1 Algorithm 3, "Modification 1"

Before running this two free groups F_1 and F_2, on alphabets which we'll call X_1 and X_2, must have been entered, as well as a free group on a third alphabet, Z say, and subgroups $H_i \le F_i$, for $i = 1, 2$. In addition alg2_pre must have been run for each H_i, to generate the foldings Γ_i, their spanning forests T_i, their doubles, D_i and spanning forests FD_i for D_i.

The input to this part of the algorithm will be a directed rooted graph Δ, with edge labels from $X_1 \cup X_2 \cup Z$. The output will be 3 graphs $\Delta_{1,0}$, $\Delta_{2,0}$ and Δ_Z, and a set ν-im(u), for each vertex of $\Delta_{1,0}$ and $\Delta_{2,0}$. That is, for $k \in \{1, 2\}$ carry out the following. We call an edge with a label from X_k an edge of *type* k, and an edge with a label from Z an edge of *type* Z.

D1 Construct $\Delta_{k,0}$ and a set ν-im(α), for each vertex α of $\Delta_{k,0}$, and build up the graph Δ_Z. In detail the steps are the following.

(a) Remove all edges of type $X_{k'}$, where $k' = 3 - k$, from Δ to give a graph $\Delta'_{k,0}$. (Using the appropriate function from graph.py.)

(b) A *shoot* is an edge incident to a leaf. Remove all shoots of type Z from $\Delta'_{k,0}$; adding edges removed to a graph Δ_Z (which starts off empty, when $k = 1$, and is not reinitialised when k is incremented to 2). Continue until there are no shoots of type Z in $\Delta'_{k,0}$. The resulting graph is $\Delta_{k,0}$.

(c) Rename vertices of $\Delta_{k,0}$: a vertex named v in Δ becomes (v, k) in $\Delta_{k,0}$. In practice it may be easiest to start at step D1a by making a copy of Δ with each vertex v renamed as (v, k) (but in this case the k will have to be removed when adding a vertex to Δ_Z).

(d) For each vertex (v, k) of $\Delta_{k,0}$, set ν-im(v, k) equal to $\{v\}$ (again, this could be done at step D1a. For each vertex v of Δ_Z, set ν-im$(v) = \{v\}$.

At the next stage $\Delta_{k,0}$ is input, and a graph $\Delta_{k,1}$ is output, $k = 1, 2$.

D2 (**Begin Modification 1**.) For all edges (α, z, β) of $\Delta_{k,0}$, where $z \in Z$, add a path $(\alpha, \phi_k(z), \beta)$ to $\Delta_{k,0}$. The resulting graph is called $\Delta'_{k,1}$. (The file graph.py has a function allowing a labelled path to be added to a graph.) For each vertex (v, k) in $V(\Delta'_{k,1}) \backslash V(\Delta_{k,0})$ set ν-im$(v, k) = \{v\}$.

D3 Construct the Stallings folding $\Delta_{k,1}$ of (each component of) $\Delta'_{k,1}$. There is more to this than appears at first sight: there will often be more than one connected component, so first the bfs function must be run to find these components. Then each component must be folded. Whenever two vertices (u, k) and (v, k) of $\Delta'_{k,1}$ are identified, by the folding map, to a vertex (w, k), set $\nu\text{-im}(w, k) = \nu\text{-im}(u, k) \cup \nu\text{-im}(v, k)$. This process, combined with the initialisation, above, of ν-im (to a single vertex) for new vertices added, will be called *updating* ν-im, in the later stages of algorithm 3.