_LCA, for the probability to be calculated.

Rule number three will handle the cases when the current gene node is beneath LCA in the gene tree. Then the species node has to be beneath S_LCA in the species tree for the probability to be calculated. This rule will stop gene splits under LCA in the gene tree from being lifted up to and above S_LCA in the species tree. An example of a situation like this can be seen in Figure 17.

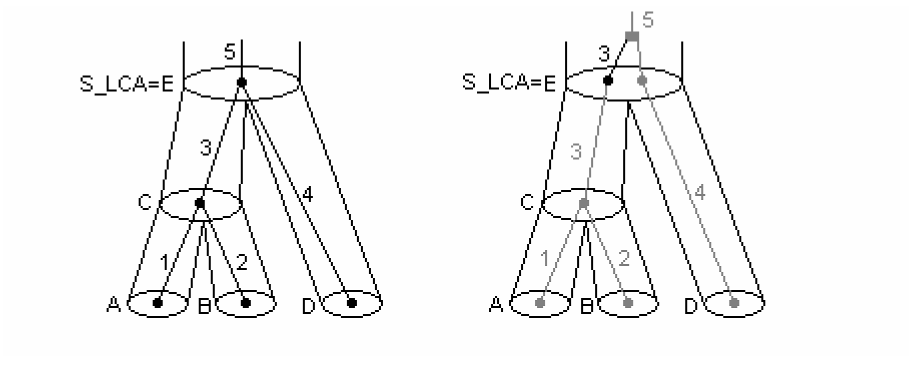

Figure 17: In the two reconciliations above genes 1 and 4 are supposed to be orthologs, making gene node 5 LCA and species node E S_LCA. When the slice probability for gene node 3 in species node C is calculated, gene node 3 can not be lifted up to species node E because then LCA can not be there, as shown in the second reconciliation. The extinct gene nodes are not included in the above reconciliations because of clarification.

The last rule in the second rule set make sure that the probability gets calculated for gene nodes in species nodes located in another part of species and gene tree then S_LCA and LCA. This rule states that if the gene node does not have LCA as a child, and LCA does not have the gene node as a child, then the probability should be calculated. A situation where this rules is needed can be seen in Figure 18.

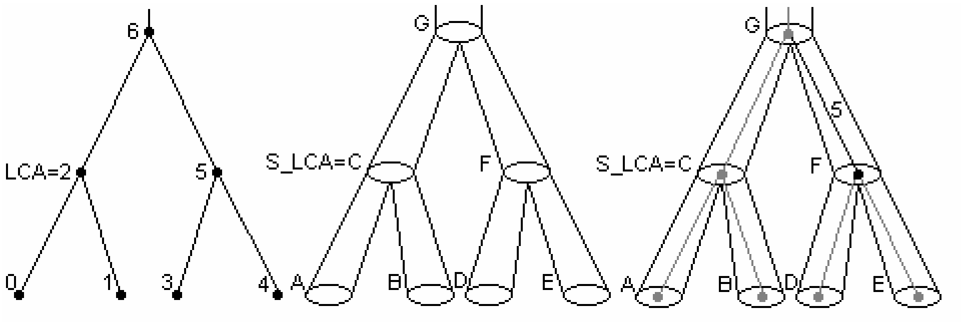

Figure 18: If the genes 0 and 1 are to be orthologs in the tree above, then gene node 2 is LCA and has to be in species node C which is S_LCA. When the slice probability for gene node 5 is calculated in F (black in the reconciliation), then the rule is going to check if F is above S_LCA in the Species tree, which it's not, then if gene node 5 is below LCA in the gene tree, which is not. The last rule of the slice rule set will then check if the nodes are in another part of the tree, which they are in this case, and the probability will be calculated.

6.1.3 Rule set for extinction calculations

The third rule set regulates the extinction likelihoods. It consists of five rules, and the pseudo code for the rules can be seen in Figure 19.

```
if (LCA != NULL AND S_LCA != NULL)
      if (S_LCA = Root)
            if (species = S_LCA AND
                 !(gene = LCA OR gene descendant to LCA))
                  then calculate probability
      if (LCA descendant to gene)
            then calculate probability
      if (species descendant to S_LCA AND
          (species parent != S_LCA) AND
          !(gene = LCA OR gene descendant to LCA))
            then calculate probability
      if(!(gene descendant to LCA) AND !(LCA descendant to gene))
            then calculate probability
```

Figure 19: Pseudo code for the extinction rule set. The symbol ! means not and != means not equal.

The first rule is the same as in the previous rule sets; if LCA can't be a speciation then the likelihood should be set to zero.

The second rule handles the case when S_LCA is equal to the species tree root node. When this occur neither the LCA node nor its children can be extinct in the roots children, so the probability is set to zero.

The next rule regards the reconciliations when the gene node has LCA as a child, and extinctions can happen as usual in all siblings to the species nodes ancestors. For example, gene nodes 1 and 2 in Figure 20 are orthologs and all the gene nodes above LCA (gene node 3) can be extinct in the siblings to the species nodes they are located in. As seen in Figure 20, gene node 5 (that has LCA as a child) is located in species node C, and can be extinct in C's sibling, node D.

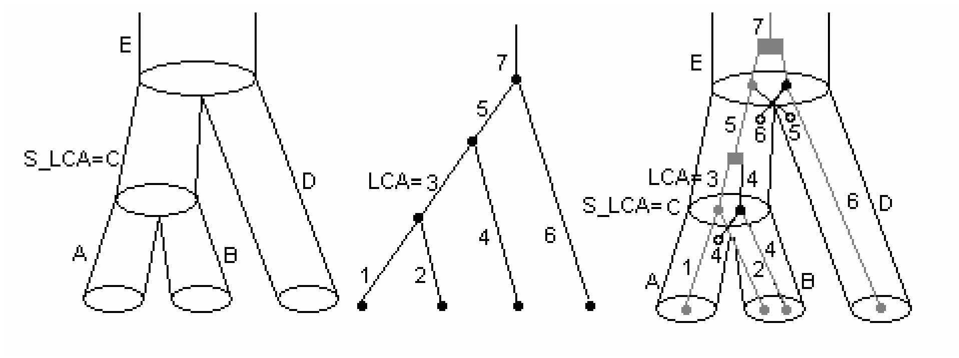

Figure 20: Shows a reconciliation where gene 1 and 2 are orthologous. The probability for gene node 5 to be extinct in species node D will be calculated, though it is above LCA according to third rule in the extinction rule set.

The fourth rule will make the following nodes probability zero; gene nodes that are in the S_LCA node or its children nodes (nodes that have S_LCA as parent) in the species tree and either are LCA or a descendent to LCA. This will prevent the extinction of gene nodes that are children of LCA, in S_LCA's children. No gene node can be extinct in the children of S_LCA because the split of LCA must be in S_LCA. For example, when gene 1 and 2 in Figure 20 are orthologs then LCA must be in S_LCA and when LCA are located in S_LCA then neither gene node 1 can go extinct in species node B, nor gene node 2 in species node A, which both has S_LCA as parent.

The last of the rules in the extinction rule set: if the current gene node neither is a descendant to LCA nor have LCA as a descendant, then the extinction likelihood should be calculated. In Figure 20 this rule would let the extinction probability be calculated for gene node 4 in species node A, and for gene node 6 in species node C.

The probability of all the possible reconciliations of the gene tree and the species tree represent the probability of the gene tree. Together with varying or fixed birth and death values both the probability of the ‘original’ gene tree and the likelihood of the gene tree that has two genes as orthologs are used in the MCMC algorithm. The likelihoods obtained for the two different gene trees, are then divided to obtain the likelihood for the two genes to be orthologs.

6.2 Test results

The implementation of the rules resulted in an algorithm that calculates the probability of two genes being orthologs. The implemented algorithm calculates the probability of a gene tree where the two genes are orthologs, given a gene tree, a species tree and birth and death parameters. The varying birth and death parameters are determined by the MCMC framework. In each iteration the probability for the gene tree that have two genes as orthologs are divided with the probability of the “complete” gene tree, to gain the wanted probability of two genes being orthologs.
To confirm the correctness of the algorithm the outputs of the calculated probabilities was used. The output shows all the calculated probabilities from all the calculation functions. Small trees were used, and a manual analysis of which probabilities that should be calculated in the tree was conducted. Then the manual results were compared to the ones outputted by the algorithm. It could be determined that the algorithm performs correct, at least when handling small trees.

Some further testing was conducted. The results from the developed algorithm where compared with the results given from the sampling algorithm (see 5.4) when using the same input. Also the CPU time for the execution for the two algorithms where compared. The CPU time was calculated by using the c++ function clock() which counted the CPU time it takes for the algorithm to do all the MCMC iterations and all probability calculations. To get as equal results as possible birth and death parameters are set from the start. The sampling algorithm is called Beep in this thesis and the developed algorithm is called Ortho. All the experiments were started with the birth rate 0.003 and death rate 0.0015 to get as equal starting-point as possible. The MCMC was run the 10000 iterations that is default, and a burn of 10 was not included when the mean value of the probabilities was counted. The trees used as input and a mean value of the probabilities can be seen in appendix A, B, C and D. Mean value and the standard deviation of the estimated birth and death rates can also be seen in the appendixes.

The first experiment was with the first trees in Table 1 (can be seen in appendix A). The CPU time for the execution is 0.79 seconds for the Ortho algorithm and 226.98 seconds for the Beep algorithm.

Second experiment (trees 2 from Table 1 that can be seen in appendix B) showed that it took 2.65 seconds for Ortho and 375.55 seconds for Beep.

In next experiment the trees number 3 in Table 1 where used. Both the gene trees and the species tree and the results are shown in appendix C. These trees took 67.71 seconds for Ortho and 1310.61 seconds for Beep.

The last experiment included the trees that are number 4 in Table 1 and can be seen in appendix D. It took Ortho 54.81 seconds of CPU time to complete the calculations mean wiles it took Beep 921.01 seconds.

Trees (Appendix)	CPU time Ortho	CPU time Beep
1 (A)	0.79 s	226.98 s
2 (B)	2.65 s	375.55 s
3 (C)	67.71 s	1310.61 s
4 (D)	54.81 s	921.01 s

Table 2: Shows the experiments done to compare the CPU time used by Ortho and Beep. The numbers in the first column referees to the numbers in Table 1, and the letters inside the parenthesis referees to the appendixes where the trees used as input and the gained probabilities can be seen.

7 Discussion

The discussion part has bee divided into two sections, the first, 7.1 discussing the rules and the second, 7.2, discussing the implementation and the test results.

7.1 The rules

The rules that were set up for the reconciliations where implemented and can be considered correct through the positive manual test. The manual test compared the expected results with the resulting calculation outputs from the implementation. This test shower that the right calculations where preformed. If the rules are correct and are correctly implemented and the outputs from the implementation truly reflect the calculations, the rules can be considered correct.

7.2 Implementation

The manual test of the implementation shows that the algorithm performs as expected on small trees (up to three leaf gene nodes), e.g. the right probabilities are included in the calculations.

As can been seen in appendix A to D the probabilities from the two algorithms, Ortho and Beep, are very similar.

The CPU time was longer for both Ortho and Beep in the third experiment then in the fourth experiment, as can be seen in Table 2. This point towards that it is the number of gene leaf nodes not the number of ortholog pairs that determines the CPU time. In Figure 21 the CPU time used by the two algorithms and the number of leaf gene nodes in the input gene tree used to draw a graph. It is obvious when looking at the graph that the Ortho algorithm is faster than the Beep algorithm.

The probability gained when doing the experiments show that the two algorithms gives about the same results. The biggest divagation between two results from the algorithms is of 0.019. This is not much when considering that Beep just approximates the probabilities.

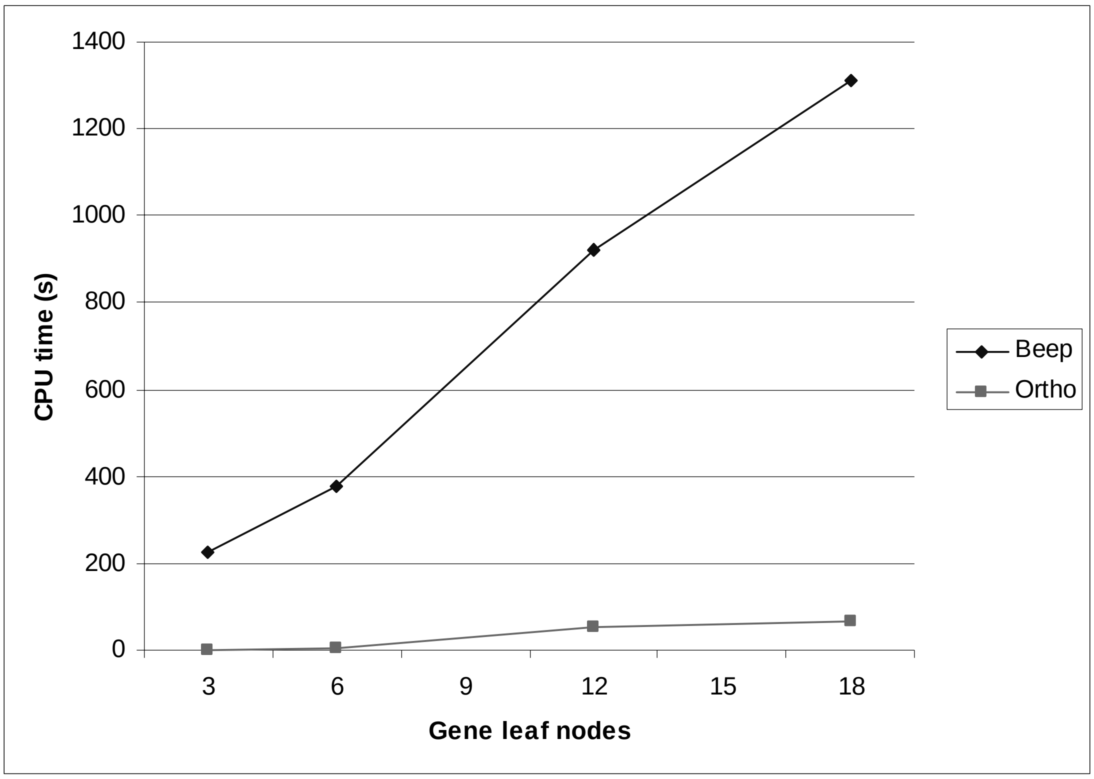

Figure 21: Here the CUP time used by the Beep and Ortho algorithms is showed. The y axis represents the CPU time and the x axis represents the number of gene leaf nodes in the input gene tree.

8 Conclusions

The algorithm constructed in this thesis can be used for investigation of orthologous genes. The result from the experiments shows that the developed algorithm gives about the same probabilities as the Beep algorithm (see 3.3) when compared. The experiments also show that the developed algorithm is faster than the Beep algorithm. The CPU time needed for execution of the two algorithms becomes bigger with the number of leaf nodes in the inputted gene tree.

The developed algorithm is the first algorithm that calculates probabilities, instead of approximating, for orthologs in a birth and death model. Hopefully the algorithm can help to get a better understanding of how orthologous genes have evolved, and finding new orthologs. In the long term, it can help to get a better understanding of how genes evolve.

9 Future Work

Some further testing of the algorithm would be necessary to be able to fully validate the results. Test where the results are known would be good. Maybe some logical test could be set up for the rules to verify them instead of verifying the implementation.

It would be interesting to examine the probabilities for genes in large trees to se which information that is gained from it.

It also would be interesting to use the implemented algorithm to investigate to which extent ortholog probability can be used when investigating novel genes. And to investigate how likely it is that orthologous genes have the same function.

Some investigation of how the time since the speciation has happened changes the ortholog probability and the resemblance of function, would also be interesting.

References

Altschul, S.F., Gish W., Miller W., Myers E.W., and Lipman D.J. (1990). *Basic local alignment search tool*. J. Mol. Biol. 215:403-410.

Arvestad, L., Berglund A., Lagergren J., and Sennblad B., 2003. *Bayesian gene/species tree reconciliation and orthology analysis using MCMC.* Bioinformatics/ISMB'03, 19:i7-i15.

Arvestad, L. Berglund A., Lagergren J., and Sennblad B.,. 2004. *Gene tree reconstruction and orthology analysis based on an integrated model for a duplications and sequence evolution.* RECOMB'04, Mars 27-31.

Attwood T. K. and Parry-Smith D. J., 1999. *Introduction to Bioinformatics.*Prentice Hall, Essex, England.

Bonizzoni P., Vedova G. D., Dondi R.**.** Reconciling Gene Trees to a Species Tree. *Algorithms and Complexity, Proceedings of the 5th Italian Conference (CIAC 2003). May 28-30 2003, Rome, Italy.* pp. 120-131.

Durbin R, Eddy S., Krogh A., and Mitchison G. 2002. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids.* (Reprint from 1998). University of Cambridge, Cambridge.

Fitch, W.M., *Distinguishing homologous from analogous proteins*. Systematic Zoology 1970 Jun; 19(2):99-113.

Functional and Comparative Genomics Fact Sheet, 2004 [online], Human Genome Project Information. Available from:
http://www.ornl.gov/sci/techresources/Human_Genome/faq/compgen.shtml#compgen
Access date: 2005-02-27

Gilks W.R., Richardson S., and Spiegelhalter D.J., 1996. *Markov Chain Monte Carlo In Practice.* Chapman & Hall/CRC, United States of America.

Huelsenbeck, J. P., Ronquist F., Nielsen R., Bollback J. P., 2001. *Bayesian inference of phylogeny and its impact in evolutionary biology.* Science's Compass, Vol. 294, Dec 2001, pp2310-2314.

Huelsenbeck, J. P. and Ronquist, F., 2001. *MRBAYES: Bayesian inference of phylogenetic trees.* Bioinformatics Vol. 17 no. 8 2001, pp 754-755.

References

Nei, M., Gu X., and Sitnikova T., 1997. Evolution by the birth-and-death process in multigene families of the vertebrate immune system. *"Genomics and the Origin of Species" 1997, Jan 30- Feb 1, at the National Academy of Sciences Beckham Canter in Irvine, CA.* Ayala F. J. and Fitch W. M., pp 7799-7806.

Page R. D. M., and Charleston M. A., 1996. *From gene to organismal phylogeny: Recinciled trees and the gene tree/species tree problem.* Molecular phylogenetics and evolution. Vol. 7 No.2, April, pp. 231-240, 1997.

Page P. D.M., and Holmes E. C., 1998.*Molecular Evolution A Phylogenetic Approach.* Blackwell Science Ltd, University press, Cambridge, Great Britain.

Radford M. Neal, January 1998. *Philosophy of Bayesian Inference.* [online], Department of Computer Science, University of Toronto. URL: http://www.cs.toronto.edu/~radford/res-bayes-ex.html, Access date: 2005-02-28.

Remm, M., Storm C. E.V., and Sonnhammer E. L.L., 2001. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of Molecular Biology:* vol. 314 (5): 1041-1052 DEC 14

Storm C. E.V. and Sonnhammer E. L.L., 2002. *Automated ortholog inference from phylogenetic trees and calculation of orthology reliability.* Bioinformatics Vol. 18 no. 1 2002 Pages 92-99.

Storm C. E.V. and Sonnhammer E. L.L., 2003. *Comprehensive Analysis of Orthologous Protein Domains Using the HOPS* Database**.** Genome Research. 2003 13: 2353-2362

Zmasek, C. M., and Eddy S. R., 2001. *A simple algorithm to infer gene duplication and speciation events on a gene tree.* Bioinformatics Vol. 17 no. 9 2001 Pages 821-828

Zmasek, C. M., and Eddy S. R., 2002. *RIO: Analyzing proteomes by automated phylogenomics using resampled inference of orthologs.* BMC Bioinformatics 16 May 2002, 3:14.

Appendix A

Gene tree: ((a, b),c)

Species tree: ((A:100,B:100):500,C:600):100;

Pairing between genes and species:

a : A

b : B

c : C

Ortho:

```
Ranked by probability
        0.9993  a1, b1
        0.9594  b1, c1
        0.9594  a1, c1

birthRate       Mean = 0.0002, SD = 0.0004
deathRate       Mean = 0.0006, SD = 0.0008
CPU time: 0.79 s
```

Beep:

```
Ranked by probability
        0.9980  b1, a1
        0.9643  c1, a1
        0.9643  c1, b1

birthRate       Mean = 0.0004, SD = 0.0006
deathRate       Mean = 0.0006, SD = 0.0006
CPU time: 226.98 s
```

Appendix B

Gene tree: (((((h1, h2), h3), h4), h5), f1);

Species tree: (Human:993, Fly:993):27;

Gene to species pairing:

h1, h2, h3, h4, h5: Human

f1: Fly

Ortho:

Ranked by probability

```
    0.9844  h3, f1
    0.9844  h2, f1
    0.9844  h5, f1
    0.9844  h4, f1
    0.9844  h1, f1
```

```
birthRate       Mean = 0.0016, SD = 0.0009
deathRate       Mean = 0.0007, SD = 0.0007
```

CPU time: 2.65 s

Beep:

Ranked by probability

```
    0.9817  f1, h1
    0.9817  f1, h4
    0.9817  f1, h5
    0.9817  f1, h2
    0.9817  f1, h3
```

```
birthRate       Mean = 0.0018, SD = 0.0007
deathRate       Mean = 0.0007, SD = 0.0008
```

CPU time: 375.55 s

Appendix C

Gene tree:

((((A010101:0.022, Gogo-A0101:0.025):0.008, (Popy-A01:0.018, Popy-A02:0.019):0.006):0.020, ((((B070201:0.028, Gogo-B0103:0.031):0.005, (Cw0102:0.019, Gogo-C0101:0.031):0.010):0.001, (Popy-B01:0.009, Popy-B02:0.008):0.035):0.013, F010101:0.086):0.003):0.001, ((G010101:0.046,(Safu-G04:0.022, (Safu-G02:0.027, (Safu-G01:0.019, Safu-G03:0.020):0.001):0.001):0.023):0.008, (E0101:0.054, Saoe-E01:0.036):0.022):0.001);

Species tree:

(((Gorilla:18,Human:18):14.0,Pongo:32.0):28.0,Saguinus:60.0):45.0;

Gene to species pairing:

Gogo-A0101, Gogo-B0103, Gogo-C0101:	Gorilla
Popy-A01, Popy-A02, Popy-B01, Popy-B02:	Pongo
Safu-G04, Safu-G02, Safu-G01, Safu-G03, Saoe-E01:	Saguinus
A010101, B070201, Cw0102, E0101, F010101, G010101:	Human

Ortho:

```
Ranked by probability
    1.0000  G010101, Safu-G04
    1.0000  G010101, Safu-G01
    1.0000  G010101, Safu-G02
    1.0000  G010101, Safu-G03
    1.0000  E0101, Saoe-E01
    0.9996  Cw0102, Gogo-C0101
    0.9966  Cw0102, Popy-B02
    0.9966  Gogo-B0103, Popy-B02
    0.9966  Gogo-C0101, Popy-B02
```

```
      0.9966  Gogo-B0103, Popy-B01
      0.9966  B070201, Popy-B02
      0.9966  Cw0102, Popy-B01
      0.9966  Gogo-C0101, Popy-B01
      0.9966  B070201, Popy-B01
      0.9874  B070201, Gogo-B0103
      0.9760  A010101, Gogo-A0101
      0.9606  Gogo-A0101, Popy-A01
      0.9606  Gogo-A0101, Popy-A02
      0.9606  A010101, Popy-A01
      0.9606  A010101, Popy-A02

birthRate     Mean = 0.0183, SD = 0.0072
deathRate     Mean = 0.0167, SD = 0.0073
CPU time: 67.71 s
```

Beep:

```
Ranked by probability
      0.9943  Saoe-E01, E0101
      0.9940  Gogo-B0103, B070201
      0.9938  Safu-G01, G010101
      0.9938  Safu-G02, G010101
      0.9938  Safu-G03, G010101
      0.9938  Safu-G04, G010101
      0.9937  Gogo-C0101, Cw0102
      0.9771  Gogo-A0101, A010101
      0.9683  Popy-B01, Gogo-C0101
      0.9683  Popy-B02, Gogo-C0101
      0.9683  Popy-B01, Cw0102
      0.9683  Popy-B02, B070201
      0.9683  Popy-B02, Cw0102
      0.9683  Popy-B02, Gogo-B0103
      0.9683  Popy-B01, Gogo-B0103
      0.9683  Popy-B01, B070201
      0.9460  Popy-A02, Gogo-A0101
      0.9460  Popy-A01, A010101
      0.9460  Popy-A01, Gogo-A0101
```

```
        0.9460  Popy-A02, A010101

birthRate       Mean = 0.0168, SD = 0.0058
deathRate        Mean = 0.0171, SD = 0.0072
CPU time: 1310.61 s
```

Appendix D

Gene tree:

((((Gogo-A0101, (Popy-A01, Popy-A02)), ((Gogo-B0103, (Popy-B01, Popy-B02)), Gogo-C0101)), (Safu-G04, (Safu-G01, (Safu-G02, Safu-G03)))), Saoe-E01)

Species tree:

((Gorilla:32.0,Pongo:32.0):28.0,Saguinus:60.0):45.0;

Gene to species pairing:

Gogo-A0101, Gogo-B0103, Gogo-C0101:	Gorilla
Popy-A01, Popy-A02, Popy-B01, Popy-B02:	Pongo
Safu-G04, Safu-G02, Safu-G01, Safu-G03, Saoe-E01:	Saguinus

Ortho:

Ranked by probability

0.9997	Gogo-B0103, Popy-B02
0.9997	Gogo-B0103, Popy-B01
0.9855	Popy-B01, Safu-G03
0.9855	Gogo-C0101, Safu-G02
0.9855	Popy-A02, Safu-G03
0.9855	Gogo-B0103, Safu-G03
0.9855	Gogo-A0101, Safu-G03
0.9855	Gogo-B0103, Safu-G02
0.9855	Popy-B01, Safu-G01
0.9855	Popy-A02, Safu-G01
0.9855	Popy-B02, Safu-G02
0.9855	Popy-A02, Safu-G04
0.9855	Popy-A01, Safu-G01
0.9855	Gogo-B0103, Safu-G04
0.9855	Gogo-A0101, Safu-G01
0.9855	Gogo-A0101, Safu-G02
0.9855	Popy-A02, Safu-G02

```
    0.9855  Popy-A01, Safu-G04
    0.9855  Popy-A01, Safu-G03
    0.9855  Gogo-C0101, Safu-G04
    0.9855  Gogo-C0101, Safu-G03
    0.9855  Popy-B01, Safu-G04
    0.9855  Popy-A01, Safu-G02
    0.9855  Gogo-A0101, Safu-G04
    0.9855  Gogo-C0101, Safu-G01
    0.9855  Popy-B02, Safu-G01
    0.9855  Popy-B02, Safu-G03
    0.9855  Gogo-B0103, Safu-G01
    0.9855  Popy-B02, Safu-G04
    0.9855  Popy-B01, Safu-G02
    0.9541  Gogo-A0101, Popy-A02
    0.9541  Gogo-A0101, Popy-A01

birthRate     Mean = 0.0220, SD = 0.0094
deathRate      Mean = 0.0131, SD = 0.0114
CPU time: 54.81 s
```

Beep:

```
Ranked by probability
    0.9827  Safu-G03, Popy-B02
    0.9827  Safu-G03, Gogo-A0101
    0.9827  Safu-G01, Gogo-A0101
    0.9827  Safu-G03, Popy-A01
    0.9827  Safu-G01, Popy-A02
    0.9827  Safu-G02, Gogo-B0103
    0.9827  Safu-G03, Popy-A02
    0.9827  Safu-G02, Popy-B02
    0.9827  Safu-G01, Popy-B01
    0.9827  Safu-G04, Popy-B02
    0.9827  Safu-G01, Gogo-C0101
    0.9827  Safu-G02, Gogo-C0101
    0.9827  Safu-G03, Gogo-C0101
    0.9827  Safu-G04, Popy-B01
    0.9827  Safu-G03, Popy-B01
    0.9827  Safu-G01, Popy-B02
    0.9827  Safu-G04, Gogo-B0103
    0.9827  Safu-G04, Gogo-C0101
    0.9827  Safu-G02, Popy-A01
    0.9827  Safu-G02, Popy-A02
    0.9827  Safu-G01, Gogo-B0103
    0.9827  Safu-G04, Popy-A02
    0.9827  Safu-G02, Popy-B01
    0.9827  Safu-G03, Gogo-B0103
    0.9827  Safu-G04, Popy-A01
    0.9827  Safu-G01, Popy-A01
    0.9827  Safu-G02, Gogo-A0101
    0.9827  Safu-G04, Gogo-A0101
    0.9807  Popy-B02, Gogo-B0103
    0.9807  Popy-B01, Gogo-B0103
    0.9619  Popy-A02, Gogo-A0101
    0.9619  Popy-A01, Gogo-A0101

birthRate     Mean = 0.0244, SD = 0.0091
deathRate     Mean = 0.0150, SD = 0.0086
CPU time: 921.01 s
```