Algorithm 2: A part of Relationship Traversal Algorithm

```
Input: Syntax tree node with subtrees
// Traverse the node with (3 or more) subtrees
// Node.LeftS: process composed of leaf nodes of the left subtree
// Node.RightS: process composed of leaf nodes of the right subtree
// Name: identifier of a defined process
// Def: ‘‘definitionLeft’’ of a subtree
if VisitEqual (Node.MiddleS) then // "="operator, meet process definition
    add Node.RightS to Init[Node.LeftS.Name];
    add Node.LeftS.Name to Node[Node.RightS];
if VisitDef (Node) then // meet the left part of a process definition
    add Node.Name to Init[Node];
    add Node to Node[Node.Name];
if VisitPME(Node) then // meet a parallel multi-instance task or process
    get InsNum, MsgNum (if exists) and TaskContent;
    add TaskContent to Init[Node];
    add Node to End[TaskContent];
if VisitSME(Node) then // meet a sequential multi-instance task or process
    get InsNum, MsgNum(if exists) and TaskContent;
    add TaskContent to Init[Node];
    add Node to End[TaskContent];
if VisitTME(Node) then // meet a time-bounded multi-instance task
    get InsNum, MsgNum(if exists),Duration and TaskContent;
    add TaskContent to Init[Node];
    add Node to End[TaskContent];
return Next, End, Init, And, Xor;
```