Manual for PrIMETV and Reconcile

Lars Arvestad Eva Schreil Bengt Sennblad

October 11, 2006

Contents

Chapter 1

Reconcile

The program RECONCILE is used to produce parsimonious reconciliations, that is, relate a given tree to a host tree with so many vertices as possible induced by the host tree. The basic usage is as follows.

```
reconcile <T> <S> [<leafmapping>]
```

1.1 Input

The mandatory input files T and S contain phylogenetic trees in Newick format. A third input file can be used to relate the leaves in T with leaves in S. This is done by, on each line in that file, have the name of leaf in T, followed by space, and then the name of the leaf in S hosting the T leaf. For example, here is a simple dataset:

T:
```
((a, b), c);
```

S:
```
(x, (y, z));
```

Leaf mapping:
```
a x
b y
c z
```

The leafmapping is mandatory, although not necessarily using a separate file. The user also has the option of embedding the leafmapping in T, in which case the special leaf-mapping file can be left out. The embedding is done using an extended Newick notation, by utilizing annotated comments. In Newick format, comments are within square brackets. For example, `[this is a comment]`. The annotated comments recognized in RECONCILE starts with the special tag `&&PRIME`, and the leaf mapping for a leaf `a` to `x` looks like

```
a [&&PRIME S=x]
```

Hence, the full example from above becomes:

T:
```
((a [&&PRIME S=x], b [&&PRIME S=y]), c [&&PRIME S=z]);
```

S:
```
(x, (y, z));
```

This input is just some of what can be expressed using PRIME comments, and the file format section (section 4) will describe the rest of the annotated comments.

1.2 Output

The output produced by RECONCILE is a further annotation of T, see section 4, which defines the reconciliation. Thus, the output can be used directly as input to PRIMETV.

Chapter 2

readReconciliation

This program intends to make the construction of general reconciliations, not just most parsimonious reconciliations, easier. By utilizing a tabular format, specifying a reconciliation is made easier.

2.1 Input

The basic usage pattern is:

```
readReconciliation [<options>] <host> <guest> <reconciliation>
```

Both the host tree (`<host>`) and the guest tree (`<guest>`) are in Newick format, with or without PRIME extensions. The reconciliation table (`<reconciliation>`) is a regular text file with two tab-delimited columns. The left column contains id numbers for the host tree and the right column is a comma-separated list of ids for guest tree vertices. This mapping from host to gene tree encodes any valid reconciliation.

The PRIME programs automatically assign vertex identities in a consistent way, and using PRIMETV with the option '-i' can help you get these identity

Host	Guest
0	0
1	1
2	2
3	3
4	4

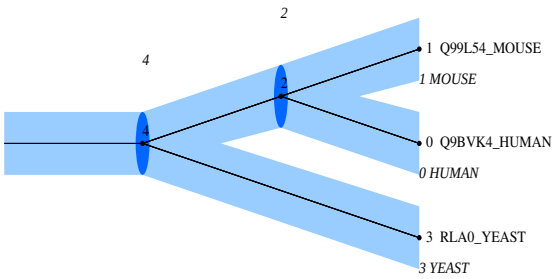

Figure 2.1: A simple example of a parsimonious reconciliation.

—Full—		—Reduced—	
Host	**Guest**	**Host**	**Guest**
0	0	0	0
1	1	1	1
2	0, 1, 2	2	0, 1
3	3	3	3
4	4	4	4

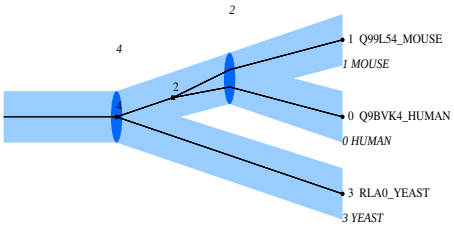

Figure 2.2: A non-parsimonious reconciliation.

numbers. However, you can also assign your own ids to vertices in both host and guest tree using the PRIME-format and the ID tag (see section 4.1).

Depending on how the guest tree vertices are given, there are two 'flavors' of the tabular format. In a *full* reconciliation, the right column comprise the guest tree vertices whose incoming edge appear on the incoming edge of the host tree vertex.

Figure 2.1 contains a small example with two three-leaf trees. The reconciliation does not imply any gene duplications/losses and we simply explains what guest tree vertices the host tree vertices should "contain".

If we want guest tree vertex 2 to *not* be a speciation, and hence appear before species vertex 2, we can write the entry for species vertex 2 to say that "guest tree vertices 0 and 1 should be mapped to this vertex, and vertex 2 comes above it". This is seen in Figure 2.2. There the second kind of notation for reconciliations can also be found, the *reduced* notation. One can notice that it is somewhat redundant to specify guest vertex 2: if both 0 and 1 are already mapped to host vertex 2, then vertex 2 is either also mapped there, or appears even higher up.

Figure 2.3 removes the vertex correspondance between the two trees even further. The value of the reduced notation becomes a little bit more clear here.

In Figure 2.4, all gene duplications appear before the first speciation and after that, only gene losses shape the gene tree. The tabular format reflects that by explaining that all gene-tree vertices are to appear at or before vertex 4 in the species tree.

—Full—		—Reduced—	
Host	**Guest**	**Host**	**Guest**
0	0	0	0
1	1	1	1
2	0, 1, 2	2	0, 1
3	3	3	3
4	0, 1, 2, 3, 4	4	2, 3

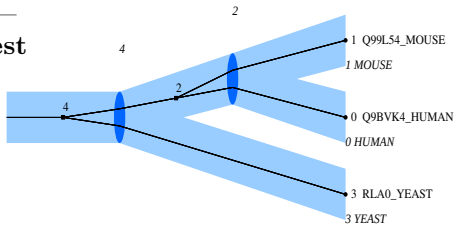

Figure 2.3: An even less parsimonious reconciliation.

—Full—		—Reduced—	
Host	**Guest**	**Host**	**Guest**
0	0	0	0
1	1	1	1
2	0, 1	2	0, 1
3	3	3	3
4	0,1,2 3, 4	4	0,1,3

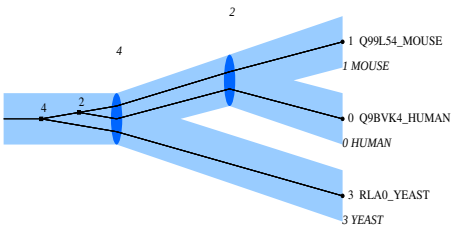

Figure 2.4: All duplication before the first speciation, then only gene losses.

2.2 Options and output

There are two main options two readReconciliation:

-q Quite mode. Without this option, redundant information (including the trees) is written to the terminal to aid interpretation of the tabular format. With this optino set, no more than the reconciled gene tree (similar to the output from Reconcile) is output.

-o Output is written to the file named by the string following '-o'.

In addition, the options -h and -u gives an explanatory text for how to use readReconciliation.

The output reconciled tree from readReconciliation can be used directly by PrIMETV.

Chapter 3

PrIMETV

The syntax for using PrIMETV is

```
primetv [options] <R> <S>
```

with the options as described below. The mandatory input files contains the reconciled tree R, perhaps produced by Reconcile, and the host tree S. For example, a command sequence such as

```
reconcile T S leafmapping > R
primetv R S
```

will produce a result such as in figure 3.1.

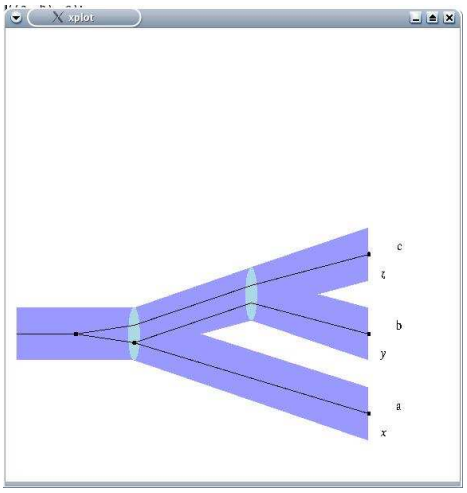

Figure 3.1: A simple reconciliation from the example

3.1 Input

As explained, the first input file R has a syntax that RECONCILE generates. If a non-parsimonious reconciliation is used, section 4 has to be studied to understand the format. The host tree S is easier to understand since it is a standard phylogenetic tree in Newick format. However, for best effect, it should have time information on the edges which will be used to scale the tree. If there is no time information, a straightforward unit length scaling of the edges in S will be used. There are two ways of submitting time information. First, the branch length field in the Newick format can be used for this. Second, there are fields in the extended comments, described below, that may contain edge or node times.

Please notice that the order of the subtrees matters for the layout. PRIMETV tries to reduce the number of crossings of guest tree edges, but there are corner cases that are not automatically identified. If you are not happy with the way either tree, R or S, is depicted, changing the order of subtrees will have the effect of rotating a vertex. This can sometimes improve the perception of the trees.

3.2 Options and output

-a Alternative layout. While the "height" of a host tree vertex is fixed, its placement on the other axis can be determined in two ways. The default way is to place a vertex where its two subtrees meet. By using option `-a` you instruct the program to place it in the middle of the leftmost and rightmost nodes in the subtrees.

-c You can set your own colors of the trees using this option. A specification is required, which is a set of three space separated color values, representing line color for the tree T, edge color for the host tree, followed by vertex color for S. A color is given either with its name, such as "black" or "lightblue", as it is described in GNU Plotutils[1], or by its hexadecimal RGB value (for example `#000000` for black and `00FF00` for green).

To exemplify, the specification

```
-c black #999999 yellow
```

draws a black tree within a host tree which has yellow vertices and grey edges.

-t Disregard branch lengths in the host tree. This option is automatically invoked where the host tree is missing time information.

-e A time scale is usually put at the side of the tree and lines mark where host tree vertices are placed. By issuing this option, edges are instead marked with their length in time.

[1] `www.gnu.org/software/plotutils`

Format name	Description
fig	Readable by xfig
gif	The common raster graphics format GIF
ps	PostScript
svg	Scalable Vector Graphics, a W3.org web standard
png	Raster graphics, "Portable Network Graphics"

Table 3.1: Output formats.

-n Do not show time annotations anywhere.

-f This option together with a string decides what kind of output shall be produced. For instance, `-f ps` makes PostScript be used. Table 3.1 summarizes the options.

-g No guest tree is given nor shown. Notice that the host tree then must be shown.

-h No host tree is given nor shown. Notice that the guest tree then must be shown.

-o Name the output file, e.g. `-o myoutput.ps`.

-i Put node numbers on the inner nodes.

-p Set pagesize. Default is 'a4'. Other options are 'letter' and 'a5'. See plotutils documentation for all choices.

-b Set bitmap size for X, GIF, and PNG. Format is WxH and default if '570x570', taken from plotutils.

-s Give a number for font scaling. By default, fonts are scaled so that text fits the illustration well. If you feel that the fonts are to small, you can for example issue `-s 1.5` to increase the size by a factor 1.5.

-v By default, trees are drawn with the root on the left and the leaves on the right. This options cause the tree to be drawn vertically instead.

-m Do not put any markers on the vertices in T.

-l You can decide how to order the leaves of the host tree using this option. The option takes the characters "r" or "l" as input, and makes the host tree vertices be rotated so that the larger subtree is on the right or on the left, respectively. Without this option, the tree is drawn in the order the sub trees were given in the input file.

Chapter 4

Tree annotations

Tree annotations are associated with leaves and subtrees and are positioned right after the leaf or subtree. The annotation starts with `[&&PRIME` and ends with a `]`, i.e., it is a simple Nexus comment. Inside the comment, separated by white space and optionally commas or colons for clarity, there should be keyword/value pairs. Table 4.1 list the currently defined keyword/value pairs.

We will now discuss the keywords and their applications.

4.1 Keywords in annotations

The `S` tag defines what leaf in the host tree (think "species") a leaf in the in-tree (think "gene") belongs to. It can also be used to name inner vertices of a tree. This is useful to put a name on a clade in the host tree, for instance.

The `ID` markup can be used to have full control over vertex id:s in the host tree S. This detailed control may make it easier to understand reconciliations. Normally, Reconcile andPrIMETV assign id:s to host tree vertices without the user's help. Since these id:s are used to define reconciliations, it is important that the automatic id assignment is the same in both programs. A user-defined reconciliation must also follow the same id assignment, and this is done by either editing a reconciliation from Reconcile, or by making sure, through user-defined id:s, that PrIMETV will use the same id:s as the reconciliation.

Here is an example of a tree with `S` and `ID` markup. Notice the irregular use of space and separators which is there to illustrate that the syntax is not rigid.

```
(
  (
    hum1[&&PRIME:S=Human ID=0],
    fly1[&&PRIME ID=1 S=Fly]
  )[&&PRIME ID=2],
  (
    hum2[&&PRIME S=Human ID=3],
    fly2[&&PRIME ID =4, S=Fly]
  )[&&PRIME ID=5]
```

`)[&&PRIME ID=6]`

The `AC` tag defines reconciliations and is somewhat complicated. The acronym comes from the mathematical term "anti-chain" and it is erroneously used for historical reasons[1]. The value associated with `AC` is a list of integers, for example `AC=(1 2 3)`, and the integers are id:s of host tree vertices that we want the tree edge to go through in the host tree. More specifically, if the `AC` tag above is associated with a leaf, then it has been defined that the leaf belongs in the host tree leaf with id 1. The edge from the leaf to its parent will, according to the reconciliation, pass through

Tag	**Argument**	**Description**
S	string	Species annotation. Used to define what host node a node in the guest tree belongs to.
AC	int_list	Anti-chains on edge. Used to define a reconciliation. See section 4.2 for instructions on this. Anti chains are numbered by their species node, and if anti-chains X, Y and Z are on a gene tree edge, the markup will contain for example `AC=(7,8,9)`, where 7, 8, 9 is the assigned ids of X, Y and Z.
ID	int	The node identifier. This cannot be arbitrarily set. Ideally, it should be as output by a tree generator application. However, as long as the ids are node unique you should be fine. Notice that it does not work to set the ID for just some of the nodes. Either all or no nodes have an ID tag.
NT	float	This is a time associated with a node, to be interpreted as the time elapsed since the root bifurcation. Notice that it is *not* the time over the edge to parent.
ET	float	This is a time associated with an edge, that is, the time passed since the *last* bifurcation.
BL	float	Branch length. This tag should not be used, but is reserved for implementation reasons. The branch length, given as for example as `<leafname>:<branchlength>`, or `(...):<branchlength>` are stored internally in PRIME using this tag. In other PRIME applications, it is possible to instruct the program to interpret branchlength as for example edge time.

Table 4.1: PRIME extended markup.

[1] We will rename this tag as soon as we have stopped arguing about what to call it.

4.2 Reconciliation examples

To make the `AC` tag more clear, some examples are necessary. Consider the host tree, here a species tree, over human, mouse and yeast.

```
(
  (
    MOUSE:0.0110,
    HUMAN:0.0110
  ):0.1466,
  YEAST:0.1576
);
```

If we have the following gene tree,

```
(
  (
    Q99L54_MOUSE [&&PRIME S=MOUSE],
    Q9BVK4_HUMAN [&&PRIME S=HUMAN]
  ),
  RLA0_YEAST [&&PRIME S=YEAST ]
);
```

the most parsimonious reconciliation can be had from running RECONCILE on it, which will produce `AC` tags.

```
(
  (
    Q99L54_MOUSE [&&PRIME S=MOUSE AC=(1)],
    Q9BVK4_HUMAN [&&PRIME S=HUMAN AC=(2)]
  ) [&&PRIME AC=(3)],
  RLA0_YEAST [&&PRIME S=YEAST AC=(0) ]
) [&&PRIME AC=(4) ]
```

Notice how we from these tags can deduce that PRIMETV and RECONCILE put the id:s 0, 1, and 2 on yeast, mouse and human. The ancestor for the vertebrates has id 3, while the root of the species tree has id 4. Running PRIMETV on the result yields an illustration such as in Figure 4.1.

Suppose we decide, using some auxilliary information such as gene order, that the ancestor of the human and mouse genes is not a speciation. We want it to be lifted up above node 3 in the species tree, and the reconciliation is hence altered slightly:

```
(
  (
    Q99L54_MOUSE [&&PRIME S=MOUSE AC=(1 3)],
    Q9BVK4_HUMAN [&&PRIME S=HUMAN AC=(2 3)]
  ),
  RLA0_YEAST [&&PRIME S=YEAST AC=(0) ]
) [&&PRIME AC=(4) ]
```

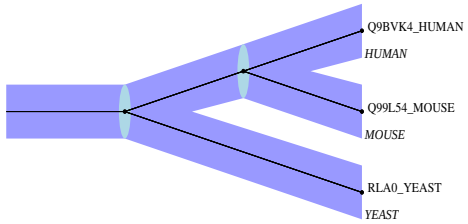

Figure 4.1: A simple parsimonious reconciliation.

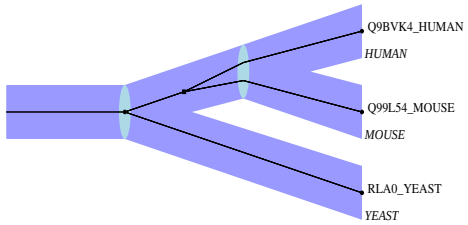

Figure 4.2: Lifting a speciation to become a duplication followed byt two independent later gene losses.

Taking this one step further, we could want both inner vertices in the gene tree to be duplications and have both appear before the root in the species tree.

```
(
  (
    Q99L54_MOUSE [&&PRIME S=MOUSE AC=(1 3 4)],
    Q9BVK4_HUMAN [&&PRIME S=HUMAN AC=(2 3 4)]
  ),
  RLA0_YEAST [&&PRIME S=YEAST AC=(0 4) ]
)
```

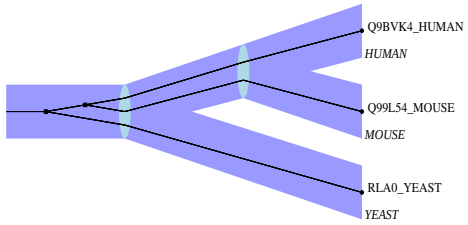

Figure 4.3: Every vertex is a duplication.

4.3 About non-binary trees

The code only supports binary trees. This is because our methods were designed for fully resolved rooted trees. However, we have also implemented *reading* of trees with node degrees higher than one. This means that binary nodes are handled in the natural way, but nodes with higher degrees are converted to several binary nodes in an arbitrary fashion. The new edges get branchlengths set to 0.