Transformation of sequence diagram into collaboration diagram using graph grammar in AToM3

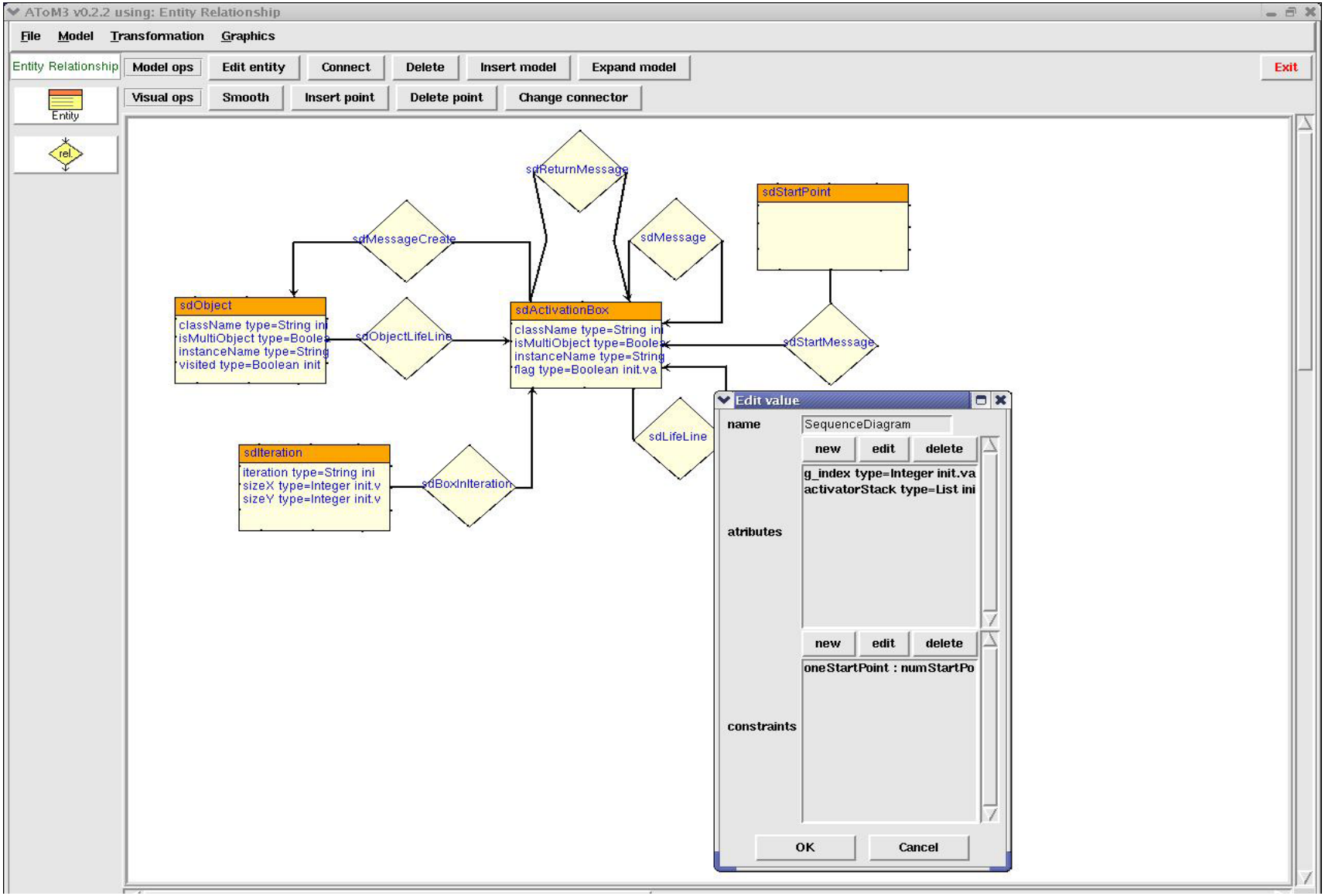
Xia shengjie
sxia@cs.mcgill.ca

Reference: Diploma Thesis
" UML Interaction Diagrams: Correct translation of Sequence Diagrams into Collaboration Diagrams“
http://www.informatik.uni-bremen.de/~hoelsch/diploma-thesis.html

OUTLINE

- ER_model
  - sequenceDiagram_ER_model
  - collaborationDiagram_ER_model
- Transformation unit
  - sd2sg
  - sg2cd
  - cd2cd1
- Conclusion

ER_model
sequenceDiagram_ER_model



Important attribute added

global_attribute: g_index = 0

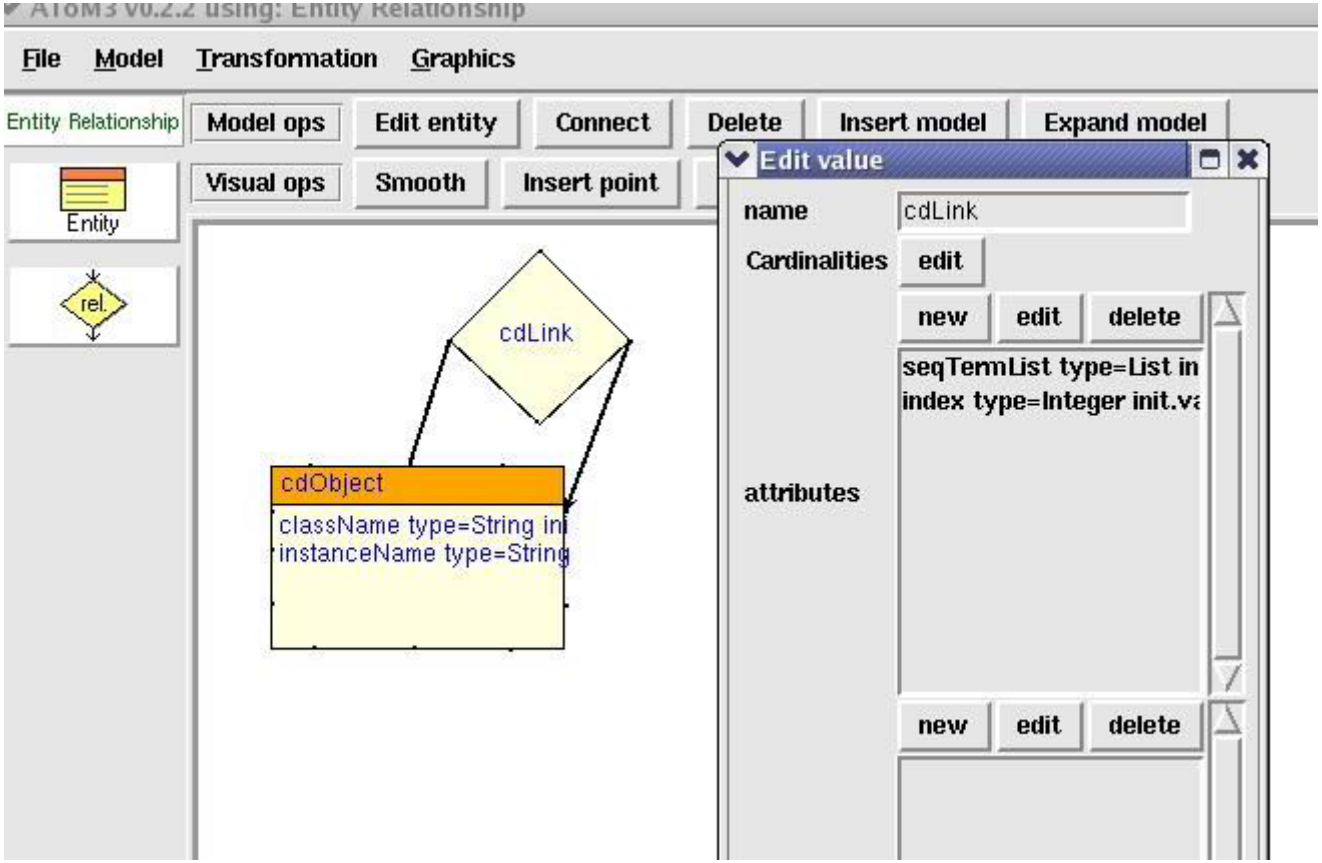
sdObject_attribute: visited = false

sdActivationBox_attribute: className
instanceName
isMultiObject

sdMessage_attribute: index = 0
visited = false
seqTermList =[[ ]]

sdReturnMessage_attribute: index = 0
visited = false

ER_model: collaborationDiagram_ER_model



note: seqTermList of cdLink: list of string

Transformation unit

Initial graph: sequence diagram

Uses:

sd2sg: traverse the sequence diagram(sd) , order the message with the index, specify sequence term list (seqTermList) of each message

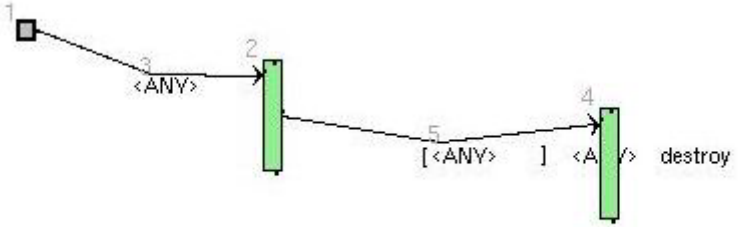
sg2cd: create the objects for the collaboration diagram(cd), add it into sd, copy messages from sd into cd, finally delete the sd

cd2cd1: tranform the cd generated from sg2cd into its standard form

Control condition: sd2sg; sg2cd; cd2cd1

Final graph: collaboration diagram

Transformation unit sd2sg



sd2sg_rule_1

sd2sg_rule_1

Condition : msg5_visited[1] == false
for all aBox
if attributes of aBox have no value
take the corresponding value of the attributes of its incoming aBox or sdObject

Action: highLight msg3

RHS:

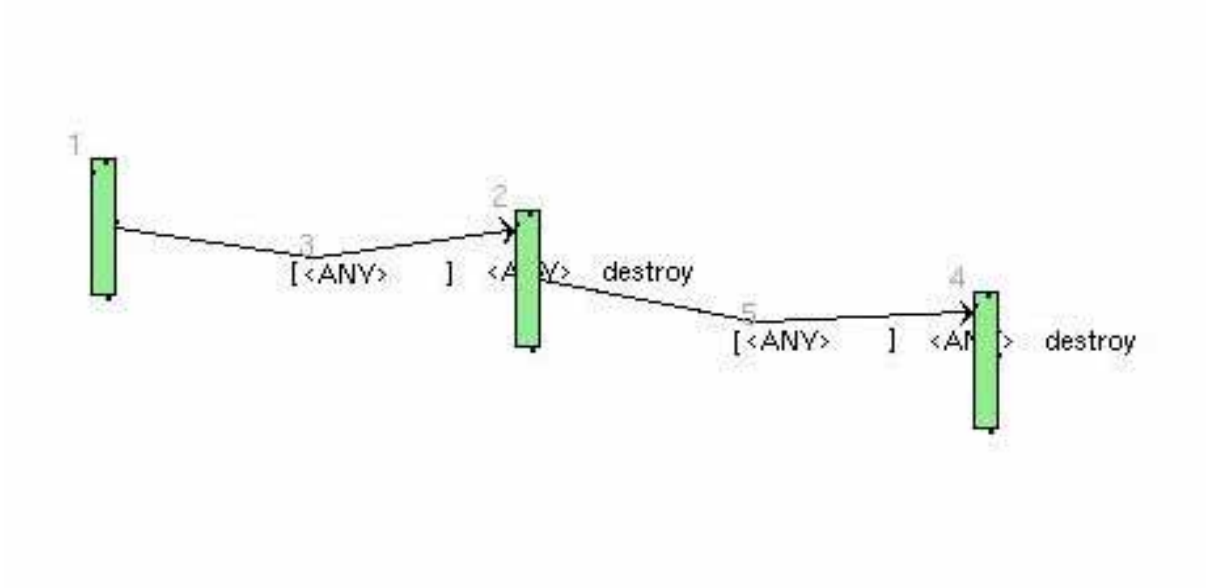
specify for msg5

visited: msg5.visited == true

index : gl_index ++
msg5.index = gl_index

seqTermList: if msg5_type == expression
seqTermList = [[1,'_']]
else msg5_type == condition
seqTermList = [[1,'a']]

Transformation unit sd2sg



sd2sg_rule_2

Case 1: Activator relationship

Case 2: Predecessor-Successor relationship

sd2sg_rule_2

condition: msg3_index == gl_index and msg5_visited == false
for all aBox, if no value for their attributes, assign value to them
action: highLight msg3
RHS:
specify for msg5:
visited = true, gl_index++, index = gl_index
seqTermList

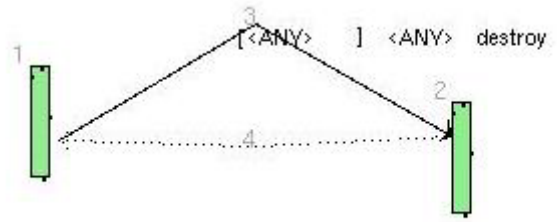
case 1: msg3 between different sdObjects, activator relationship

if msg5.type == expression
msg5.seqTermList = msg3.seqTermList.append([1,'_'])
if msg5.type == condition
msg5.seqTermList = msg3.seqTermList.append([1,'a'])

case 2: msg3 between same sdObjects, predecessor-successor relationship

change the last term of msg3.seqTermList
msg5.seqTermList = msg3.seqTermList
assume last_term_msg3_seqTermList = [int1, str2]
if msg5.type == expression
last_term = [int1+1,'_']
if msg3.type == expression and msg5.type == condition
last_term =[int1+1,'a']
if msg3.type == condition and msg5.type == condition
last_term = [int1, successor(str2)]

Transformation unit sd2sg



sd2sg_rule_3

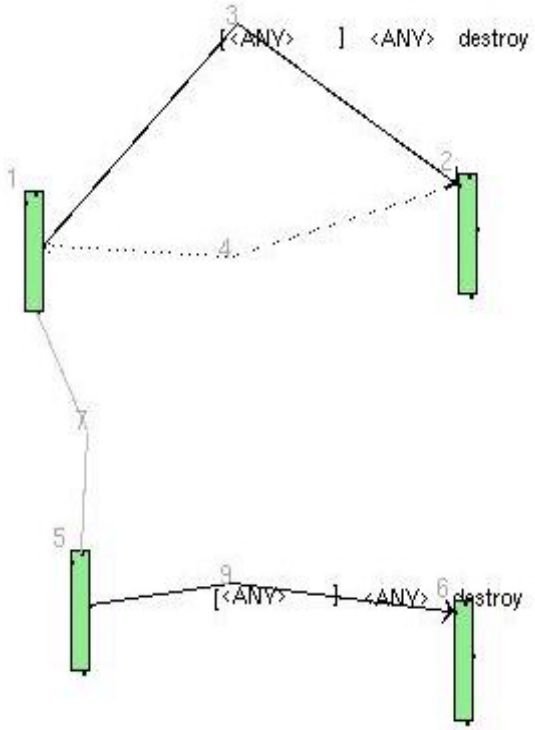
condition: msg3_index == gl_index and msg4_visited == false
assignment for the attributes of all aBox if necessary

RHS

specify for msg4

visited = true, gl_index++, index = gl_index

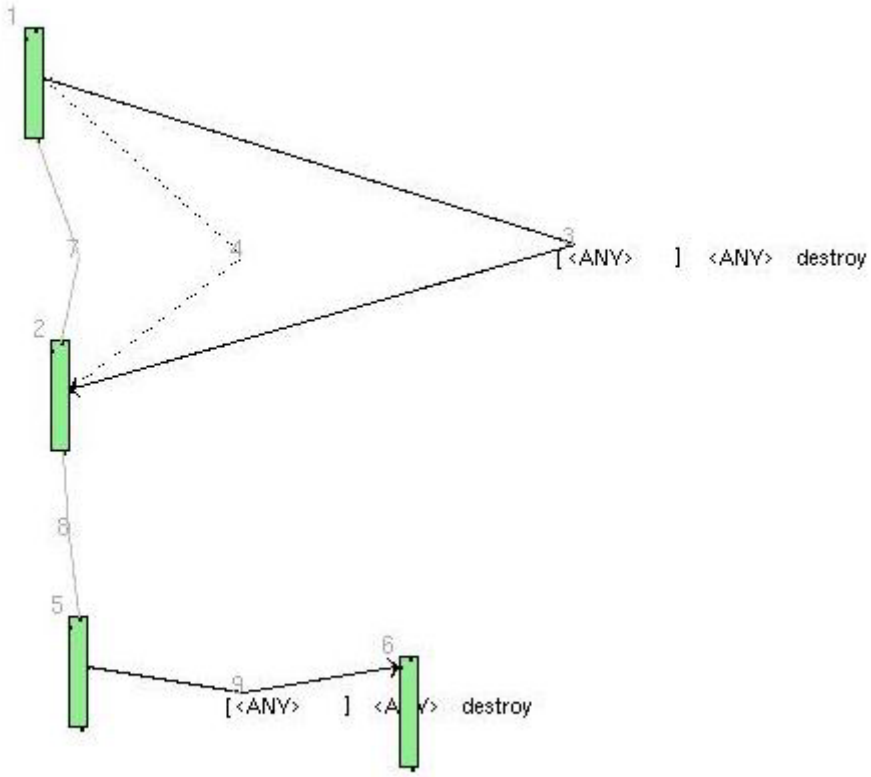
Transformation unit sd2sg



sd2sg_rule_4

Deal with Predecessor - successor relationship

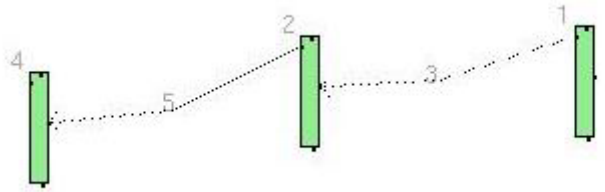
Transformation unit sd2sg



sd2sg_rule_4_1

Deal with the Predecessor-successor relationship

Transrmafotion unit sd2sg



sd2sg_rule_5

condition: msg3_index == gl_index and msg5_visited == false
assignment for the attributes of all aBox if necessary

RHS

specify for msg4

visited = true, gl_index++ , index = gl_index

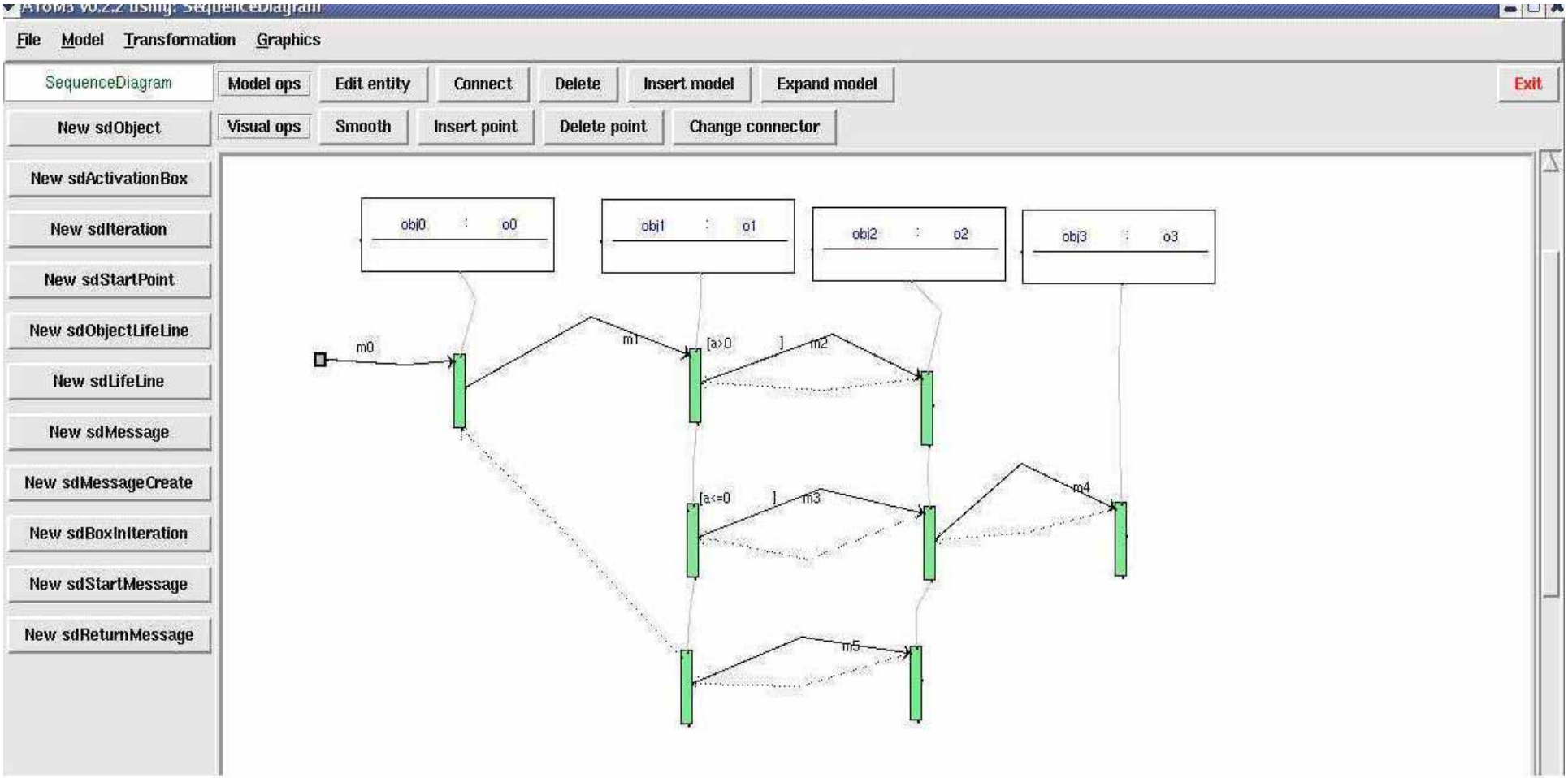


Figure 1 Example: sequence diagram

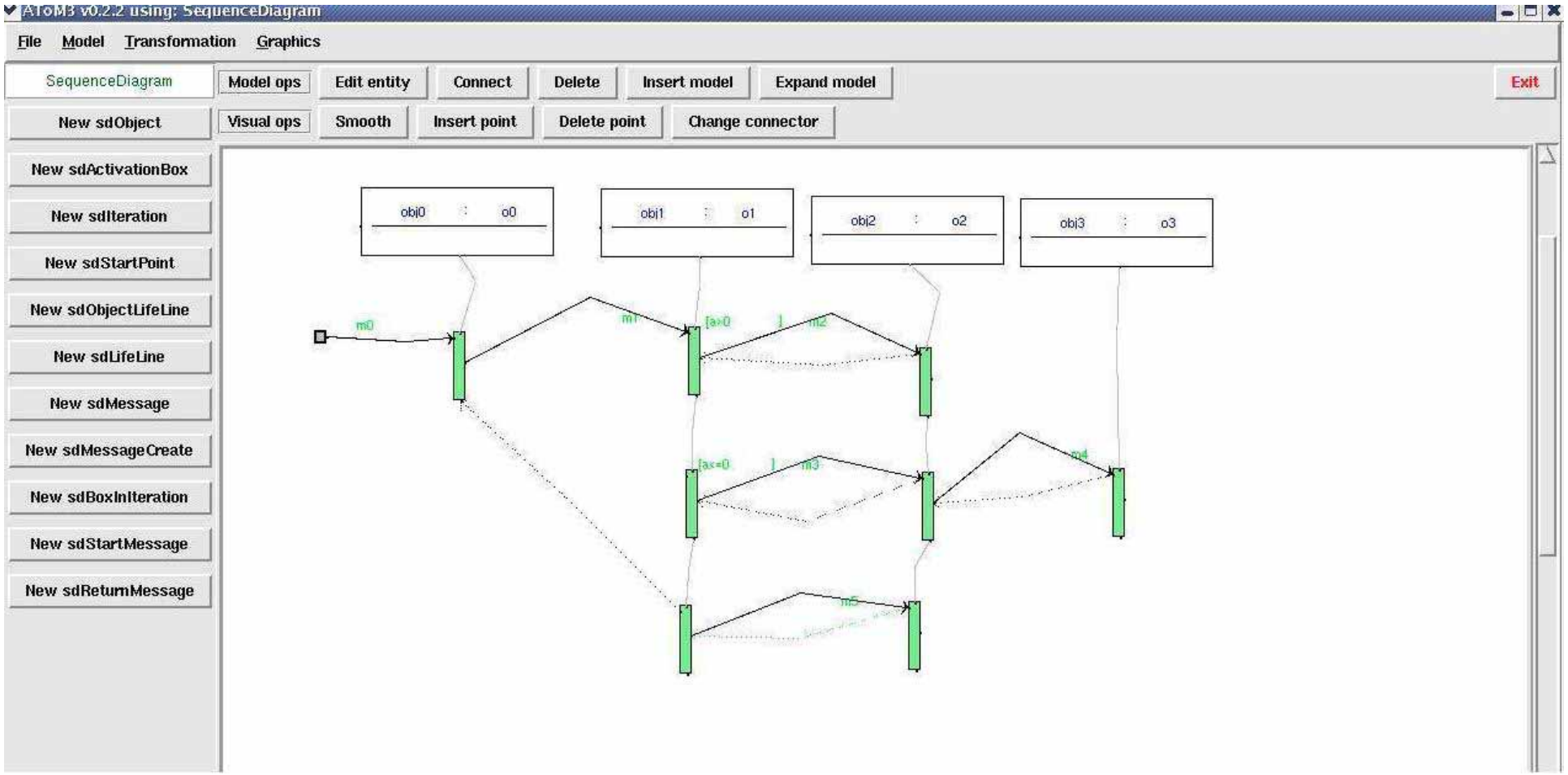
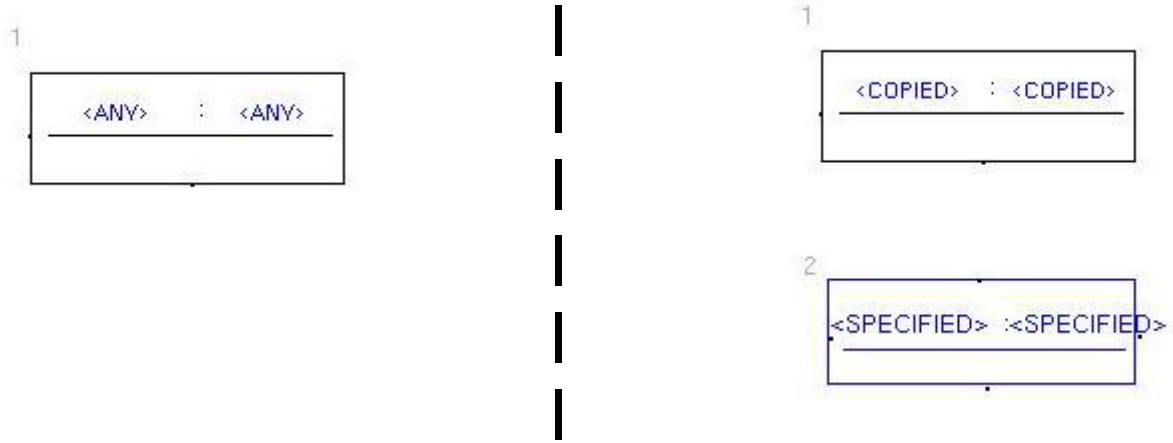


Figure 2 Example: the resulting diagram after the application of sd2sg to the sequence diagram in figure 1

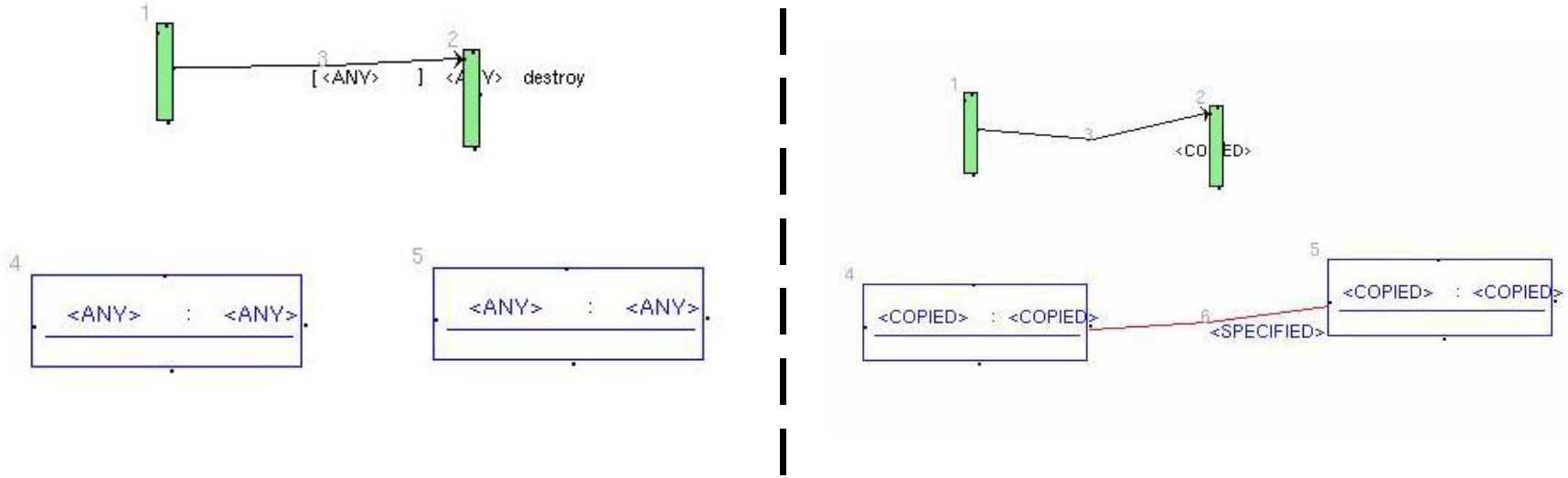
Transformation unit sg2cd



sg2cd_rule_1

condition: sdObject.visited == false
RHS:
specify for cdObject
cdObject.className = sdObject.className
cdObject.instanceName = sdObject.instanceName

Transformation unit sg2cd



sg2cd_rule_2

sg2cd_rule_2:

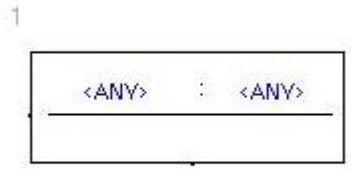
condition: abox1_instanceName == cdObj4_instanceName
abox3_instanceName == cdObj5_instanceName
gl_index == msg3_index
action: gl_index ++
RHS
specify for cdLink
seqTermList: contain all the information about msg3
including its sending direction, type, expression,
condition if any, seqTermList

Transformation unit sg2cd

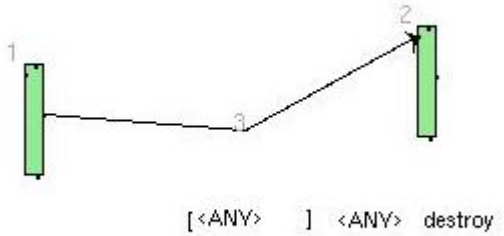
sg2cd_rule_3

action: gl_index++

Transformation unit sg2cd

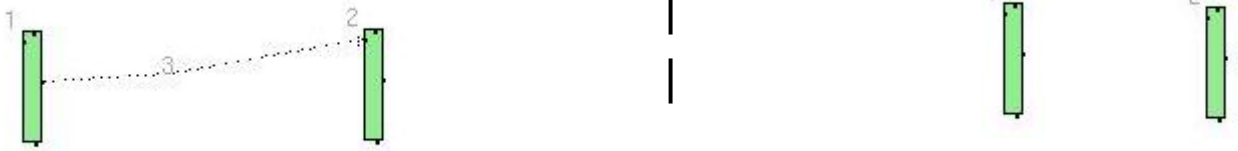


sg2cd_rule_3

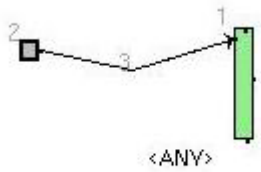


sg2cd_rule_4

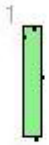
Transformation unit sg2cd



sg2cd_rule_5



sg2cd_rule_6



sg2cd_rule_7

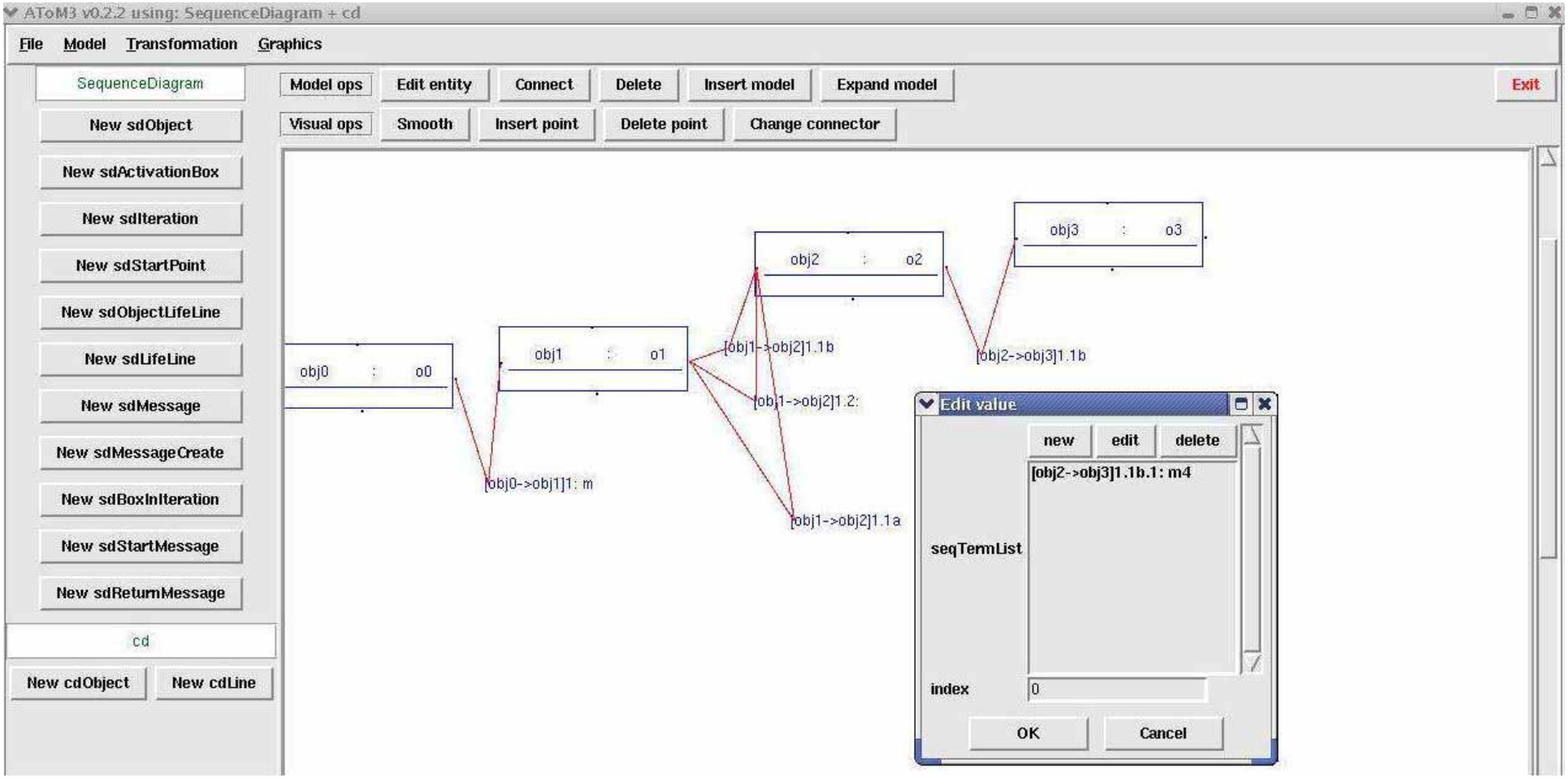
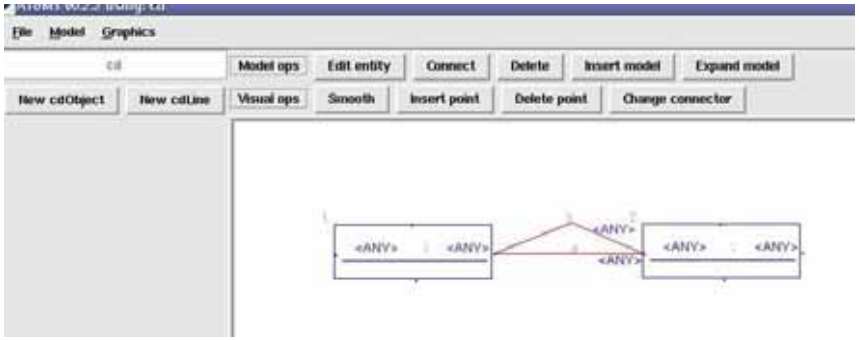
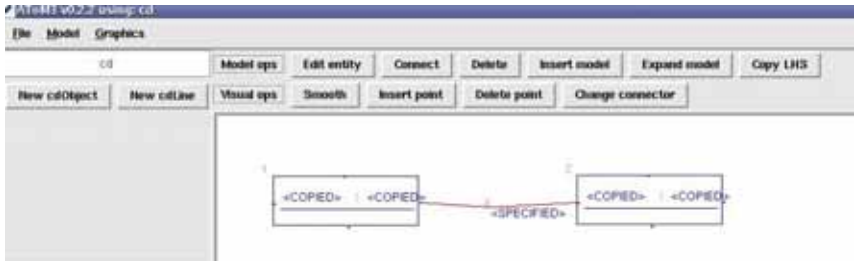


Figure 3 Example: the resulting diagram after the application of sg2cd to the diagram in figure 2

Transformation unit cd2cd1





cd2cd1_rule_1

RHS

Specify for cdLink4

seqTermList = seqTermList + cdLink3.seqTermList

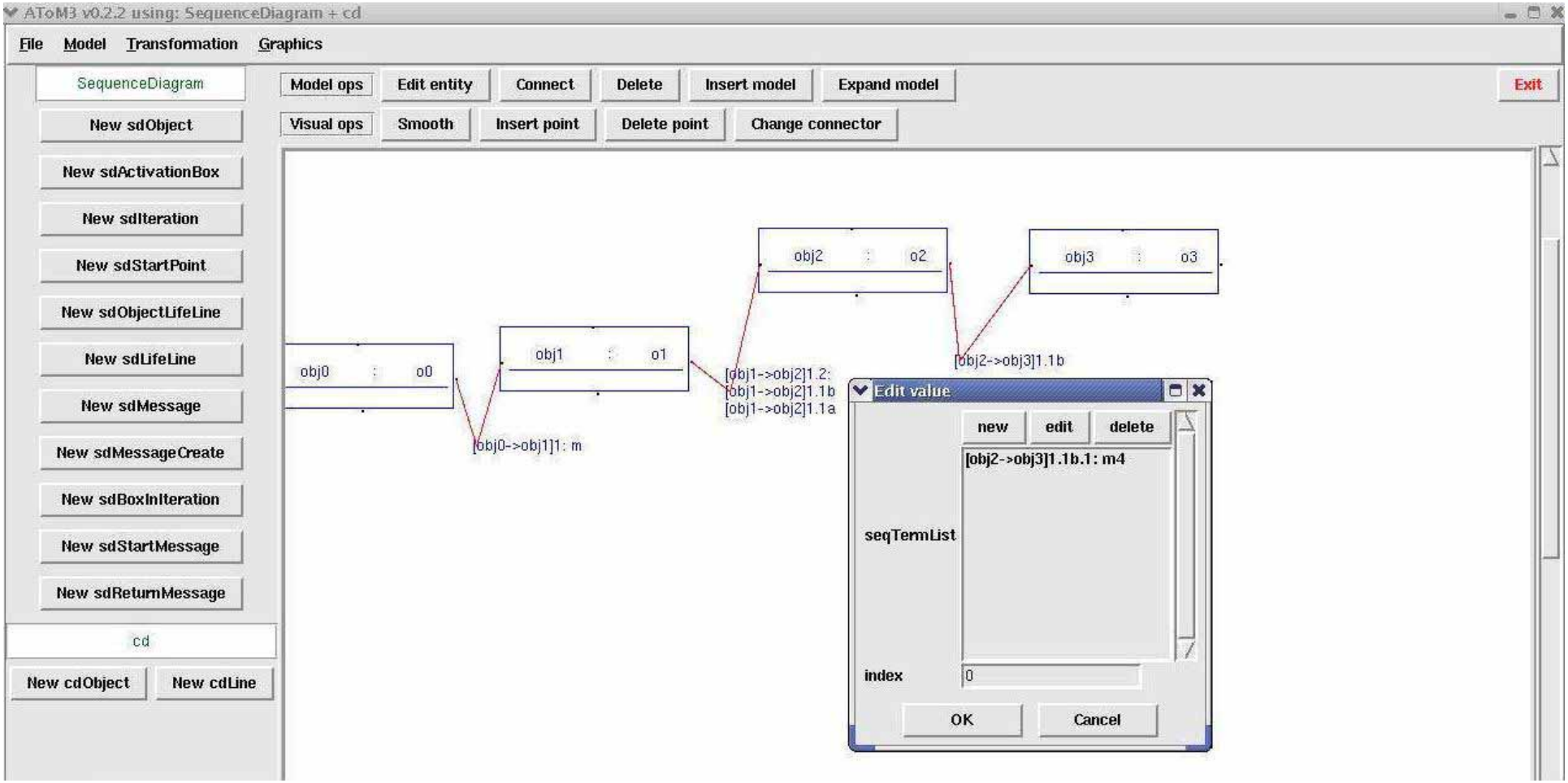


Figure 4 Example: the collaboration diagram after the application of cd2cd1 to the diagram in figure 3

Conclusion

- sd2sg has taken two cases into account
  - case 1: communication between the different objects
  - case 2: communication between the same objects
- sg2cd and cd2cd1 only deal with case 1
- As a result, this project can only deal with the case 1, but it’s very easy to add the rule into sg2cd and cd2cd1 into order to implement case 2