

# 1 Semistructured Merge

## 1.1 Early Concepts

1.  $A_L$  and  $A_R$  are the sets of all the nodes added by left and right, respectively
2.  $D_L$  and  $D_R$  are the sets of all the nodes deleted by left and right, respectively

## 1.2 Merge Algorithms

### Algorithm 1: Merge Files

```
Input: l, b, r, o
1 if  $l.content = b.content$  then
2   |  $o.content \leftarrow r.content$ ;
3 else if  $b.content = r.content \vee l.content = r.content$  then
4   |  $o.content \leftarrow l.content$ ;
5 else
6   |  $L \leftarrow fileToTree(l)$ ;
7   |  $B \leftarrow fileToTree(b)$ ;
8   |  $R \leftarrow fileToTree(r)$ ;
9   |  $M \leftarrow mergeTrees(L, B, R)$ ;
10  |  $H \leftarrow getActiveHandlers()$ ;
11  | foreach  $h \in H$  do
12  |   |  $h.handle(M)$ ;
13  | end
14  |  $o.content \leftarrow treeToText(M)$ ;
15 end
```

### Algorithm 2: Merge Trees

```
Input: L, B, R
Output: result of merging left, base and right trees
1  $L.index = 1$ ;
2  $B.index = 2$ ;
3  $R.index = 3$ ;
4  $LB \leftarrow mergeNodes(L, B, 1)$ ;
5  $M \leftarrow mergeNodes(LB, R, 2)$ ;
6  $D \leftarrow D_L \cap D_R$ ;
7  $removeRemainingBaseNodes(M, D)$ ;
8  $mergeMatchedContent(M)$ ;
9 return  $M$ ;
```

**Algorithm 3:** Remove Remaining Base Nodes

**Input:**  $T, D$

```

1 if  $D = \emptyset$  then return;
2 foreach  $d \in D$  do
3   if  $T = d$  then
4      $P \leftarrow T.parent$ ;
5      $P.children \leftarrow P.children - T$ ;
6     return;
7   end
8 end
9 foreach  $t \in T.children$  do
10   $removeRemainingBaseNodes(t, D)$ ;
11 end

```

**Algorithm 4:** Merge Matched Content

**Input:**  $T$

```

1 function  $mergeMatchedContent(T)$ 
2   foreach  $t \in T.children$  do
3      $mergeMatchedContent(t)$ ;
4   end
5   if  $T.children = \emptyset$  then
6     if  $SEPARATOR \in T.body$  then
7        $l, b, r \leftarrow split(T.body, SEPARATOR)$ ;
8        $l \leftarrow l - MARKER$ ;
9        $T.body \leftarrow textualMerge(l, b, r)$ ;
10    end
11  end
12 end

```

**Algorithm 5:** Merge Nodes**Input:** A, B, step**Output:** result of merging node A and node B

```

1 if  $A.type \neq B.type \vee A.id \neq B.id$  then
2   | return null;
3 end
4  $M \leftarrow A$ ;
5 if  $A.children = \emptyset \wedge B.children = \emptyset$  then
6   | if MARKER  $\in A.body$  then
7     |  $M.body \leftarrow A.body + B.body$ ;
8   | else if  $step = 1$  then
9     |  $M.body \leftarrow \text{MARKER} + A.body + \text{SEPARATOR} + B.body + \text{SEPARATOR}$ ;
10  | else if  $A.index = 1$  then
11    |  $M.body \leftarrow \text{MARKER} + A.body + \text{SEPARATOR} + \text{SEPARATOR} + B.body$ ;
12  | else
13    |  $M.body \leftarrow \text{MARKER} + \text{SEPARATOR} + A.body + \text{SEPARATOR} + B.body$ ;
14  | end
15  | return  $M$ ;
16 else if  $A.children \neq \emptyset \wedge B.children \neq \emptyset$  then
17   | foreach  $b \in B.children$  do
18     |  $a \leftarrow \text{find}(a \in A.children \rightarrow a.type = b.type \wedge a.id = b.id)$ ;
19     | if  $a = \text{null}$  then
20       |  $M.children \leftarrow M.children \cup b$ ;
21       | if  $step = 1$  then  $D_L \leftarrow D_L \cup b$  ;
22       | else  $A_R \leftarrow A_R \cup b$  ;
23     | else
24       | if  $a.index = -1$  then  $a.index \leftarrow A.index$  ;
25       | if  $b.index = -1$  then  $b.index \leftarrow B.index$  ;
26       | if  $step = 1 \wedge a \in A_L$  then  $A_R \leftarrow A_R \cup b$  ;
27       |  $m \leftarrow \text{mergeNodes}(a, b, step)$ ;
28       |  $M.children \leftarrow M.children \cup m$ ;
29     | end
30   | end
31   | foreach  $a \in A.children$  do
32     |  $b \leftarrow \text{find}(b \in B.children \rightarrow a.type = b.type \wedge a.id = b.id)$ ;
33     | if  $b = \text{null}$  then
34       |  $ls \leftarrow \text{leftSiblings}(a)$ ;
35       |  $rs \leftarrow \text{rightSiblings}(a)$ ;
36       |  $M.children \leftarrow ls \cup a \cup rs$ ;
37       | if  $step = 1$  then  $A_L \leftarrow A_L \cup a$  ;
38       | else  $D_R \leftarrow D_R \cup a$  ;
39     | end
40   | end
41   | return  $M$ ;
42 end
43 return null;

```