1 Semistructured Merge

1.1 Early Concepts

- Every node's origin is set to UNKNOWN beforehand

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 $L.origin = LEFT$;
2 $B.origin = BASE$;
3 $R.origin = RIGHT$;
4 $LB \leftarrow$ `mergeNodes`(L, B);
5 $M \leftarrow$ `mergeNodes`(LB, R);
6 **foreach** $d \in D_L \cap D_R$ **do**
7   `removeNode`(d, M);
8 **end**
9 `runTextualMergeOnLeaves`(M);
10 **return** M;

Algorithm 3: Run Textual Merge On Leaves

Input: T

1 **foreach** $t \in T.children$ **do**
2   `runTextualMergeOnLeaves`(t);
3 **end**
4 **if** $T.children = \emptyset \wedge SEPARATOR \in T.body$ **then**
5     $l, b, r \leftarrow split(T.body, SEPARATOR)$;
6     $l \leftarrow l - MARKER$;
7     $T.body \leftarrow$ `textualMerge`(l, b, r);
8 **end**

Algorithm 4: Merge Nodes

Input: A, B
Output: result of merging nodes A and B

1 if $A = null$ then return B;
2 if $B = null$ then return A;
3 if $A.type \neq B.type \vee A.id \neq B.id$ then return $null$;

4 $M.id \leftarrow B.id$;
5 $M.type \leftarrow B.type$;
6 $M.origin \leftarrow B.origin$;
7 $M.children \leftarrow \emptyset$;

8 if $A.children = \emptyset \wedge B.children = \emptyset$ then
9   if $MARKER \in A.body$ then
10 $M.body \leftarrow A.body + B.body$;
11   else if $A.origin = LEFT \wedge B.origin = BASE$ then
12 $M.body \leftarrow MARKER + A.body + SEPARATOR + B.body + SEPARATOR$;
13   else if $A.origin = LEFT$ then
14 $M.body \leftarrow MARKER + A.body + SEPARATOR + SEPARATOR + B.body$;
15   else
16 $M.body \leftarrow MARKER + SEPARATOR + A.body + SEPARATOR + B.body$;
17   end

18   return M;
19 else if $A.children \neq \emptyset \wedge B.children \neq \emptyset$ then
20   foreach $b \in B.children$ do
21 $a \leftarrow find(a \in A.children \rightarrow a.type = b.type \wedge a.id = b.id)$;
22   if $a.origin = UNKNOWN$ then $a.origin \leftarrow A.origin$;
23   if $b.origin = UNKNOWN$ then $b.origin \leftarrow B.origin$;
24 $M.children \leftarrow M.children \cup \texttt{mergeNodes(}a\texttt{, }b\texttt{, }step\texttt{)}$;
25   end
26   foreach $a \in A.children$ do
27 $b \leftarrow find(b \in B.children \rightarrow a.type = b.type \wedge a.id = b.id)$;
28   if $a.origin = UNKNOWN$ then $a.origin \leftarrow A.origin$;
29   if $b = null$ then $M.children \leftarrow M.children \cup a$;
30   end
31   return M;
32 end

33 return $null$;