1 Multiple Initialization Blocks Handler

1.1 Handler Algorithm

Algorithm 1: Handle

Input: L, B, R, M

$A_L \leftarrow \{l \in L \mid (\neg\exists b \in B)(l.id = b.id)\}$;
$A_R \leftarrow \{r \in R \mid (\neg\exists b \in B)(r.id = b.id)\}$;
$D_B \leftarrow \{b \in B \mid (\neg\exists l \in L)(b.id = l.id) \wedge (\neg\exists r \in R)(b.id = r.id)\}$;

$IB_L \leftarrow \{n \in A_L \mid n.type = INITBLOCK\}$;
$IB_R \leftarrow \{n \in A_R \mid n.type = INITBLOCK\}$;
$IB_B \leftarrow \{n \in D_B \mid n.type = INITBLOCK\}$;

$E_L \leftarrow$ `editedNodes`(IB_L, IB_B);
$E_R \leftarrow$ `editedNodes`(IB_R, IB_B);

$D_L \leftarrow$ `deletedNodes`(IB_L, IB_B, E_L);
$D_R \leftarrow$ `deletedNodes`(IB_R, IB_B, E_R);

foreach $b \in IB_B$ **do**
    $l \leftarrow E_L.getValue(b)$;
    $r \leftarrow E_R.getValue(b)$;
    if $l \neq null \wedge r \neq null$ **then**
        $m \leftarrow find(m \in M \rightarrow m.body = l.body)$;
        $m.body \leftarrow$ `textualMerge`$(l.body, b.body, r.body)$;
        $m \leftarrow find(m \in M \rightarrow m.body = r.body)$;
    else if $l \neq null \vee r \neq null$ **then**
        if $l \neq null$ **then**
            $r \leftarrow find(r \in D_R \rightarrow r.body = b.body)$;
            if $r \neq null$ **then** `removeNode`(b, M) ;
        else
            $l \leftarrow find(l \in D_L \rightarrow l.body = b.body)$;
            if $l \neq null$ **then** `removeNode`(b, M) ;
        end
        $m \leftarrow find(m \in M \rightarrow m.body = l.body)$;
        $m.body \leftarrow$ `textualMerge`$(l.body, b.body, r.body)$;
        $m \leftarrow find(m \in M \rightarrow m.body = r.body)$;
    else
        $m \leftarrow find(m \in M \rightarrow m.body = b.body)$;
    end
    `removeNode`(m, M);
end

$A_L \leftarrow$ `addedNodes`(IB_L, IB_B, E_L);
$A_R \leftarrow$ `addedNodes`(IB_R, IB_B, E_R);

Algorithm 2: Edited Nodes

Input: IB, IB_B
Output: map associating a deleted base node b in IB_B and its correspondent added branch node a in IB

$D \leftarrow \{d \in IB_B \mid (\neg\exists a \in IB)(d.body = a.body)\}$;
$A \leftarrow \{a \in IB \mid (\neg\exists d \in IB_B)(a.body = d.body)\}$;
$matches \leftarrow \emptyset$;
foreach $a \in A$ **do**
    $S \leftarrow \{d \in D \mid a.body \approx d.body\}$;
    $b \leftarrow \underset{s \in S}{\text{argmax}}$ (`similarity(`$s.body$, $a.body$`)`);
    if $b \neq null$ **then** $matches \leftarrow matches \cup \{b : a\}$;
end
return $matches$

Algorithm 3: Added Nodes

Input: IB, IB_B, E
Output: set of initilization block nodes added by branch

$A \leftarrow \{n \in IB \mid (\neg\exists b \in IB_B)(n.body = b.body)\}$;
$A \leftarrow \{n \in A \mid (\neg\exists e \in E)(n.body = e.value.body)\}$;
return A;

Algorithm 4: Deleted Nodes

Input: IB, IB_B, E
Output: set of initialization block nodes deleted by branch

$D \leftarrow \{b \in IB_B \mid (\neg\exists n \in IB)(b.body = n.body)\}$;
$D \leftarrow \{n \in D \mid (\neg\exists e \in E)(n.body = e.key.body)\}$;
return D;