PHYREX

User Manual

Roald Rossnes

© Copyright 2003 - 2004 by Roald Rossnes/Computational Biology Unit/University of Bergen. The software package is provided "as is" without warranty of any kind. In no event shall the author or his "employer" be held responsible for any damage resulting from the use of this software, including but not limited to the frustration that you may experience in using the package. The program package, including source codes, example data sets, executables, and this documentation, is distributed free of charge for academic use only. Permission is granted to copy and use programs in the package provided no fee is charged for it and provided that this copyright notice is not removed.

Contents

1 Introduction **3**

2 Installation **4**
2.1 Distribution 4
2.2 Linux 4
2.3 Path 4
2.4 Java 5

3 File Formats **5**
3.1 Treeformat 5
3.1.1 Schreiber format 5
3.1.2 Newick format 7
3.1.3 Parsing 7
3.2 Sequence Format 8
3.2.1 Fasta format 8
3.2.2 Collecting Input 8
3.2.3 Procedure 9
3.3 Gene Expression Format 11

4 Usage **13**
4.1 Command Panel 13
4.1.1 Input 14
4.1.2 Settings 14
4.1.3 Run 15
4.1.4 Menu 15
4.2 Tree Panel 16
4.3 Node Panel 16
4.4 Progress Panel 17

5 Constraints **17**
5.1 Purpose 17
5.2 Other Documentation 17

1 Introduction

As genomes evolve under selective pressure, not only do protein coding sequences evolve, but the gene regulatory sites affecting expression profiles and alternative splice site usage also evolve. To understand the evolutionary history of gene and genome functionality and the selective pressures that have affected the evolution of genomes, it is desirable to reconstruct the ancestral state of gene expression and splice site usage. At a first level, I have developed a Minimum Evolution approach based on the use of gene expression profiles. The approach is implemented to work with large scale datasets like microarrays, e.g. the Affymetrix genechip technology, and is correlated with an analysis of the actual regulatory sequence reconstruction done by similar methods, where such information is available.

My objectives with this research is to highlight the changes made by selective pressure by use of Minimum evolution methods to reconstruct continous data at the ancestral states in a phylogenetic tree.

To reconstruct the ancestral states of the continuous data traits I have developed a brute force algorithm that constructs an interval of allowed values on each internal node in a phylogenetic tree (Schreiber format), and chooses the best value to represent each node. The algorithm runs trough the tree two times, hence an order O(2n) time complexity. In the first run the intervals of allowed values are constructed and in the second run the intervals are narrowed and the representing value is chosen. This is done for every gene represented in the large scale data set.

An organism can by accumulating substitutions divide into two closely related organisms. By comparing sequences from a set of homologous genes from different species it is possible to find the sequence of their closest common ancestor. Algorithms that do such calculation on sequences have allready been developed. In this thesis ClustalW and BaseML are used. ClustalW is used to align sequences found in the leaf nodes of the tree and BaseML is used to calculate sequences at the ancestral nodes of the phylogenetic tree based on the alignment done by ClustalW. The sequences used are the upstream regions of the genes collected from EnsEmbl. Our work has also produced an algorithm and methods for constructing ancestral gene expression profiles. and a framework that can be used to display and compare them to the sequence calculation done as described above.

Simultaneous reconstruction of regulatory sequences and expression profiles reduces the signal to noice ratio by using long branched significant classes to correlate substitutions with functional effects. By comparing the two calculations mentioned above and by isolating candidate genes where a clear change in gene expression is correlated with a high number of point mutations in the upstream region, the signal to noice ratio is reduced.

2 Installation

2.1 Distribution

PHYREX is distributed as a zipped catalog structure. For the program to run properly it is dependant of an internal catalog structure in the software package. This means that the catalog structure that is constructed in the program download file must be maintained when unzipped onto your local system. This is important because the program use a static catalog structure below the root directory (EvolutionFolder).

2.2 Linux

PHYREX is made to work under a Linux environment with an installation of a Java Virtual Machine. The first thing to do is to check if these properties is fulfilled.

Since the program is running on a Java Virtual Machine you could think that it should be platform independent. This is not correct since the program does perform some platform dependent calls to the operative system and because of the CLisp module used by the program is implemented in a Linux binary file.

PHYREX is developed and runned under both whitebox and redhat linux.

Enclosed with PHYREX there is a treeconverter that convert trees from Schreiber to Nevick format and the other way around. The treeconverter is developed in CLisp and compiled in to a Linux binary file that is run by a script file. This makes the treeconverter platform dependent but it does not need a CLisp compiler. If you want some documentation on CLisp you can find some information on http://www.cons.org/cmucl.

2.3 Path

PHYREX is using externally developed software like PAML and BaseML. To make these programs available to the program the path to the executable files must be set in the path. I'm currently running a whitebox linux version which is using a bash shell. Under this configuration I use the .bashrc file in my root catalog to set the PATH and other variables. The file must at least contain these declarations :

export JDK_HOME=/Home/stud/roaldr/java/j2sdk1.4.2_06/lib/
export JAVA_HOME=/Home/stud/roaldr/java/j2sdk1.4.2_06/
exportPATH=./:/Home/stud/roaldr/hfoppg/evolutionFolder/clustalw1.83/:
/Home/stud/roaldr/hfoppg/evolutionFolder/bnb/:/Home/stud/roaldr:
/Home/stud/roaldr/java/j2sdk1.4.2_06/bin/
export CLASSPATH=./:/Home/stud/roaldr/database/hsqldb/lib/hsqldb.jar:/Home/stud/
export CVSROOT=/net/bccs/cbu/cvsroot xset b off

This is just example data and the users has to set up their own catalog structure and path files.

2.4 Java

PHYREX is developed using Java 1.4.2 and newer API. This does not mean that it only works using this API, but I cannot guarantee that all functions will compile using older APIs. I therefore recomend an installation of Java 1.4.2 or newer berfore using PHYREX. Java API's can be obtained from http://www.sun.com.

When this is done and all the above mentioned resources and properties is collected PHYREX should be compiled this can be done by entering the root folder and runing:

javac rrevolution/*.java
java -Xmx250m rrevolution/Rrevolution

3 File Formats

Since PHYREX is using data collected from many different sources it depends on a variety of different file formats. The file formats are often described by the organization constructing them or based on other formats that are well documented.

3.1 Treeformat

Schreiber vs. Newick

Trees as a datastructure is normally represented as a string of characters encapsulated by various kinds of brackets constructing the branches between the nodes, normally represented by a node number e.t.c. Different formats have different ways of formating the string.

3.1.1 Schreiber format

The Schreiber tree format is very flexible and can be expanded by just adding tags, and can therefore be used to store information in a tree structure, a kind of a low level XML code. It makes use of square brackets and commas ("[","]",",") to build the tree structure. Meaning that each external node is represented by a number and each internal node is represented by a pair of paranteses that enclose all the decendents of that internal node.

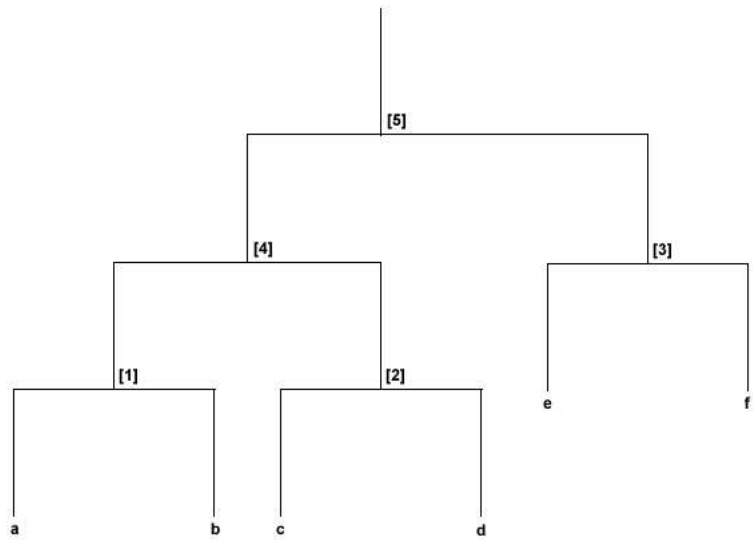

Figure 1: Schreiber tree

For instance in figure 1 the internal node [1] would be represented like [a,b] in the string representing the tree.

The tree in the figure above is represented by this string "n:-[[a,b],[c,d],[e,f],[[1],[2]],[[3],[4]]]". The string is read from left to right constructing parent nodes for each pair of parantheses that are closed and iteratively giving the parent nodes numbers. When all the leaf nodes are combined, parent nodes are combined in the same fashion as the leaf nodes by pairing their number (enclosed in brackets) inside another set of square parentheses. In general, one could describe the format by giving each external node a number related to the position of the node, and each internal node could be viewed like a list of children denoted [X], where X is a number and [X] is a list of numbers separated by commas (","). Only the childnodes of a internal node must be combined before the internal node could be constructed. Meaning that "n:-[[a,b],[c,d],[[1],[2]],[e,f],[[3],[4]]]" would also be a legal statement, since both internal node 1 and 2 are constructed one can construct internal node 3 berfore combining external nodes e and f. This would result in the tree shown in figure 2.

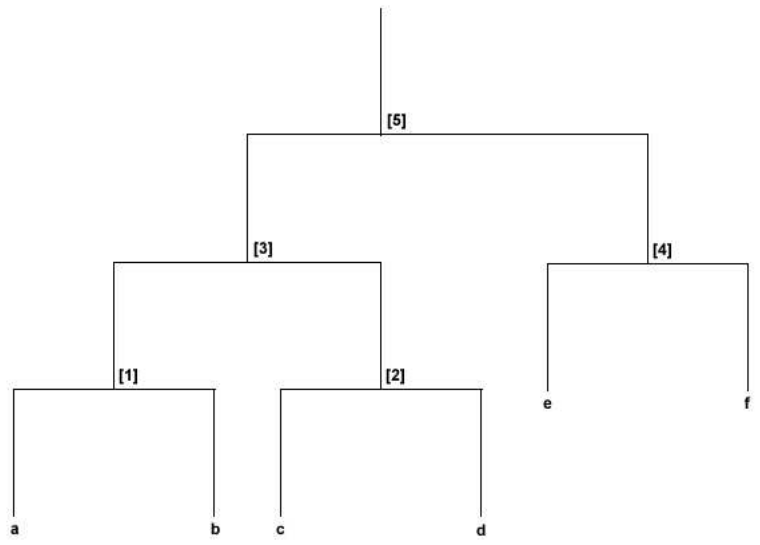

Figure 2: Alternative Schreiber tree

3.1.2 Newick format

The newick format is a more common format for representing trees. This format uses regular parantheses and a nesting of these to represent the tree stuctures. A representation of the tree in figure 1 in newick format would be "((((a,b),(c,d)),(e,f)));" All newick formated trees must end with a semicolon (";"). There exists an extension of the newick format annotated NHX. This extension is oftens used to represent annotated trees because of the tags introduced in the format. This is a more flexible solution than the original format, but not as flexible as the schreiber format. A more complete description of the newick format is available from Joseph Felsenstein's website.

3.1.3 Parsing

As mentioned earlier PHYREX have a treeparser that can convert trees between the two different tree formats, but since the treeparser is implemented in CLisp and system dependent one could get around it by maintaining a copy of the tree in both Schreiber and Nevick format.This needs knowledge of the Schreiber format. The Nevick Format is the same format that is used in the Phylip package and documentation can be found with a simple google search on nevick and/or phylip. The Schreiber format is not so much used and can be harder to find information about. The suggested way to deal with this is to use the treeconverter to make the schreiber tree based on a nevick tree, this can be done

as following :

treeconverter <nevick_tree> <output_file> <.>

if you are in the converter catalog in the catalog structure.

3.2 Sequence Format

Sequences used in the program are collected from the EnsEMBL database and written in a FASTA - like format. One needs to collect one file per species and one file containing a list of homologs between all the species from EnsEMBL. A procedure on how to collect these files can be found in Section 3.2.2.

Ensembl is freely accessible and consists of several levels from which data can be collected. Ensembl can be accessed via an html interface, a java API, a batch query tool (EnsMart) and trough different tools that can be locally installed on your computer. The data can be retrived as flat files, FASTA formated files or raw dumps. All these different ways of accessing Ensembl make it extremely flexible and easy to work with. In this project the user can access Ensembl through EnsMart.

3.2.1 Fasta format

The fasta format is a specific way to format database retrievals. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data.

- The description line starts with a greater than symbol (">").
- The word immediately following the greater than symbol (">") is the "ID" (name) of the sequence, the rest of the line is the description.
- The "ID" and description is often divided by a token e.g. the '|' character.
- The "ID" and the description are optional.
- All lines of text should be shorter than 80 characters.
- The sequence ends if there is another greater than symbol (">") symbol at the beginning of a line and another sequence begins.

3.2.2 Collecting Input

EnsMart is an EnsEmbl tool that allows the user to run batch queries against the EnsEmbl database. It is structured to allow genome queries and therefore a very useful tool in this project. This section describes how the user should use EnsMart to retrieve the data from the EnsEmbl databases in a manner that suits this program. There are two different data files that need to be downloaded from the EnsEmbl database.

1) The sequences of the upstream regions (One for each organism).

2) A file that contains a mapping of homolog genes between the species.

EnsMart is an online tool and can be found at http://www.ensembl.org/Multi/martview. It consists of several HTML pages used to construct queries to be used against the EnsEmbl database.

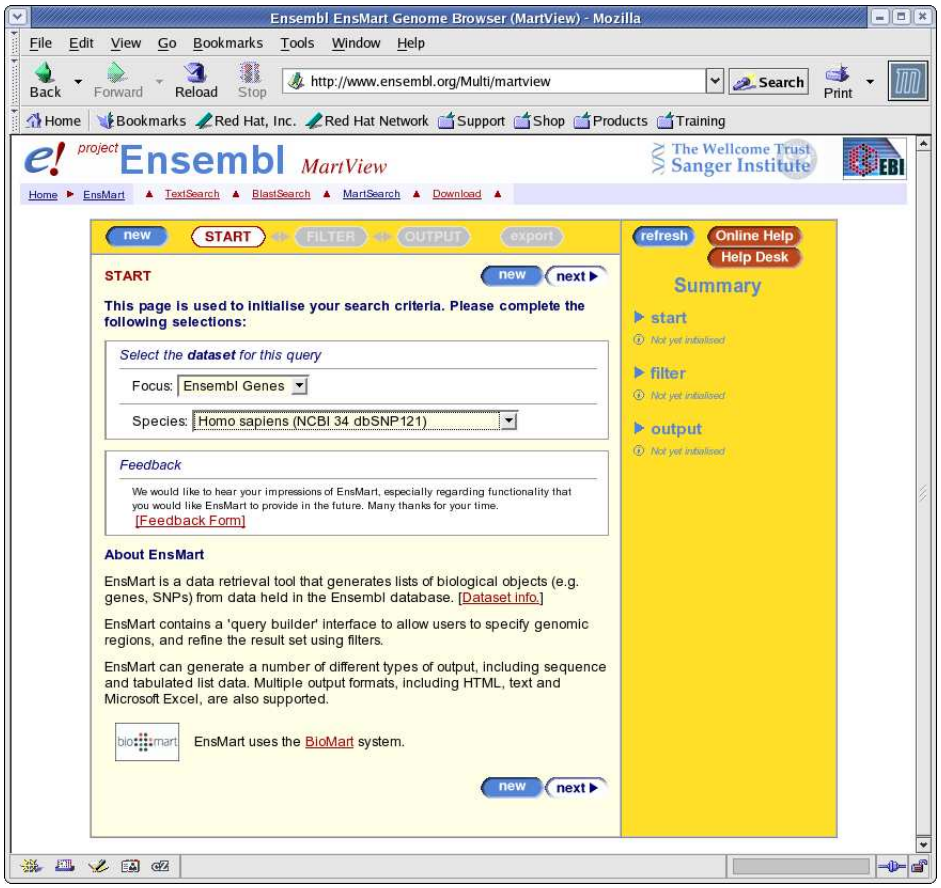

Figure 3: EnsMart

3.2.3 Procedure

First, the user has to collect a file containing all the sequences of the upstream regions of the genes in study. This can be done by following the recipe described in Algorithm 1.

Algorithm 1 Collecting Sequence files

for each species do
1: On the ensmart startpage choose "focus" : "ensmart genes"
2: and "species" : your species of study
3: Press "next"
4: In "region" - frame, uncheck "limit to" -box
5: Press "next"
6: Choose "sequence tab"
7: In "type" box Check "Genes - transcript information ignored"
8: In "type" box Check "5' upstream only"
9: In "type" box Enter the number of upstream basepairs in "5' Flank (bp)" - box
10: In "Select the output format" - box Check "Text, Fasta"
11: In " File compression" - box Check "none"
12: In "Name of result" - box type inn a name
13: Press "export"
14: Save pop up as a file
end for loop

After collecting sequence files for all the species the user must collect a file containing "maps of homologs". This is a mapping of EnsEmbl id's of the genes from each species that are homolog with each others. This can be done by following the recipe described in Algorithm 2.

Algorithm 2 Collecting "map of homologs"

for each analysis do
1: On the ensmart startpage choose "focus" : "ensmart genes"
2: and "species" : one of your species of study
3: Press "next"
4: In "region" - frame, uncheck "limit to" -box
5: In "MULTI SPECIES COMPARISONS" - frame, check the species of study
6: In "MULTI SPECIES COMPARISONS" - frame, and the "only" - box for each species of study
7: Press "next"
8: In "MULTI SPECIES COMPARISONS" - frame, check the "Ensembl Gene ID" -box of every species of study
9: In "Select the output format" - box Check "Text, comma separated"
10: In " File compression" - box Check "none"
11: In "Name of result" - box type inn a name
12: Press "export"
13: Save pop up as a file
end for loop

A more thorough explanation of Ensmart and EnsEmbl in general can be found on their website http://www.ensembl.org.

3.3 Gene Expression Format

The system input files have different formats, some of them are species dependent and exists in many copies, others follow the analysis and exists in only one copy. The main provider of these files are the Affymetrix experiment and the EnsEmbl database.

From the AffyMetrix experiment, the following files must be collected:

cdf - file

The <name>.cdf file holds information about the structure of the genechip used in the experiment. It maps the perfect match and mismatch probes against each other to construct a probe pair and specifies the "algorithms" used to construct the probe sets. The .cdf file is formated in a special tab separated format developed by AffyMetrix and needs a specially developed parser to be read properly. For each experiment to be analysed in PHYREX one .cdf file is needed. The file uses a name convention that specifies the GeneChip it is developed for. In the test set used in this thesis the file is called HG_U95A.cdf

cel - file

The .cel - files holds the values for each probe on the GeneChip. The values are represented with a number, positive or negative, describing the amount of target DNA that did bind to the probe. To run a PHYREX analysis you need one .cel file per species in the analysis (one per leaf node in the tree). The file is tab separated following a similar but not identical format as used in the .cdf file and needs a specially developed parser.

annotation - file

The annotation file either follows the AffyMetrix experiment or can be downloaded from the AffyMetrix wesite (http://www.affymetrix.com). It contains a list of all the id's used on a GeneChip and maps them to other id's in public available databases such as EnsEmbl. The file is formated as comma separated values and follows a similar naming convention as the .cdf file (testset : HG_U95A_annot.csv) This file can be obtained in other formats as well, but support for these formats are not developed in PHYREX.

4 Usage

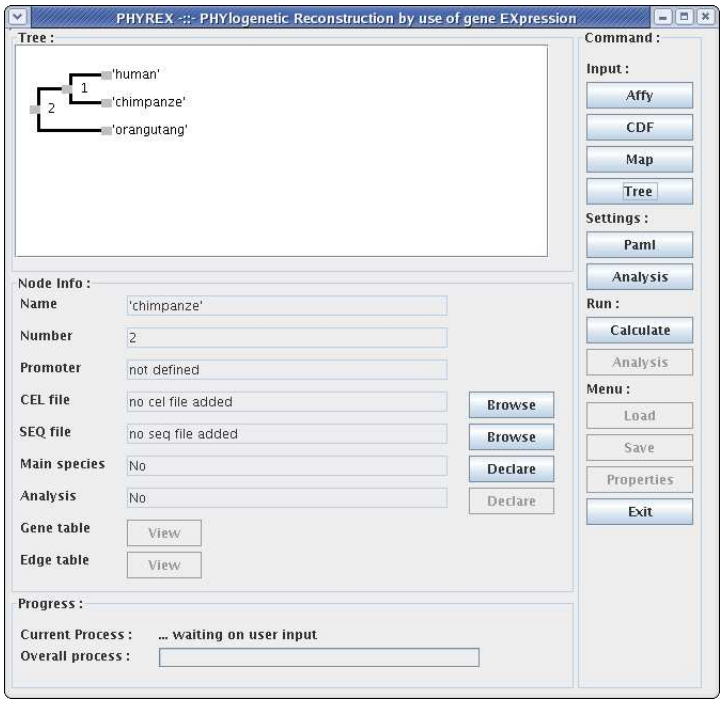

Figure 4: PHYREX

The graphical user interface of PHYREX is divided into four parts: the command panel, the tree panel, the node panel and the progress panel. I'll give a quick introduction to all three panels here and a description of what they contains.

The description below is done separate for each button or component in the GUI, some components must be initialised before others to make the program work as intended. By following the consecutive order the components are described in below these interactions will work.

4.1 Command Panel

The command panel is the menu system of PHYREX. In this panel almost all the user interactions and tuning of the system is done. The command panel is divided into four parts each taking care of different types of user interaction.

4.1.1 Input

The input part of the command panel takes care of the input needed to set up the analysis and initialize the tree and node panels.

Affy

The Affy button is used to load the AffyMetrix annotation file into the system. It starts processes for reading and parsing the file into a local "database" that is used later to populate the leaf nodes with data.

CDF

When clicking the CDF button the user is asked to specify the path to the CDF file of this analysis. The CDF file contains information used by PHYREX to set up the system for reading in gene expression data from the microarray experiment data files.

Map

The Map button is used to specify the file containing the map of homologs used by the system. This file contains information on wich genes in the leaf nodes that are homolog to each other. The file can be obtained as described in Section 3.2.3.

Tree

To guide the reconstruction process done by PHYREX the system needs a phylogenetic tree. By clicking on the Tree button the user specifies a path to the tree file. The tree file must be in Schreiber format, and is parsed immediately after loaded. When the tree is loaded the Tree panel is initialized and made active.

4.1.2 Settings

This part of the command panel is used to set different propeties used by the system when reconstructing sequences or analysing the results.

Paml

The Paml button opens a panel that can be used to tune the BaseML parameters used in the sequence reconstruction. The tuning of these parameters assume knowledge of the BaseML system, the user is refered to the PAML user manual.

Analysis

The Analysis button is used to give parameters to the analysis that should be performed after the reconstruction and population of the internal nodes of the tree is done. In the same way as the Paml button it opens a panel containing all the porperties that can be altered.

4.1.3 Run

This part of the command panel is used to perform the population of the tree and to analyse the results. Before running these buttons all input data must be collected both in the command, tree and node panel.

Calculate

After all the data is collected the Calculate button performs the population and reconstruction algorithms for both gene expression values and sequence data.

Analysis

The analysis button is used to isolate candidates for further investigation by e.g. TESS. It uses the parameters given by the analysis properties given in the settings section of the command panel.

4.1.4 Menu

The Menu consists of Save, Load, Properties and Exit buttons.

Load

The Load button can be used to retrieve previously calculated results to perform new analysis on them.

Save

The Save button saves the calculations in an XML - like format.

Properties

Not active in this version of PHYREX.

Exit

Exits the system.

4.2 Tree Panel

The Tree Panel becomes active after the Tree button in the Command Panel has loaded a tree in Schreiber format. The tree displayed is used to populate the leaf nodes in the tree with data. Before running any calculations each leaf node must be visited and data must be loaded in the Node panel for each leaf.

In addition the user must specify one internal node to analyse, See Node Panel Section.

4.3 Node Panel

The Node panel takes care of the display and collection of leaf specific data. For each analysis some files must be collected for all the leafs in the tree, this is done by the Node panel.

Cel File

The Cel File button is used to specify the path to the file containing micro array experiment values for the leaf. This is stored in the .cel files collected from Affymetrix.

Seq File

The Seq File button specifies the upstream region sequences found from EnsEmbl for each species.

Main Species

The button is used to specify the species used to guide the normalisation of the gene expression data. Normally this is the species the AffyMetrix GeneChip is designed for. Only one leaf should have the value "Yes" in each calculation.

Analysis

The Analysis button is used to specify the tripplett that should be analysed. Only internal nodes may have the value "Yes". The analysis is performed on the node specified and its two childs.

Gene Table

The Gene Table button displays the calculation results of this node. It gets active after the calculation is performed.

Edge Table

The Edge Table button displays the changes along the branch between this node and its parent.

4.4 Progress Panel

The Progress panel is used to display the progress in the calculations after the user has clicked on the calculation button in the Command panel. The calculations take some time and the Progress panel displays an approximation of how much work that has been done.

5 Constraints

5.1 Purpose

The purpose of this User Maual is to give the user a brief introduction to how PHYREX work and how to install it. For documentation on how to install other software that PHYREX needs to work I refer the reader to that particulare software user manual.

5.2 Other Documentation

A more thorough documentation of PHYREX is available in the thesis document found on http://www.rossnes.org/phyrex.