Inapplicable algorithm description

1 First downpass

1. Enter the tree on a tip and get on its node.
2. Compare the character states between the left and the right descendant of the node.
   (a) If there is a **union** between both descendants set the node state as this union else go to 2b.
      i. If there is an inapplicable token in the ancestral state *and* that both descendants have also an inapplicable token, set the node state as the **intersection** of both descendants then go to 4.

   (b) Else set the node state as the **intersection** between both descendants.
      i. If any of the descendants have an inapplicable token, add the inapplicable token to the node then go to 4.
3. @@@ Activate so subtree states?
4. Exit the first downpass.

2 First uppass

1. Enter the tree on the root.
2. @@@ Do some activation business.
3. If the node has both an inapplicable token go to 3a else go to 4.
   (a) If the node has an applicable token, go to 3(a)i else go to 3b.
      i. If the ancestor has only the inapplicable token, set the node state to be only the inapplicable token; then go to 4. Else go to 3(a)ii.
      ii. Remove the inapplicable token from the node; then go to 4.

   (b) The node is an inapplicable token; go to 3(b)i.

        i. If the ancestor has only the inapplicable token, set the node state to be only the inapplicable token; then go to 4. Else go to 3(b)ii.
        ii. If there is an **intersection** between the node's descendants states that is not inapplicable, set the node state to that **intersection**; then go to 4. Else go to 3(b)iii.
        iii. Set the node state to the inapplicable token only; then go to 4.

4. Exit the first uppass.

3 Second downpass

1. Enter the tree on a tip and get on its node.

2. If the node contains at least one applicable state go to 2a else, go to 4.

    (a) If the **union** between both descendants have an applicable character, set the node state to this union without any inapplicable characters; then go to 4. Else go to 2b.

    (b) Set the node state to be the **intersection** between the descendants and remove any eventual inapplicable tokens; then go to 4.

3. @@@ Some activation business

4. Exit the second downpass.

4 Second uppass

1. If the node has no inapplicable token go to 1a else go to 1b.

    (a) If ancestral state also doesn't have an inapplicable token go to 1(a)i, else go to 1(a)ii.

        i. If the **union** between the node and the ancestral state is equal to the ancestral state then set the node state to this union; then go to 3. Else go to 3.
        ii. If there is a **union** between the descendants then set the node to be the **intersection** between the **intersection** of the descendants and the **union** of the ancestral state (node = node — ((left — right) & ancestral)); then go to 3. Else go to 1(a)iii.

iii. If there is an **intersection** between the descendants that has an inapplicable token then go to 1(a)iiiA. Else go to 1(a)iiiD.

A. If this intersection **unions** with the ancestral state then set the node state to be equal to the **intersection** between the descendants **unioned** with the ancestral state and **intersecting** with the ancestral state (((left — right) & ancestral) — ancestral); then go to 3. Else go to 1(a)iiiB.

B. Set the node state to be the **intersection** between the descendants **intersecting** with the ancestral character (left — right — ancestral) without any eventual inapplicable tokens; then go to 3. Else go to 1(a)iiiC.

C. Set the node state to be the **intersection** between the node state and the ancestral state; then go to 3.

D. If the **union** between node state and ancestral state equals the ancestral state set the node state to be this union; then go to 3.

(b) If there is a **union** between the descendants, set the node state to be equal to this union; then go to 3. Else go to 2.

2. do some counting

3. Exit second uppass.