Exploring the space of gene/species reconciliations with transfers

Yao-Ban Chan, Vincent Ranwez, Celine Scornavacca

Noname manuscript No.
(will be inserted by the editor)

Exploring the space of gene/species reconciliations with transfers

Yao-ban Chan · Vincent Ranwez · Céline Scornavacca

Received: date / Accepted: date

Abstract Reconciliations between gene and species trees have important applications in the study of genome evolution (e.g. sequence orthology prediction or quantification of transfer events). While numerous methods have been proposed to infer them, little has been done to study the underlying reconciliation space. In this paper, we characterise the reconciliation space for two evolutionary models: the $\mathbb{DTL}$ (duplication, loss and transfer) model and a variant of it — the no-$\mathbb{TL}$ model — which does not allow $\mathbb{TL}$ events (a transfer immediately followed by a loss).We provide formulae to compute the size of the corresponding spaces and define a set of transformation operators sufficient to explore the entire reconciliation space. We also define a distance between two reconciliations as the minimal number of operations needed to transform one into the other and prove that this distance is easily computable in the no-$\mathbb{TL}$ model. Computing this distance in the $\mathbb{DTL}$ model is more difficult and it is an open question whether it is NP-hard or not. This work constitutes an important step toward reconciliation space characterisation and reconciliation comparison, needed to better assess the performance of reconciliation inference methods through simulations.

Keywords reconciliation, counting, distance, gene duplication, gene transfer, sampling, phylogenomics

1 Introduction

Gene families evolve through a complex process involving, among other things, incomplete lineage sorting and evolutionary events such as speciation ($\mathbb{S}$), gene duplication ($\mathbb{D}$), horizontal gene transfer ($\mathbb{T}$) and gene loss ($\mathbb{L}$). Genomes of related species share a common evolutionary history that leaves shared footprints on them that are used to infer the species phylogeny (depicted as a *species tree*). However, some evolutionary events (such as losses, duplications, or transfers) act only on some genes, and so their resulting gene histories (depicted as *gene trees*) may differ from the overall species tree. Each series of evolutionary events which explains the observed discrepancy between gene and species trees is called a reconciliation. See Figure 1 for an example of a gene and species tree and a reconciliation between the two.

Yao-ban Chan
School of Mathematics and Physics, The University of Queensland, St. Lucia, QLD 4072, Australia, E-mail: yaoban.chan@uq.edu.au

Vincent Ranwez
Montpellier SupAgro (UMR AGAP), 2 place Pierre Viala 34060 Montpellier Cedex 02, France, E-mail: ranwez@supagro.inra.fr

Céline Scornavacca
Institut des Sciences de l'Evolution (ISEM, UMR 5554 CNRS), Université Montpellier II, Place E. Bataillon - CC 064 - 34095 Montpellier Cedex 5, France, E-mail: celine.scornavacca@univ-montp2.fr

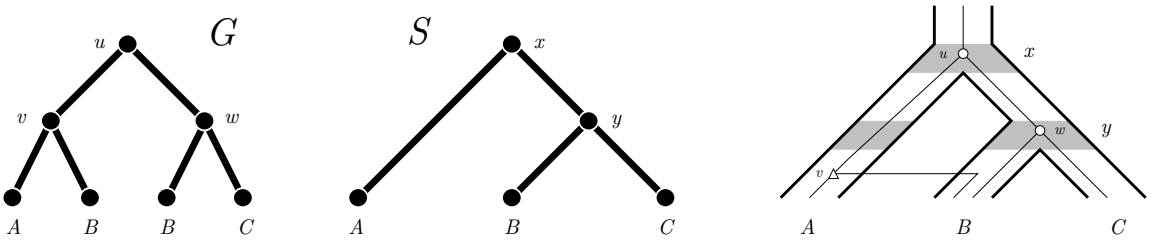


Fig. 1 An example of a gene and species tree and a reconciliation between the two. Here, there are two copies of the gene in species B, and a single copy in species A and C.

Reconciliation methods aim to infer the macro events (e.g. gene duplication and gene transfer) undergone by a gene family, by constructing reconciliations which explain the differences between the gene and species trees. Standard models either consider only speciation, gene duplication and loss as macro events — the $\mathbb{DL}$ model — or additionally consider transfer events — the $\mathbb{DTL}$ model. In some parts of the Tree of Life, gene transfers are very uncommon; in these cases, the $\mathbb{DL}$ model is a reasonable proxy of gene evolution, having the advantage of reducing false transfer prediction. However, in other kingdoms gene transfers are the main mechanism for introducing genomic novelties and cannot be ignored. Hence, depending on the studied species, one should favor the $\mathbb{DL}$ model, e.g. for mammals, or the $\mathbb{DTL}$ one, e.g. for bacteria that are known to regularly exchange genomic material through transfers [2].

Accurate inference of these evolutionary events plays an important role in studying genome evolution as well as in inferring orthology relationships [17,11], and may also help to improve phylogeny inference and datation [3,1,5,15,18]. However, although considerable work has been done on inferring reconciliations, comparatively less study has been made of the underlying space of all possible reconciliations.

In [8], Doyon *et al.* explored the space of all possible reconciliations in the $\mathbb{DL}$ model. They gave an algorithm to count the number of possible reconciliations in this space and defined an operator set which allows us to transform any reconciliation into any other. The minimal number of operations needed for this transformation defines a distance between reconciliations. These operators allowed them to explore the space of reconciliations; they showed how to construct a tree which allows an efficient traversal of all or some reconciliations in the space, using this to analyse the distribution of reconciliations and the relationship between cost, number of reconciliations in the space and their distance to optimality. They conclude that parsimony is indeed a justified paradigm for the inference of evolutionary scenarios for the $\mathbb{DL}$ model.

In this paper, we extend the basis of the work of Doyon *et al.* to the more complex $\mathbb{DTL}$ model [10,14,9,20]. We aim both at exploring the space of all possible $\mathbb{DTL}$ reconciliations and at defining a distance measure between two $\mathbb{DTL}$ reconciliations. In order to do this, we shall define an operator set which allows us to transform one $\mathbb{DTL}$ reconciliation into another. Doyon *et al.* defined operators for this purpose for the $\mathbb{DL}$ model; some of our operators are natural extensions of theirs, and we introduce new operators to obtain a complete set which can transform any reconciliation into any other. We anticipate that our work here will enable the development of techniques to draw similar conclusions to Doyon *et al.* about the space of $\mathbb{DTL}$ reconciliations.

This paper provides a deeply needed theoretical framework for the $\mathbb{DTL}$ model, presenting the first formal characterisation of the valid reconciliation space; this is a prerequisite for proving the correctness of any algorithm that aims to compute $\mathbb{DTL}$ reconciliations. This framework allows for many possible applications; as a demonstration, we present an algorithm to count the number of all possible reconciliations between a gene tree and a dated species tree under the $\mathbb{DTL}$ model. Finally, we prove that the operators introduced here are sufficient to transform any valid reconciliation into any other one, hence allowing a full exploration of the valid reconciliation space and defining a distance between reconciliations. This will, among other things, allow a better comparison of

reconciliations — for example for software validation — and to design Bayesian reconciliation methods for complex gene evolution models.

In the $\mathbb{DTL}$ model [9], loss events are not considered separately but only as part of speciation-loss ($\mathbb{SL}$) or transfer-loss ($\mathbb{TL}$) events (duplication-loss events leave no trace and are therefore undetectable). As we shall show, allowing $\mathbb{TL}$ events expands the space of possible reconciliations greatly and changes its structure. For this reason, we consider here two related models: one where $\mathbb{TL}$ events are not allowed and one where they are. We refer to the former model as the no-$\mathbb{TL}$ model.

2 Notation

Let $T = (V(T), E(T))$ be a (rooted) tree with labelled leaf vertices. We denote the root of T by $r(T)$ and the leaves of T by $L(T)$. If $x \in L(T)$, then we denote the label of x by $s(x)$. The (multi)set of all labels of leaves in T is denoted by $\mathcal{L}(T)$.

If $x \in V(T)$, we define T_x to be the subtree of T with root x and the *height* of x (denoted $h(x)$) to be the number of edges on the unique path from $r(T)$ to x (so for example, $h(r(T)) = 0$). If $x \in V(T)$ is not the root of T, then x_p and $e(x)$ denote its parent vertex and parent edge respectively. If $x \in V(T)$ is not a leaf, then x_l and x_r are its left and right child vertices respectively. The child vertices are unordered, so they are interchangeable. For edges $e \in E(T)$, we define e_t to be the target vertex of e, and the height of e as $h(e) = h(e_t)$.

We define an *element* of T to be a member of $V(T) \cup E(T)$. For two elements x, y of T, we say that x is *below* y (and y is *above* x) and write $x \leq y$ if y lies on the unique path from $r(T)$ to x.

We define a gene tree G as a tree where each leaf represents an extant gene. Likewise, we define a species tree S as a tree in which each leaf represents an extant species. The labels of the leaves of S are the identifiers of the species, so they are unique, while internal vertices represent speciation events. We suppose that gene and species trees are rooted and binary and that $\mathcal{L}(G)$ is a (multi)set such that each label of $\mathcal{L}(G)$ appears in $\mathcal{L}(S)$ (denoting in which species each gene is found).

A species tree S is said to be *dated* if it is associated to a time function θ_S which represents the time separating a vertex from any extant species descending from it, i.e. $\theta_S : V(S) \to \mathbb{R}^+$ such that if $y \leq x$ then $\theta_S(y) \leq \theta_S(x)$ and if $x \in L(S)$ then $\theta_S(x) = 0$. In order to have time-consistent scenarios [9], we *subdivide* S by creating *artificial* vertices in every branch where a speciation occurs at the same time in another branch. These artificial vertices represent time boundaries, which helps us ensure that transfer events in the reconciliation only occur between co-existing entities. We also add an edge which is the parent of $r(S)$. We denote the resulting unary-binary tree by S'. For any vertex $x \in V(S')$, we define $\Lambda(x) = 1$ if x is artificial and 0 otherwise. See [9, Figure 3] for an example of subdivision. If not specified otherwise, a reconciliation is always between the gene tree G and the subdivided species tree S'.

In the $\mathbb{DTL}$ model [9], seven elementary (combinations of) events are considered: $\mathbb{S}$ speciation, $\mathbb{D}$ duplication, $\mathbb{T}$ transfer, $\mathbb{SL}$ speciation-loss, $\mathbb{TL}$ transfer-loss, $\varnothing$ no event (indicating that a gene lineage has crossed a time boundary), and $\mathbb{C}$ contemporary event (associating an extant gene copy to its corresponding species). In the next section we consider a simplified version of this model where $\mathbb{TL}$ events are not allowed. Note that this model is not biologically meaningful, and is analysed here because of its theoretical properties and as an intermediate step toward the full $\mathbb{DTL}$ model.

3 The no-$\mathbb{TL}$ model

For the no-$\mathbb{TL}$ model, a key observation is that reconciliations are completely determined by the images of the vertices of G. For example, if we know that a particular gene is transferred from another species, the transfer must have occurred where its parent vertex in the gene tree is mapped.

We follow the notation of [8] and define a reconciliation as being a mapping of the vertices of G onto the elements of S'.

Definition 1 A reconciliation is a mapping $\alpha : V(G) \to V(S) \cup E(S')$. For any $u \in V(G)$, if $u \in L(G)$, then $\alpha(u)$ must be the unique extant species which contains the gene u ($\mathbb{C}$ event). Otherwise:

- if $\alpha(u) \in V(S)$, then $\alpha(u_l) \leq e(\alpha(u)_l)$ and $\alpha(u_r) \leq e(\alpha(u)_r)$, or vice versa; ($\mathbb{S}$ event)
- if $\alpha(u) \in E(S')$, then one of the following conditions is satisfied:
  - $\alpha(u_l), \alpha(u_r) \leq \alpha(u)$; or ($\mathbb{D}$ event)
  - $\alpha(u_l) \leq \alpha(u)$ and $\alpha(u_r) \not\leq \alpha(u), h(\alpha(u_r)) \geq h(\alpha(u))$, or vice versa. ($\mathbb{T}$ event)

Moreover, $\mathbb{SL}$ and $\varnothing$ events can be deduced from α in the following way: for every vertex u, an $\mathbb{SL}$ (respectively $\varnothing$) event is associated to each non-artificial (resp. artificial) vertex of S' on the path from $\alpha(u_p)$ to $\alpha(u)$, extremities excluded.

Since $\mathbb{SL}$ and $\varnothing$ events can be deduced from α, in this section we will focus only on $\mathbb{S}$, $\mathbb{D}$ and $\mathbb{T}$ events. If u is a $\mathbb{T}$ event, the target of the transfer can be inferred by observing the positions of $\alpha(u_l)$ and $\alpha(u_r)$. Note that by this definition we cannot map a vertex of G to an artificial vertex in S'. This is because mapping a vertex of G to a vertex of S' represents an $\mathbb{S}$ event, and $\mathbb{S}$ events can occur only at vertices of S.

This definition enforces certain relations between the heights of the images of u and its children. More precisely, if x and y are elements of S' and either $h(x) < h(y)$, or $h(x) = h(y)$ and $x \in E(S')$, we say that x *allows* y. By Definition 1, $\alpha(u)$ must allow both $\alpha(u_l)$ and $\alpha(u_r)$.

Observe that, due to subdivision, two events belonging to two different reconciliations located in different branches of S' may correspond to the same event in the original species tree S. This implies that two reconciliations that are distinct according to Definition 1 in S' may be equivalent in S. Because of this, we define a "canonical" reconciliation which is a representative member of this equivalent set. In essence, a $\mathbb{D}$ or $\mathbb{T}$ event can be "moved lower" if the corresponding vertex of G is mapped just above an artificial vertex, and its children also allow it to be mapped to the edge below that vertex. In a canonical reconciliation [16,7], no $\mathbb{D}$ or $\mathbb{T}$ event can be moved lower:

Definition 2 A reconciliation α is *canonical* if neither of the following two cases happens for any $u \in V(G)$:

- $\alpha(u)$ is a $\mathbb{D}$ event, $(\alpha(u))_t$ is artificial, and $h(\alpha(u)) < \min(h(\alpha(u_l)), h(\alpha(u_r)))$;
- $\alpha(u)$ is a $\mathbb{T}$ event transferring to the edge x, both $(\alpha(u))_t$ and x_t are artificial, and $h(\alpha(u)) < \min(h(\alpha(u_l)), h(\alpha(u_r)))$.

3.1 Reconciliation space

In order to describe the reconciliation space, we define a *partial* reconciliation α to be a reconciliation where we have assigned $\alpha(v)$ for all vertices v of G with height greater than some number, say h. Now let u be a vertex of height h.

Definition 3 Given a partial reconciliation α, if $\alpha(u_l) \leq \alpha(u_r)$ or vice versa, we say u is a *forced duplication*. Otherwise, the *speciation point* of u is the lowest vertex of S that is an ancestor of both $\alpha(u_l)$ and $\alpha(u_r)$.

In a model with no transfers, the speciation point (where it exists) of a vertex is identical to the so-called LCA mapping of the vertex [10], defined as the lowest common ancestor of all species which contain an extant gene descended from that vertex. It is known that the LCA mapping induces a most parsimonious reconciliation in this model.

Definition 4 Given a partial reconciliation α, we define the set $A_\alpha(u)$ as follows:

- If u is a forced duplication, $A_\alpha(u)$ consists of all edges above both $\alpha(u_l)$ and $\alpha(u_r)$. ($\mathbb{D}$ event)

- Otherwise, $A_\alpha(u)$ is the union of:
  - all edges above the speciation point of u; ($\mathbb{D}$ event)
  - the speciation point of u; ($\mathbb{S}$ event)
  - any edge lying on the path between the speciation point of u and $\alpha(u_l)$ which allows $\alpha(u_r)$, or vice versa. ($\mathbb{T}$ event)

Lemma 1 *Let α be a partial reconciliation to height $h+1$ and u an internal vertex of G with height h. Then $A_\alpha(u)$ is the set of all possible mappings for u, i.e. $A_\alpha(u)$ is the set of images of u which appear in at least one full reconciliation between G and S' that extends α.*

Proof. We need to show that u is a valid event (according to Definition 1) if it is mapped to an element of $A_\alpha(u)$ and not a valid event if it is mapped to any other element of S'. Certainly if it is not mapped to a valid event, no reconciliation which extends α can include that image. Conversely, if it is mapped to a valid event, we can trivially extend the partial reconciliation to a full one by mapping all un-assigned vertices of G to $e(r(S'))$ as $\mathbb{D}$ events. There are two cases:

- u is a forced duplication, i.e. $\alpha(u_l) \leq \alpha(u_r)$ (or vice versa). From Definition 1, neither $\mathbb{S}$ nor $\mathbb{T}$ events have this property, and since it is not a leaf, u can only be mapped to a $\mathbb{D}$ event (hence the name). It is clear that any edge which lies above both $\alpha(u_l)$ and $\alpha(u_r)$ satisfies the conditions of a $\mathbb{D}$ event, and no other element of S' does.
- u is not a forced duplication. Then there are three cases corresponding to the type of event that u could be mapped to in S':
  - u is mapped to a $\mathbb{D}$ event. Then $\alpha(u) \geq \alpha(u_l), \alpha(u_r)$ and so $\alpha(u)$ must lie above or at the speciation point. Since $\alpha(u) \in E(S')$ by definition, u can only be mapped to an edge above the speciation point. Any such edge satisfies the conditions of a $\mathbb{D}$ event (and no other element of S' does).
  - u is mapped to an $\mathbb{S}$ event. Again $\alpha(u) \geq \alpha(u_l), \alpha(u_r)$ and so $\alpha(u)$ must lie above or at the speciation point. However, we also require that the paths from $\alpha(u_l)$ and $\alpha(u_r)$ to $\alpha(u)$ be mutually exclusive; otherwise the child edges of u would be mapped to the same branch in S' and it would not be a valid $\mathbb{S}$ event. This property holds for the speciation point of u, so we can map it to there, but not for any other vertex above the speciation point (and we cannot map u to an edge of S' as a speciation).
  - u is mapped to a $\mathbb{T}$ event. We can take $\alpha(u) \geq \alpha(u_l)$ without loss of generality. Then, since $\alpha(u) \not\geq \alpha(u_r)$, $\alpha(u)$ must be between the speciation point and $\alpha(u_l)$. Moreover, $h(\alpha(u_r)) \geq \alpha(u)$ so $\alpha(u)$ must allow $\alpha(u_r)$. It can be checked that any edge of S' which satisfies these conditions also satisfies the conditions of a $\mathbb{T}$ event (and by the above reasoning, no other element does).

  As there are no other types of events that u can be mapped to, u cannot be mapped as a valid event to any other element of S'.

□

Lemma 1 implies that we can construct all possible reconciliations using a bottom-up approach. We can also count the number of possible reconciliations in this fashion, as we prove next.

Theorem 1 *Let $R(u,x)$ be the number of reconciliations between G_u and S' where $\alpha(u) = x$. If $u \in L(G)$, then $R(u,x) = 1$ if $x \in L(S)$ and $s(u) = s(x)$, and 0 otherwise. If $u \notin L(G)$, then*

$$R(u,x) = \begin{cases} \displaystyle\sum_{\substack{x_1 \leq e(x_l),\\ x_2 \leq e(x_r)}} [R(u_l,x_1)R(u_r,x_2) + R(u_l,x_2)R(u_r,x_1)] & \text{if } x \in V(S), \\ \displaystyle\sum_{x_1,x_2 \leq x} R(u_l,x_1)R(u_r,x_2) + \sum_{\substack{x_1 \leq x, x_2 \not\leq x,\\ h(x_2) \geq h(x)}} [R(u_l,x_1)R(u_r,x_2) + R(u_l,x_2)R(u_r,x_1)] & \\ & \text{if } x \in E(S'). \end{cases}$$

Proof. The case for $u \in L(G)$ is obvious, so suppose that $u \notin L(G)$. If $x \in V(S)$, then u is an $\mathbb{S}$ event, so u_l maps to an element below the left child edge of x and u_r maps to an element below the right child edge of x, or vice versa. The expression follows by calculating the number of possibilities for reconciling the child subtrees. If $x \in E(S')$, then either u is a $\mathbb{D}$ event, in which case both u_l and u_r must map to an element below x, or it is a $\mathbb{T}$ event. In the latter case, u_l maps to an element below x and u_r maps to an element which is at a greater height than x, but not below it, or vice versa. The expression again follows from the possibilities for reconciling the child subtrees. □

The total number of possible reconciliations between G and S' is given by $\sum_{x \in V(S) \cup E(S')} R(r(G), x)$. This can be calculated from Theorem 1 by recursively computing $R(u, x)$ for all $u \in V(G)$ and $x \in V(S) \cup E(S')$, considering the vertices of G in post-order (the order of vertices of S' does not matter). For an example of this, see Appendix A.

Once $R(u, x)$ is computed for all (u, x) pairs, the values can be stored in a directed acyclic graph, with each vertex of the graph corresponding to a (u, x) pair and each edge representing the fact that the target vertex was used to compute the $R(u, x)$ value of the parent vertex. A uniformly random reconciliation can then be drawn from the space of all possible reconciliations by a top-down traversal of this graph, weighting each child vertex by its $R(u, x)$ value. This is a modification of Algorithm 1 of [8].

With a little more care, we can also count the number of canonical reconciliations. As noted before, reconciliations which belong to the same equivalence class are not really "different", in the sense that they represent the same evolutionary scenario. As such, it provides a more biologically meaningful picture of the reconciliation space if only canonical reconciliations are included. Thus, is it more interesting to count the number of canonical reconciliations rather than simply the total number of reconciliations; the former is free of distortions induced in the latter by the subdivision of S.

Theorem 2 *Let $R'(u, x)$ be the number of canonical reconciliations between G_u and S' where $\alpha(u) = x$. If $u \in L(G)$, then $R'(u, x) = 1$ if $x \in L(S)$ and $s(u) = s(x)$, and 0 otherwise. If $u \notin L(G)$, then*

$$R'(u,x) = \begin{cases} \displaystyle\sum_{\substack{x_1 \leq e(x_l),\\ x_2 \leq e(x_r)}} [R'(u_l,x_1)R'(u_r,x_2) + R'(u_l,x_2)R'(u_r,x_1)] & \text{if } x \in V(S) \\ \displaystyle\sum_{x_1,x_2 \leq x} R'(u_l,x_1)R'(u_r,x_2) & \text{if } x \in E(S') \\ \quad -\Lambda(x_t) \displaystyle\sum_{x_1,x_2 < x} R'(u_l,x_1)R'(u_r,x_2) & \\ \quad + \displaystyle\sum_{\substack{x_1 \leq x, x_2 \not\leq x,\\ h(x_2) \geq h(x)}} [R'(u_l,x_1)R'(u_r,x_2) + R'(u_l,x_2)R'(u_r,x_1)] & \\ \quad - \displaystyle\sum_{\substack{x_1 < x, x_2 \not\leq x,\\ h(x_2) > h(x)}} \Lambda(x_t)\Lambda(x_3)\,[R'(u_l,x_1)R'(u_r,x_2) + R'(u_l,x_2)R'(u_r,x_1)] & \end{cases} \tag{1}$$

where in the last line, x_3 denotes the vertex ancestor of x_2 of the same height as x.

Proof. The reasoning is the same as for Theorem 1, except we must take care not to count non-canonical reconciliations. This can be done by ensuring that when counting $R'(u, x)$, the mapping $\alpha(u) = x$ satisfies Definition 2. There are three cases:

- $\alpha(u)$ is an $\mathbb{S}$ event. This can never cause a reconciliation to be non-canonical.
- $\alpha(u)$ is a $\mathbb{D}$ event. If x_t is artificial, then potentially the $\mathbb{D}$ event could be moved downwards through it. However, we also need both children of u to be mapped to elements strictly below x, as otherwise the move would be "blocked" by one of them. Therefore, if $\Lambda(x_t) = 1$, and

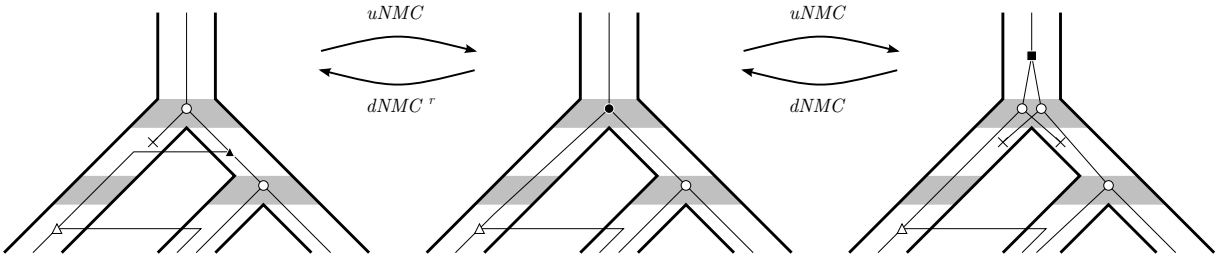


Fig. 2 The effect of the $uNMC$ and $dNMC$ operators. Here we operate on the filled vertex.

$x_1, x_2 < x$, the $\mathbb{D}$ event can be moved downwards and the reconciliation is non-canonical. This results in the term on the third line of (1).
- $\alpha(u)$ is a $\mathbb{T}$ event. If x_t and x_3 are both artificial and both children of u are mapped to elements with height strictly greater than $h(x)$, the $\mathbb{T}$ event can be moved downwards through x_t and x_3 to the next time-step and the reconciliation is non-canonical. This results in the term on the fifth line of (1).

□

As before, the total number of canonical reconciliations is given by $\sum_{x \in V(S) \cup E(S')} R'(r(G), x)$, and this can be computed by recursively computing $R'(u, x)$ in the order previously described.

3.2 Operators

We define two operators which change one reconciliation to another. These operators are the natural equivalent of the up and down Nearest Mapping Change operators introduced in [8], denoted respectively as $uNMC$ and $dNMC$, and so we also use these names for our operators.

Definition 5 Let α be a reconciliation and u an internal vertex of G. Let ω_u and ω_d be two elements of S' defined as follows:

- ω_u is the lowest element of $A_\alpha(u)$ above $\alpha(u)$ which is allowed by $\alpha(u_p)$;
- ω_d is one of the highest elements of $A_\alpha(u)$ below $\alpha(u)$.

If $\alpha(u)$ is an $\mathbb{S}$ event, then its two child edges both lie in $A_\alpha(u)$ and are legal choices for ω_d. In this case we denote the corresponding operators by $dNMC_u^l$ and $dNMC_u^r$. We define $uNMC_u(\alpha)$ and $dNMC_u(\alpha)$ as two mappings $V(G) \to V(S) \cup E(S')$, such that $uNMC_u(\alpha)(u) = \omega_u$ and $dNMC_u(\alpha)(u) = \omega_d$, and $uNMC_u(\alpha)(v) = dNMC_u(\alpha)(v) = \alpha(v)$ for any vertex $v \neq u$. If there is no legal choice for ω_u or ω_d, the corresponding operator cannot be applied.

We illustrate these operators in Figure 2.

Lemma 2 *If $uNMC_u(\alpha)$ and $dNMC_u(\alpha)$ are defined, they are legal reconciliations.*

Proof. By definition, ω_u and ω_d lie in $A_\alpha(u)$ and are allowed by $\alpha(u_p)$, so they are valid images for u. Since we have not changed the image of any other vertex, we only need to show that $\alpha(u_p)$ still satisfies one of the conditions of Definition 1, i.e. it is a valid event. But it can be seen that if $\alpha(u_p)$ satisfies one of these conditions in α, then moving $\alpha(u)$ to an ancestor or descendant does not invalidate that condition, as long as the new image is still allowed by $\alpha(u_p)$. Thus $\alpha(u_p)$ is still a valid event of the same type in both $uNMC_u(\alpha)$ and $dNMC_u(\alpha)$ and so these mappings are indeed valid reconciliations. □

It is intuitively easy to see that repeated application of $uNMC$ and $dNMC$ operators can change any reconciliation into any other (for example, by moving all vertices of G to $e(r(S'))$, and then moving them downwards to their desired images). A formal proof of this fact is an easy consequence of Theorem 3, presented in the next section.

3.3 Distance measure

In [8], Doyon *et al.* define a distance measure between two reconciliations, which they call $d(\alpha, \beta)$, as the smallest number of NMC operators needed to change α to β. We extend this definition to our model.

Definition 6 Let α and β be two reconciliations. We define $d(\alpha, \beta)$ to be the smallest number of $uNMC$ and $dNMC$ operators needed to convert α to β.

If we disallow transfers, this distance measure reduces to the measure in [8]. Note that as every $uNMC$ operator is the inverse of a $dNMC$ operator and vice versa, it is clear that this distance is indeed a metric.

Given two reconciliations, we can calculate the distance between them by defining a third reconciliation which can be reached from both by moving vertices upwards only.

Definition 7 Let α and β be two reconciliations. The *midpoint* γ of α and β is the reconciliation for which $\gamma(u)$ is an element of S' above both $\alpha(u)$ and $\beta(u)$ for all $u \in V(G)$, and there exists no reconciliation $\gamma' \neq \gamma$ with this property where $\gamma'(u) \leq \gamma(u)$ for all $u \in V(G)$.

Lemma 3 *Given $\gamma(u_l)$ and $\gamma(u_r)$, $\gamma(u)$ is the lowest element of S' above both $\alpha(u)$ and $\beta(u)$ which allows both $\gamma(u_l)$ and $\gamma(u_r)$. In particular, the midpoint is unique.*

Proof. It is certainly true from the definition that $\gamma(u)$ is above $\alpha(u)$ and $\beta(u)$, and it must allow $\gamma(u_l)$ and $\gamma(u_r)$ since γ is a valid reconciliation. Let y be the lowest element of S' with these properties. Since $y \geq \alpha(u)$ and $\gamma(u) \geq \alpha(u)$, we must have $y \leq \gamma(u)$.

Now consider the mapping γ' which is equal to γ everywhere except at $\gamma'(u) = y$. It suffices to show that γ' is a valid reconciliation, as the definition of the midpoint then gives $y = \gamma(u)$. To do this, we only need to check that $\gamma'(u_p)$ and $\gamma'(u)$ are valid events. The former follows from the fact that $y \leq \gamma(u)$. For the latter, there are two possibilities:

- $y \in E(S')$. Without loss of generality, we can take $\alpha(u) \geq \alpha(u_l)$. Then $y \geq \alpha(u) \geq \alpha(u_l)$ and $\gamma(u_l) \geq \alpha(u_l)$, so $y \geq \gamma(u_l)$ since y allows $\gamma(u_l)$ by definition. Since y also allows $\gamma(u_r)$, $\gamma'(u)$ must satisfy the conditions for either a $\mathbb{D}$ or a $\mathbb{T}$ event.
- $y \in V(S)$. Since any edge allows the same elements as its parent vertex, we see that y must be the lowest common ancestor of $\alpha(u)$ and $\beta(u)$ — otherwise, we could move it to one of its child edges. As in the previous case, $y \geq \alpha(u_l)$ and so $\alpha(u_l) \leq \gamma(u_l) < y$ (since a vertex does not allow itself) without loss of generality.
  Now if $\beta(u) = y$, it must be an $\mathbb{S}$ event and so one of $\beta(u_l)$ and $\beta(u_r)$ lies in the child subtree of y not containing $\alpha(u_l)$. Otherwise, $\beta(u)$ itself lies in that subtree, and one of $\beta(u_l)$ and $\beta(u_r)$ also does. If $\beta(u_l)$ lies in this subtree, then $\alpha(u_l)$ and $\beta(u_l)$ lie in different child subtrees of y, and so $\gamma(u_l) \geq y$, a contradiction. Thus $\beta(u_r) \leq \beta(u)$ and by the same argument as above we have $\beta(u_r) \leq \gamma(u_r) < y$. Morever, since $\alpha(u_l)$ and $\beta(u_r)$ lie in different child subtrees of y, so must $\gamma(u_l)$ and $\gamma(u_r)$, and so $\gamma'(u)$ is a valid $\mathbb{S}$ event.

The uniqueness of the midpoint follows immediately from the fact that there is a full ordering of all elements of S' above, say, $\alpha(u)$. □

This lemma allows us to easily calculate γ recursively from the leaves upward.

Theorem 3 *Let α and β be two reconciliations with midpoint γ. Let $A^\gamma_\alpha(u)$ be the subset of $A_\alpha(u)$ that lies strictly above $\alpha(u)$ and below $\gamma(u)$ (and includes $\gamma(u)$ itself). Then*

$$d(\alpha, \beta) = d(\alpha, \gamma) + d(\gamma, \beta) = \sum_{u \in V(G)} |A^\gamma_\alpha(u)| + \sum_{u \in V(G)} |A^\gamma_\beta(u)|.$$

In particular, $d(\alpha, \beta)$ is always finite, so any reconciliation can be transformed into any other by a finite number of $uNMC$ and $dNMC$ operators.

Proof. Firstly, we show by induction that in any sequence of reconciliations that transforms α to β, the vertex u must be mapped to $\gamma(u)$ in at least one reconciliation. This is certainly true for $u \in L(G)$, as here $\alpha(u) = \beta(u) = \gamma(u)$ by definition. Now let $u \notin L(G)$ and suppose that this property is true for all vertices $v \in V(G)$ with height greater than $h(u)$.

Let R be a sequence of reconciliations which transforms α into β but never maps u to $\gamma(u)$. Let y be the element of S' with lowest height to which u is mapped over all reconciliations in R. Since the $uNMC_u$ and $dNMC_u$ operators can only move $\alpha(u)$ to an element of S' above or below it (and no other $uNMC_v$, $dNMC_v$ with $v \neq u$ move $\alpha(u)$ at all), all images of u in reconciliations in R are descendants of y, or y itself.

By induction, R contains two reconciliations γ_1 and γ_2 (not necessarily distinct) such that $\gamma_1(u_l) = \gamma(u_l)$ and $\gamma_2(u_r) = \gamma(u_r)$. But $y \geq \gamma_1(u), \gamma_2(u)$, and by the definition of a reconciliation $\gamma_1(u)$ allows $\gamma_1(u_l)$ and $\gamma_2(u)$ allows $\gamma_2(u_r)$, so y must allow both $\gamma(u_l)$ and $\gamma(u_r)$. Now consider the mapping γ' with $\gamma'(u) = y$ and $\gamma'(v) = \gamma(v)$ for $v \neq u$. If $y < \gamma(u)$ then γ' is a valid reconciliation, but this contradicts the definition of γ. So we must have $y \geq \gamma(u)$ and that u must be mapped to $\gamma(u)$ at some stage in R, since γ is also valid. This proves that u is mapped to $\gamma(u)$ in any sequence of reconciliations that transforms α to β.

Now consider a sequence of operators that transforms α to γ by applying $uNMC$ operators to every vertex until it reaches its image in γ, starting with $r(G)$ and moving each vertex of G only if its parent has been moved. Because the children of any vertex u are fixed at $\alpha(u_l)$ and $\alpha(u_r)$ when we move u, we require $|A_\alpha^\gamma(u)|$ moves for this vertex. Therefore it is clear that

$$d(\alpha, \gamma) \leq \sum_{u \in V(G)} |A_\alpha^\gamma(u)|.$$

We still need to show that $|A_\alpha^\gamma(u)|$ is the minimum number of $uNMC_u$ operators required to move $\alpha(u)$ to $\gamma(u)$. Since the $uNMC_u$ operator moves u to either its parent vertex or edge, in any sequence of reconciliations transforming α to γ, u must be mapped to every edge between $\alpha(u)$ and $\gamma(u)$ at least once.

Now consider the set of images of u in the sequence defined above (including $\gamma(u)$ but not $\alpha(u)$), which is $A_\alpha^\gamma(u)$. This set must contain all edges between $\alpha(u)$ and $\gamma(u)$, and possibly also the speciation point of u in α. This means that either it is minimal among all sequences which transform $\alpha(u)$ to $\gamma(u)$, or that $A_\alpha^\gamma(u)$ contains the speciation point and the minimal set does not. In the latter case, since $\alpha(u)$ lies below its speciation point, it is a $\mathbb{T}$ event. Since $A_\alpha^\gamma(u)$ contains the speciation point of u in α, u must be an $\mathbb{S}$ or $\mathbb{D}$ event in γ. But if the minimal set does not contain an element of $V(S)$, there must exist a sequence of NMC operators which transforms α into γ without changing the type of u. This means that u is also a $\mathbb{T}$ event in γ, a contradiction. Therefore $A_\alpha^\gamma(u)$ is minimal, so we require at least $|A_\alpha^\gamma(u)|$ operators to change $\alpha(u)$ to $\gamma(u)$. Thus

$$d(\alpha, \gamma) \geq \sum_{u \in V(G)} |A_\alpha^\gamma(u)|,$$

and the equality of these two terms is shown.

Since u must always pass through $\gamma(u)$, and there exists a sequence of operators which transforms α to γ in the minimum number of operators required to transform $\alpha(u)$ to $\gamma(u)$ for all $u \in V(G)$, we see that the minimum-length sequence of reconciliations from α to β can be taken to pass through γ. This means that $d(\alpha, \beta) = d(\alpha, \gamma) + d(\gamma, \beta)$. However, because every $uNMC_u$ is the inverse of a $dNMC_u$ operator and vice versa, we know that $d(\gamma, \beta) = d(\beta, \gamma)$. This proves the theorem. □

We note that it is clear that Theorem 3 would not hold if we took a strict subset of $uNMC_u$ and $dNMC_u$ operators for any vertex u, as we require all of these operators to move u to all elements in $A_\alpha(u)$ for any reconciliation α.

4 The $\mathbb{DTL}$ model

Exploring the space of reconciliations when $\mathbb{TL}$ events are allowed is more complicated. Without $\mathbb{TL}$ events, a reconciliation is determined by the images of the vertices of G in S' (i.e. by the placement of $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{S}$ events). On the other hand, $\mathbb{TL}$ events are not unambiguously determined by these images, and in fact it is almost always possible to have more than one reconciliation with the same $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{S}$ events, but with different $\mathbb{TL}$ events. From now on, we therefore follow the notation of [16] and define a reconciliation as a mapping α from $V(G)$ to a sequence of elements of $V(S')$. The sequence $\alpha(u) = (\alpha_1(u), \alpha_2(u), \ldots, \alpha_\ell(u))$ represents the sequence of edges to which $e(u)$ is assigned (in both the argument and the value of α, a vertex represents its parent edge). We reproduce the full formal definition of a reconciliation [16, Definition 1] in Appendix B.

As before, we are also interested in counting the number of canonical reconciliations. In this model, a canonical reconciliation is one in which no $\mathbb{D}$, $\mathbb{T}$ or $\mathbb{TL}$ event can be moved lower. We also reproduce the formal definition of a canonical reconciliation [16, Definition 2] in Appendix B.

The space of reconciliations that we analyse is not theoretically the full space of all possible reconciliations. In particular, we impose five conditions:

- $\mathbb{DL}$ events are not allowed;
- $\mathbb{T}$ events followed immediately by an $\mathbb{L}$ event in the transferred branch are not allowed;
- consecutive $\mathbb{TL}$ events in the same time-step are not allowed;
- non-trivial subtrees of genes which all end in $\mathbb{L}$ events are not allowed (e.g. an $\mathbb{S}$ event followed by $\mathbb{L}$ events in both its children); and
- the gene family appears where the root of G is placed and not before (i.e. $|\alpha(r(G))| = 1$).

Disallowing $\mathbb{DL}$ events is necessary, because this event sequence can occur an arbitrary number of times in a reconciliation. If we were to allow it, the reconciliation space would be infinite. The same applies to a $\mathbb{T}$ followed by an $\mathbb{L}$ in the transferred branch, and consecutive $\mathbb{TL}$s in a time-step. The other two assumptions (and, indeed, the first three) are standard in the literature (for example, in [9]), even though disregarding them does not result in an infinite space. All of these requirements are also consistent with a parsimony approach: no reconciliation breaking any of these requirements can possibly be a most parsimonious reconciliation.

We lastly make one final assumption: in the species tree S, no two speciations occur at the exact same time. This is reasonable if you consider the species evolving as a continuous-time process; the probability of two events occurring at exactly the same time is zero. For our purposes, this assumption simplifies our counting theorems.

4.1 Reconciliation space

We construct a reconciliation in two stages: firstly we allocate the placement and types of the vertices of G and their immediate outgoing edges. These can, in some sense, be chosen freely, as we show in Lemma 4. We define an *undetermined* reconciliation α as a reconciliation where we have not assigned $\alpha(u)$ for any vertex $u \in V(G)$.

Lemma 4 *Let α be an undetermined reconciliation. For all $u \in V(G)$, we allocate the first and the last elements of $\alpha(u)$, i.e. $\alpha_1(u)$ and $\alpha_\ell(u)$ — but not the length of $\alpha(u)$ — such that condition (a) of Definition 15 is satisfied. In addition, we require that $h(\alpha_1(u)) \leq h(\alpha_\ell(u))$. Then there always exists at least one reconciliation α which satisfies these allocations.*

Proof. We need to show that there exists a valid path (sequence of vertices satisfying the conditions of $\varnothing$, $\mathbb{SL}$ or $\mathbb{TL}$ events) in S' between $\alpha_1(u)$ and $\alpha_\ell(u)$ for all u. Let $\alpha'_1(u)$ be the ancestor of $\alpha_\ell(u)$ which is at the same height as $\alpha_1(u)$ (a $\mathbb{TL}$ event). Then we complete $\alpha(u)$ by adding — between $\alpha_1(u)$ and $\alpha_\ell(u)$ — the ordered sequence of vertices composing the path in S' from $\alpha'_1(u)$ to $(\alpha_\ell(u))_p$. If $\alpha_1(u) = \alpha'_1(u)$, we do not add a second copy to $\alpha(u)$. All events on this path are either $\mathbb{SL}$ or $\varnothing$ events. It is easy to verify that the resulting sequence satisfies condition (b) of Definition 15. □

Next, we choose for each edge of G the path that it follows in S'. As we have already allocated the beginning and end of each edge, i.e. $\alpha_1(u)$ and $\alpha_\ell(u)$, we only have to deal with possible $\mathbb{TL}$ events, since $\mathbb{SL}$ and $\varnothing$ events will be determined according to whether or not there is a speciation in S'. Furthermore, these edges do not interact, so they can be allocated independently. The following lemma formally describes the possibilities for allocating these edges.

Lemma 5 *Let α be a partial reconciliation obtained as described in Lemma 4. Then α can be completed into a full reconciliation by independently completing the mapping of each edge as follows:*

1: $i \leftarrow 1$;
2: ***repeat***
3: ***if*** $h(\alpha_i(u)) = h(\alpha_\ell(u))$ ***then***
4: ***if*** $\alpha_i(u) = \alpha_\ell(u)$ ***then***
5: *we set $|\alpha(u)| = i$ and declare $\alpha(u)$ completed;*
6: ***else***
7: *we set $\alpha_{i+1}(u) = \alpha_\ell(u)$, $|\alpha(u)| = i + 1$ and declare $\alpha(u)$ completed;*
8: ***else***
9: ***if*** $i > 1$ *and* $h(\alpha_{i-1}(u)) = h(\alpha_i(u))$ ***then***
10: *we set $\alpha_{i+1}(u)$ to be a child of $\alpha_i(u)$;*
11: ***else***
12: *we set $\alpha_{i+1}(u)$ to be a child of $\alpha_i(u)$ or any vertex other than $\alpha_i(u)$ with the same height as $\alpha_i(u)$;*
13: $i \leftarrow i + 1$;
14: ***until*** *$\alpha(u)$ is completed.*

Furthermore, all possible reconciliations can be constructed in this manner.

Proof. As stated above, this is merely an enumeration of all possibilities. At every time-step, the path may contain at most one $\mathbb{TL}$ to any branch at the same height. If it already has a $\mathbb{TL}$ in the current time-step (line 9), it must descend to a child edge (line 10); otherwise $\alpha_{i+1}(u)$ can be a $\mathbb{TL}$, $\mathbb{SL}$ or $\varnothing$ event (line 12). Once it reaches the height of its final vertex (line 4), it must either have one final $\mathbb{TL}$ event (line 7) or simply finish, if it is already in the correct branch (line 5).

Given any reconciliation α, it is clear that α satisfies the conditions given in Lemma 4. The algorithm given above does not exclude any possible path between $\alpha_1(u)$ and $\alpha_\ell(u)$, so α can be constructed in this way. □

This complete characterisation of all possible reconciliations allows us to efficiently count their number using a bottom-up approach. We first define a notion which makes it easier to envision the image of an edge of G.

Definition 8 For a reconciliation α and a vertex $u \in V(G)$, we say $x \in V(S')$ is the *representative vertex* of $\alpha(u)$ at height h if $h(x) = h + 1$ and for some $i \in \{1, 2, \ldots, |\alpha(u)| - 1\}$ we have $\alpha_i(u) = x_p, \alpha_{i+1}(u) = x$.

The representative vertex determines the behaviour of $\alpha(u)$ at height h. The path corresponding to $e(u)$ must pass through the parent vertex and parent edge of the representative vertex. It is easy to see that $\alpha(u)$ must have exactly one representative vertex at every height (where it exists). We emphasise for the sake of clarity that the representative vertex of $\alpha(u)$ at height h does *not* itself have height h, but height $h + 1$; this is necessary because it also determines the first edge that $\alpha(u)$ occupies after height h.

Lemma 6 *If $h(\alpha_1(u)) = h_1$ and $h(\alpha_\ell(u)) = h_2 \geq h_1$, then there are*

$$\prod_{h=h_1}^{h_2-1} (h + 2) = \frac{(h_2 + 1)!}{(h_1 + 1)!}$$

possibilities for completing $\alpha(u)$.

Proof. As no two speciation events can occur simultaneously, there are $h+1$ different vertices with height h in S'. Now, at each height $h \in \{h_1, h_1+1, \ldots, h_2-1\}$, we choose the representative vertex of $\alpha(u)$ at height h. This representative vertex is itself at height $h+1$ and can be placed at any of the $h+2$ different vertices present at this height.

Once all representative vertices are chosen, at each height h there are two possibilities. The first is that $\alpha(u)$ already contains two different vertices of height h, in which case the first vertex must be a $\mathbb{TL}$ event and no further vertices of height h can be added. The second is that $\alpha(u)$ contains one vertex of height h, which is preceded immediately by a vertex of smaller height and followed by a vertex of greater height (since it is impossible for a single vertex to appear twice in $\alpha(u)$). Again, no further vertices of height h can be added. So once the representative vertices have been chosen, no further vertices can be added to $\alpha(u)$ and so $\alpha(u)$ is determined uniquely. The result follows immediately. □

Theorem 4 *Let $u \in V(G), x \in V(S')$, and let $f(u,x)$ denote the number of reconciliations between G_u and S' where $\alpha(u) = (x)$. If $u \in L(G)$, then $f(u,x) = 1$ if $x \in L(S)$ and $s(u) = s(x)$, and 0 otherwise. If $u \notin L(G)$, then*

$$f(u,x) = 2(1-\Lambda(x)) \sum_{\substack{x_1,x_2,\\ h(x_1),h(x_2)>h(x)}} \left(\frac{(h(x_1)+1)!(h(x_2)+1)!}{(h(x)+2)!^2} f(u_l,x_1) f(u_r,x_2) \right)$$
$$+(2h(x)+1) \sum_{\substack{x_1,x_2,\\ h(x_1),h(x_2)\geq h(x)}} \left(\frac{(h(x_1)+1)!(h(x_2)+1)!}{(h(x)+1)!^2} f(u_l,x_1) f(u_r,x_2) \right).$$

Proof. If $u \in L(G)$, the initial conditions are obvious. Now suppose $u \notin L(G)$. Since $\alpha_\ell(u) = x$, we know that u is either an $\mathbb{S}$ event at x or a $\mathbb{D}$ or $\mathbb{T}$ event on $e(x)$.

The first term corresponds to the case that u is an $\mathbb{S}$ event. Since it is a speciation, x cannot be an artificial vertex, which accounts for the factor of $1-\Lambda(x)$, and the children of x must be mapped to elements with strictly greater height. We let $\alpha_\ell(u_l) = x_1$ and $\alpha_\ell(u_r) = x_2$. Now we can either allocate $\alpha_1(u_l) = x_l$ and $\alpha_1(u_r) = x_r$ or vice versa, which accounts for the preceding factor of 2. The fractional term in the sum is the number of possibilities for completing $\alpha(u_l)$ and $\alpha(u_r)$, given by Lemma 6 — for example, if $\alpha_1(u_l) = x_l$ and $\alpha_\ell(u_l) = x_1$, we have $h_1 = h(x_l) = h(x)+1$ and $h_2 = h(x_1)$ in the Lemma. The remainder of the summand is the number of possibilities for allocating the subtrees G_{u_l} and G_{u_r} respectively to S'.

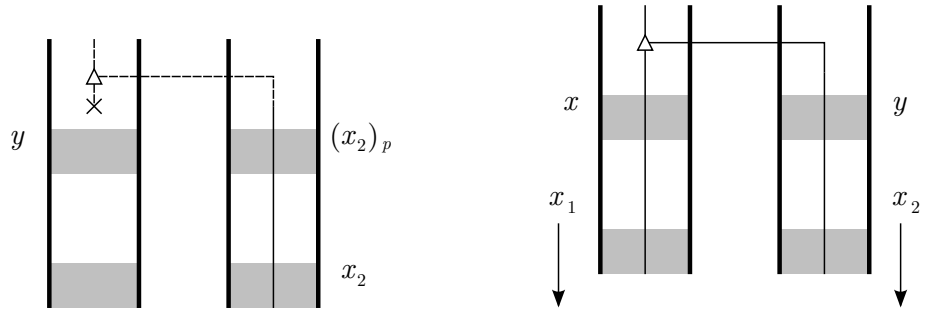
The second term corresponds to the case that u is a $\mathbb{D}$ or $\mathbb{T}$ event. In this case, its children must be mapped to elements with height at least $h(x)$. Again we let $\alpha_\ell(u_l) = x_1$ and $\alpha_\ell(u_r) = x_2$. Now if u is a $\mathbb{D}$ event, we must allocate $\alpha_1(u_l) = \alpha_1(u_r) = x$ (1 possibility). If u is a $\mathbb{T}$ event, we can either allocate $\alpha_1(u_l) = x$ and $\alpha_1(u_r)$ to be a different vertex of S' with $h(\alpha_1(u_r)) = h(x)$ ($h(x)$ possibilities) or vice versa (another $h(x)$ possibilities). This gives the prefactor of $2h(x)+1$. As before, the fraction in the sum counts the number of possibilities for completing $\alpha(u_l)$ and $\alpha(u_r)$, and the remainder of the summand counts the number of possibilities for allocating the child subtrees. □

The total number of possible reconciliations between G and S' is given by $\sum_{x \in V(S')} f(r(G), x)$, and we compute this value by calculating $f(u,x)$ for all x and u, again considering vertices of G in post-order.

A surprising result is easily derived from this theorem.

Corollary 1 *The number of reconciliations between G and S' is independent of the labels of $L(G)$.*

Proof. We show this by induction on the number of vertices in G. If $|V(G)| = 1$, there is always exactly one possible reconciliation (since the species in S are unique). Otherwise, any change of labels of $L(G)$ does not change $f((r(G))_l, x)$ and $f((r(G))_r, x)$ for any $x \in V(S')$ by induction, and by Theorem 4, $f(r(G), x)$ is also unchanged for any x. □



(a) The reconciliation will not be canonical if y and $(x_2)_p$ are artificial and there is a $\mathbb{TL}$ event at height $h(y)$

(b) The reconciliation will not be canonical if x and y are artificial and x_1 and x_2 are placed lower in S'

Fig. 3 Notations used in Lemma 7 and Theorem 5.

4.2 Counting canonical reconciliations

In this section we extend the results of the previous section to count canonical reconciliations. Firstly, we count the possible ways of completing paths so that no $\mathbb{TL}$ event can be moved downwards through two artificial vertices.

Lemma 7 *Let $q(x_1, x_2)$ be the number of ways of allocating $\alpha(u)$ from $\alpha_1(u) = x_1$ up to $\alpha_i(u) = x_2$ for some i, where x_2 is the representative vertex of $\alpha(u)$ at some height, so that no $\mathbb{TL}$ event can be moved downwards. Then*

$$q(x_1, x_2) = \begin{cases} 1 - \Lambda(x_1)\Lambda((x_2)_p)I(x_1 \neq (x_2)_p) & \text{if } h(x_2) = h(x_1) + 1, \\ \displaystyle\sum_{y, h(y)=h(x_2)-1} q(x_1, y)\left[1 - \Lambda(y)\Lambda((x_2)_p)I(y \neq (x_2)_p)\right] & \text{otherwise.} \end{cases} \tag{2}$$

Proof. Firstly, we must have $h(x_2) > h(x_1)$. In particular, we cannot have $h(x_2) = h(x_1)$ as $\alpha(u)$ cannot have a representative vertex at height $h(x_1) - 1$ by definition.

If $h(x_2) = h(x_1) + 1$, then the path up to x_2 is (x_1, x_2) if $x_1 = (x_2)_p$ and $(x_1, (x_2)_p, x_2)$ otherwise. The only way in which this partial path can contravene the definition of a canonical reconciliation is if $\alpha_1(u) = x_1$ is a $\mathbb{TL}$ event (i.e. $x_1 \neq (x_2)_p$) which can be moved downwards. This will happen if both x_1 and $(x_2)_p$ are artificial vertices, as the event will be moved through them. This gives the first case of (2).

If $h(x_2) > h(x_1) + 1$, we consider all possible choices of the representative vertex at height $h(x_2) - 2$, which we denote by y. This vertex can be chosen freely, unless it implies the existence of a $\mathbb{TL}$ event at height $h(y)$ (i.e. $y \neq (x_2)_p$) which can be moved downwards (i.e. y and $(x_2)_p$ are artificial). See Figure 3(a) for a depiction of this situation. This, combined with the fact that paths are completely determined by their representative vertices, gives the second case of (2). □

Lemma 8 *Let $p(x_1, x_2)$ be the number of possibilities for completing $\alpha(u)$, given $\alpha_1(u) = x_1$ and $\alpha_\ell(u) = x_2$, so that no $\mathbb{TL}$ event can be moved downwards. Then*

$$p(x_1, x_2) = \begin{cases} 1 & \text{if } h(x_1) = h(x_2), \\ \displaystyle\sum_{y, h(y)=h(x_2)} q(x_1, y) & \text{otherwise.} \end{cases}$$

Proof. If $h(x_1) = h(x_2)$, then the only possibility is that $\alpha(u) = (x_1)$ if $x_1 = x_2$ and $\alpha(u) = (x_1, x_2)$ otherwise (i.e. x_1 is a $\mathbb{TL}$ event). Otherwise, we let y be the representative vertex of $\alpha(u)$ at height $h(x_2) - 1$. This vertex can be chosen freely, as any $\mathbb{TL}$ event at height $h(y)$ can never

be moved downwards because the path ends at that height. Again combining with the fact that paths are completely (and uniquely) determined by their representative vertices gives the desired equation. □

In order to accurately count canonical reconciliations, we must also disregard reconciliations which have $\mathbb{D}$ or $\mathbb{T}$ events that can be moved downwards. In order to do this, we must be able to control whether the first event in a path is a $\mathbb{TL}$ event, which would prevent the event corresponding to the vertex at the start of the path from being moved downwards.

Lemma 9 *Let $p'(x_1, x_2)$ be the number of ways of allocating $\alpha(u)$ under the same conditions as in Lemma 8, with the added condition that $\alpha_1(u)$ must not be a $\mathbb{TL}$ event. Then the formulae of Lemmas 7 and 8 apply, with p and q replaced by p' and (say) q' respectively, except for the initial conditions, which are replaced by:*

$$\begin{aligned} q'(x_1, x_2) &= I(x_1 = (x_2)_p) && \text{if } h(x_2) = h(x_1) + 1 && \text{(for Lemma 7),} \\ p'(x_1, x_2) &= I(x_1 = x_2) && \text{if } h(x_1) = h(x_2) && \text{(for Lemma 8).} \end{aligned}$$

Proof. The only place in which the added condition can occur is at the very start of the path, which is covered by the initial conditions in the Lemmas. We require that $\alpha_1(u)$ not be a $\mathbb{TL}$ event, and so $x_1 = (x_2)_p$ in Lemma 7, and $x_1 = x_2$ in Lemma 8. □

We are finally ready to express a recurrence for the number of canonical reconciliations.

Theorem 5 *Let $f'(u, x)$ denote the number of canonical reconciliations between G_u and S' where $\alpha(u) = (x)$. If $u \in L(G)$, then $f'(u, x) = 1$ if $x \in L(S)$ and $s(u) = s(x)$, and 0 otherwise. If $u \notin L(G)$, then*

$$\begin{aligned} f'(u,x) = (1 - \Lambda(x)) & \sum_{\substack{x_1,x_2 \\ h(x_1),h(x_2)>h(x)}} [p(x_l,x_1)p(x_r,x_2) + p(x_r,x_1)p(x_l,x_2)]\, f'(u_l,x_1)f'(u_r,x_2) \\ + & \sum_{\substack{x_1,x_2 \\ h(x_1),h(x_2)\geq h(x)}} p(x,x_1)p(x,x_2)f'(u_l,x_1)f'(u_r,x_2) \\ - & \sum_{\substack{x_1,x_2 \\ h(x_1),h(x_2)>h(x)}} \Lambda(x)p'(x,x_1)p'(x,x_2)f'(u_l,x_1)f'(u_r,x_2) \\ + & \sum_{\substack{x_1,x_2,y \\ h(x_1),h(x_2)\geq h(x), \\ y\neq x, h(y)=h(x)}} [p(x,x_1)p(y,x_2) + p(y,x_1)p(x,x_2)]\, f'(u_l,x_1)f'(u_r,x_2) \\ - & \sum_{\substack{x_1,x_2,y \\ h(x_1),h(x_2)>h(x), \\ y\neq x, h(y)=h(x)}} \Lambda(x)\Lambda(y)\,[p'(x,x_1)p'(y,x_2) + p'(y,x_1)p'(x,x_2)]\, f'(u_l,x_1)f'(u_r,x_2). \end{aligned} \tag{3}$$

Proof. The case for $u \in L(G)$ is again obvious, so we assume $u \notin L(G)$. There are three cases:

- u is an $\mathbb{S}$ event at x. This is covered by the first line of (3). Firstly x must not be artificial. We let $x_1 = \alpha_\ell(u_l)$ and $x_2 = \alpha_\ell(u_r)$. These must have height strictly greater than $h(x)$. There are two possibilities, that either $\alpha_1(u_l) = x_l$ and $\alpha_1(u_r) = x_r$ or vice versa. Once these are assigned, the number of ways to complete the paths is given by the term in square brackets, and the number of ways to complete the subtrees is given by the remaining terms. Note that all $\mathbb{S}$ events are canonical.
- u is a $\mathbb{D}$ event at $e(x)$. This is covered by the second and third lines of (3). As in the first case, we let $x_1 = \alpha_\ell(u_l)$ and $x_2 = \alpha_\ell(u_r)$, and these must have height greater than or equal to $h(x)$. We now must have $\alpha_1(u_l) = \alpha_1(u_r) = x$. The second line counts the possibilities for allocating the paths and the subtrees.

However, there is a possibility that this event is able to be moved downwards through an artificial vertex, i.e. it is a non-canonical $\mathbb{D}$ event. This will happen if both x_1 and x_2 have height strictly greater than $h(x)$, x is artificial, and neither $\alpha(u_l)$ nor $\alpha(u_r)$ start with a $\mathbb{TL}$ event. In particular, if x is artificial then neither $\alpha(u_l)$ nor $\alpha(u_r)$ can start with an $\mathbb{SL}$ event. This gives rise to the third line of (3).

– u is a $\mathbb{T}$ event at $e(x)$. This is covered by the fourth and fifth lines of (3). We again let $x_1 = \alpha_\ell(u_l)$ and $x_2 = \alpha_\ell(u_r)$. We also let y be the target of the transfer, so $y \neq x$ but $h(y) = h(x)$. Now, there are again two cases, where $\alpha_1(u_l) = x$ and $\alpha_1(u_r) = y$ or vice versa. The possibilities for completing the paths and the subtrees are given in the fourth line.
Again there is a possibility that this event and its target may be able to be moved downwards through two artificial vertices. This will happen if both x_1 and x_2 have height strictly greater than $h(x)$, x and y are artificial, and neither $\alpha(u_l)$ and $\alpha(u_r)$ start with a $\mathbb{TL}$ event. (See Figure 3(b) for a depiction of this case.) This is expressed in the fifth line.

□

The number of possible canonical reconciliations between G and S' is $\sum_{x \in V(S')} f'(r(G), x)$. We compute this number recursively using the order previously described.

4.3 Operators

As before, we define operators which can transform any reconciliation into any other. A key observation here is that there are three parts of a reconciliation which can, in some sense, be considered separately from each other:

1. the locations of the (images of the) vertices of G in S' (determined by $\alpha_\ell(u)$ for $u \in V(G)$);
2. the event types ($\mathbb{S}$, $\mathbb{D}$ or $\mathbb{T}$) associated to the vertices of G (determined by $\alpha_1(u_l)$ and $\alpha_1(u_r)$ for internal vertices u of G); and
3. the images of the edges of G in S' (determined by the remaining values of α).

Informally, the choice of each of these parts does not constrain the choice of parts below it in the list:

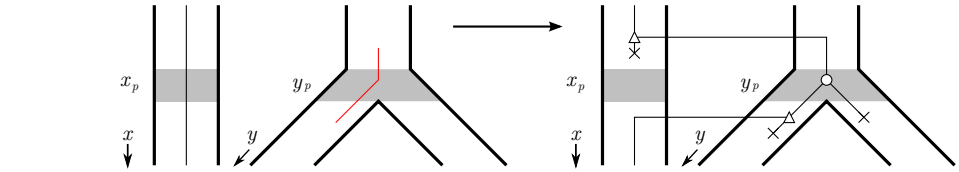
1. the locations of the vertices of G in S' can be chosen freely, as long as they are height-consistent;
2. given the locations, the event types can be chosen freely, as long as each vertex represents an $\mathbb{S}$, $\mathbb{T}$ or $\mathbb{D}$ event;
3. given the locations and event types, the images of the edges of G in S' can be chosen freely and independently from each other.

Therefore we define three classes of operators which each affect one of these parts of the reconciliation. The $UP/DOWN$ operators change the location of the vertices by moving them up and down in S', the OUT operators change event types by changing the immediate out-edges of the events, and the $PATH$ operators change the images of the edges by changing representative vertices. The $UP/DOWN$ operators are the analogs of the $uNMC$ and $dNMC$ operators for the no-$\mathbb{TL}$ model; the remaining operators are chosen so that all parts of the reconciliation can be controlled in a natural way. We note that certainly these are not the only operators which can be taken, but they are a natural extension of the no-$\mathbb{TL}$ model, and also retain the useful property that they are local operators only.

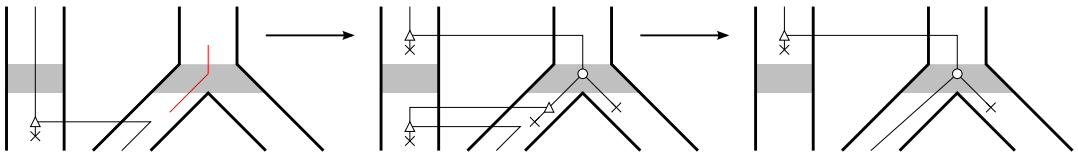
Before giving formal definitions of these operators, we define a reduction function which removes consecutive $\mathbb{TL}$ events in a time-step, should they appear.

Definition 9 Let γ be a sequence of vertices of S', sorted by height (from smallest to largest). We define $\rho(\gamma)$ as the sequence obtained by doing the following, for each height h for which γ contains (consecutive) vertices $x_1, \ldots, x_n$ with height h:

– if $x_1 = x_n$, replace $x_1, \ldots, x_n$ by x_1;
– if $n \geq 3$ and $x_1 \neq x_n$, replace $x_1, \ldots, x_n$ by x_1, x_n;



(a) $PATH_{u,y}$ sets y as a representative vertex of $\alpha(u)$ by the addition of $\mathbb{TL}$ events



(b) PATH operators can give rise to multiple $\mathbb{TL}$ events in a time-step, which must be reduced

Fig. 4 Examples of the $PATH$ operator. The path induced by the new representative vertex is marked in the diagrams on the left.

– otherwise, do nothing.

The $PATH$ operators (inspired by the BFACF algorithm in combinatorics [4,6]) modify the image of an edge of G without affecting the locations or event types of the vertices of G, or any other edge. They do this by changing the representative vertex of the path at a given height to any other vertex at the same height by adding $\mathbb{TL}$ events (which may then be reduced).

Definition 10 Let α be a reconciliation and $u \in V(G)$. If x is the representative vertex of $\alpha(u)$ at height h, then for any vertex $y \neq x$ in $V(S')$ with $h(y) = h(x)$, we define $PATH_{u,y}(\alpha)$ as the mapping where:

$$PATH_{u,y}(\alpha)(u) = \rho(\alpha_1(u), \ldots, x_p, y_p, y, x, \ldots, \alpha_\ell(u)),$$
$$PATH_{u,y}(\alpha)(v) = \alpha(v) \text{ for } v \neq u.$$

Observe that y is now the representative vertex of $PATH_{u,y}(\alpha)(u)$ at height h. See Figure 4 for an illustration of this operator.

In future definitions, we use the '$\cdot$' operator as a concatenation of sequences (in this case of vertices of S'). Single elements are equivalent to a sequence containing only that element.

The OUT operators modify the immediate out-edges of an internal vertex u of G, namely $\alpha_1(u_l)$ and $\alpha_1(u_r)$. Naturally this changes the paths $\alpha(u_l)$ and $\alpha(u_r)$, but it leaves the location of the vertices of G in S' unchanged. Figure 5 provides examples illustrating the following definition of these operators.

Definition 11 Let α be a reconciliation and u an internal vertex of G. Then:

– If u is mapped to an $\mathbb{S}$ event, we define $OUT_u(\alpha)$ as the mapping where:

$$OUT_u(\alpha)(u_l) = \rho(\alpha_1(u_r) \cdot \alpha(u_l)),$$
$$OUT_u(\alpha)(u_r) = \rho(\alpha_1(u_l) \cdot \alpha(u_r)),$$
$$OUT_u(\alpha)(v) = \alpha(v) \text{ for } v \neq u_l, u_r.$$

This switches the children of u, adding a $\mathbb{TL}$ at the beginning of both $\alpha(u_l)$ and $\alpha(u_r)$ back to their original starting branches. u remains an $\mathbb{S}$ event.

– If u is mapped to a $\mathbb{D}$ event, for any vertex $y \neq \alpha_1(u_l)$ with $h(y) = h(\alpha_1(u_l))$, we define $OUT^l_{u,y}(\alpha)$ as the mapping where:

$$OUT^l_{u,y}(\alpha)(u_l) = \rho(y \cdot \alpha(u_l)),$$
$$OUT^l_{u,y}(\alpha)(v) = \alpha(v) \text{ for } v \neq u_l.$$

This changes the left child of u to a transfer to the branch $e(y)$ (so u is changed from a $\mathbb{D}$ to a $\mathbb{T}$), and adds a $\mathbb{TL}$ event at the beginning of $\alpha(u_l)$ back to its original starting branch. The operator $OUT^r_{u,y}$ is defined similarly.

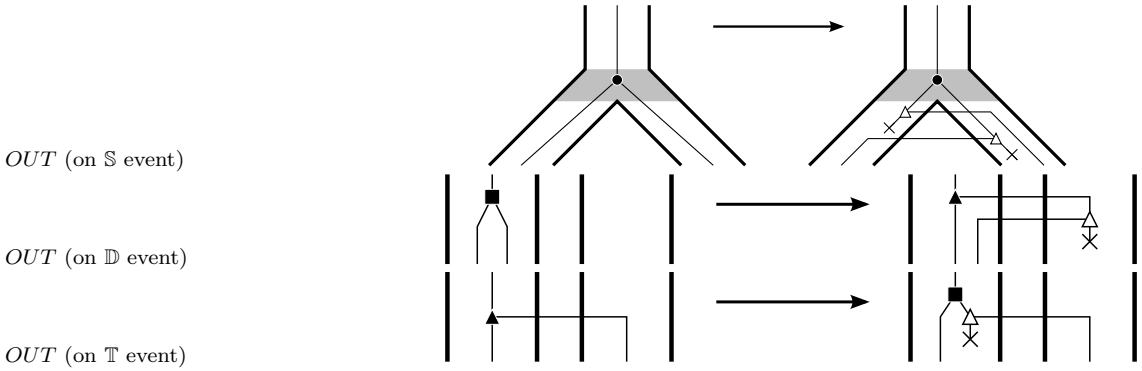


Fig. 5 Examples of the OUT operators.

– If u is mapped to a $\mathbb{T}$ event so that u_l (respectively u_r) is the transferred branch, we define $OUT_{u,y}$ identically to $OUT^l_{u,y}$ (resp. $OUT^r_{u,y}$) in the case where u is a $\mathbb{D}$ event. This changes the target of the transfer (possibly to the branch of u), so u either stays a $\mathbb{T}$ event or changes from a $\mathbb{T}$ to a $\mathbb{D}$.

To aid with the definition of the final class of operators, we make a preliminary definition which mirrors that of a reconciliation in the no-$\mathbb{TL}$ model.

Definition 12 Let α be a reconciliation. We define the *vertex mapping* $\bar{\alpha}$ of α to be a mapping $V(G) \to V(S) \cup E(S')$ where:

– if u is mapped to an $\mathbb{S}$ event, $\bar{\alpha}(u) = \alpha_\ell(u)$;
– if u is mapped to a $\mathbb{D}$ or $\mathbb{T}$ event, $\bar{\alpha}(u) = e(\alpha_\ell(u))$.

The vertex mapping of a reconciliation corresponds exactly to the format of a reconciliation for the no-$\mathbb{TL}$ model. We note that the vertex mapping determines the value of $\alpha_\ell(u)$ and the height of $\alpha_1(u)$ for all $u \in V(G)$. By Lemma 4, any mapping $\bar{\alpha}$ where $\bar{\alpha}(u)$ allows $\bar{\alpha}(u_l)$ and $\bar{\alpha}(u_r)$ is a valid vertex mapping (i.e. is the vertex mapping of a valid reconciliation).

The UP and $DOWN$ operators move images of the vertices of G to immediate ancestors and descendants in S'. In this way, they are the analogs of the $uNMC$ and $dNMC$ operators in the no-$\mathbb{TL}$ case. By necessity, they potentially also change the type of the vertex and the paths adjacent to it. Figure 6 provides examples illustrating the following definition of these operators.

Definition 13 Let α be a reconciliation and u an internal vertex of G. Then we define the UP and $DOWN$ operators as follows. In all cases, if the value of the changed reconciliation is not given for a vertex v, it is assumed to be equal to $\alpha(v)$.

– If u is mapped to an $\mathbb{S}$ event,

$$\begin{aligned} UP_u(\alpha)(u_l) &= \rho(\alpha_\ell(u) \cdot \alpha(u_l)), \qquad UP_u(\alpha)(u_r) = \rho(\alpha_\ell(u) \cdot \alpha(u_r)), \\ DOWN^l_u(\alpha)(u) &= \rho(\alpha(u) \cdot (\alpha_\ell(u))_l), \\ DOWN^r_u(\alpha)(u) &= \rho(\alpha(u) \cdot (\alpha_\ell(u))_r). \end{aligned}$$

The UP_u operator moves u into a $\mathbb{D}$ event in its parent edge and adds an $\mathbb{SL}$ event to the beginning of both $\alpha(u_l)$ and $\alpha(u_r)$. The $DOWN^l_u$ and $DOWN^r_u$ operators move u into a $\mathbb{T}$ event in its left and right child edges respectively and add an $\mathbb{SL}$ event to the end of $\alpha(u)$.
– If u is mapped to a $\mathbb{D}$ or $\mathbb{T}$ event, UP_u can be applied if $\bar{\alpha}(u_p)$ allows $(\alpha_\ell(u))_p$. Define $\alpha' = PATH_{u,\alpha_\ell(u)}(\alpha)$. Note that $\alpha_\ell(u)$ is a representative vertex of $\alpha'(u)$ and that $\alpha'_{\ell-1}(u) = (\alpha_\ell(u))_p$ is not a $\mathbb{TL}$ event.
  – If $(\alpha_\ell(u))_p$ is an artificial vertex,

$$\begin{aligned} UP_u(\alpha)(u) &= \rho(\alpha'_1(u), \ldots, \alpha'_{\ell-1}(u)), \\ UP_u(\alpha)(u_l) &= \rho((\alpha'_1(u_l))_p \cdot \alpha'(u_l)), \qquad UP_u(\alpha)(u_r) = \rho((\alpha'_1(u_r))_p \cdot \alpha'(u_r)). \end{aligned}$$

This moves u upwards through an artificial vertex (adding $\varnothing$ or $\mathbb{SL}$ events at the beginning of both $\alpha(u_l)$ and $\alpha(u_r)$), but does not change its event type. If $u = r(G)$, then take $UP_u(\alpha)(u) = ((\alpha(u))_p)$ instead.

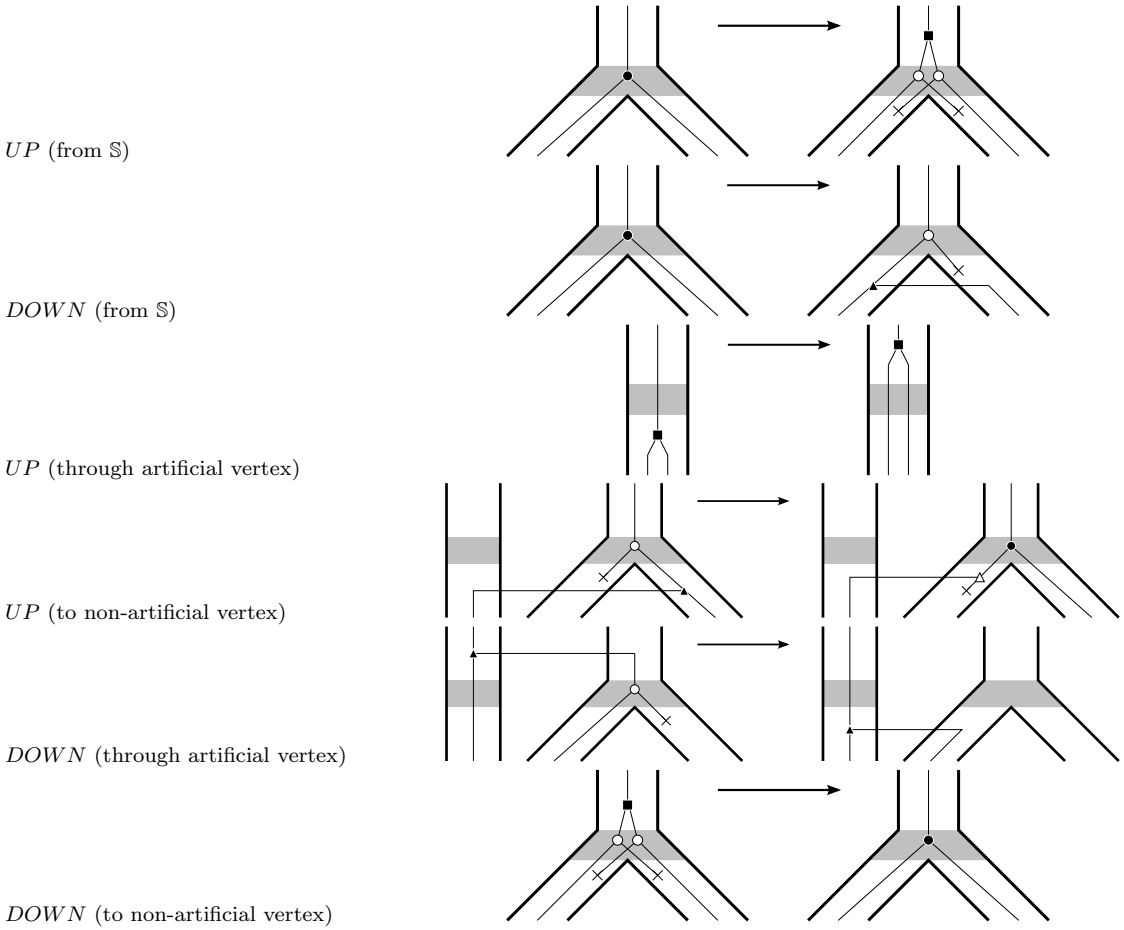


Fig. 6 Examples of the UP and $DOWN$ operators. The operators move the filled vertex.

- If $(\alpha_\ell(u))_p$ is not an artificial vertex,

$$UP_u(\alpha)(u) = \rho(\alpha'_1(u), \ldots, \alpha'_{\ell-1}(u)),$$
$$UP_u(\alpha)(u_l) = \rho((\alpha'_{\ell-1}(u))_l \cdot \alpha'(u_l)), \qquad UP_u(\alpha)(u_r) = \rho((\alpha'_{\ell-1}(u))_r \cdot \alpha'(u_r)).$$

  This moves u upwards into an $\mathbb{S}$ event at its parent vertex, potentially adding $\mathbb{TL}$ events at the start of each of its child edges to transfer them back to their original starting branches. If $u = r(G)$, then take $UP_u(\alpha)(u) = ((\alpha(u))_p)$ instead.

- If u is mapped to a $\mathbb{D}$ or $\mathbb{T}$ event, $DOWN_u$ can be applied if $\alpha_\ell(u)$ allows both $\bar{\alpha}(u_l)$ and $\bar{\alpha}(u_r)$. Define $\alpha' = PATH_{u_l,(\alpha_1(u_l))_l} PATH_{u_r,(\alpha_1(u_r))_l}(\alpha)$, so that neither $\alpha'_1(u_l)$ or $\alpha'_1(u_r)$ is a $\mathbb{TL}$ event.
  - If $\alpha_\ell(u)$ is an artificial vertex,

$$DOWN_u(\alpha)(u) = \rho(\alpha'(u) \cdot (\alpha'_\ell(u))_l),$$
$$DOWN_u(\alpha)(u_l) = \rho(\alpha'_2(u_l), \ldots, \alpha'_\ell(u_l)), \qquad DOWN_u(\alpha)(u_r) = \rho(\alpha'_2(u_r), \ldots, \alpha'_\ell(u_r)).$$

    This moves u downwards through an artificial vertex (adding a $\varnothing$ event at the end of $\alpha(u)$), but does not change its event type.

– If $\alpha_\ell(u)$ is not an artificial vertex,

$$DOWN_u(\alpha)(u_l) = \rho((\alpha'_\ell(u))_l, \alpha'_2(u_l), \ldots, \alpha'_\ell(u_l)),$$
$$DOWN_u(\alpha)(u_r) = \rho((\alpha'_\ell(u))_r, \alpha'_2(u_r), \ldots, \alpha'_\ell(u_r)).$$

This moves u downwards into an $\mathbb{S}$ event at its child vertex (which is non-artificial by assumption). Its child edges will potentially start with $\mathbb{TL}$ events to transfer them to their original *second* branches.

The preliminary $PATH$ operators appearing within the UP and $DOWN$ operators in the case that u is a $\mathbb{D}$ or a $\mathbb{T}$ event ensure that these latter operators can always be applied disregarding $\mathbb{TL}$ events, by forcing there to be no $\mathbb{TL}$ events in between the event and its nearest vertex of S' in the appropriate direction (above or below).

We note that the $PATH$ and OUT operators do not change the vertex mapping of their operand, and UP_u and $DOWN_u$ only change the vertex mapping at u, to an ancestor or a descendant. In particular, the UP_u operator moves the vertex mapping of u to the lowest possible element of S' (i.e. an element which is not an artificial vertex) above it, and the $DOWN_u$ moves the vertex mapping of u to the highest possible element of S' below it.

Lemma 10 *The UP, $DOWN$, OUT and $PATH$ operators produce valid reconciliations.*

Proof. This is a straightforward but repetitive matter of comparing the definitions of the operators to the definition of a valid reconciliation, given in Definition 15. In each case we must ensure that every element of every path satisfies the definition of one of the events.

As an example, we show that if u is mapped to an $\mathbb{S}$ event, then $\alpha' = UP_u(\alpha)$ is a valid reconciliation. Since only $\alpha'_1(u_l)$ and $\alpha'_1(u_r)$ have changed from α, there are only three elements that we need to check:

– $\alpha'_\ell(u)$: since $\alpha'_1(u_l) = \alpha'_1(u_r) = \alpha'_\ell(u)$, $\alpha'_\ell(u)$ satisfies the conditions of a $\mathbb{D}$ event.
– $\alpha'_1(u_l)$: Since $\alpha_\ell(u)$ is an $\mathbb{S}$ event, $\alpha_1(u_l)$ is a child of $\alpha_\ell(u)$, and so $\alpha'_1(u_l)$ satisfies the conditions of an $\mathbb{SL}$ event (we know that it is a non-artificial vertex).
– $\alpha'_1(u_r)$: likewise, $\alpha_1(u_r)$ is a child of $\alpha_\ell(u)$ and so $\alpha'_1(u_r)$ satisfies the conditions of an $\mathbb{SL}$ event.

Since all the elements of α' which are changed from α are still valid events, it is a valid reconciliation. □

Theorem 6 *Let α and β be two reconciliations. By repeated applications of $PATH$, OUT, UP and $DOWN$ operators, α can be transformed into β.*

Proof. Firstly, we use UP and $DOWN$ operators to transform α into a reconciliation α' where $\bar{\alpha'} = \bar{\beta}$. Starting from α, we apply UP_u operators to each internal vertex u of G in pre-order until we reach a reconciliation γ where $\bar{\gamma}(u) = e(r(S'))$. This is always possible because when we move u, by construction the parent of u is already mapped to $e(r(S'))$, which allows itself. By a similar reasoning, we can apply a sequence of UP operators to β to reach γ', where $\bar{\gamma'} = \bar{\gamma}$. But the effect of any UP_u operator on the vertex mapping can be reversed by some $DOWN_u$ operator, so we take this second sequence of UP operators and apply the corresponding $DOWN$ operators to γ. This gives us a reconciliation α' where $\bar{\alpha'} = \bar{\beta}$.

Next, we use OUT operators to transform α' into α'', where $\alpha''_1(u) = \beta_1(u)$ and $\alpha''_\ell(u) = \beta_\ell(u)$ for all $u \in V(G)$. Since $\bar{\alpha'} = \bar{\beta}$, the latter condition is already satisfied. There are two possibilities at each internal vertex:

– u is mapped to $\mathbb{S}$ events by both α' and β. Then either $\alpha'_1(u_l) = \beta_1(u_l)$ and $\alpha'_1(u_r) = \beta_1(u_r)$, in which case nothing needs to be done, or $\alpha'_1(u_l) = \beta_1(u_r)$ and $\alpha'_1(u_r) = \beta_1(u_l)$, in which case we apply a single OUT_u operator.
– u is mapped to $\mathbb{D}$ or $\mathbb{T}$ events by α' and β. We apply an $OUT_{u,\alpha'_\ell(u)}$ operator to transform $\alpha'_\ell(u)$ into a $\mathbb{D}$ event if it is not already one, and then an $OUT_{u,y}$ operator to transform it into an event with the same out-edges as $\beta(u)$ if this is not already the case. For example, if $\alpha'_\ell(u)$ is a $\mathbb{T}$ event transferring its left branch, and $\beta(u)$ is a $\mathbb{T}$ event transferring its right branch, we apply the operators $OUT_{u,\alpha'_\ell(u)}$ followed by $OUT^r_{u,\beta_1(u_r)}$.

Call the resulting reconciliation α''. By construction, $\alpha''_1(u_l) = \beta_1(u_l)$ and $\alpha''_1(u_r) = \beta_1(u_r)$ for each internal vertex u, so α'' has the required property.

Finally, we use $PATH$ operators to transform $\alpha''(u)$ into $\beta(u)$ for each vertex $u \in V(G)$. By construction, $\alpha''_1(u) = \beta_1(u)$ and $\alpha''_\ell(u) = \beta_\ell(u)$. Now, at each height $h = h(\alpha''_1(u)), \ldots, h(\alpha''_\ell(u)) - 1$, we apply the $PATH_{u,x}$ operator, where x is the representative vertex of $\beta(u)$ at height h. From the definition of this operator, the resulting path has the same representative vertex at each height as $\beta(u)$, and by the reasoning in the proof of Lemma 6, a path is uniquely defined by its representative vertices. Therefore the resulting path is $\beta(u)$. When we have transformed all the paths in this way, the resulting reconciliation is β and so the theorem is proved. □

The reasoning in the proof gives the following corollary.

Corollary 2 *Let α and β be two reconciliations with $\bar{\alpha} = \bar{\beta}$. Then α can be transformed into β using only $PATH$ and OUT operators.*

It is not as straightforward as the no-$\mathbb{TL}$ model to see that the set of operators we use here is minimal. Certainly, the UP and $DOWN$ operators are necessary in order transform any reconciliation into any other. But it is possible to construct scenarios where the effect of a particular OUT operator can be replicated by a series of UP, $DOWN$ and $PATH$ operators, and likewise certain $PATH$ operators can be replaced by a series of UP, $DOWN$ and OUT operators. However, it is equally possible to construct simple reconciliations (see Figure 7) which cannot be reached from others without OUT operators, and likewise with $PATH$ operators. As such, we do require all of these operators to reach every possible reconciliation.

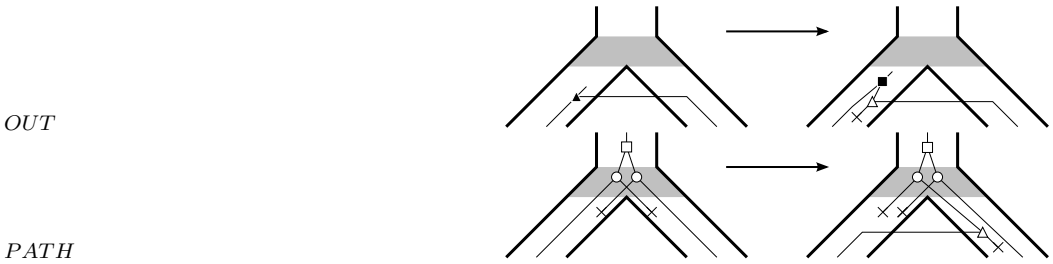


Fig. 7 Examples of OUT and $PATH$ operators which cannot be replaced by (a series of) operators of different type. Note that these are complete reconciliations, not local views.

As before, the operators we define induce a distance measure naturally between two reconciliations, defined as the smallest number of operators needed to transform one reconciliation into the other:

Definition 14 Let α and β be two reconciliations. We define $d(\alpha, \beta)$ to be the smallest number of UP, $DOWN$, $PATH$ and OUT operators needed to convert α to β.

Unfortunately, how to determine this distance is still an open problem. In Appendix C we show how to determine the minimum number of UP and $DOWN$ operators required to transform one reconciliation into another. These operators correspond directly to the $uNMC$ and $dNMC$ operators in the no-$\mathbb{TL}$ model, and indeed our result is proved in a similar fashion. However, this number may be strictly smaller than the number of UP and $DOWN$ operators used in the minimal-length sequence of all operators.

Because the UP and $DOWN$ operators change the type of a vertex when moving through a non-artificial vertex, and the new type is fixed, it is possible to have an operator whose effect cannot be reversed by a single operator (see Figure 8). Therefore the distance defined in Definition 14 is not a metric. Note however that the distance measure induced by UP and $DOWN$ operators treated in Appendix C is indeed a metric.

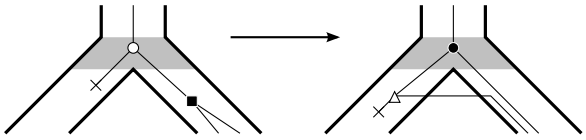


Fig. 8 An example of an UP operator which cannot be reversed by a single operator (the corresponding $DOWN$ operator changes the vertex into a $\mathbb{T}$).

5 Conclusion

In this paper, we have presented an algorithm to count the number of all possible reconciliations between a gene tree and a dated species tree. Moreover, we have defined a set of operators which allow us to transform any $\mathbb{DTL}$ reconciliation between a given pair of trees into any other reconciliation of the same pair. This operator set enables us to explore the space of all possible reconciliations between these two trees, and to define a distance measure between two such reconciliations.

Being able to characterise the space of all possible reconciliations is critical in the study and inference of reconciliations. Up to now, theoretical work related to reconciliation inference has often been undermined by an absence of formal definition of the valid reconciliation space, apart from ad-hoc definitions of this space as the set of reconciliations explored by a given reconciliation algorithm. For example, in a recent paper [13], some of the authors proved the importance of filtering reconciliations in order to obtain a reliable subset of evolutionary events. The best strategy they found to compute the event supports used to filter reconciliations relied on observed event frequencies in Near-optimal Parsimonious Reconciliations (NPRs). Since no definition of the space of the reconciliations close to optimality exists currently, a sample of NPRs was obtained by computing MPRs for different cost vectors. This approach is costly in terms of time, and the choice of altered costs also impacts the final result. The theoretical framework provided here allows us to formally define the set of near-optimal reconciliations as the subset of the reconciliation space having a cost below a given threshold. Since we are now able to define the set of near-optimal reconciliations in a formal way, we will now be able to design algorithms to compute them efficiently [19].

Being able to explore the space of all possible reconciliations is also crucial. For example, the operator set that we have defined in this paper can be used to explore the neighborhood of MPRs to obtain a representative set of NPRs efficiently. More generally, our operators provide the missing tool to design Bayesian reconciliation methods based on Monte Carlo sampling. Such methods have already proved their efficacy in phylogenomics where, for instance, they can handle complex models for which a maximum likelihood approach becomes intractable [12]. This paves the way toward reconciliation methods based on more complex gene evolution models, e.g. models with a loss penalty that depends on the number of remaining copies of the gene — as biologically expected.

This work is also an important step towards comparison of reconciliations. Being able to compute a distance between two reconciliations would, among other things, allow us to study the diversity of equally parsimonious reconciliations on the same species tree-gene tree pair, in order to cluster and eventually summarise them through a consensus reconciliation. It would also help to better assess the accuracy of reconciliation software through simulation, by allowing a fine-grained comparison of simulated reconciliations against inferred ones. Whether or not the computation of the proposed distance measure is a NP-hard problem when $\mathbb{TL}$ events are allowed (see Definition 14) is still an open problem that we intend to explore in a follow-up paper.

Acknowledgements

This work was partially funded by the French *Agence Nationale de la Recherche Investissements d'avenir / Bioinformatique* (ANR-10-BINF-01-02, *Ancestrome*).

This publication is contribution no. 2014-XXX of the Institut des Sciences de l'Evolution de Montpellier (ISEM, UMR 5554).

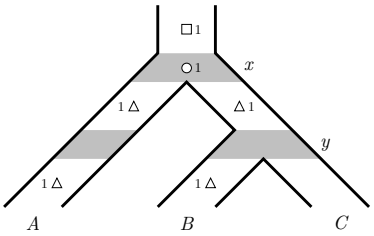
References

1. Abby, S., Tannier, E., Gouy, M., Daubin, V.: Detecting lateral gene transfers by statistical reconciliation of phylogenetic forests. BMC Bioinformatics **11**(1), 324 (2010)
2. Abby, S.S., Tannier, E., Gouy, M., Daubin, V.: Lateral gene transfer as a support for the tree of life. Proceedings of the National Academy of Sciences **109**(13), 4962–4967 (2012)
3. Åkerborg, Ö., Sennblad, B., Arvestad, L., Lagergren, J.: Simultaneous Bayesian gene tree reconstruction and reconciliation analysis. Proceedings of the National Academy of Sciences of the United States of America **106**(14), 5714–5719 (2009)
4. Berg, B., Foerster, D.: Random paths and random surfaces on a digital computer. Physics Letters B **106**(4), 323–326 (1981)
5. Boussau, B., Szöllősi, G.J., Duret, L., Gouy, M., Tannier, E., Daubin, V.: Genome-scale coestimation of species and gene trees. Genome Research **23**(2), 323–330 (2013)
6. Aragao de Carvalho, C., Caracciolo, S., Fröhlich, J.: Polymers and $g|\varphi|^4$ theory in four dimensions. Nuclear Physics B **215**(2), 209–248 (1983)
7. Doyon, J., Berry, V., Scornavacca, C., Ranwez, V.: DTLS-trees: proving the parsimony model for reconciling gene and species trees (2013). In preparation
8. Doyon, J., Chauve, C., Hamel, S.: Space of gene/species trees reconciliations and parsimonious models. Journal of Computational Biology **16**(10), 1399–1418 (2009)
9. Doyon, J.P., Scornavacca, C., Gorbunov, K.Y., Szöllősi, G.J., Ranwez, V., Berry, V.: An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In: Proceedings of the 2010 international conference on Comparative genomics, RECOMB-CG'10, pp. 93–108. Springer-Verlag, Berlin, Heidelberg (2011). URL http://dl.acm.org/citation.cfm?id=1927857.1927866
10. Goodman, M., Czelusniak, J., Moore, G.W., Herrera, R.A., Matsuda, G.: Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. Syst. Zool. **28**, 132–163 (1979)
11. van der Heijden, R., Snel, B., van Noort, V., Huynen, M.: Orthology prediction at scalable resolution by phylogenetic tree analysis. BMC Bioinformatics **8**(1), 83 (2007)
12. Lartillot, N., Philippe, H.: A bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. Molecular biology and evolution **21**(6), 1095–1109 (2004)
13. Nguyen, T.H., Ranwez, V., Berry, V., Scornavacca, C.: Support measures to estimate the reliability of evolutionary events predicted by reconciliation methods (2013). Accepted at PLOS ONE
14. Page, R.D.: Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. Syst. Biol. **43**, 58–77 (1994)
15. Rasmussen, M.D., Kellis, M.: A Bayesian Approach for Fast and Accurate Gene Tree Reconstruction. Molecular Biology and Evolution **28**(1), 273–290 (2010). URL papers://c574a988-041f-48f0-a387-1407660e65a3/Paper/p3804 papers2://publication/uuid/F8EE29D6-AD8B-4D48-B909-D42B119141B0 papers2://publication/uuid/8708F7F2-A362-40D9-91D8-D5E6D829AAD0 http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?dbfrom=pubmed&id=20660489&retmode=ref&cmd=prlinks papers2://publication/doi/10.1093/molbev/msq189
16. Scornavacca, C., Paprotny, W., Berry, V., Ranwez, V.: Representing a set of reconciliations in a compact way. Journal of Bioinformatics and Computational Biology **11** (2013)
17. Storm, C.E.V., Sonnhammer, E.L.L.: Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. Bioinformatics **18**(1), 92–99 (2002)
18. Szöllősi, G.J., Rosikiewicz, W., Boussau, B., Tannier, E., Daubin, V.: Efficient exploration of the space of reconciled gene trees. Systematic Biology **62**(6), 901–912 (2013)
19. To, T.H., Ranwez, V., Jacox, E., Scornavacca, C.: Better event supports using all suboptimal reconciliations (In prep)
20. Tofigh, A., Hallett, M., Lagergren, J.: Simultaneous identification of duplications and lateral gene transfers. IEEE/ACM TCBB **8**(2), 517–535 (2011). DOI http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.14

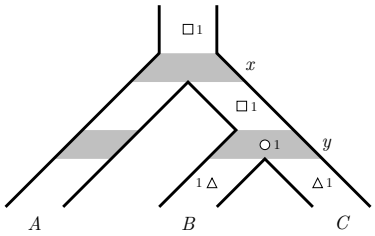
A — A reconciliation counting example

We illustrate Theorem 1 by counting the number of reconciliations between the trees shown in Figure 1. It is trivial to calculate $R(l, z)$ for $l \in L(G)$, so we do not illustrate this. In Figure 9(a), we calculate $R(v, z)$ for all elements z of S'; when this number is non-zero, it is written in the tree at z next to a symbol indicating the type of event that v is mapped to — in accordance with convention we use circles, squares and triangles for $\mathbb{S}$, $\mathbb{D}$ and $\mathbb{T}$ events respectively. For example, $R(v, x) = 1$, and if v is mapped to x it must be an $\mathbb{S}$ event. Because v_l and v_r are both leaves, $R(v, z)$ is at most 1 for any z. In Figure 9(b), we do likewise with $R(w, z)$ for all elements z of S'.

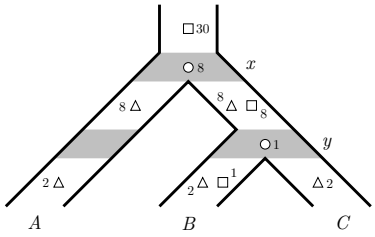
In Figure 9(c), we calculate $R(u, z)$ for all elements z of S' in the same fashion. Here, if u can be mapped as two different events in the same branch, we separate the cases (for clarity) even though they are added together in



(a) All possible placements for vertex v. Each placement has 1 possibility



(b) All possible placements for vertex w. Each placement has 1 possibility



(c) All possible placements for vertex u. The number of possibilities is indicated

Fig. 9 Counting the number of reconciliations between the trees in Figure 1.

$R(u, z)$. For example, suppose $z = e(y)$. Here, if u is mapped to z as a $\mathbb{T}$ event, then the branch containing v must be the transferred branch. From Figure 9(a), there are 2 possible images for v that are allowed by z but not below it, and for each of these images (called, say, z') $R(v, z') = 1$. Likewise, from Figure 9(b) there are 4 possible images for w that are below z, and each of these images again have $R(w, z') = 1$. Thus there are 8 possibilities for u to be mapped to z as a $\mathbb{T}$ event. It can be determined likewise that there are also 8 possibilities for u to be mapped to z as a $\mathbb{D}$ event, and so $R(u, z) = 16$.

The total number of reconciliations is found by summing all the possibilities in all branches of Figure 9(c), which gives a total of 70 reconciliations.

B — Reconciliation definitions

In this section, we reproduce (for completeness) the formal definitions of a $\mathbb{DTL}$ reconciliation and a canonical reconciliation in [16].

Definition 15 (Definition 1, [16]) Consider a gene tree G, a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, and its subdivision S'. Let α be a function that maps each node u of G onto an ordered sequence of nodes of S', denoted $\alpha(u) = (\alpha_1(u), \alpha_2(u), \ldots, \alpha_\ell(u))$. The function α is said to be a *reconciliation* between G and S' if and only if exactly one of the following events occurs for each couple of nodes u of G and $\alpha_i(u)$ of S' (denoting $\alpha_i(u)$ by x' below):

a) if x' is the last node of $\alpha(u)$, one of the cases below is true:
   1. $u \in L(G)$, $x' \in L(S')$ and $s(x') = s(u)$; ($\mathbb{C}$ event)
   2. $\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x'_l, x'_r\}$; ($\mathbb{S}$ event)
   3. $\alpha_1(u_l) = x'$ and $\alpha_1(u_r) = x'$; ($\mathbb{D}$ event)
   4. $\alpha_1(u_l) = x'$, and $\alpha_1(u_r)$ is any node other than x' having height $h(x')$
      or $\alpha_1(u_r) = x'$, and $\alpha_1(u_l)$ is any node other than x' having height $h(x')$; ($\mathbb{T}$ event)

b) otherwise, one of the cases below is true:
   5. x' is an artificial node and $\alpha_{i+1}(u)$ is its only child; ($\varnothing$ event)
   6. x' is not artificial and $\alpha_{i+1}(u) \in \{x'_l, x'_r\}$; ($\mathbb{SL}$ event)
   7. $\alpha_{i+1}(u)$ is any node other than x' having height $h(x')$. ($\mathbb{TL}$ event)

Definition 16 (Definition 2, [16]) Consider a gene tree G, a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, and its subdivision S'. A reconciliation α between G and S' is said to be *canonical* if and only if:

1. for each node u of G and index $1 \leq i \leq |\alpha(u)|$, the node $\alpha_i(u)$ satisfies one of the following conditions:
   (a) $\alpha_i(u)$ is a $\mathbb{C}/\mathbb{S}/\varnothing/\mathbb{SL}$ event;
   (b) $\alpha_i(u)$ is a $\mathbb{D}/\mathbb{T}$ event such that at least one of $\alpha_1(u)$ and $\alpha_1(u_r)$ is not a $\varnothing$ event;
   (c) $\alpha_i(u)$ is a $\mathbb{TL}$ event such that $\alpha_i(u)$ is a non-artificial node of S' or $\alpha_{i+1}(u)$ is not a $\varnothing$ event.
2. $\alpha_1(r(G))$ is not a $\varnothing$ event.

C — Distance measure induced by UP and $DOWN$ operators

Definition 17 Let α and β be two reconciliations. We define $d^v(\alpha, \beta)$ to be the smallest number of UP and $DOWN$ operators needed to transform α into β (combined with any number of $PATH$ and OUT operators).

It is possible that $d^v(\alpha, \beta)$ may in fact be less than the number of UP and $DOWN$ operators in the overall shortest sequence of operators. We note that it is certainly a lower bound for this number.

Definition 18 Let α and β be two reconciliations. The *midpoint mapping* $\hat{\gamma}$ of α and β is a mapping $V(G) \to V(S) \cup E(S')$ where, for all $u \in V(G)$, $\hat{\gamma}(u)$ is above both $\bar{\alpha}(u)$ and $\bar{\beta}(u)$, and if u is an internal vertex, $\hat{\gamma}(u)$ allows $\hat{\gamma}(u_l)$ and $\hat{\gamma}(u_r)$. Furthermore, there exists no reconciliation γ' with a vertex mapping $\bar{\gamma}'$ which satisfies these properties and where $\bar{\gamma}'(u) \leq \hat{\gamma}(u)$ for all $u \in V(G)$ and the inequality is strict in at least one case.

By definition, the midpoint mapping is a valid vertex mapping, so there exists at least one reconciliation with vertex mapping equal to $\hat{\gamma}$.

Definition 19 Let α and β be two reconciliations. The *midpoint* γ of α and β is a reconciliation with vertex mapping $\bar{\gamma} = \hat{\gamma}$. This specifies $\gamma_\ell(u)$ for all $u \in V(G)$. For each internal vertex u of G, we then choose:

- if $\hat{\gamma}(u) \in V(S)$, then $\gamma_1(u_l) = (\gamma_\ell(u))_l$ and $\gamma_1(u_r) = (\gamma_\ell(u))_r$;
- if $\hat{\gamma}(u) \in E(S')$, then $\gamma_1(u_l) = \gamma_1(u_r) = \gamma_\ell(u)$ (a $\mathbb{D}$ event),

and take $\gamma_1(r(G)) = \gamma_\ell(r(G))$. We take the remaining values of $\gamma(u)$ as specified in the proof of Lemma 4.

Theorem 7 *Let α and β be two reconciliations with midpoint γ. Let $d^v_u(\alpha, \gamma)$ be the number of edges and non-artificial vertices that lie between $\bar{\alpha}(u)$ and $\bar{\gamma}(u)$. Then*

$$d^v(\alpha, \beta) = d^v(\alpha, \gamma) + d^v(\gamma, \beta) = \sum_{u \in V(G)} d^v_u(\alpha, \gamma) + \sum_{u \in V(G)} d^v_u(\beta, \gamma).$$

Proof. The reasoning used in the first three paragraphs of the proof of Theorem 3 can be used again here, replacing "reconciliation" by "vertex mapping", to show that for any vertex u, in any sequence of reconciliations transforming α to β, at least one must have a vertex mapping which maps u to $\bar{\gamma}(u)$.

Because an UP_u operator will only move the vertex mapping of u to its nearest non-artificial ancestor, it is clear that we require at least $d^v_u(\alpha, \gamma)$ such operators to reach a reconciliation with a vertex mapping which maps u to $\bar{\gamma}(u)$. Likewise, we require at least $d^v_u(\beta, \gamma)$ $DOWN_u$ operators to go from this reconciliation to one with a vertex mapping which maps u to $\bar{\beta}(u)$. Therefore we have

$$d^v(\alpha, \beta) \geq \sum_{u \in V(G)} d^v_u(\alpha, \gamma) + \sum_{u \in V(G)} d^v_u(\beta, \gamma).$$

It remains to show that there is a valid sequence of operators with this many UP and $DOWN$ operators which transforms α to β. Consider the sequence which starts from α and applies UP operators to each internal vertex u of G in pre-order until it is mapped (via the vertex mapping) to $\bar{\gamma}(u)$. This is always possible because $\bar{\gamma}$ is a valid vertex mapping and so $\bar{\gamma}(u)$ allows $\bar{\gamma}(u_l)$ and $\bar{\gamma}(u_r)$. By Corollary 2, we can apply only $PATH$ and OUT operators to transform the resulting reconciliation into γ. This requires exactly $\sum_{u \in V(G)} d^v_u(\alpha, \gamma)$ UP operators.

Now we can apply a similar procedure to γ, applying $DOWN$ operators in post-order (choosing $DOWN^l$ and $DOWN^r$ appropriately) until each vertex u is mapped (via the vertex mapping) to $\bar{\beta}(u)$. As before, this is always possible, and we then apply $PATH$ and OUT operators to transform the resulting reconciliation into β. This requires exactly $\sum_{u \in V(G)} d^v_u(\beta, \gamma)$ $DOWN$ operators, and so the sequence we describe is the required sequence. □