0.1 Deletions Handler

0.1.1 Handler Algorithm

Algorithm 1: Handle

Input: L, B, R, M

```
1  T_L ← treeToText(L);
2  T_B ← treeToText(B);
3  T_R ← treeToText(R);
4  foreach d_l ∈ D_L do
5      if d_l.children ≠ ∅ then
6          r ← find(r ∈ R → r.id = d_l.id);
7          m ← find(m ∈ M → m.id = d_l.id);
8          if sameShape(d_l, r) ∧ d_l.body = r.body then removeNode(m, M);
9          else if newReference(d_l.id, T_B, T_R) then m.parent.addChild(r, m.index);
10         else
11             removeNode(m, M);
12             a_l ← renamingMatch(A_L, d_l, T_B, T_L);
13             if a_l ≠ null then
14                 r.id ← a_l.id;
15                 removeNode(a_l, M);
16                 m.parent.addChild(r, m.index);
17             else
18                 n.id ← r.id;
19                 n.type ← r.type;
20                 n.body ← conflict(ε, d_l, r);
21                 m.parent.addChild(n, m.index);
22             end
23         end
24     end
25 end
26 foreach d_r ∈ D_R do
27     if d_r.children ≠ ∅ then
28         l ← find(l ∈ L → l.id = d_r.id);
29         m ← find(m ∈ M → m.id = d_r.id);
30         if sameShape(d_r, l) ∧ d_r.body = l.body then removeNode(m, M);
31         else if newReference(d_r.id, T_B, T_L) then m.parent.addChild(l, m.index);
32         else
33             removeNode(m, M);
34             a_r ← renamingMatch(A_R, d_r, T_B, T_R);
35             if a_r ≠ null then
36                 l.id ← a_r.id;
37                 removeNode(a_r, M);
38                 m.parent.addChild(l, m.index);
39             else
40                 n.id ← l.id;
41                 n.type ← l.type;
42                 n.body ← conflict(l, d_r, ε);
43                 m.parent.addChild(n, m.index);
44             end
45         end
46     end
47 end
```

Algorithm 2: Same Shape

Input: A, B
Output: wether nodes A and B have same shape

```
1 if A.children = ∅ ∧ B.children = ∅ then return A.type = B.type;
2 if A.children = ∅ ∨ B.children = ∅ then return FALSE;
3 if |A.children| ≠ |B.children| then return FALSE;
4 result ← TRUE;
5 foreach (a, b) ∈ (A.children, B.children) do
6 |   result ← result ∧ sameShape(a, b);
7 end
8 return result;
```

Algorithm 3: New Reference

Input: id, T_B, T
Output: wether there is a new reference to id in T

```
1 return countReferences(id, T) > countReferences(id, T_B);
```

Algorithm 4: Renaming Match

Input: A, d, T_B, T
Output: added node a in A with the same shape and similar body as deleted node d, such that there are no new references to a's id in T

```
1 return find(a ∈ A → sameShape(a, d) ∧ a.body ≈ d.body ∧ ¬newReference(a.id, T_B, T));
```