Graph Rewriting: Translation of Sequence Diagram into Collaboration Diagram

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

1. Foundation
    - 1.1 Interaction diagram (metamodel object diagram)
    - 1.2 Graph transformation

2. Translation of sequence Diagram into the Collaboration Diagram
    - 2.1 Overview
    - 2.2 Sequence graph (SG)
    - 2.3 Generate sequence graph
    - 2.4 Translation of sequence graph into metamodel object graph
    - 2.5 Collaboration graph
    - 2.6 Translation of metamodel object graph into collaboration graph

3. Conclusion

1.Foundations

- The language architecture of UML :

  4-layer metamodel structure, meta-metamodel, metamodel,model and user-objects layer. Sequence and collaboration diagram are placed in the model and user-objects layer.

- Translation uses those part of the metamodel that provide the means to specify interaction diagram, i.e metamodel object diagram

1.1 Interaction diagram
(metaModel object diagram MOG)

- The UML Notation Guide abstracts sequence and collaboration diagram as interaction diagram. Both are based on the same underlying info but emphasize on different aspects of behavior, which are collaboration and Interaction.

- Structural aspect of behavior

  (1) Collaboration (on specification level)

  (2) CollaborationInstanceSet (on instance level)

- The other aspect of behavior: Communication patten

  (1) Interaction (on specification level)

  (2) InteractionInstanceSet (on instance level)

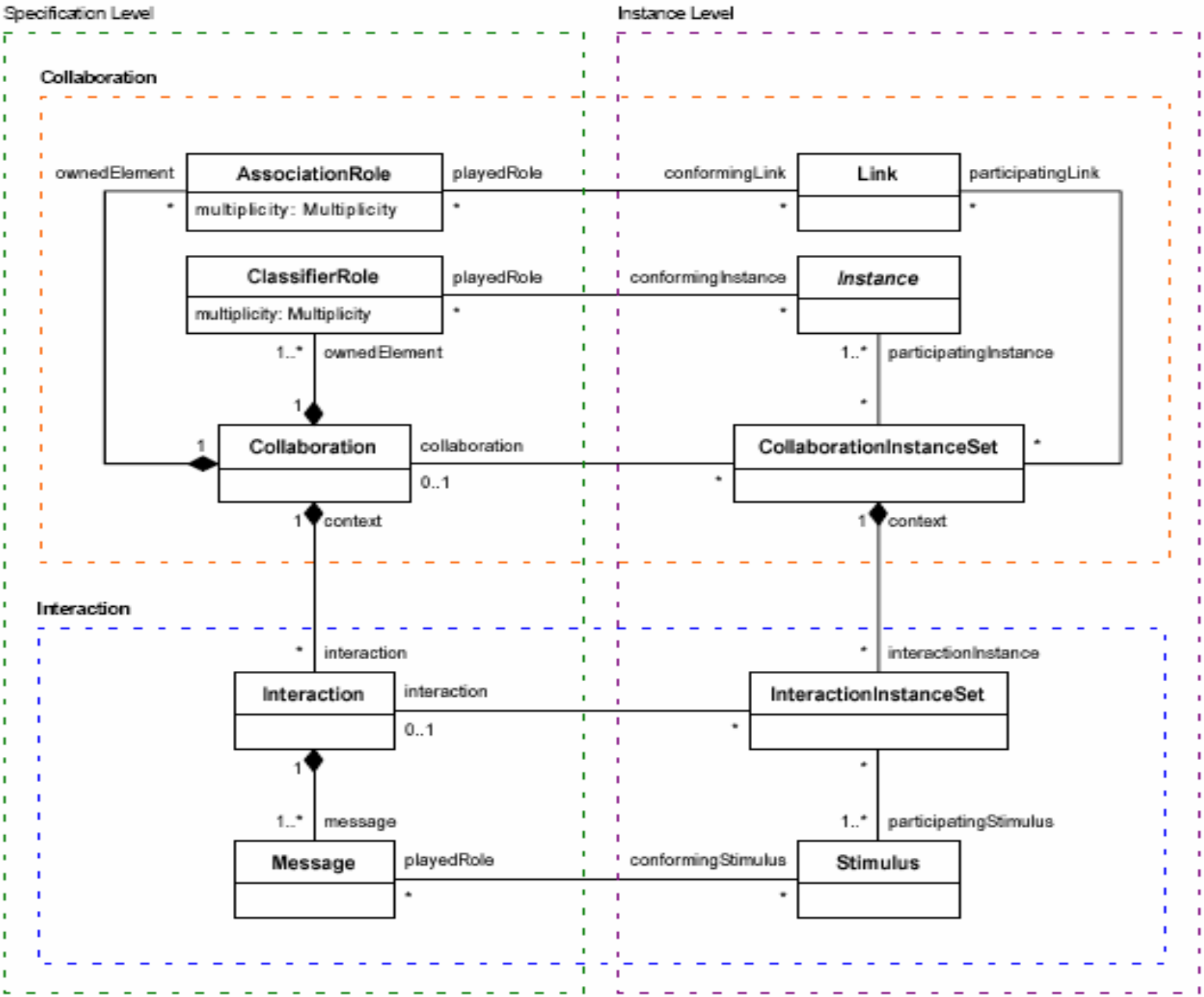


Figure 2.1: Collaboration and Interaction

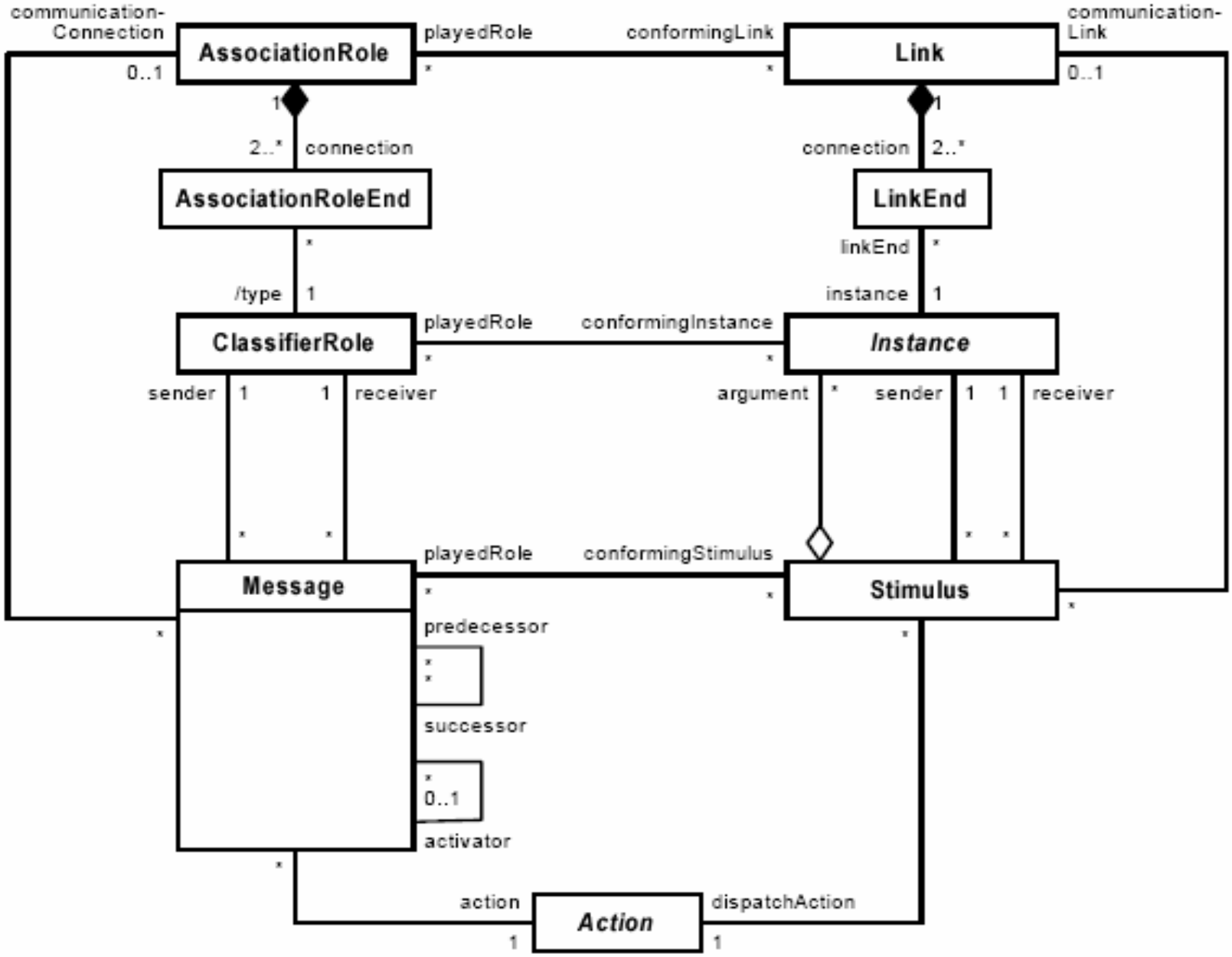


Figure 2.2: Messages and Stimuli

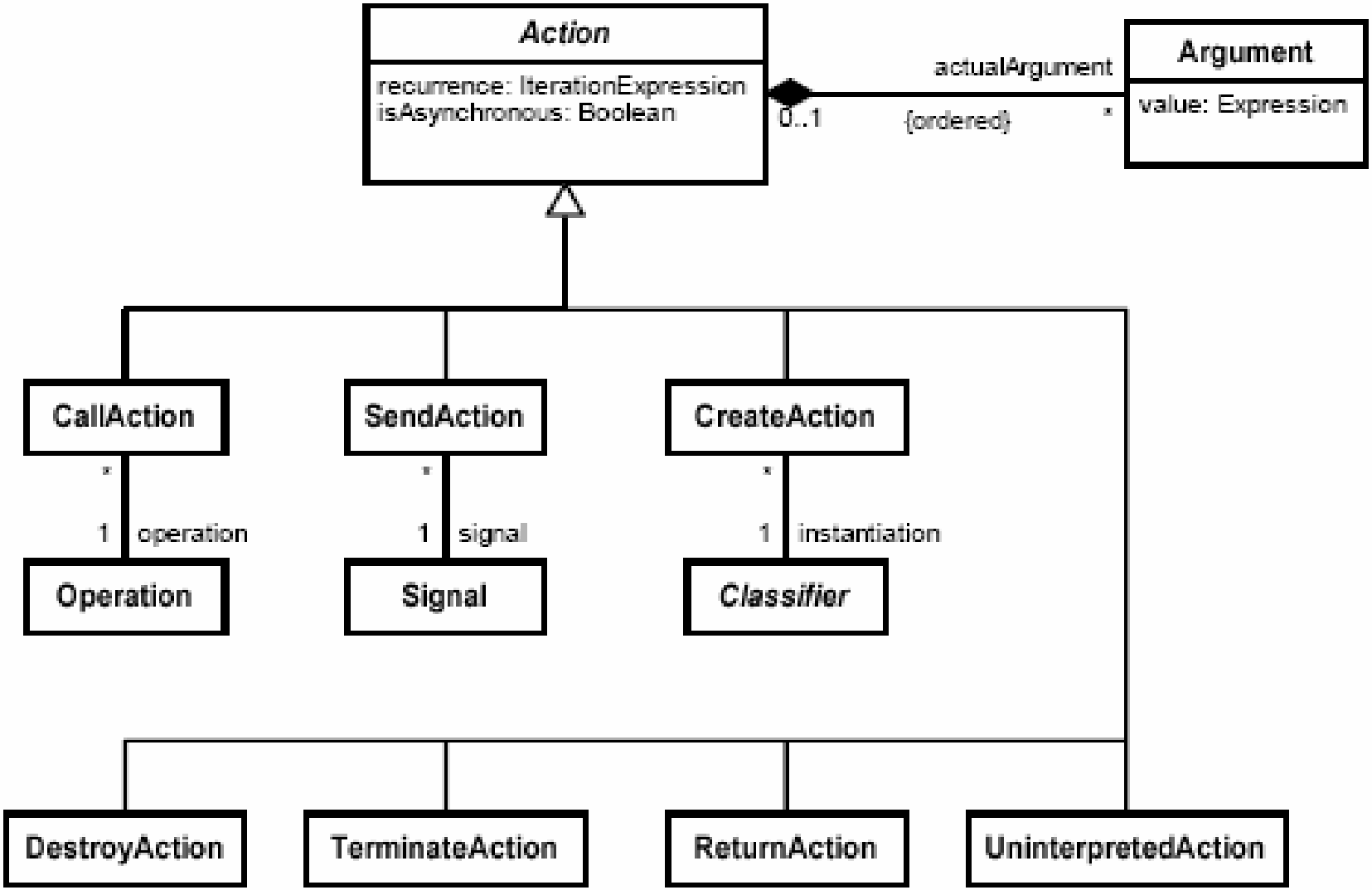


Figure 2.3: Actions

- Activator relationship: nested communication
- Predecessor relationship : sibling messages with same activator
- Sequence expression :
  describe the activator and predecessor relationships of a Message

  a) a dot-separated list of sequence terms that followed by a colon. Sequence term consists of an integer possibly followed by a lower case letter

  b) Each term of the list represents a level of nesting within the interaction

  c) Example:
  --- message 1: is the activator of message 1.1: and 1.2:
  --- message 1.1: is the predecessor of message 1.2:
  --- message 1.3a: and 1.3b: corresponds to two branches of the conditional branch

1.2 Graph transformation

- Graph transformation rule:

  Replace parts of a given graph with another graph, based on a set of rules.

  (1) A rule r = (L,R,p)

  L: left-hand side graph; R: right-hand side graph; p: L->R a partial morphism

  (2) Negative Application Graph:

  describe a situation that's not wanted in the host graph

  All the items of N that aren't part of l(L) represents the forbidden structure

- Transformation unit tu = (I, U, R, C,T)

  I : describe the initial graph

  U: a finite set of imported transformation unit

  R: a finite set of graph transformation rules

  T: describe the terminal graph

  C: control condition

- Attributed graph

  Attribute is a triple consisting of the type, the name, the value of the attribute.

  can be assigned to both nodes and edges of a graph

2. Translation of Sequence Diagrams into Collaboration Diagram

2.1 Overview

GenerateSG: generate the graph representation of sequence diagram

SG2MOG: operate on SG and generate MOG

MOG2CG: operate on MOG and generate CG

CG2CD: translate the CG into CD

2.2 Sequence Graph (SG)

- One node type: SGObject

- Eight edge type:
  group 1: denote the communication
  SGStimulus, SGReturn, SGVisited

  group 2: marker in the process of translating the Interaction
  Current: attr index is used to number the message in object graph
  attr activatorStack stores and indicate the corresponding activator msg
  SGIf: used to mark the beginning of the conditional branching
  attr join specifies where the different branches join
  EndIf: used to mark the end of conditional branching
  End: indicate the end of the communication sequence

  group 3: Activation
  Activation path: same multiple SGObject nodes and activation edge
  incoming SGStimulus: activator message
  outcoming SGStimulus: pairly related to each other througth predecessor relationship

2.3 generate sequence graph

- GenerateSG
  init: initial sequence graph
  rules: GenerateSG_R1 insert Stimulus and Return
  GenerateSG_R2 insert conditional branching
  GenerateSG_R3 add another branch
  -- note: no control condition, the transformation order depends on the sequence order of the message in the sequence diagram

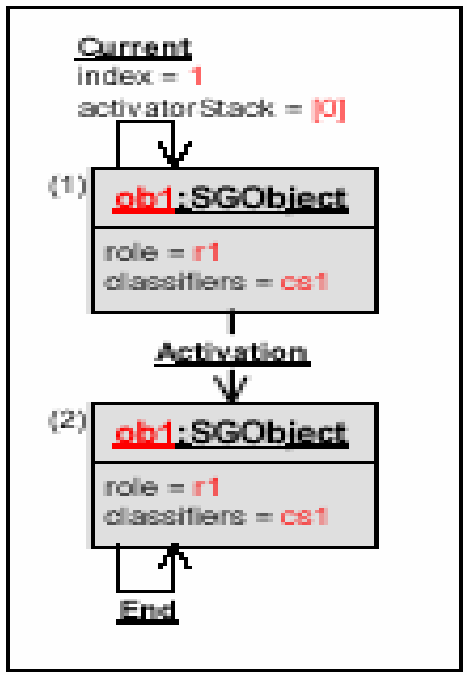


Figure 3.8: Initial Sequence Graph

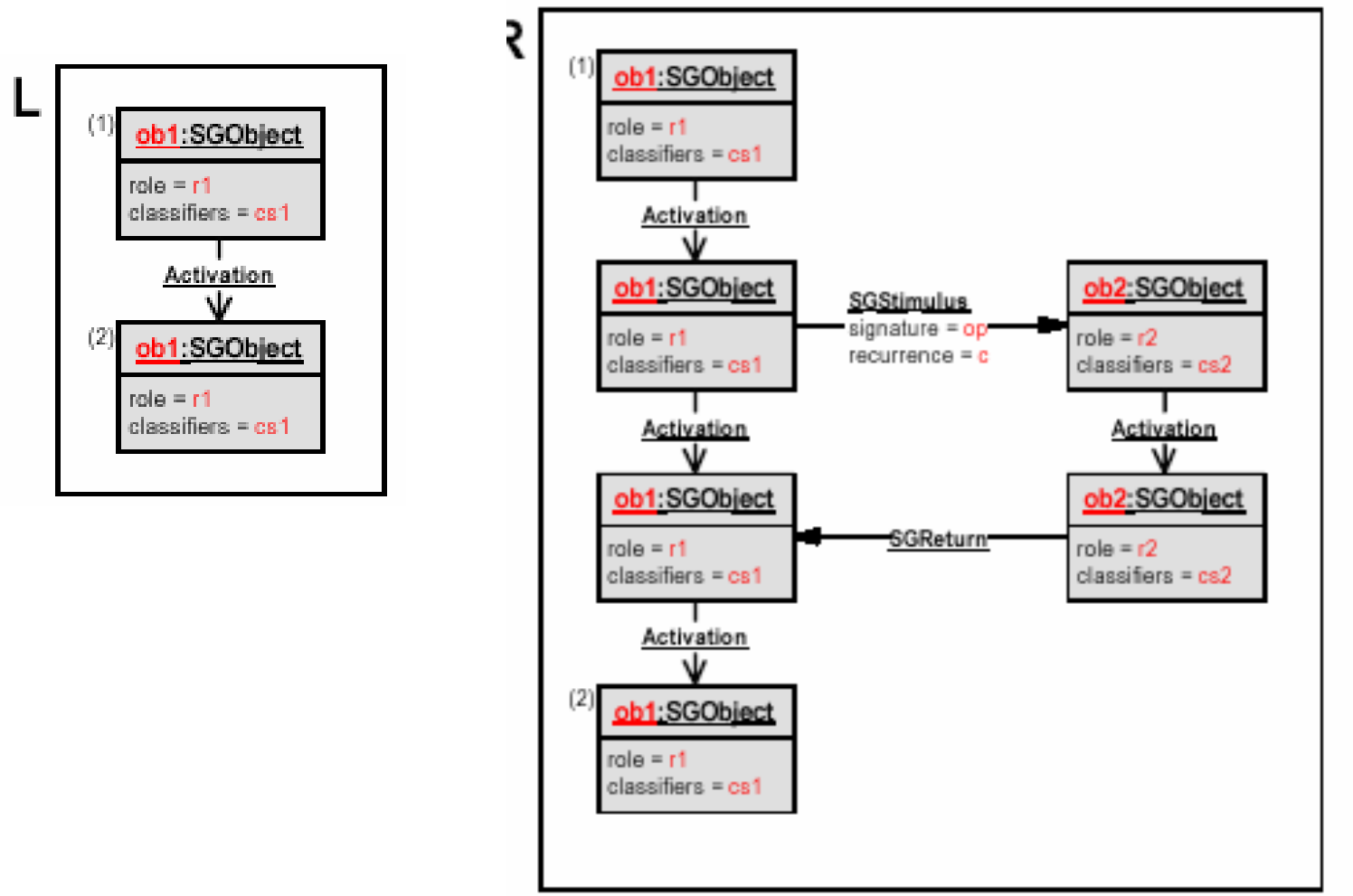


Figure 3.9: GenerateSG_R1: insert an SGStimulus and a matching Return

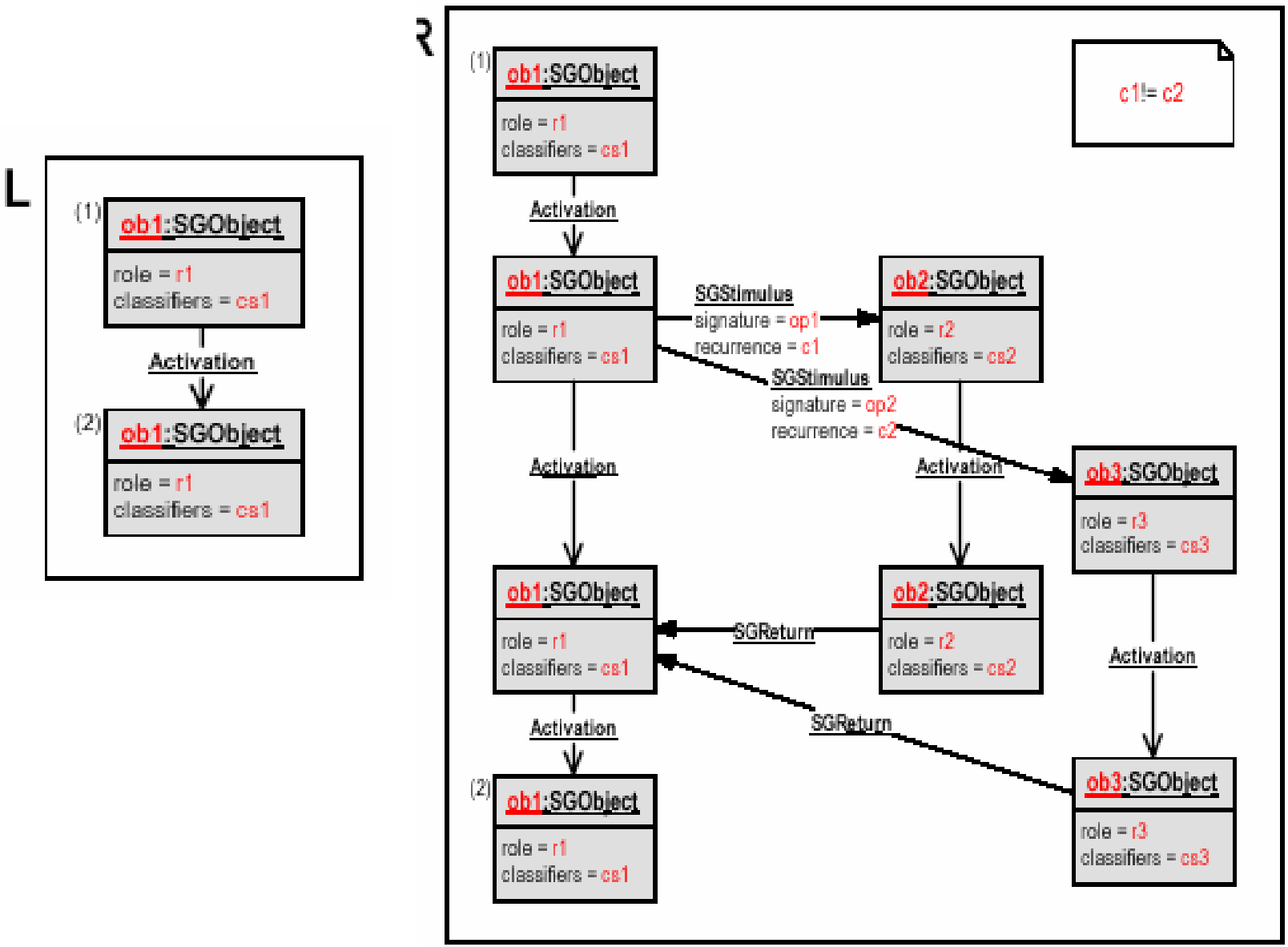


Figure 3.10: GenerateSG_R2: insert a conditional branching with two branches

3. Example : Moving the queen

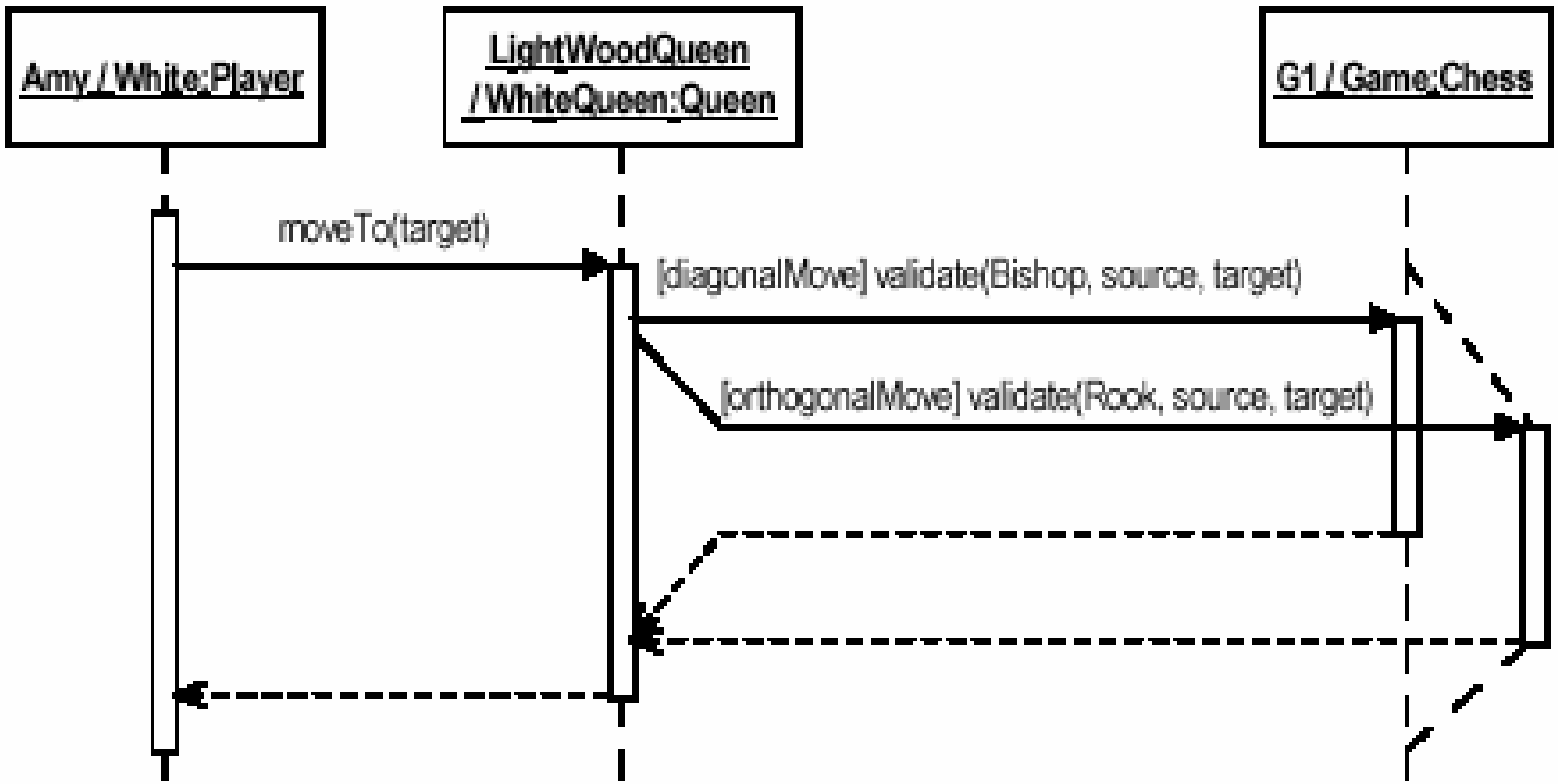


Figure 4.9: Moving the Queen: sequence diagram

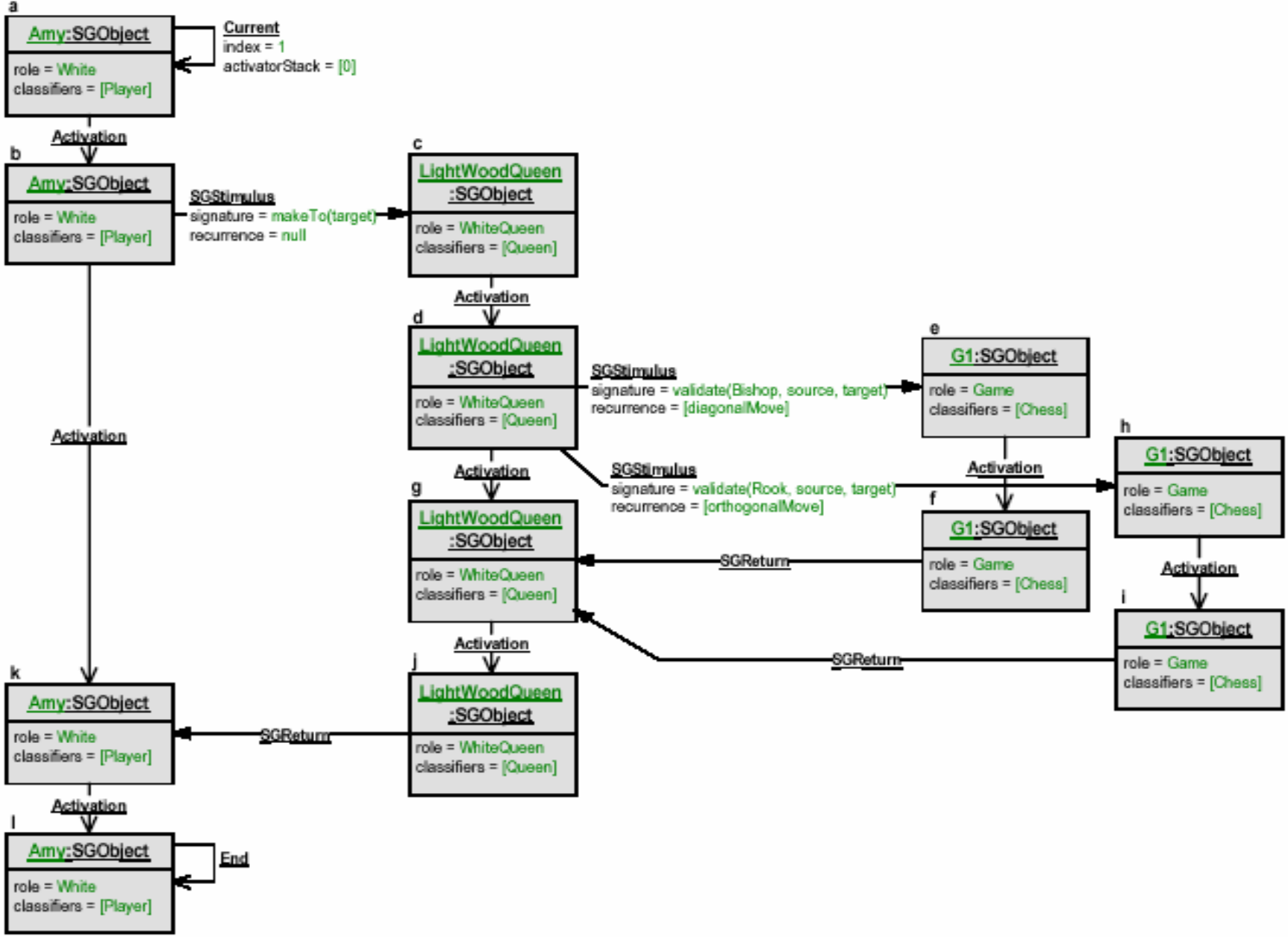


Figure 4.10: Moving the Queen: sequence graph before the application of **SG2MOG-Interaction**

2.4 Translation of Sequence Graph into Metamodel Object Graph (MOG)

SG2MOG

rules: SG2MOG_R1

uses: SG2MOG_Collaboration
SG2MOG_Interaction
DeleteSG

cond: SG2MOG_R1;
SG2MOG_Collaboration;
SG2MOG_interaction;
DeleteSG

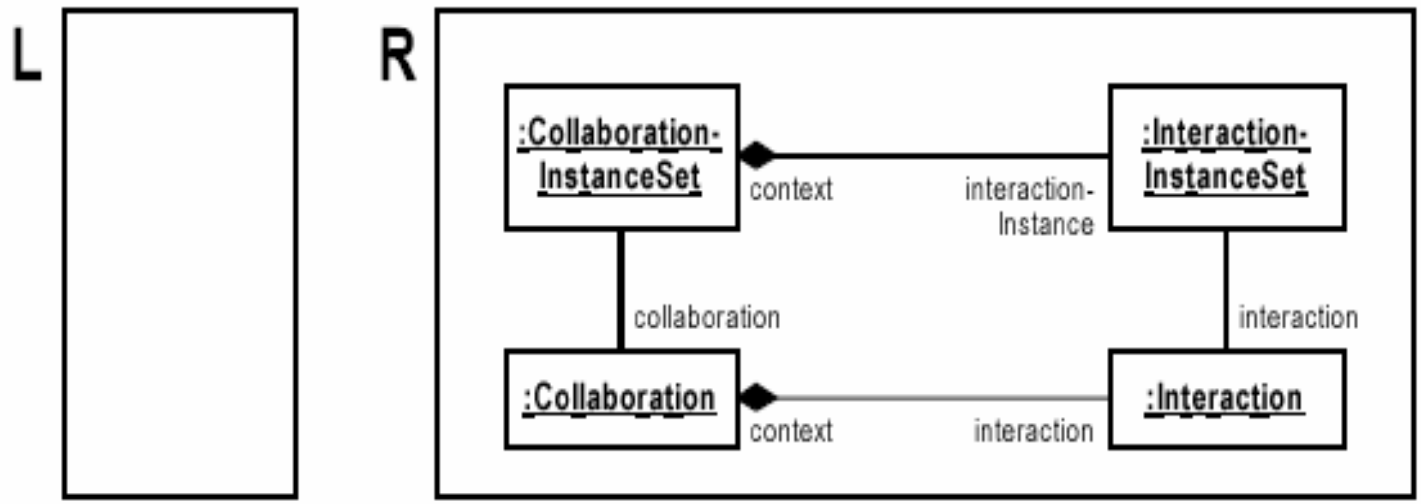


Figure 3.14: Rule SG2MOG_R1: Generating the initial metamodel object graph

- SG2MOG_Collaboration

  rules: Adding ClassifierRoles and conforming object to Collaboration
  SG2MOG_Coll_R1a: add new ClassifierRole and conforming Object
  SG2MOG_Coll_R1b: add new ClassifierRole
  Adding base Classifier to a ClassfierRole
  SG2MOG_Coll_R2a: add new base classifier
  SG2MOG_Coll_R2b: link existing base Classifier
  SG2MOG_Coll_R2c: end base Classifier addition
  Add root Message to Interaction
  SG2MOG_Coll_R3: add root Message to Interaction

  cond: ((SG2MOG_Coll_R1a|SG2MOG_Coll_R1b);
  (SG2MOG_Coll_R2a|SG2MOG_Coll_R2b)!;
  SG2MOG_Coll_R2c)!; SG2MOG_Coll_R3

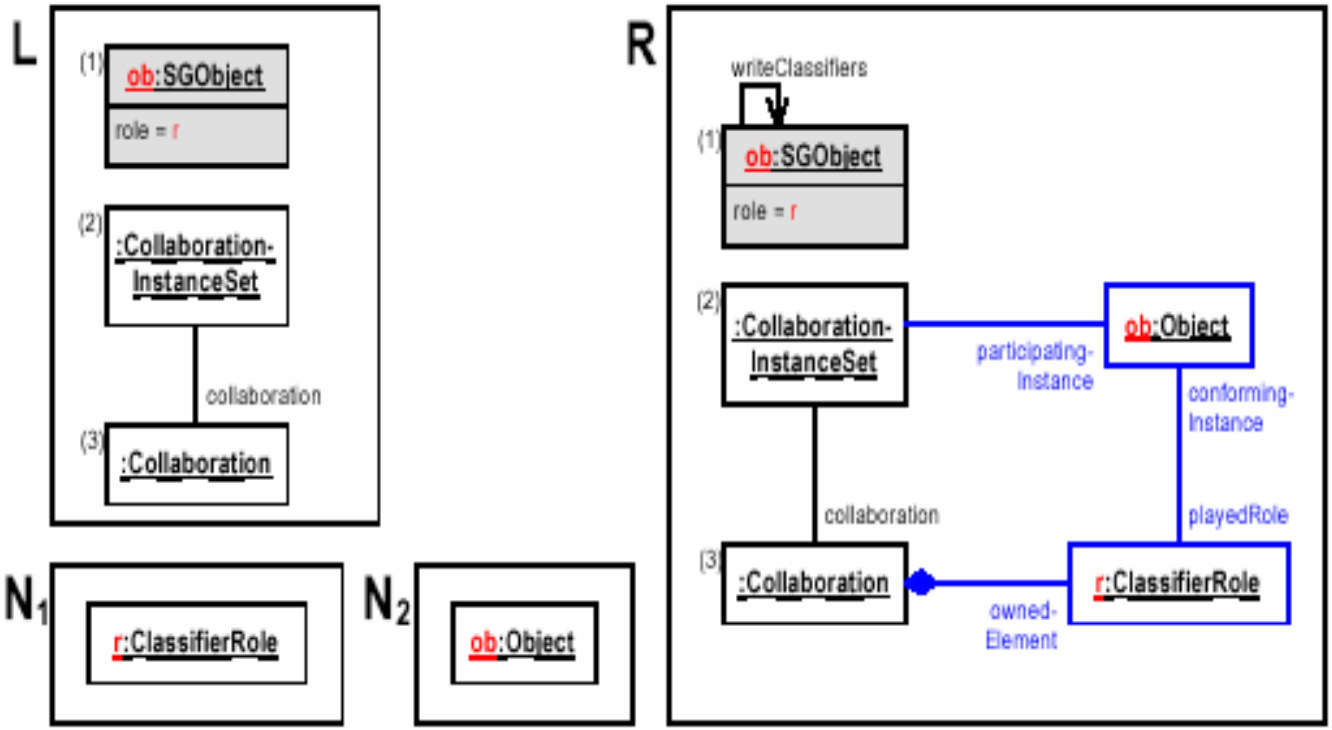


Figure 3.16: Rule SG2MOG_Coll_R1a: addition of a new ClassifierRole and a new Object

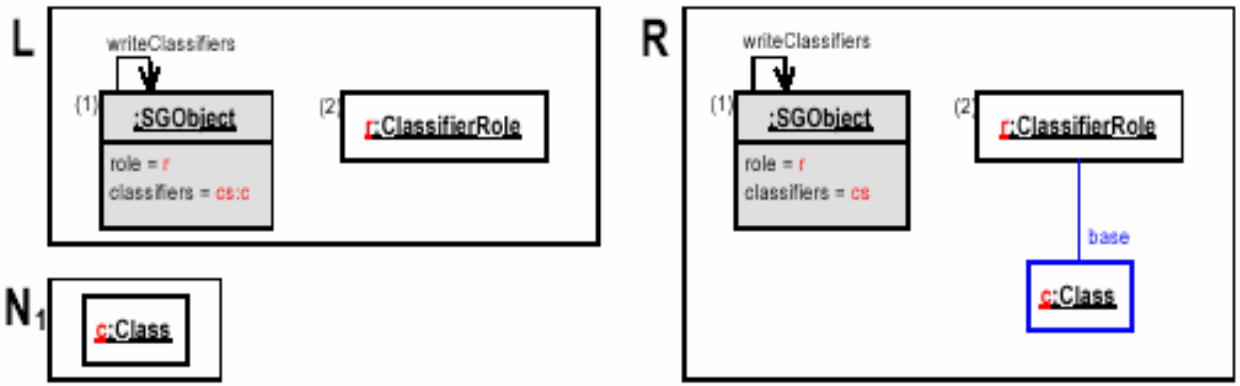


Figure 3.19: Rule SG2MOG_Coll_R2a: adding a new base Classifier and attaching it to the ClassifierRole in consideration

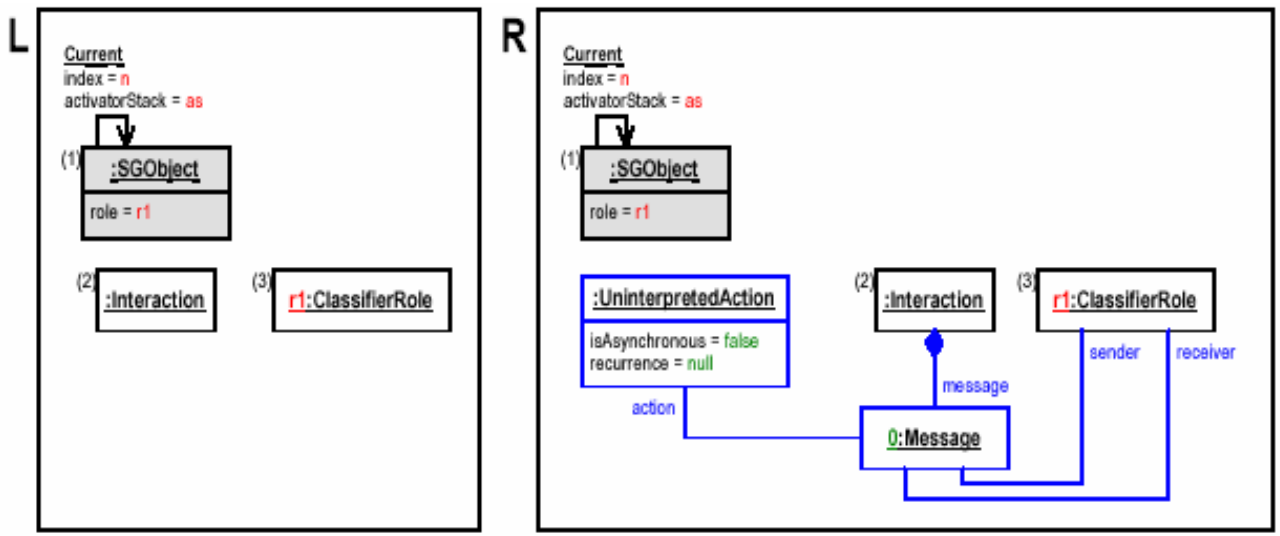


Figure 3.24: Rule SG2MOG_Coll_R3: addition of the root Message to the Interaction

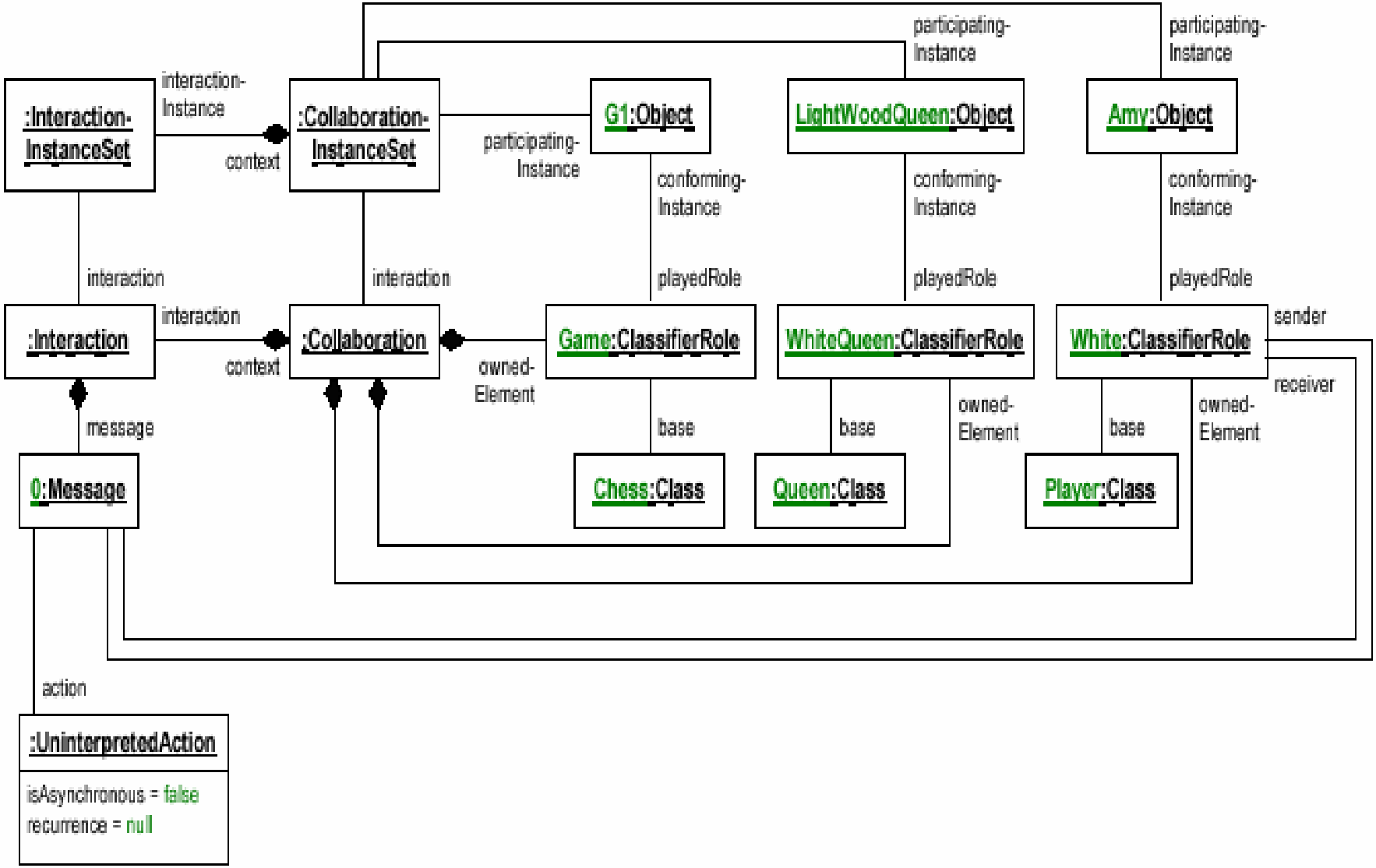


Figure 4.11: Moving the Queen: metamodel object graph after the application of SG2MOG_Collaboration

- SG2MOG_Interaction

  rules: Adding a Stimulus

  SG2MOG_Int_R1a: add Stimulus without predecessor

  SG2MOG_Int_R1b: add Stimulus with predecessor

  Adding a Return

  SG2MOG_Int_R2a: add Return without predecessor

  SG2MOG_Int_R2b: add Return with predecessor

  Following Activation Path

  SG2MOG_Int_R3a: prepare translation of a branching

  SG2MOG_Int_R3b: move along activation path

  Translate a Branch

  SG2MOG_Int_R4a: translate branch without predecessor

  SG2MOG_Int_R4b: translate branch with predecessor

  End of Branch

  SG2MOG_Int_R5a: completion of a branch

  SG2MOG_Int_R5b: completion of a branching

Note: control condition isn't necessary, The rules are designed to process the sequence graph in depth first order

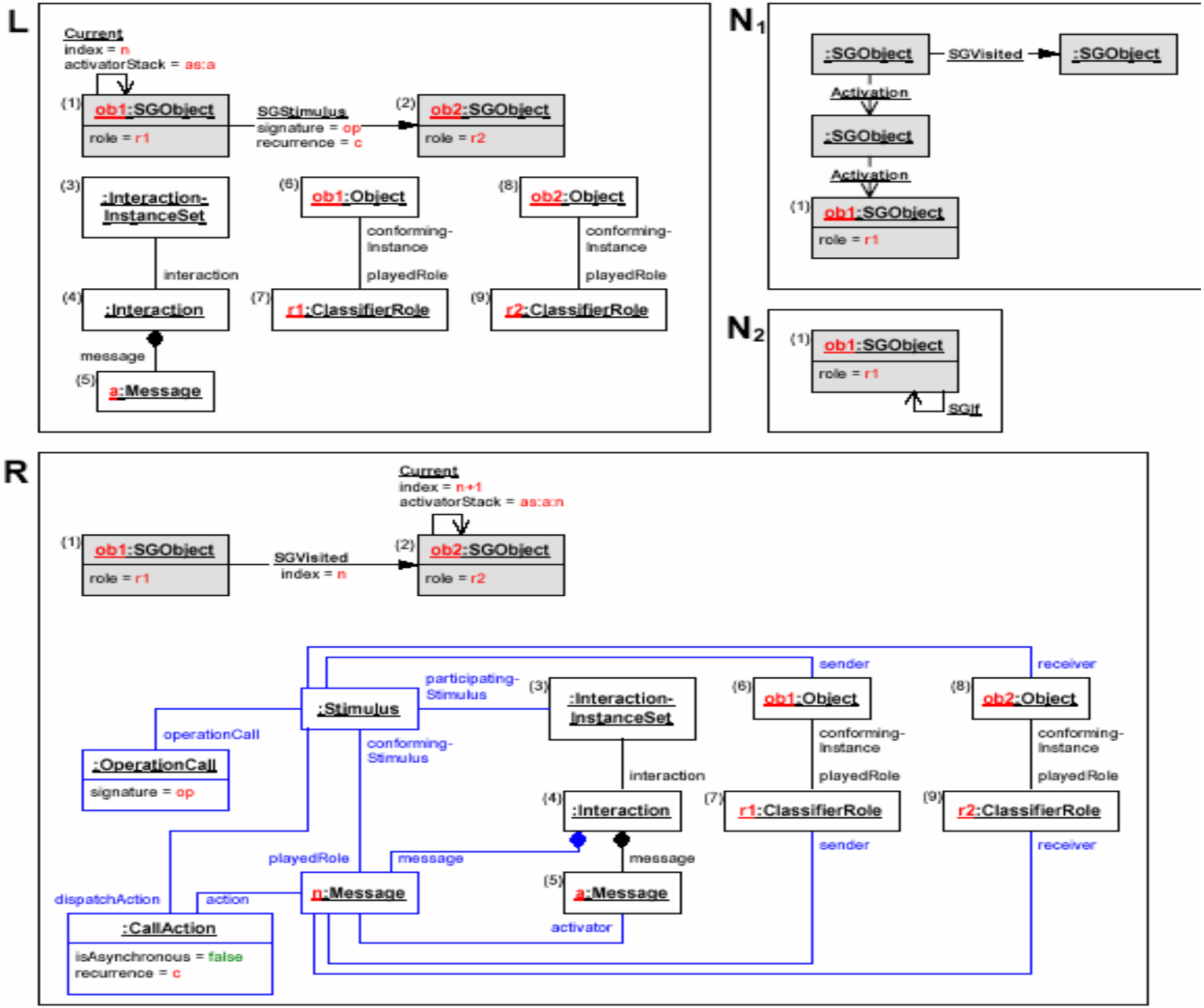


Figure 3.26: Rule SG2MOG_Int_R1a: Stimulus without predecessor

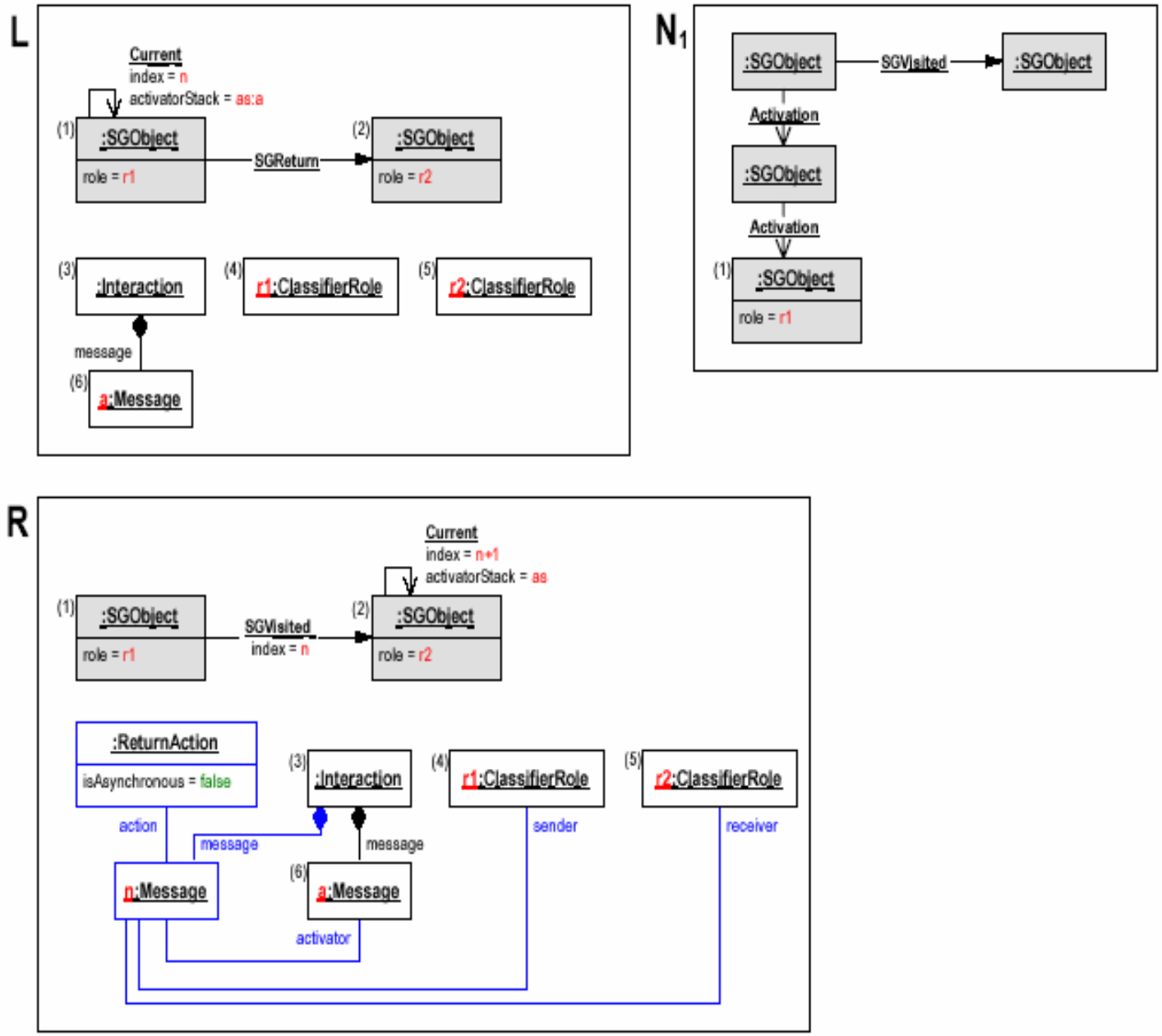
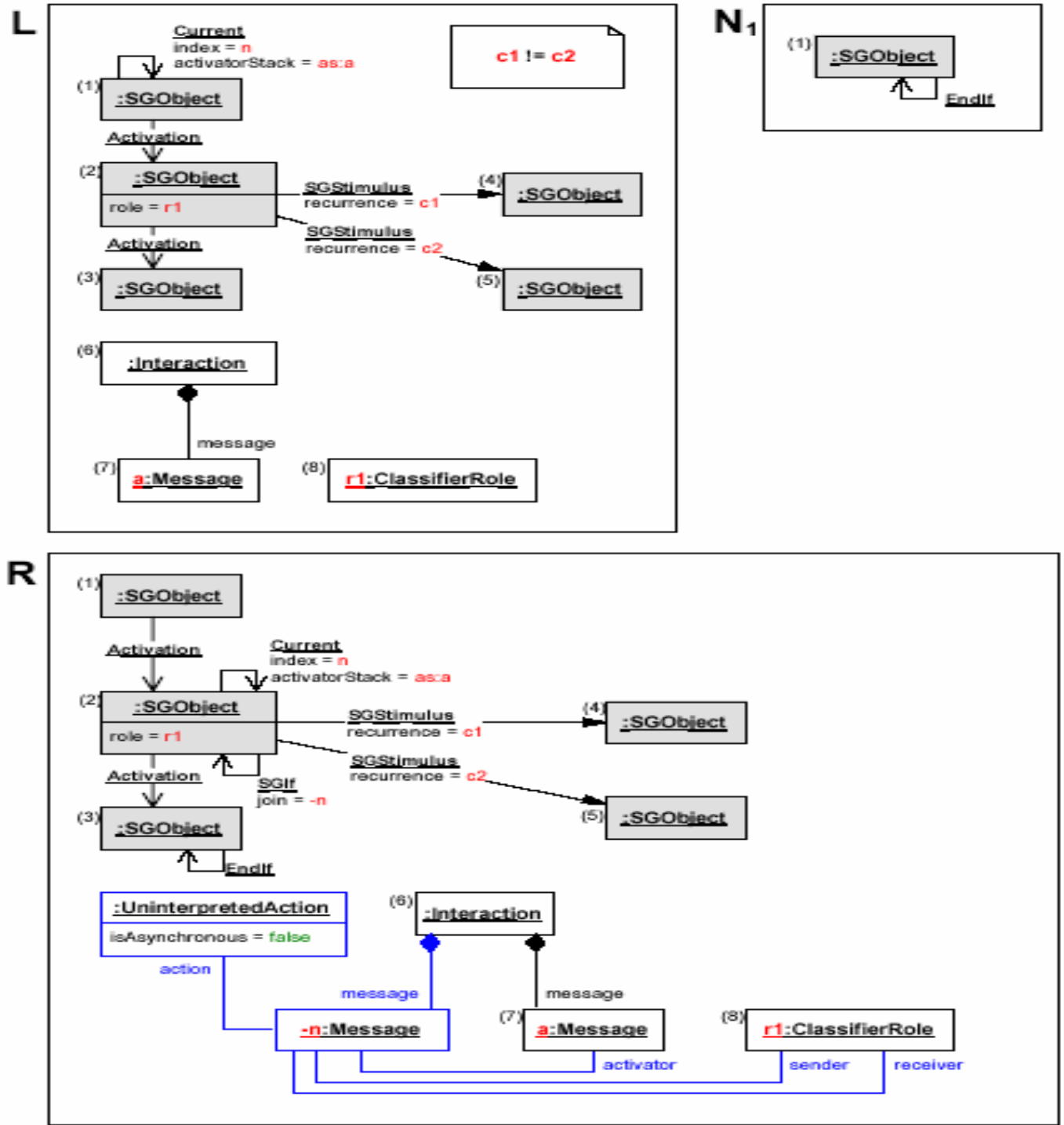


Figure 3.29: Rule SG2MOG_Int_R2a: Return without predecessor

Figure 3.33 Rule SG2MOG_Int_R3a: preparation for translating a ranchin



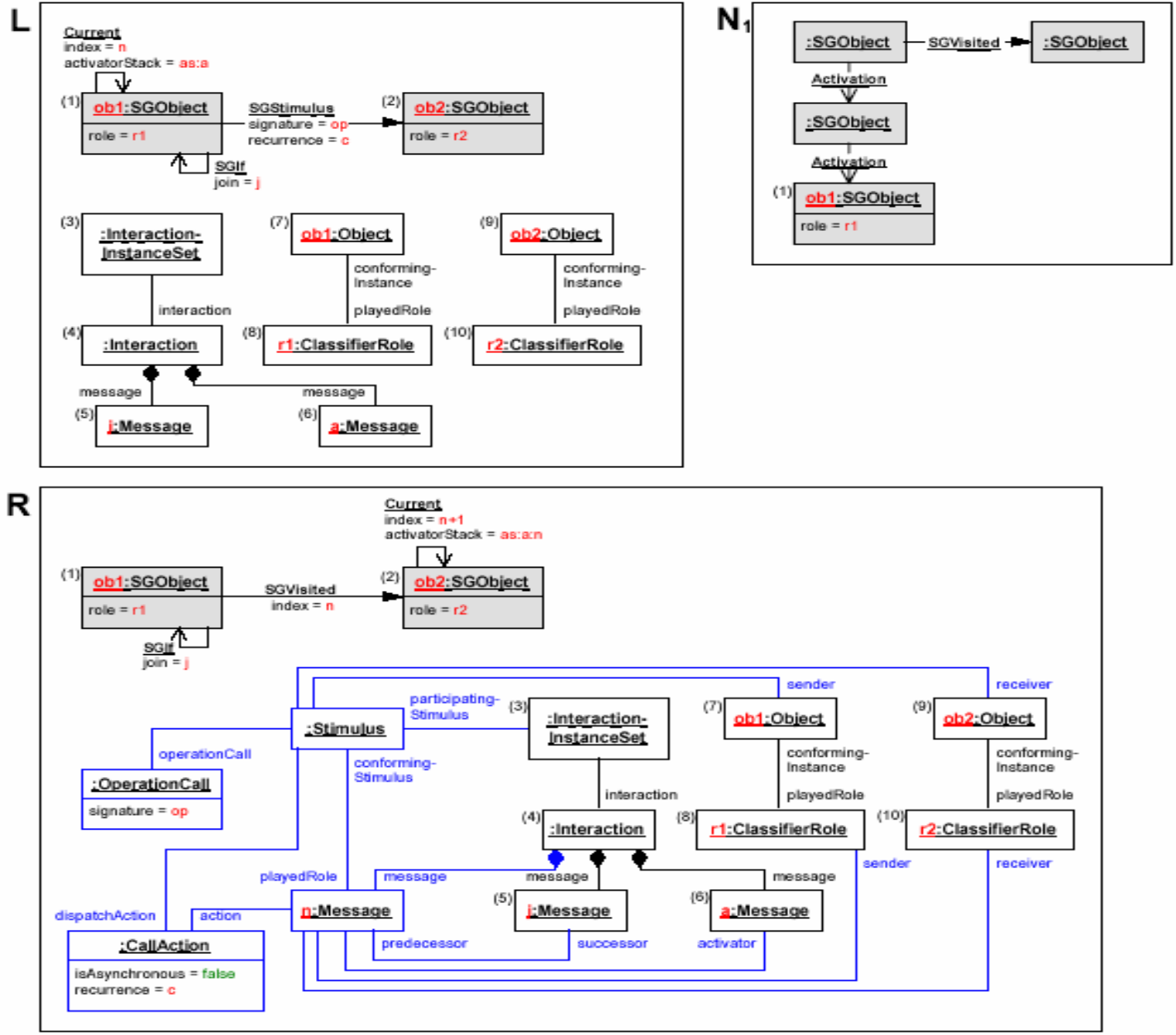


Figure 3.38: Rule SG2MOG_Int_R4a: conditional branching without predecessor

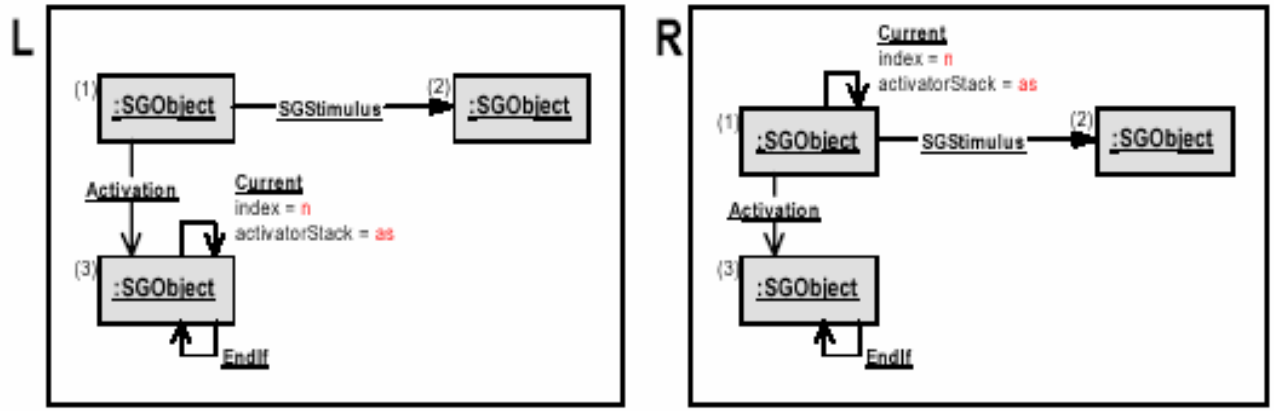


Figure 3.41: Rule SG2MOG_Int_R5a: completion of a branch and return from its end to the beginning of the next branch

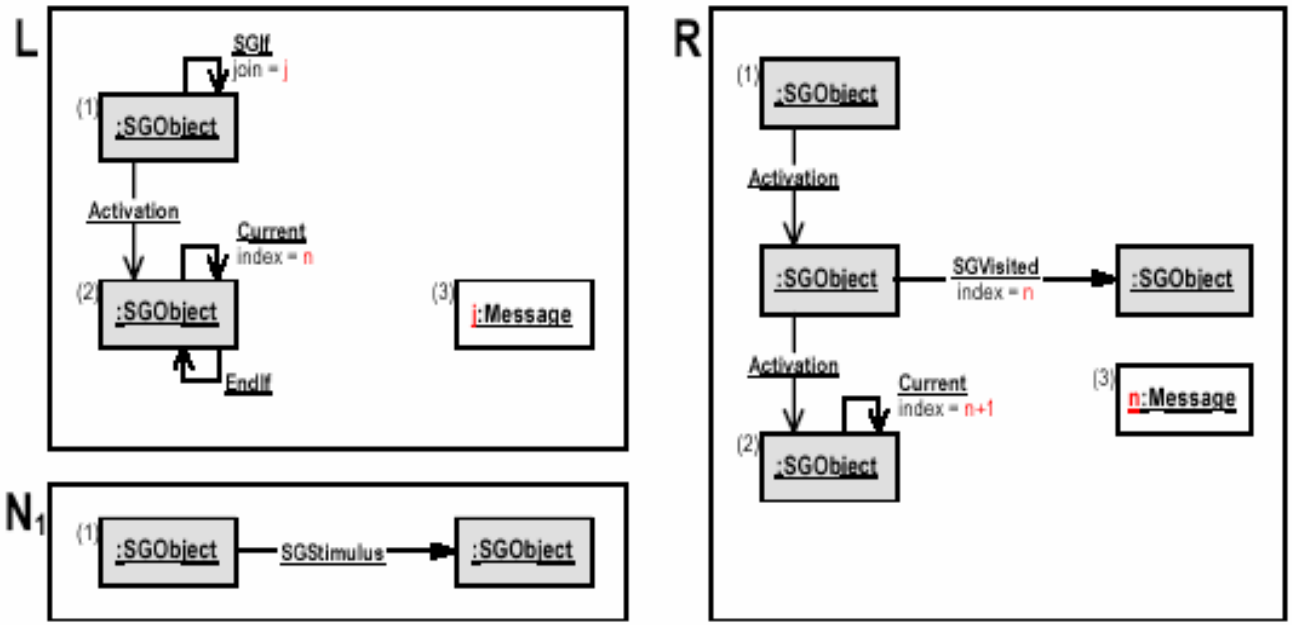


Figure 3.42: Rule SG2MOG_Int_R5b: completion of a branching

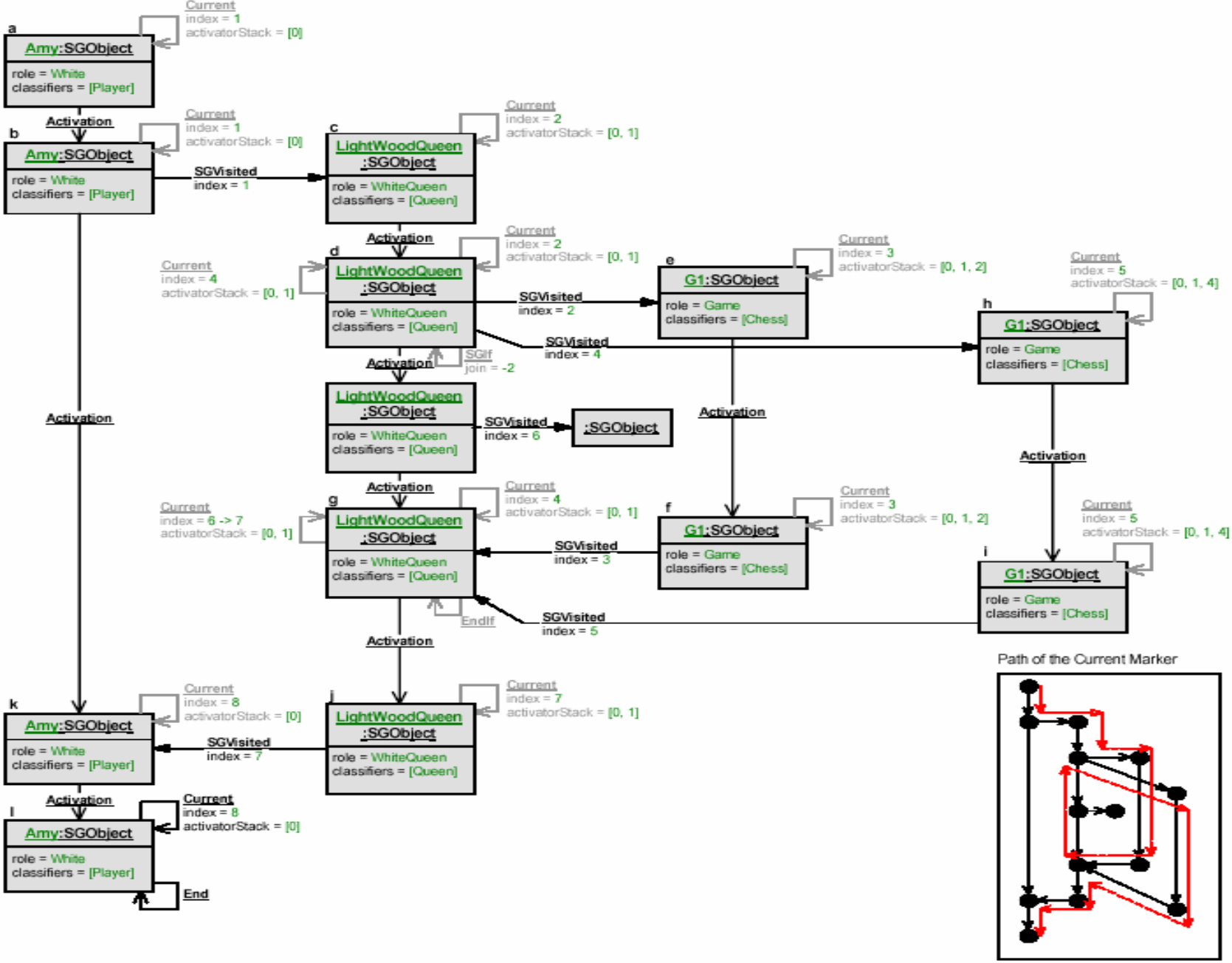


Figure 4.12: Moving the Queen: sequence graph after the application of **SG2MOG_Interaction**

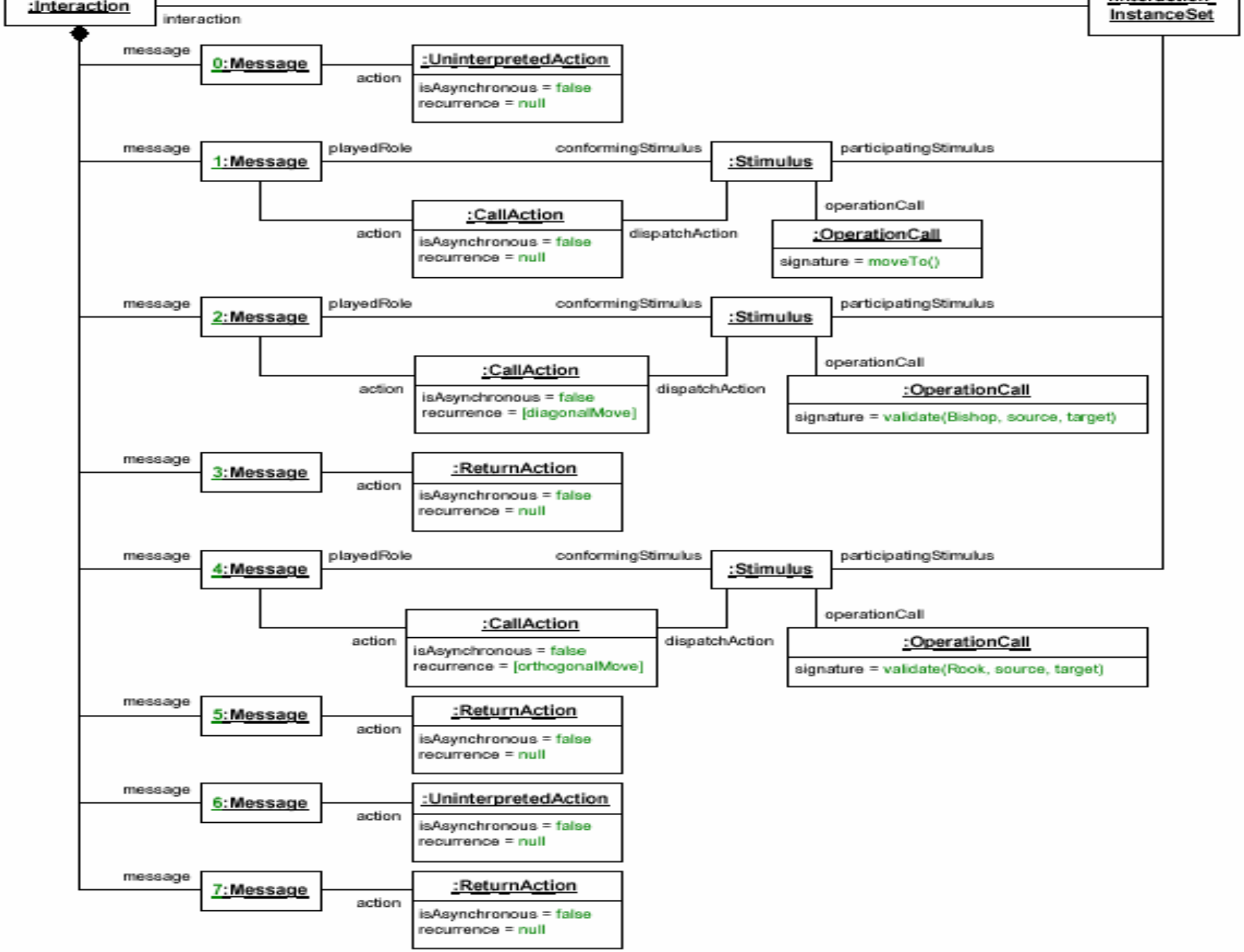


Figure 4.13: Moving the Queen: Messages, Stimuli, Actions, and OperationCalls

2.5 Collaboration Graph (CG)

- Node type: CGObject (correspond to the SGObject)
- Edge type:
  1. CGStimulus (correspond to the SGStimulus):
     signature
     recurrence
     seqExpr: impose an order on the stimuli
     a stack of sequence term (st#, st@)

  2. CGReturn (correspond to the SGReturn)
     seqExpr

2.6 Translation of Metamodel Object Graph into Collaboration Graph

MOG2CG

uses: MOG2CG_Collaboration
MOG2CG_Interaction
DeleteMOG

cond: MOG2SG_Collaboration; MOG_Interaction; DeleteMOG

- MOG2CG_Collaboration

  Rules: Adding Objects to the Collaboration Graph
  MOG2CG_Coll_R1: add new Object
  Assignment of the Base Classifiers
  MOG2CG_Coll_R2a: assign a new base classifier
  MOG2CG_Cloll_R2b: completion of the assignment
  Set Current Edge
  MOG2CG_Coll_R3: set Current edge

  Cond: (MOG2CG_Coll_R1; MOG2CG_Coll_R2a!;
  MOG2CG_Coll_R2b)!;
  MOG2CG_Coll_R3

- The resulting graph:
  (1) a MOG and CG which consists solely of nodes
  (2) a current edge marks the message 1 of MOG
  attributes: index
  seqTermList for numbering the CGStimulus edges

- MOG2CG_Interation

  Rules: Adding a Stimulus

  MOG2CG_Int_R1: add Stimulus

  Adding a Return

  MOG2CG_Int_R2: add Return

  Go to next Message

  MOG2CG_Int_R3: go to successor Message

  Translate a Branch

  MOG2CG_Int_R4a: translate branch

  MOG2CG_Int_R4b: go to join Message

  End of a Branch

  MOG2CG_Int_R5a: completion of a branching

  MOG2CG_Int_R5b: completion of a branching

*note: the translation is achieved by traversing the activator tree in depth-first order. The translation is completed when msg 0 is marked as current

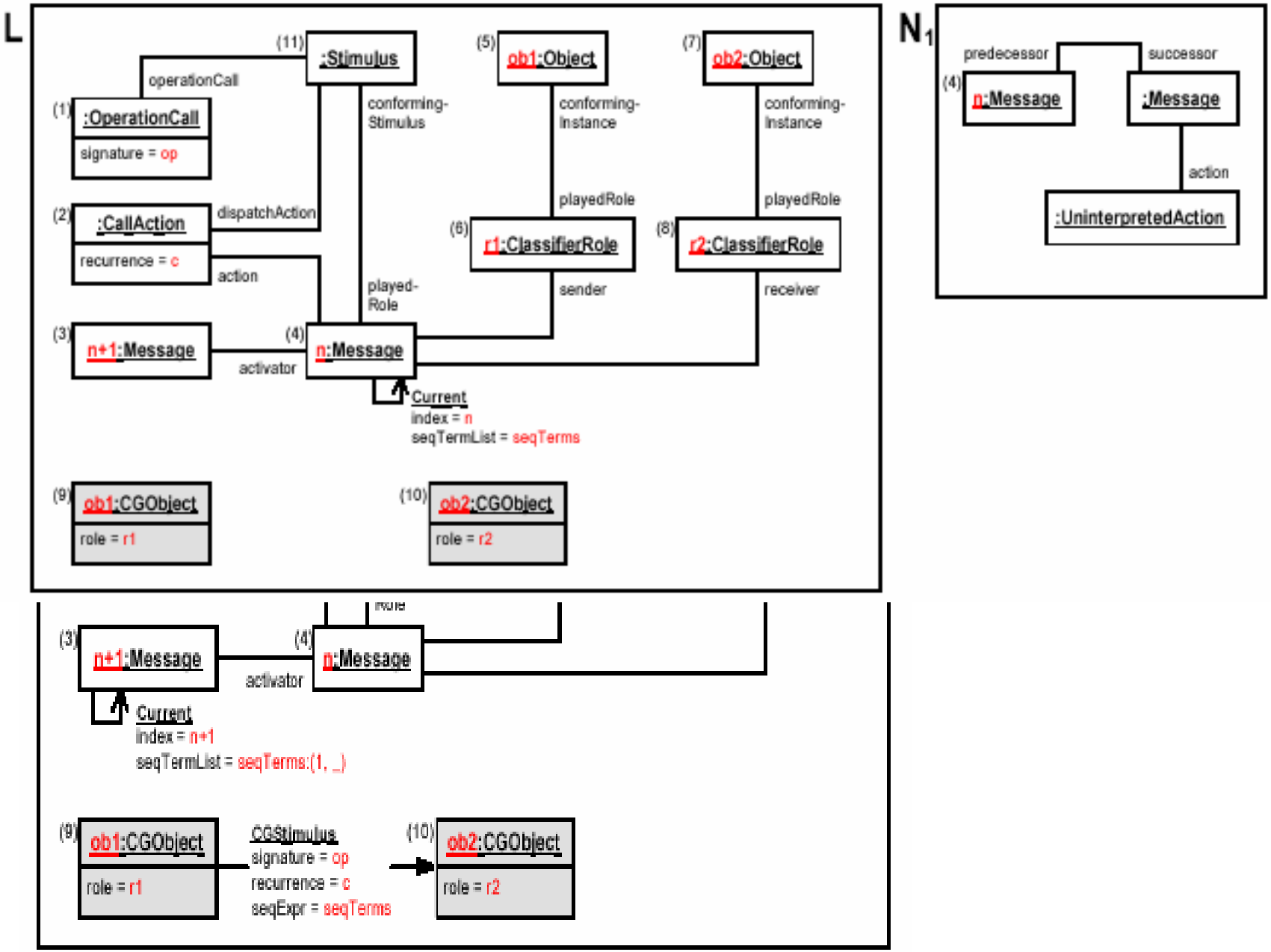


Figure 3.53: Rule MOG2CG_Int_R1: translation of a Stimulus representing a procedure call

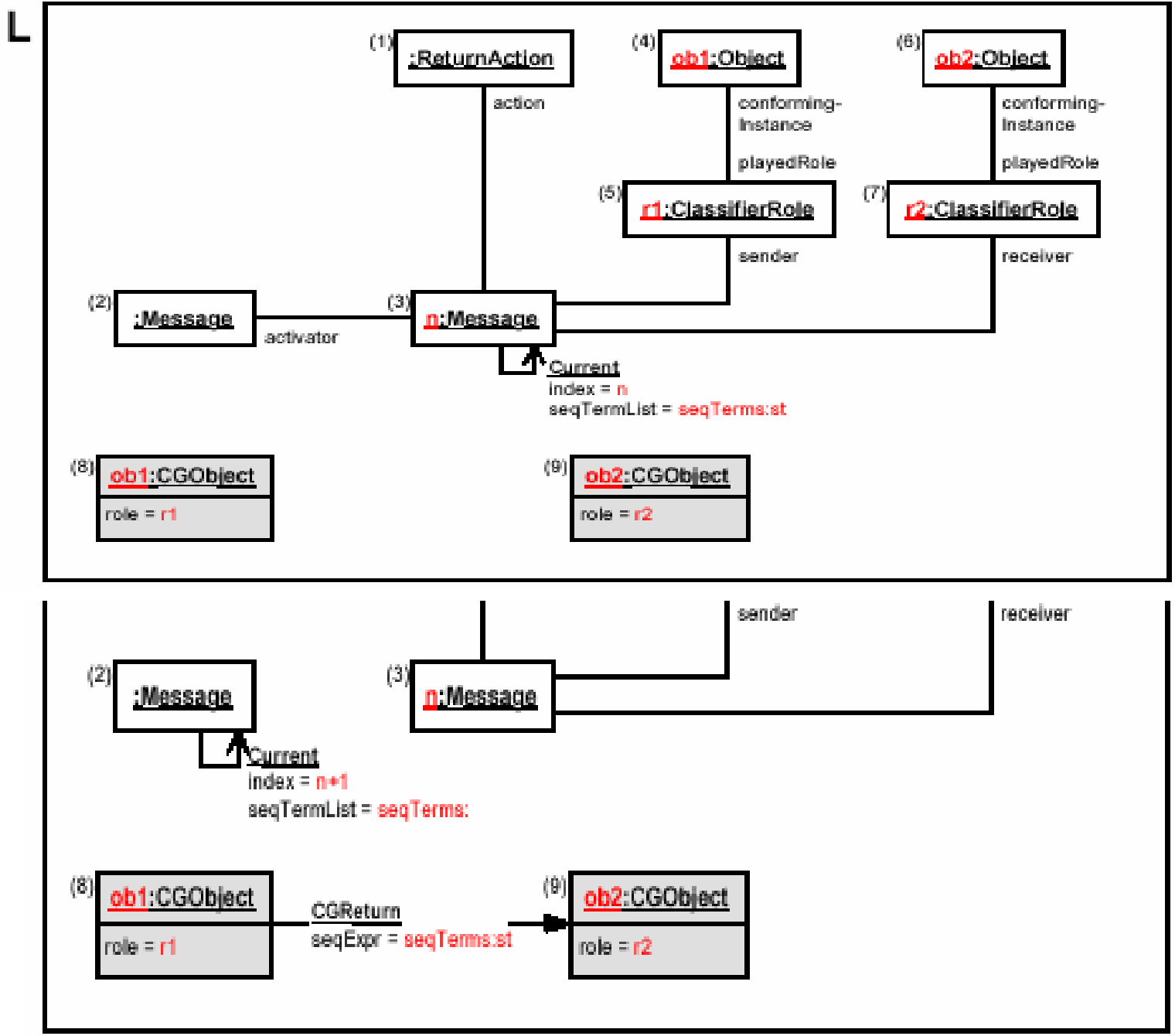


Figure 3.55: Rule MOG2CG_Int_R2: Return from a procedure call

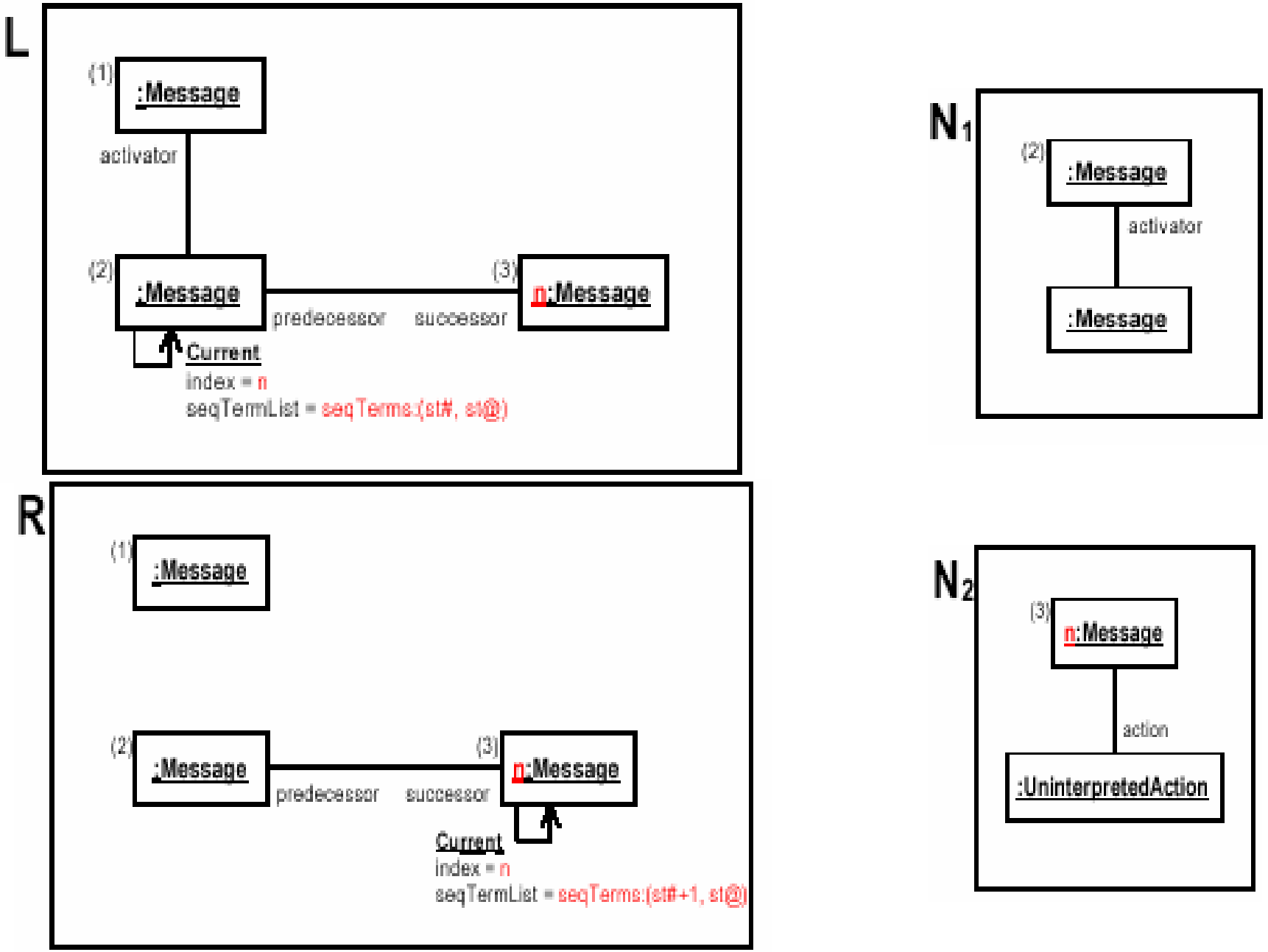


Figure 3.57: Rule MOG2CG_Int_R3: tracking to the successor Message

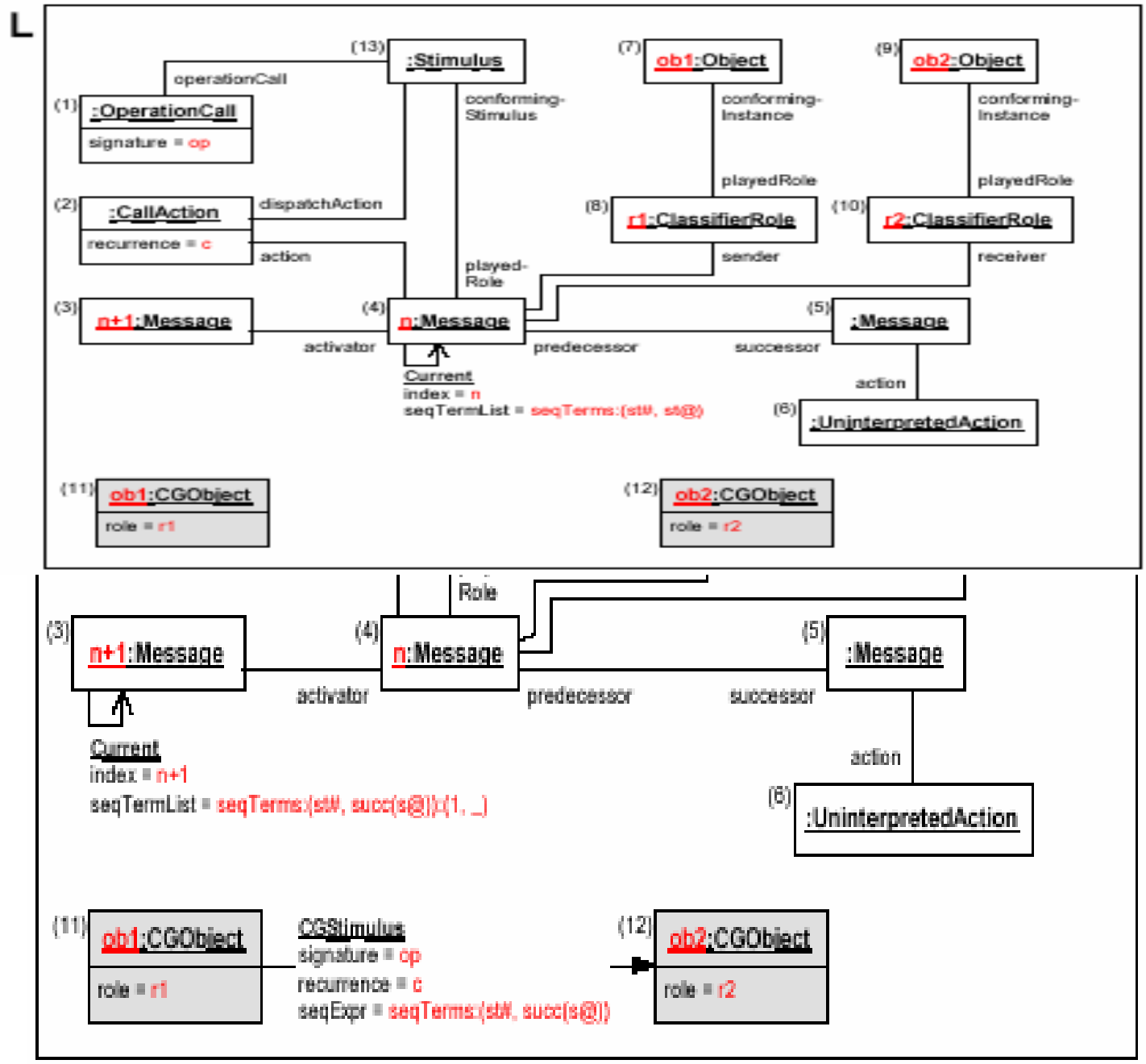


Figure 3.59: Rule MOG2CG_Int_R4a: translation of an initial Message of a conditional branching

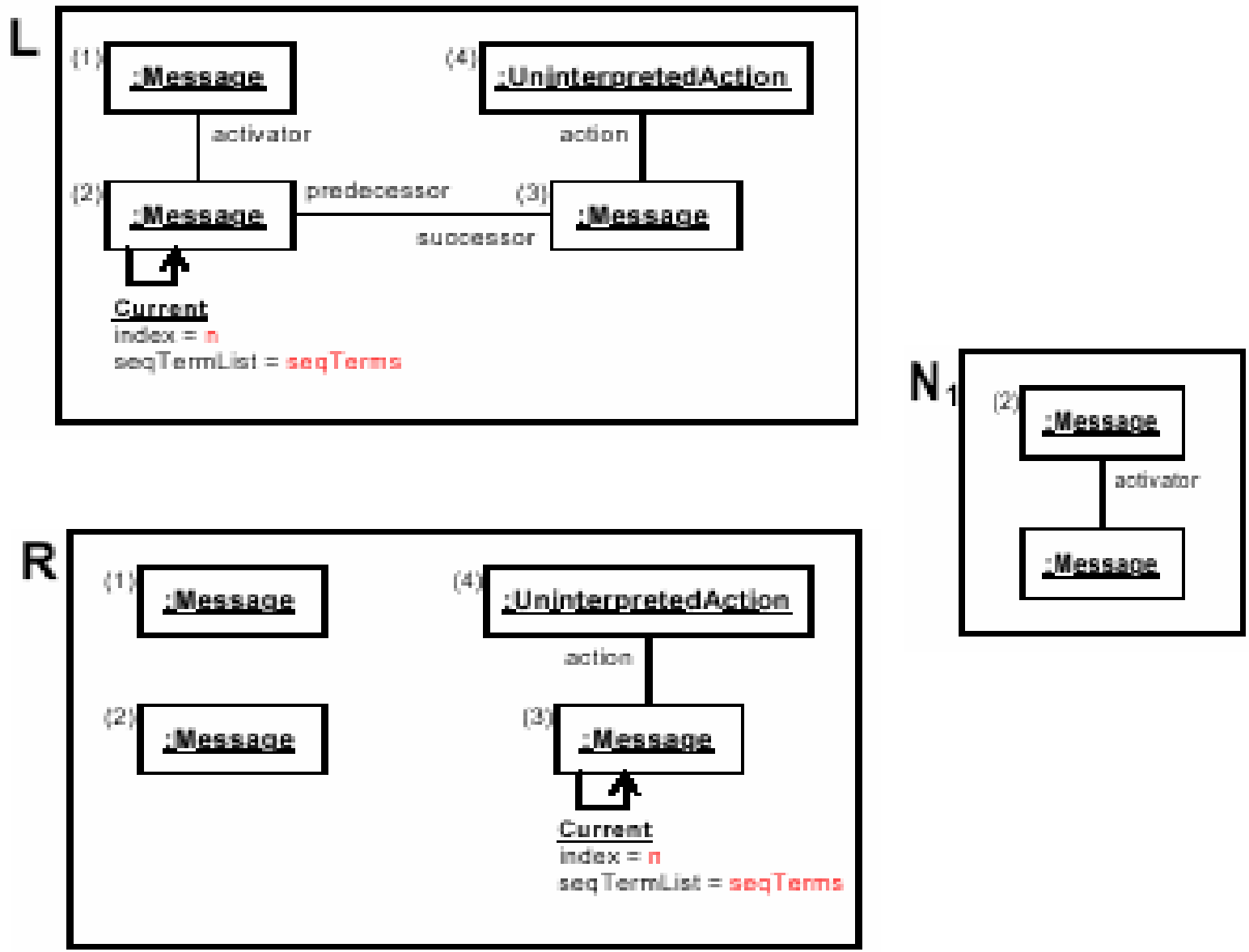


Figure 3.60: Rule MOG2CG_Int_R4b: tracking to the join Message

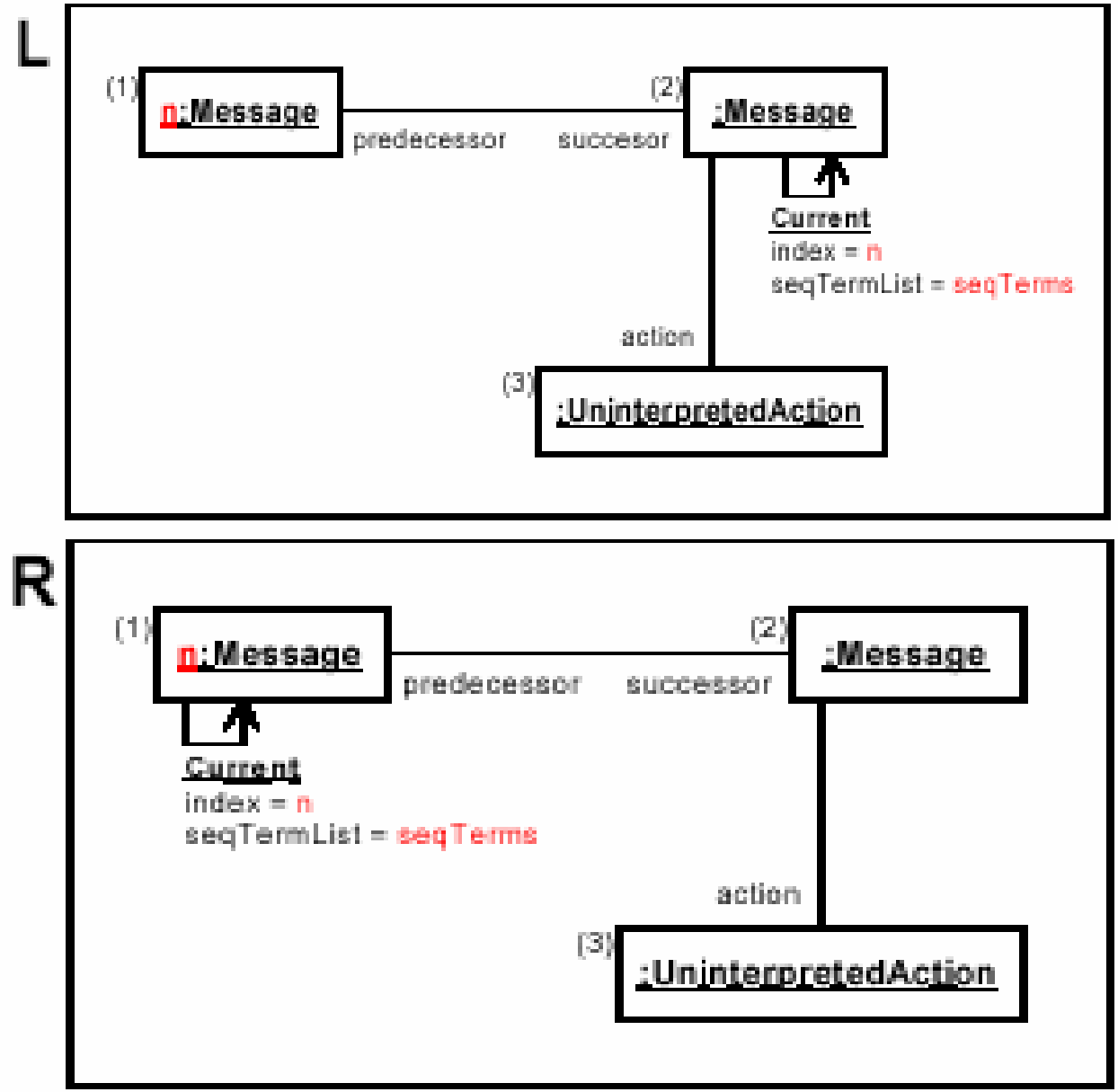
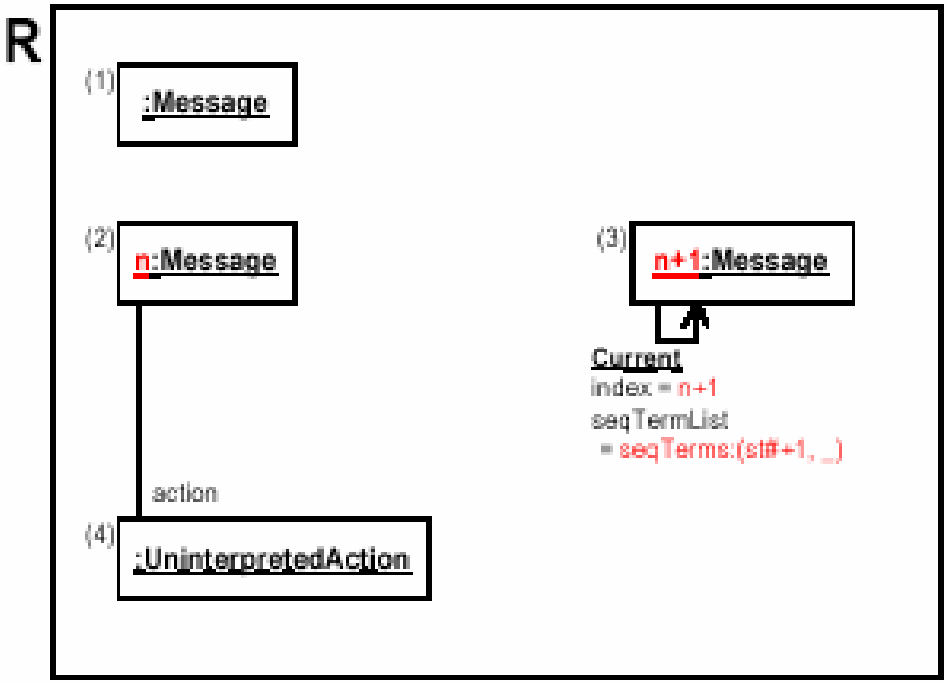


Figure 3.63: Rule MOG2CG_Int_R5a: completion of a branch and tracking back to the beginning of the next one



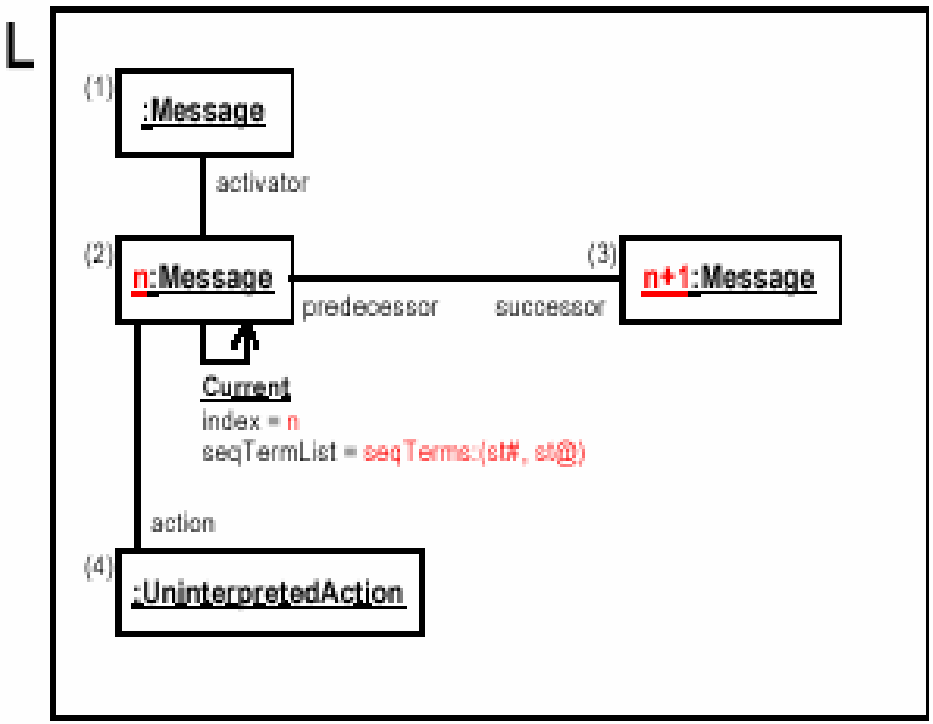


Figure 3.64: Rule MOG2CG_Int_R5b: completion of a branching and tracking to the next Message

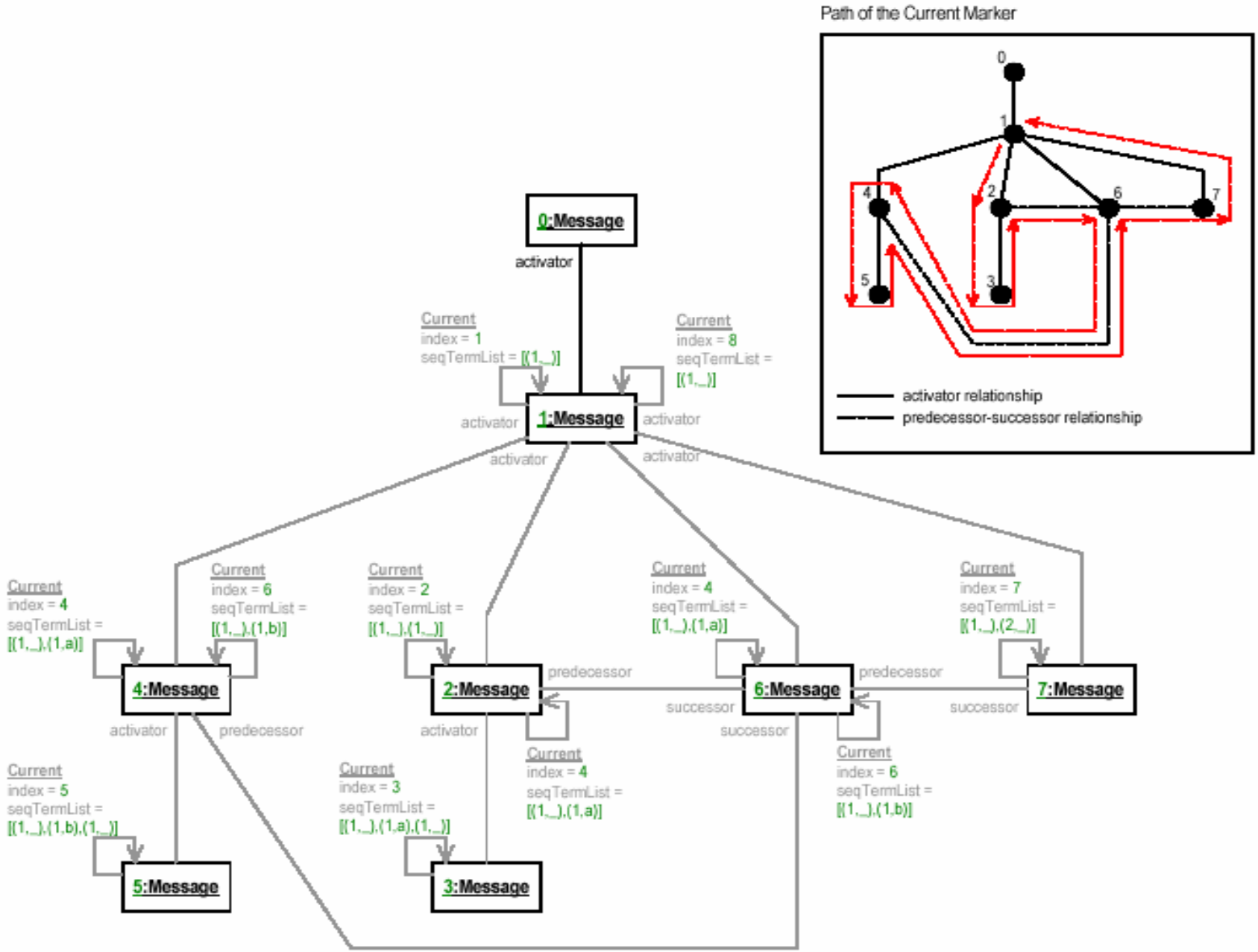


Figure 4.17: Moving the Queen: activator tree of the metamodel object graph after the application of **MOG2CG**

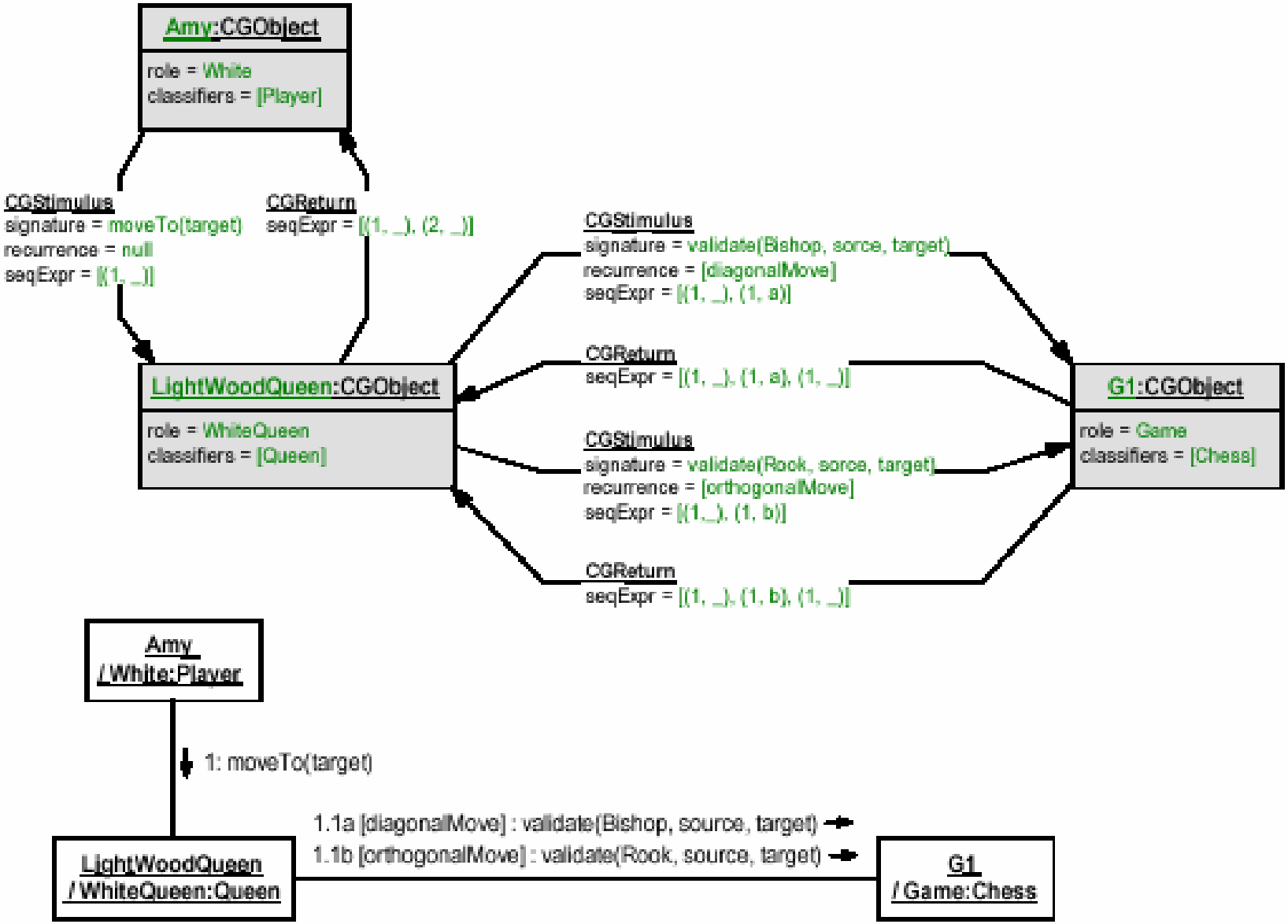


Figure 4.16: Moving the Queen: collaboration graph and collaboration diagram

3 Conclusion

- Investigate an appoach to translate SD into CD by means of graph transformation
- Consider nonconcurrent, synchronous procedural SD on instance level