Inapplicable algorithm description

1 First downpass

1. Enter on any cherry (i.e. pair of tips) on the tree and move to its most recent common ancestor; **then**, *go* to 2.

2. **If** there is any state in common between both descendants, *go* to 3; **else** *go* to 4.

3. **If** the state in common is only the inapplicable state, and both descendants have an applicable state, *set* the node's state to be the union of the descendants' states. **Else**, *set* the node's state to be the state in common between both descendants **then** *go* to 5.

4. **If** both descendants have an applicable state, *set* the node's state to be the union of both descendants states without the inapplicable state. **Else**, *set* the node's state to be the union of its descendants states. **Then** *go* to 5.

5. **If** possible, move to the node's ancestor and *go* to 2; **else** move to the next unvisited cherry's ancestor and *go* to 2. Once all nodes have been visited, end the first downpass.

2 First uppass

1. Enter the tree on its root. **If** the root has any applicable state, remove the eventual inapplicable state. **Then** move to one of the root's descendants and *go* to 2.

2. **If** the node has the inapplicable state, *go* to 3; **else**, leave the node's state unchanged and *go* to 7.

3. **If** the node also has an applicable state, *go* to 4; **else**, *go* to 5.

4. **If** the node's ancestor has the inapplicable state, *set* the node's state to be the inapplicable state only and *go* to 7; **else** remove the inapplicable state from the current node states. **Then** *go* to 7.

5. **If** the node's ancestor has the inapplicable state, *set* the node's state to be the inapplicable state only and *go* to 7; **else** *go* to 6.

6. **If** any of the descendants have an applicable state, **then** *set* the node's state to be the union of the applicable states of its descendants; **else** *set* the node's state to be the inapplicable state only. **Then** *go* to 7.

7. **If** one of the node's descendants is an unvisited tip, *go* to 8; **else** move to the closest non-visited node and *go* to 2. Once all nodes have been visited, end the first uppass.

8. **If** the unvisited tip has both inapplicable and applicable states, **then** *go* to 9 **else** *go* to 7

9. **If** the current node is inapplicable, solve the tip as inapplicable only; **else** remove the inapplicable state from the tip **then** *go* to 7.

3 Initialise tracker

1. Start at any tip and *go* to 2.

2. **If** the tip only contains an inapplicable state, *set* its tracker to "off" **then** move to the next tip and *go* to 2; **else** *go* to 3.

3. **If** the tip does not contain the inapplicable state, *set* its tracker to "on" **then** move to the next tip and *go* to 2. **Else** *go* to 4.

4. **If** the tip's ancestor contains an inapplicable state, *set* the tip's tracker to "off" **else**, *set* the tip's tracker to "on". **Then** *go* to the next tip and *go* to 2.

4 Second downpass

1. Enter on any cherry on the tree and move to its most recent common ancestor. **If** the trackers of either descendants is "on", *set* this node's tracker to "on". **Else** *set* it to "off". **Then**, *go* to 2

2. **If** the node had an applicable state in the previous pass (first up), *go* to 3; **else** leave the node state unchanged and *go* to 8.

3. **If** there is any state in common between both descendants, *go* to 4; **else**, *go* to 5.

4. **If** the states in common are applicable, *set* the node's state to be these states in common without the eventual inapplicable token; **else** *set* the node's state to be the inapplicable state. **Then** *go* to 8.

5. *Set* the node's state to be the union of the applicable states of both descendants (if present) and *go* to 6.

6. **If** both descendants have an applicable state, ***increment*** the tree length (change increment) and *go* to 8; **else** *go* to 7.

7. **If** both of the node's descendants' trackers are "on", ***increment*** the tree length (applicable region increment) **then** *go* to 8; **else** just *go* to 8.

8. **If** possible, move to the node's ancestor and *go* to 3; **else** move to the next unvisited cherry's ancestor and *go* to 3. Once all nodes have been visited, end the second downpass.

5 Second uppass

1. Enter the tree on its root and move to one of the root's descendants. **Then** *go* to 2.

2. **If** the node has any applicable state, *go* to 3; **else**, *go* to 10.

3. **If** the node's ancestor has any applicable state, *go* to 4; **else**, *go* to 11.

4. **If** the node states are the same as its ancestor, *go* to 11; **else**, *go* to 5.

5. **If** there is any state in common between the node's descendants, *go* to 6; **else** *go* to 7.

6. *Add* to the current node any state in common between its ancestor and its descendants. Then *go* to 11.

7. **If** the union between the node's descendants contains the inapplicable state, *go* to 8; **else** *go* to 9.

8. **If** there is any state in common between either of the node's descendants and its ancestor, *set* the node's states to be its ancestor's; **else** *add* to the current node states the applicable states also found in its descendants and ancestor. **Then** *go* to 11.

9. *Add* to the node's states the states of its ancestor. **Then** *go* to 11.

10. **If** both of the node's descendants' trackers are "on", ***increment*** the tree length (applicable region increment) **then** *go* to 8; **else** just *go* to 11.

11. **If** any one of the node's descendants is not a tip, move to the next node and *go* to 2. **If** both descendants are tips, move to the closest non-visited node and *go* to 2. Once all nodes have been visited, end the second uppass.

The tree length is then equal to the number of state changes and the number of additional applicable regions.