The liveness2BPMN transformation algorithm

The transformation algorithm is a recursive algorithm that takes the liveness formula expression elements (right hand side of the formula) from left to right and applies the templates shown in Figure 1, gradually building the BPMN process model. For applying templates, keep in mind that the control flows from left to right, thus, if a template follows another, then it is connected to its rightmost element.

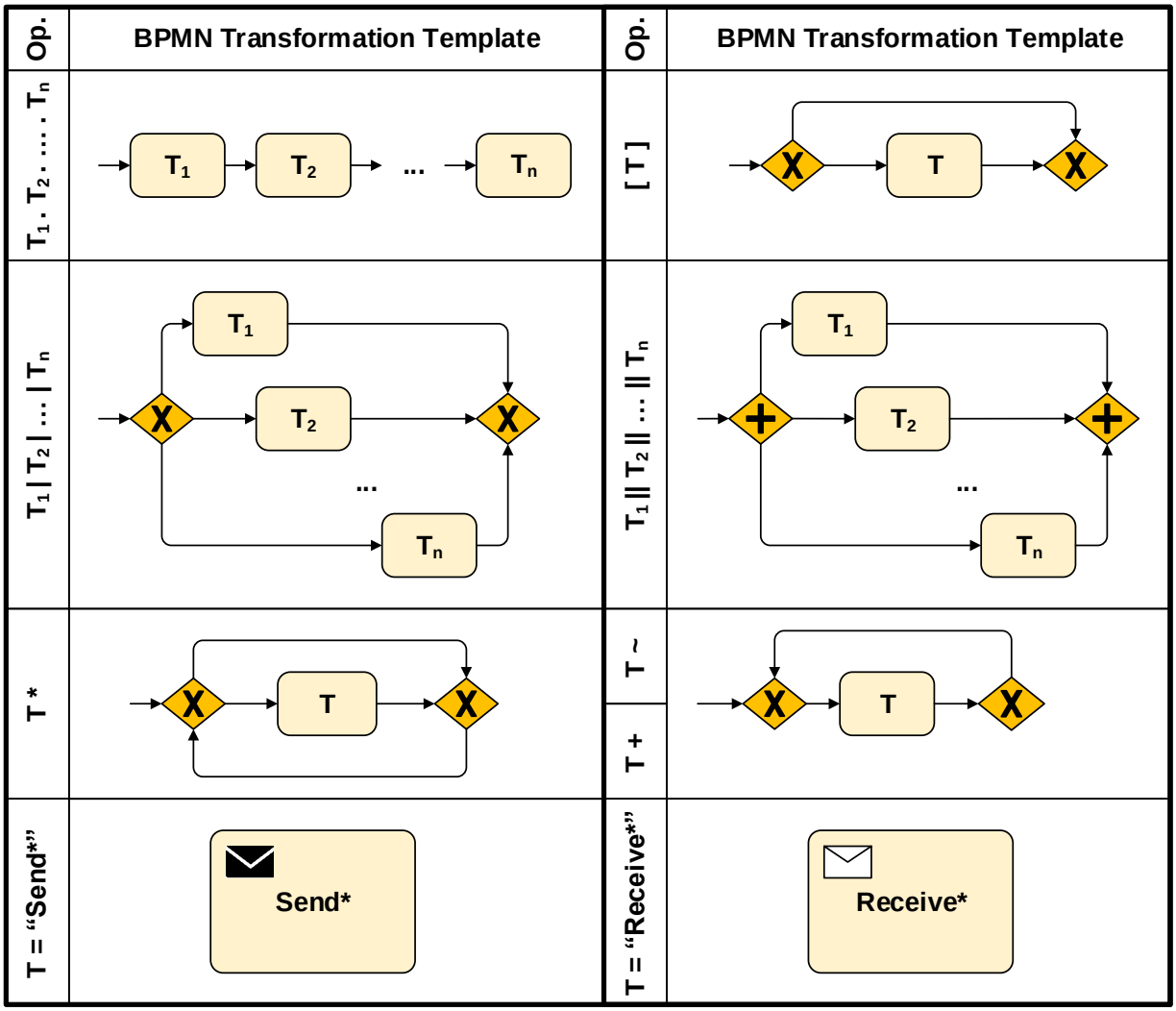

Figure 1. The BPMN 2.0 transformation templates for the liveness formula operators and task names

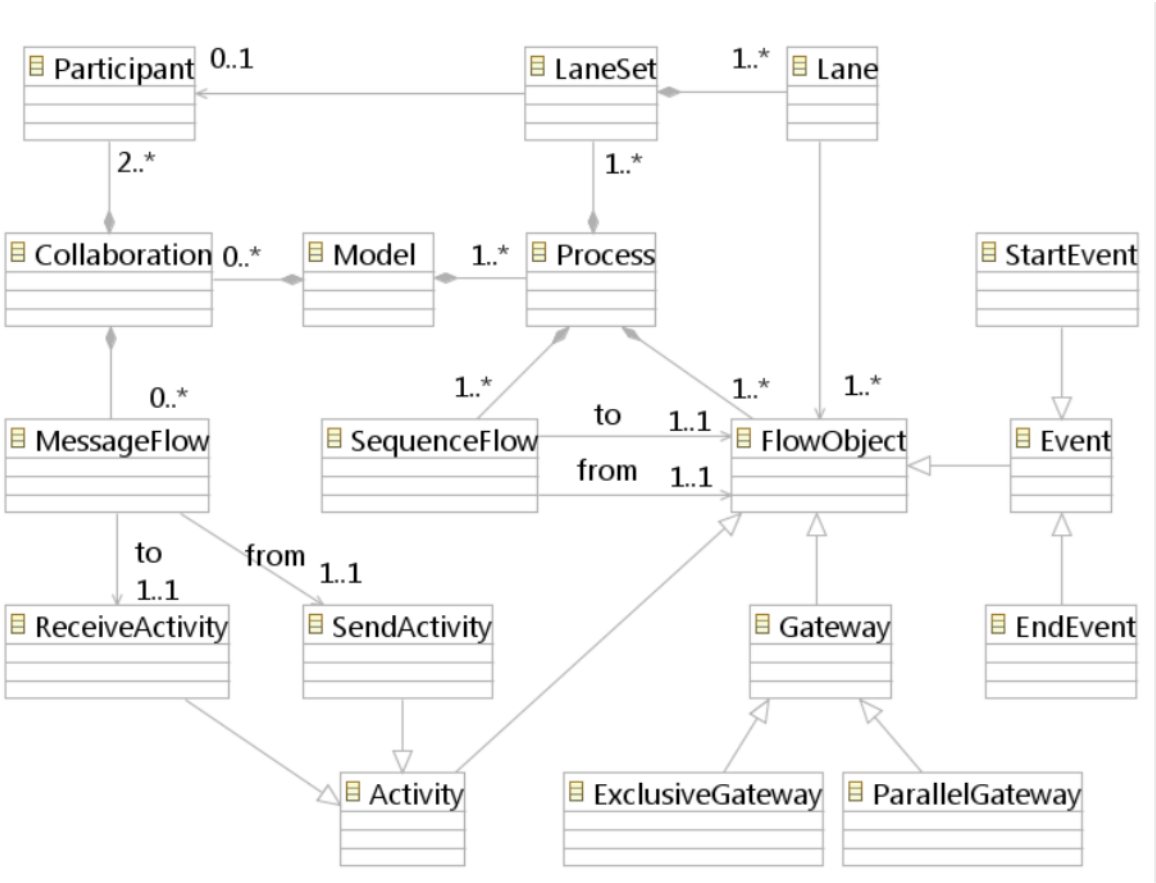

Figure 2. Abstract BPMN 2.0 metamodel

The transformation algorithm uses the elements depicted in Figure 2 in order to provide a complete BPMN model out of a simple liveness formula. It is important to note that all BPMN elements inherit from FlowObject therefore they can be connected with SequenceFlows. The algorithm is provided with a liveness formula and an empty BPMN model in order to recursively produce the desired result. The complete algorithm is provided for the reader.

```
Program transform(String liveness, Model model)
      Process process = new Process
      model.processes.add(process)
      StartEvent startEvent = new StartEvent
      startEvent.type = start
      process.add(startEvent)

      FlowObject lastActivity = createProcess(liveness.formula1.expression,
workflowProcess, startEvent)
      EndEvent endEvent = new EndEvent
      endEvent.type = end
      process.add(endEvent)
      SequenceFlow transition = new SequenceFlow
      transition.from = lastActivity
      transition.to = endEvent
      process.add(transition)
End Program
```

```
Function FlowObject createProcess(String expression, Process process,
FlowObject activity)
      List terms = new List
            For Each term_i In expression
                  terms.add(term_i)
            End For
            If terms.size() > 1 Then
                  If expression Is sequentialExpr Then
                        For Each term_i In expression
                              FlowObject newActivity = createProcess(term_i,
                              process, activity)
                              activity = newActivity
                        End for
                  Else If expression Is orExpr
                        Gateway xorEntryGateway = new Gateway
                        xorEntryGateway.gatewayType = XOR
                        process.add(xorEntryGateway)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = activity
                        transition.to = xorEntryGateway
                        process.add(transition)
                        Gateway xorExitGateway = new Gateway
                        xorExitGateway.gatewayType = XOR
                        process.add(xorExitGateway)
                        For Each term_i In expression
                              FlowObject newActivity = createProcess(term_i,
                              workflowprocess, xorEntryGateway)
                              transition = new Transition
                              transition.from = newActivity
                              transition.to = xorExitGateway
                              process.add(transition)
                        End for
                        activity = xorExitGateway
                  Else If expression Is parallelExpr
                        Gateway parallelEntryGateWay = new Gateway
                        parallelGateWay.gatewayType = Parallel
                        process.add(parallelEntryGateway)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = activity
                        transition.to = parallelEntryGateway
                        process.add(transition)
                        Gateway parallelExitGateway = new Gateway
                        parallelExitGateway.gatewayType = Parallel
                        process.add(parallelExitGateway)
                        For Each term_i In expression
                              FlowObject newActivity = createProcess(term_i,
                              process, parallelEntryGateway)
                              transition = new Transition
                              transition.from = newActivity
                              transition.to = xorExitGateway
                              process.add(transition)
                        End For
                        activity = parallelGateway
                  End If
            For Each term_i In expression
                  If term_i Is basicTerm
                        boolean foundLeftHandSideEqualsBasicTerm = false
```

```
For Each formula_i In liveness
      If formula_i.leftHandside = term_i Then
            FlowObject newActivity =
            createProcess(formula_i.expression,
            process, activity)
            activity = newActivity
            foundLeftHandSideEqualsBasicTerm = true
      End If
      If foundLeftHandSideEqualsBasicTerm = false
            Activity newActivity = new Activity
            process.add(newActivity)
            SequenceFlow transition=new SequenceFlow
            transition.from = activity
            transition.to = newActivity
            process.add(transition)
            activity = newActivity
      End If
Else If (term_i is of type '(' term ')' ) Then
      FlowObject newActivity = createProcess(term, process,
      activity)
      activity = newActivity
Else If (term_i is of type '[' term ']')Then
      Gateway xorEntryGateway = new Gateway
      xorEntryGateway.gatewayType = XOR
      process.add(xorEntryGateway)
      Gateway xorExitGateway = new Gateway
      xorEntryGateway.gatewayType = XOR
      process.add(xorEntryGateway)
      SequenceFlow transition = new SequenceFlow
      transition.from = activity
      transition.to = xorEntryGateway
      process.add(transition)
      FlowObject newActivity = createProcess(term, process,
      xorEntryGateway)
      SequenceFlow transition = new SequenceFlow
      transition.from = newActivity
      transition.to = xorExitGateway
      process.add(transition)
      SequenceFlow transition = new SequenceFlow
      transition.from = xorEntryGateway
      transition.to = xorExitGateway
      activity = xorExitGateway
Else If (term_i is of type '*') Then
      Gateway xorEntryGateway = new Gateway
      xorEntryGateway.gatewayType = XOR
      process.add(xorEntryGateway)
      Activity xorExitGateway = new Activity
      xorEntryGateway.gatewayType = XOR
      process.add(xorEntryGateway)
      SequenceFlow transition = new SequenceFlow
      transition.from = activity
      transition.to = xorEntryGateway
      process.add(transition)
      FlowObject newActivity = createProcess(term, process,
      xorEntryGateway)
      SequenceFlow transition = new SequenceFlow
      transition.from = newActivity
```

```
                        transition.to = xorExitGateway
                        process.add(transition)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = xorEntryGateway
                        transition.to = xorExitGateway
                        process.add(transition)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = xorExitGateway
                        transition.to = startof(term)
                        process.add(transition)
                        activity = xorExitGateway
                  Else If (term_i is of type '~') Then
                        FlowObject newActivity = createProcess(term, process,
                        activity)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = newActivity
                        transition.to = startof(term)
                        process.add(transition)
                        activity = newActivity
                  Else If (term_i is of type '+') Then
                        Gateway xorExitGateway = new Gateway
                        xorExitGateway.gatewayType = XOR
                        process.add(xorExitGateway)
                        FlowObject newActivity = createProcess(term_i, process,
                        activity)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = newActivity
                        transition.to = xorExitGateway
                        process.add(transition)
                        SequenceFlow transition = new SequenceFlow
                        transition.from = xorExitGateway
                        transition.to = startof(term)
                        process.add(transition)
                        activity = xorExitGateway
                  End If
            End If
      End For
return activity
```