**ForEach Transformation for Story Diagrams**

The forEach Transformation transforms an *input story diagram* into a *forEach story diagram*.

**Added Parameters:**

The forEach story diagram contains the following Parameters:

* step : StepGraph  
  This is the StepGraph on which the story diagram is to be applied.
* successors : SuccessorStepGraphs  
  This is the result container. Each successor that is produced by the current story diagram will be added to the successors reference of this object.

**New first activity:**

The forEach story diagram has an additional activity node (compared to the input story diagram) which serves as the first activity node. This node is created by the following steps:

* The activity is a forEach activity.
* It contains all object and link variables as well as all pattern constraints that are contained in the left hand side of the first activity node of the input story diagram. The names of all of these object variables are suffixed by “\_step”. For example, “o1” becomes “o1\_step”.
* It contains a bound object variable step representing the parameter step.
* All object variables are bound via the step object.
  + If the object variable binds an unchangeable node, then it is bound via the unchangeableNodes reference.
  + If the object variable binds some other node, then it is bound via the changeableNodes reference.
* It contains an object variable trans : SDMTransition which has modifier CREATE.
* It has two outgoing activity edges:
  + An activity edge with edge guard END to the final node of the story diagram.
  + An activity edge with edge guard EACH\_TIME to the copy statement node.

**Copy Statement Node:**

The copy statement node is responsible for copying the current step graph. Thus, it is called for each match of the first activity. It contains the following call:

SDMReachabilityComputation.copyStepGraph(step, trans)

It returns a copy of the current StepGraph step and connects it via the SDMTransition trans which has been created in the first activity node. The result of the call is stored in the variable succ : StepGraph. After this node, the execution continues with the original story diagram.

**Original Story Diagram:**

After the copy statement node, the original story diagram follows and is executed on the copy succ. Therefore, all activity nodes and edges of the input story diagram are copied to the forEach story diagram and then modified. In general, the following modification are necessary on the activity node level:

* The transition from the copy statement node goes to the first activity node of the input story diagram.
* All activity edges that target a final node are redirected to the first activity node of the forEach story diagram (obtain the next possible matching for the story diagram).

The execution must use the same match that was identified in the first activity node. Thus, the match is recreated using the information provided by the copy function. That requires the following changes in the first node of the original story diagram:

* All object variables of the left hand side are bound using the index map of the SDMTransition trans which was used for copying. The index maps objects of step to their respective copy in succ. Thus, the match obtained in the first activity of the forEach story diagram is recreated using the index map: o1 := trans.index(o1\_step).
* It is extended by a bound object variable successors which represents the second parameter of the forEach story diagram. Then, succ is added to the successors reference of this object.

In all subsequent activity nodes of the input story diagram, the following modification must be applied:

* If an object is created by the story diagram, then it must be added to the changeableNodes reference of succ.
* If an object is deleted by the story diagram, then it must be removed from the changeableNodes reference of succ.