



Multi-View Modelling (Languages in bits and pieces)

| | | |
|---------------|--|---|
| Esther Guerra | Escuela Politécnica Superior Ingeniería Informática Universidad Carlos III, Madrid (Spain) |  |
| Juan de Lara | Escuela Politécnica Superior Ingeniería Informática Universidad Autónoma, Madrid (Spain) |  |

Motivation

- Multi-View Visual Languages, made of a set of notations.
- Needed in order to make the (visual) language more appropriate for specifying in the large.
- Smaller, more comprehensible models.
- Multiple perspectives.
- Examples: UML, SySML, Labyrinth, ...

Questions/Goals

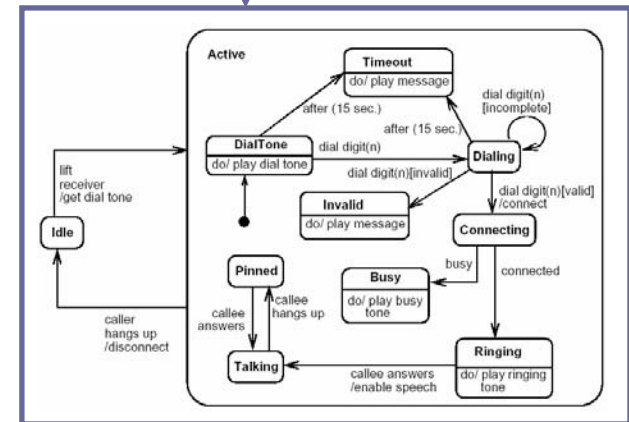
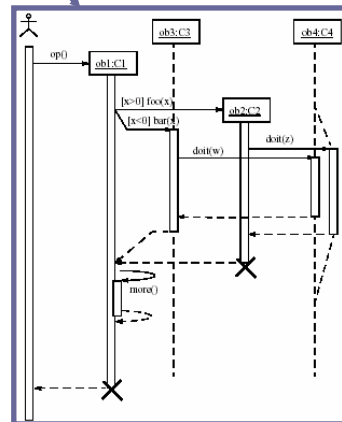
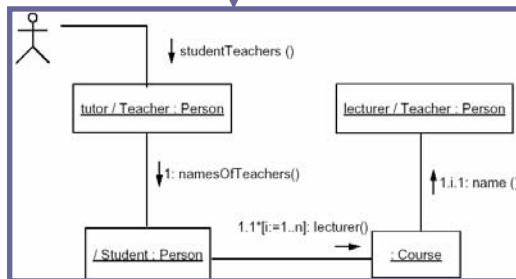
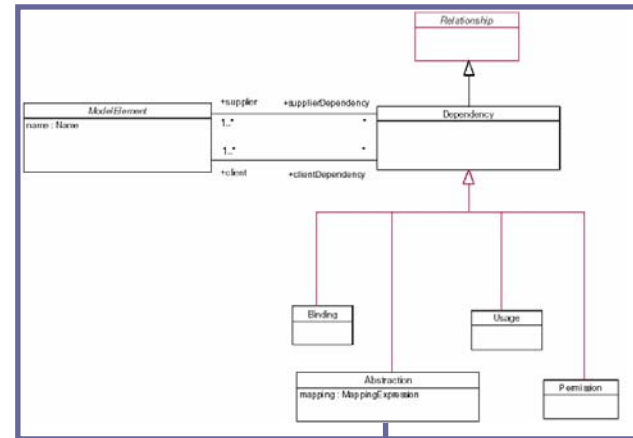
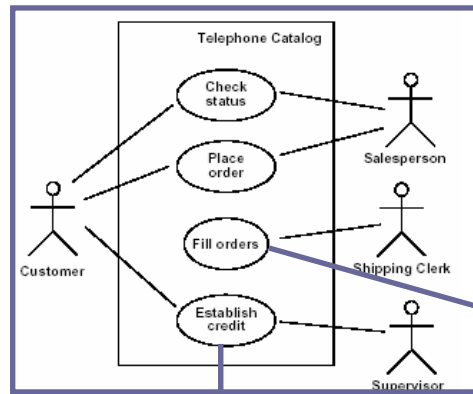
- How to specify such languages? (and obtain a customized modelling environment).
- Support for different types of views:
 - *System* views.
 - *Derived* and *Audience* oriented views.
 - *Semantic* views.
- Support for different behaviours.
- Support for Consistency:
 - Syntactic.
 - Semantic.

Agenda



- **Defining Multi-view visual languages.**
 - Background: Triple Graph Grammars
 - Syntactic consistency. Behavioural patterns.
- Derived and Audience-oriented views.
- Semantic views.
- Example.
- Conclusions and Future work.

Multi-View (Visual) Languages

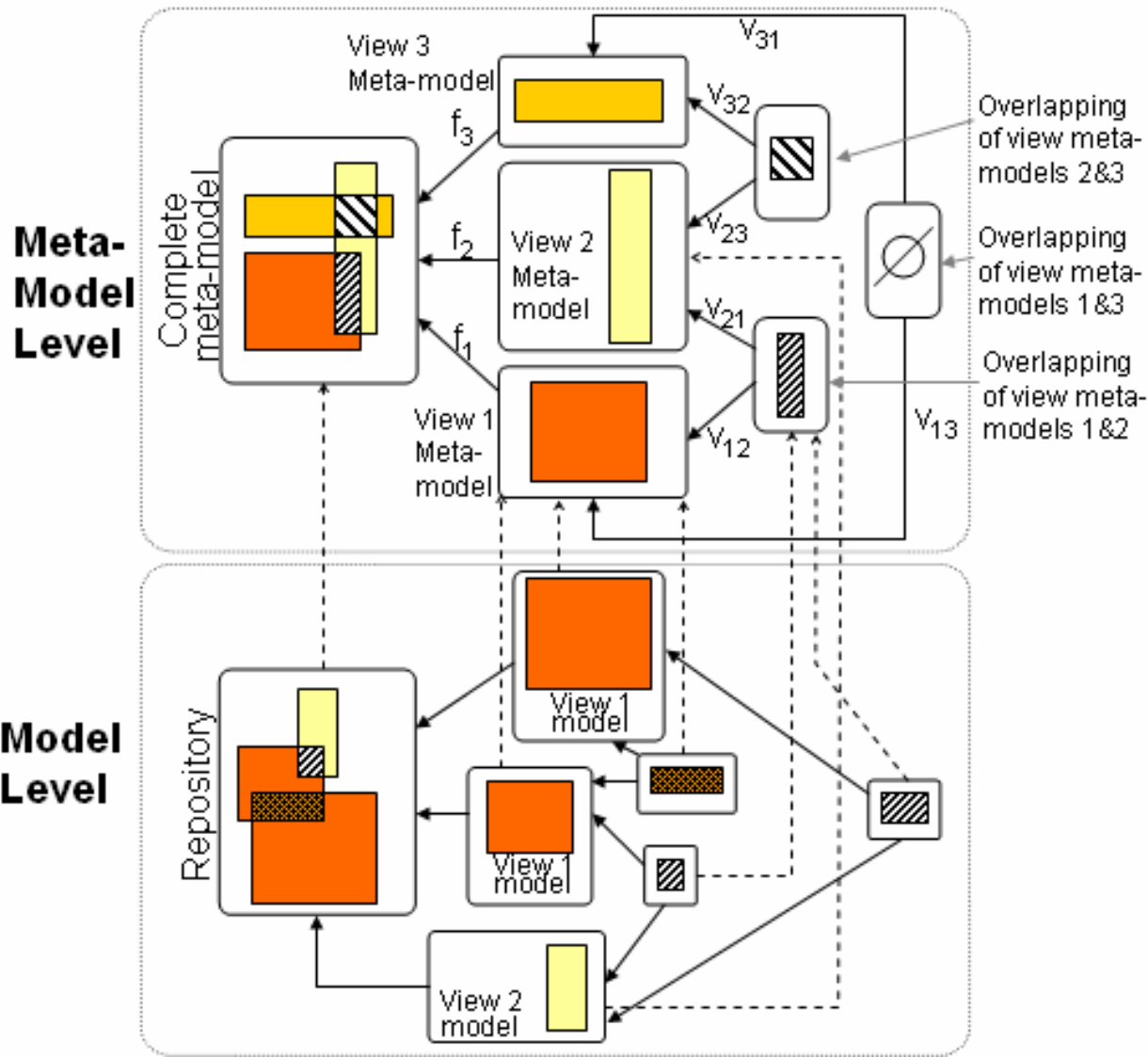


Multi-view (visual) languages

- A family of (visual) notations, which can be used in combination to describe different aspects of a system.
- They are usually related through a common specification (the meta-model).
- Different diagram types (*viewpoints*) as restrictions of the global meta-model.

Multi-view languages.

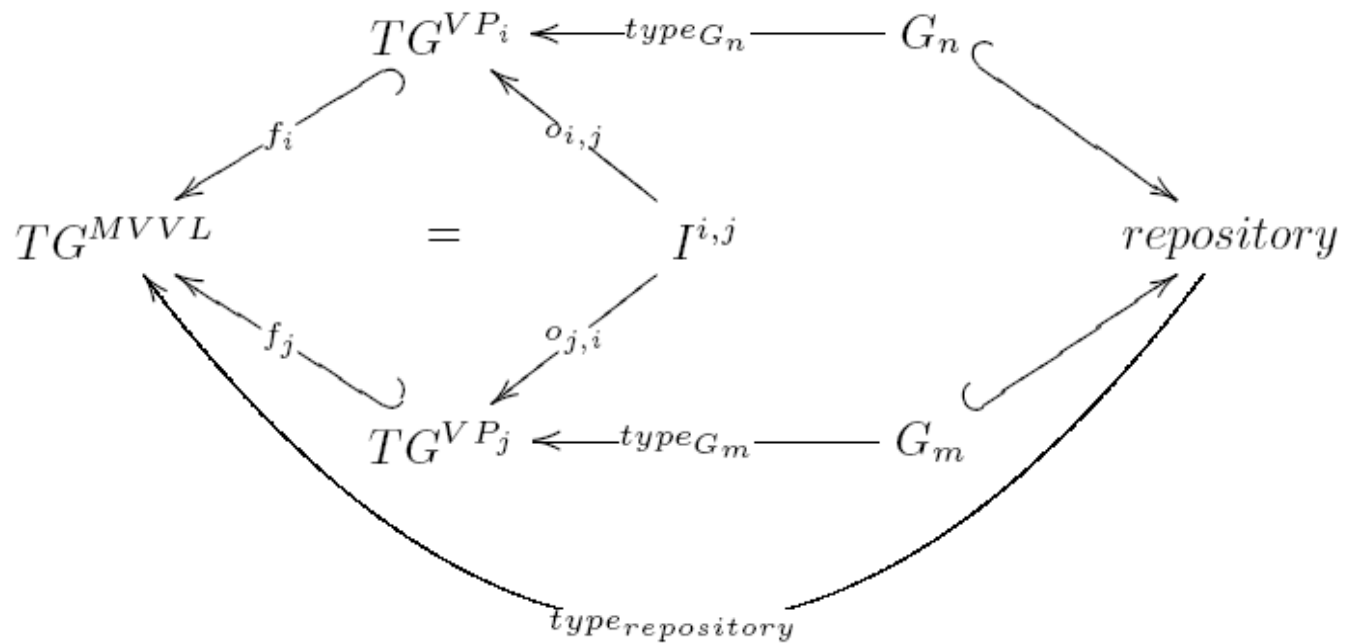
Formalization.



- Each square is a pullback.
- This also holds at the model level.
- Pullback objects give information about what elements should be updated in other views due to a change.
- Inclusions vs. other functions for the f 's.
- Mappings implemented as triple graph grammars.

Multi-view languages.

Formalization.



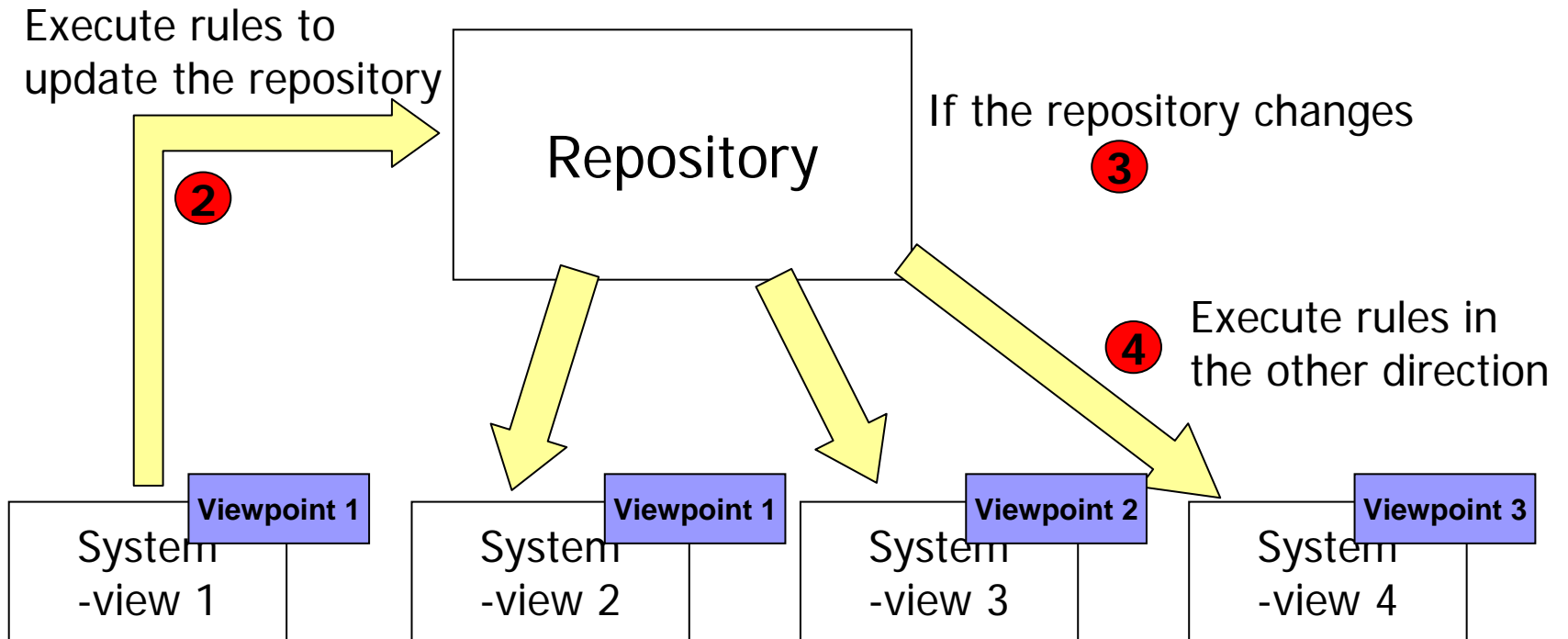
Multi-view languages.

Consistency.

- **Syntactic consistency:** rules are automatically derived from the whole meta model. Similar to Model View Controller architecture.
- Additional, domain specific rules can be added by the VL designer.
- **Semantic consistency:** transformations from the repository (Semantic views)
- The user decides when to perform these checks.

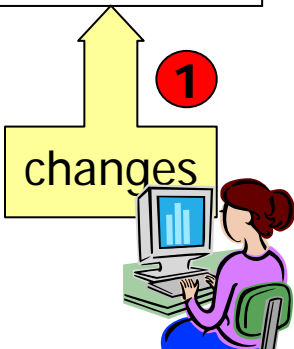
Multi-view languages.

Syntactic consistency



- Two kinds of grammars:

- To update the repository from a change in a view.
- To update a view from a change in the repository.



Agenda

- Defining Multi-view visual languages.
 - **Background: Triple Graph Grammars**
 - **Syntactic consistency. Behavioural patterns.**
- Derived and Audience-oriented views.
- Semantic views.
- Example.
- Conclusions and Future work.

Multi-view languages.

Graph Grammars in the DPO approach

- Category theory. Theory valid for any (weak) adhesive HLR category (graphs, attributed graphs, Petri nets, triple graphs, etc.)
- production: $L \xleftarrow{l} K \xrightarrow{r} R$
- K contains the preserved elements. Morphisms l and r are injective.
- Direct derivation as 2 *pushouts*.

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ \downarrow m & & \downarrow d & & \downarrow m^* \\ G & \xleftarrow{l^*} & D & \xrightarrow{r^*} & H \end{array}$$

$$D = G - m(L - l(K))$$

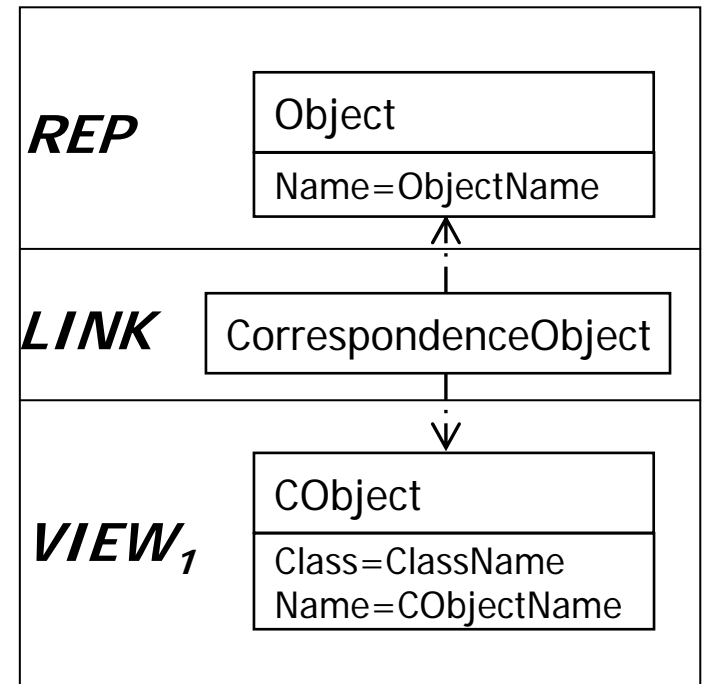
$$H = D + m^*(R - r(K))$$

("gluing" at $d(K)$)

Multi-view languages.

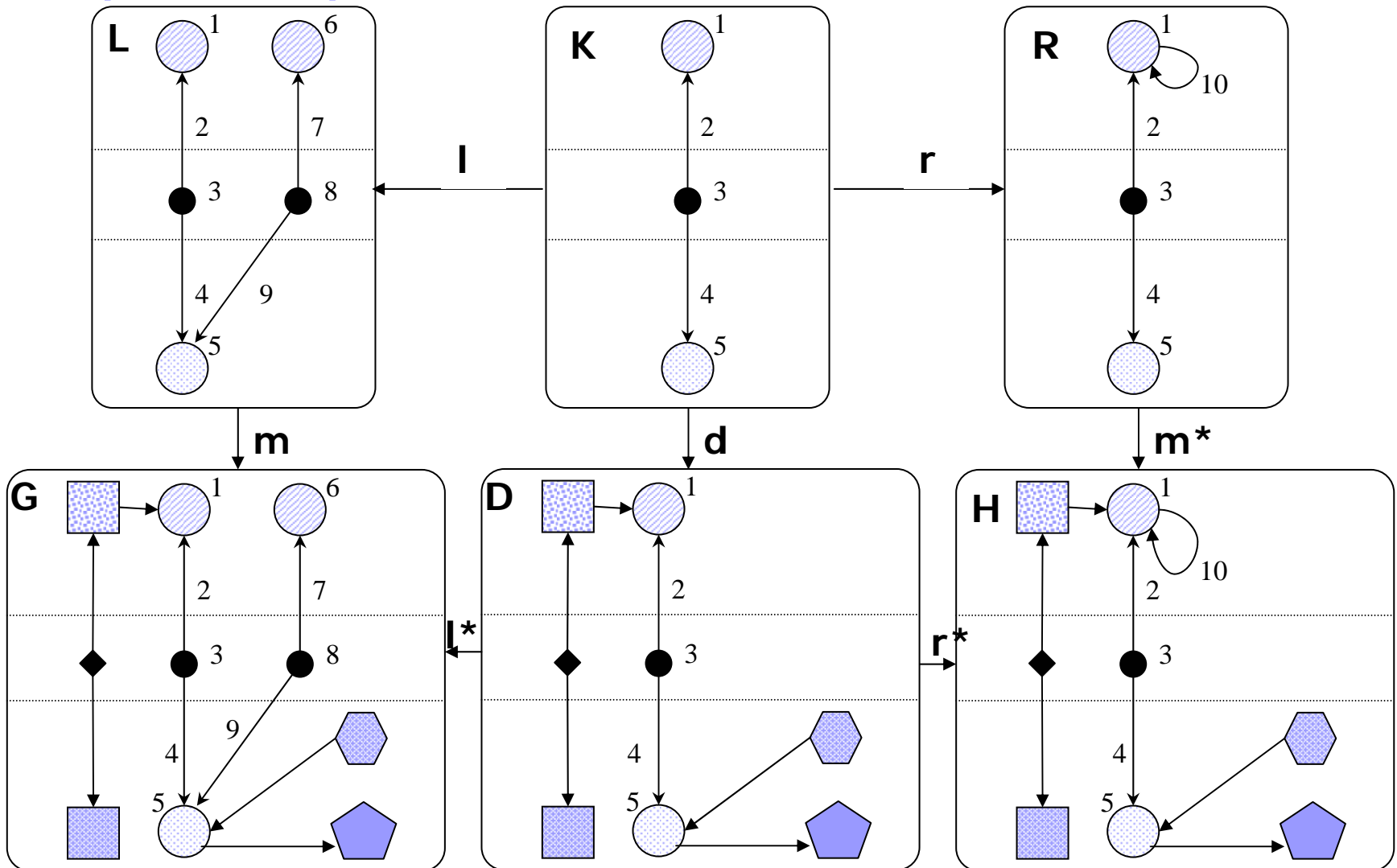
Triple Graphs

- ◆ By Andy Schürr.
- ◆ Rewriting of triple graphs.
- ◆ $G = (G_1 \xleftarrow{g_1} \text{LINK} \xrightarrow{g_2} G_2)$
- ◆ Morphisms g_1 and g_2 represent m to n relationships between nodes in G_1 and G_2 . We take them partial.



Multi-view languages.

Triple Graph Grammars

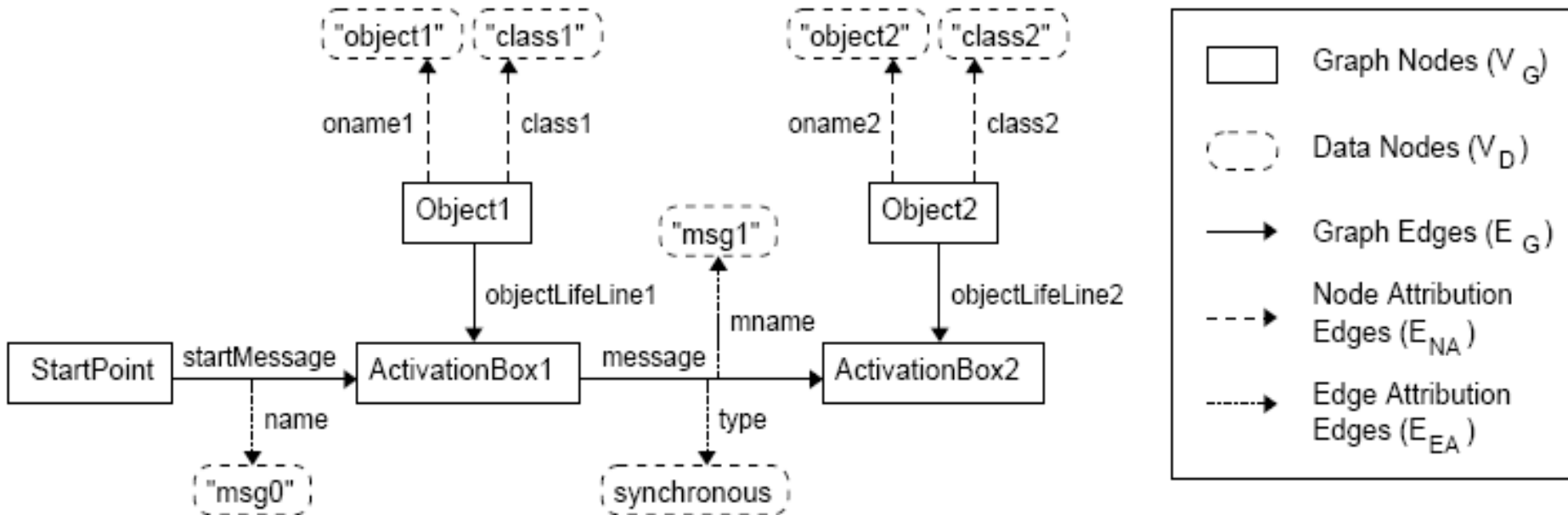


Multi-view languages.

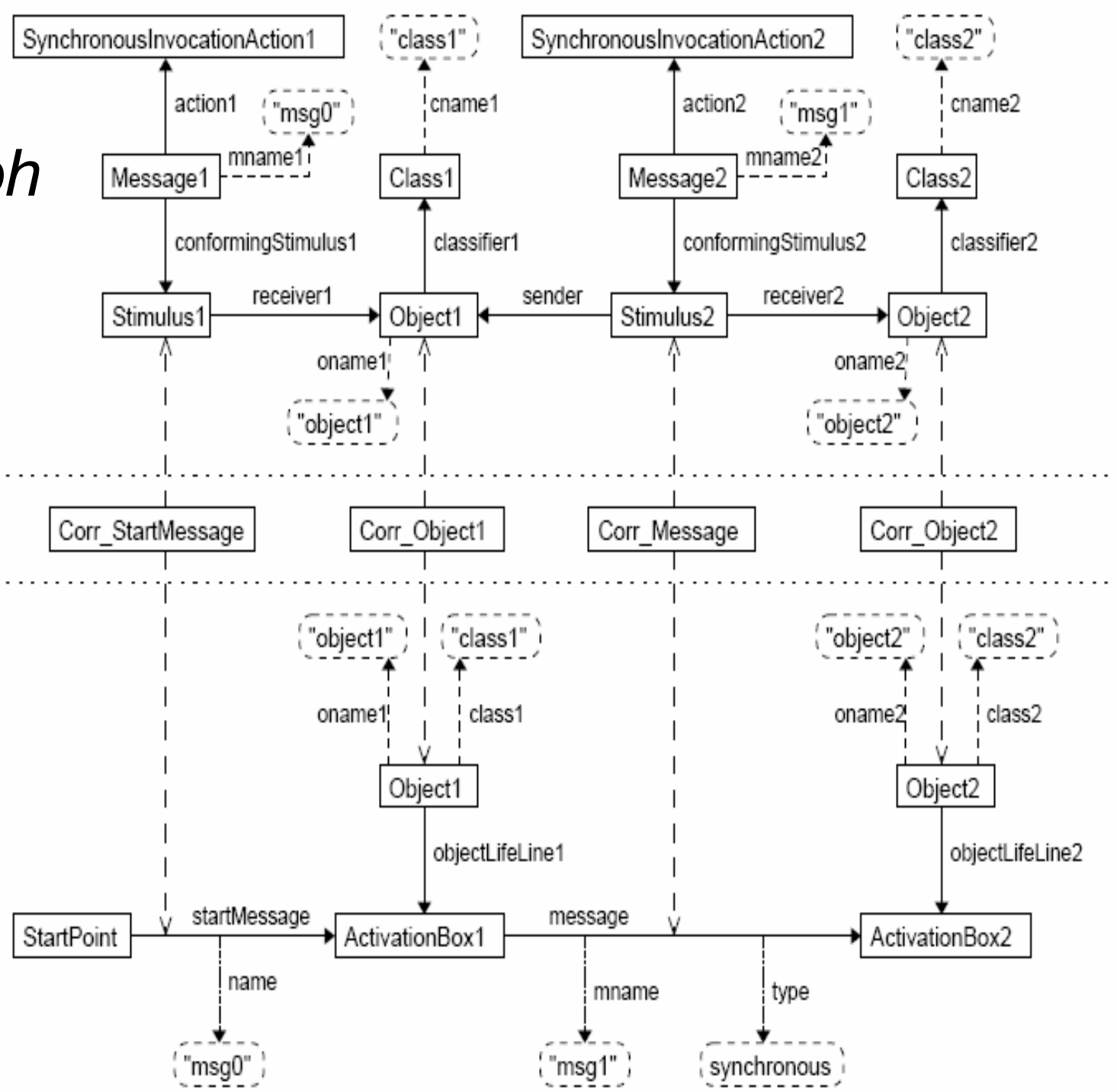
Formalization: Triple Graphs

- Built on the notion of *E-graphs* (graphs with node and edge attribution).

$$G = (V_G, V_D, E_G, E_{NA}, E_{EA}, (source_j, target_j)_{j \in \{G, NA, EA\}})$$



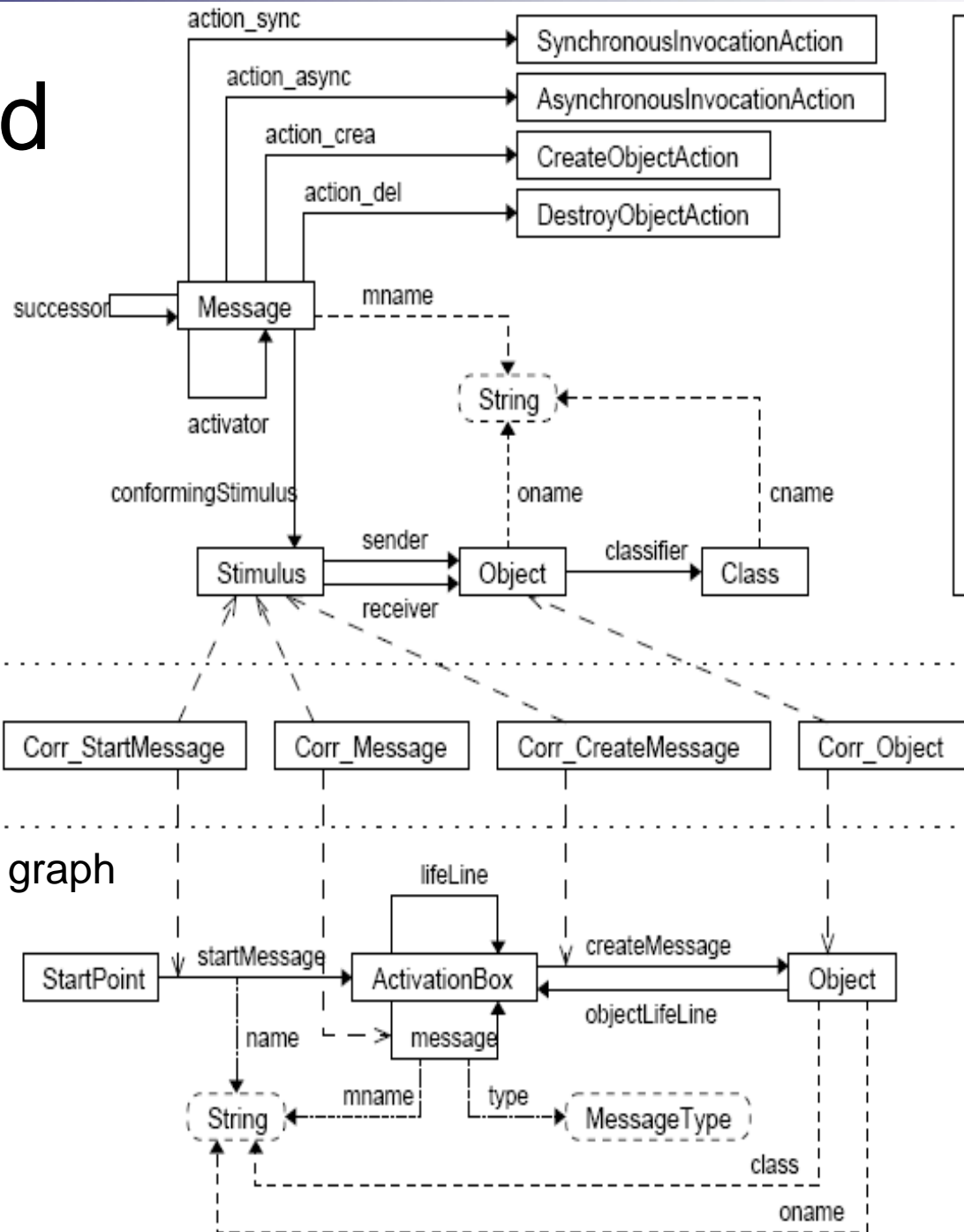
TriE-Graph



TriA-Graph (Attributed Triple Graphs)

- *Tri-Egraphs* are provided with an algebra in order to structure the attribute set into sorts and provide appropriate operations for attribute computation.
- Data signature $DSIG=(S_D, OP_D)$, with sorts for attribution $S'_D \subseteq S_D$.
- $TriAG=(TriEG, D)$, with:
 - $TriEG= (G_1, G_2, G_C, c_1, c_2)$
 - D a $DSIG$ - algebra with $\bigcup_{s \in S'_D} D_s = \bar{V}_{D_i}$ for $i \in \{1, 2, C\}$

Attributed Type Triple Graph



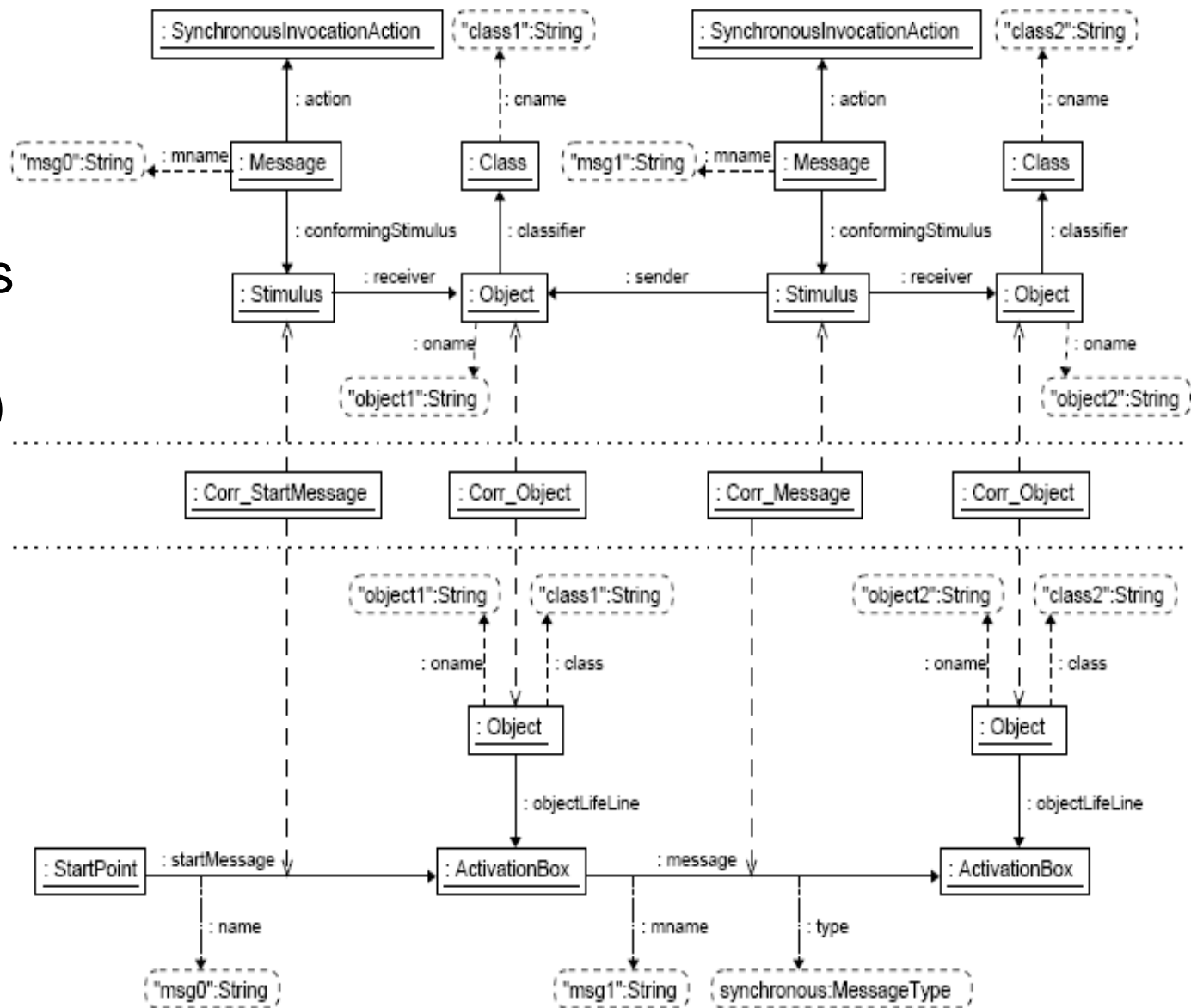
- The associated algebra is final.

- A part of the type graph Also for the correspondence graph.

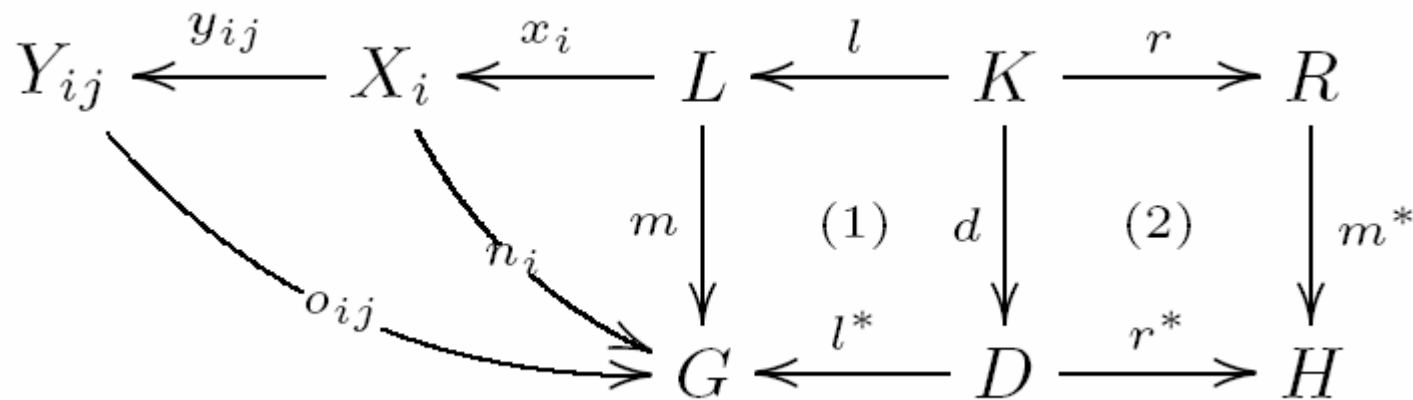
Attributed Typed Triple Graphs

Objects are tuples
(graph, typing):
TriTAG=(TriAG, t)

We showed that
it is an adhesive
HLR category.



Application Conditions

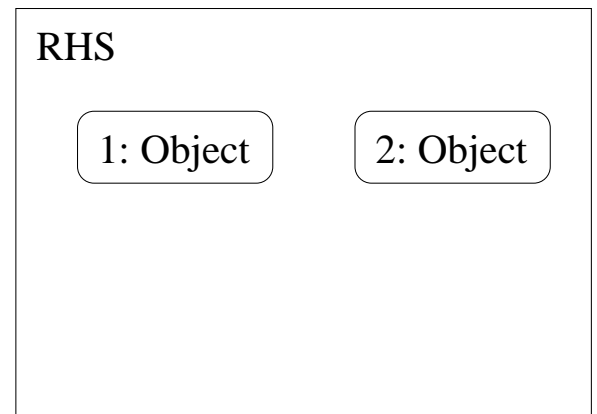
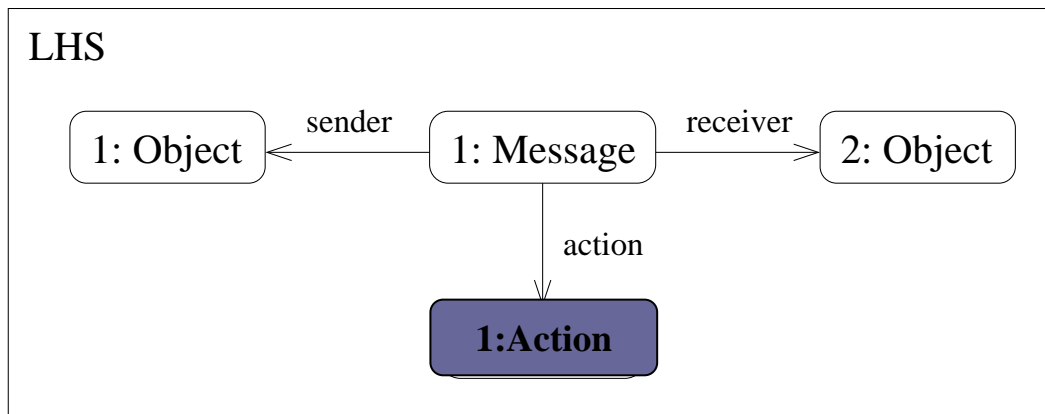
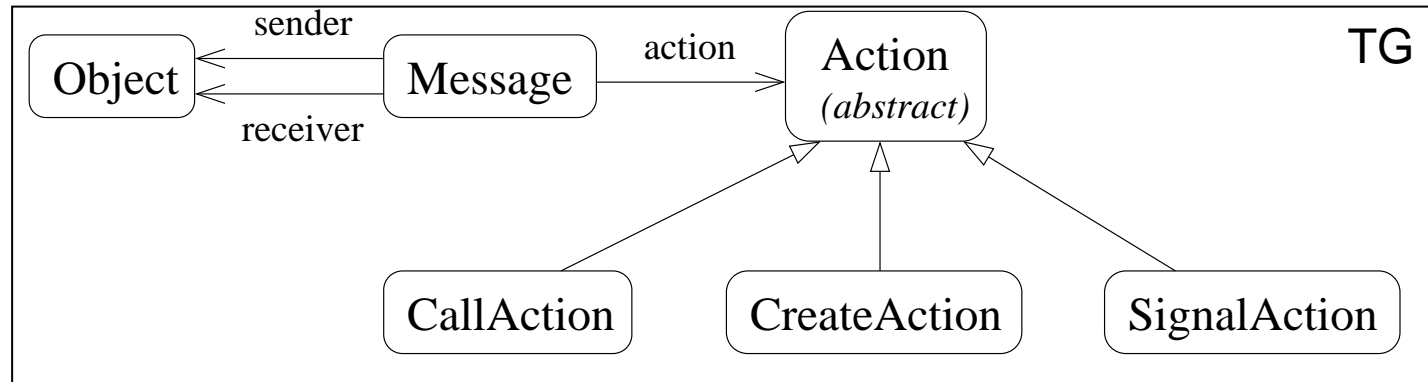


- Negative Application Condition (NAC) if Y_{ij} is empty.
- Positive Application Condition (PAC) if $X_i \cong L$ and Y_{ij} not empty.

Integration with meta-modelling

- “Instances” of abstract classes are allowed to appear in rules.
- The rule is equivalent to a set of concrete rules, resulting from the valid substitution of the abstract elements by concrete ones.
- If the abstract element appears in the RHS, then it should also appear in the LHS (i.e. elements with abstract typing cannot be created).

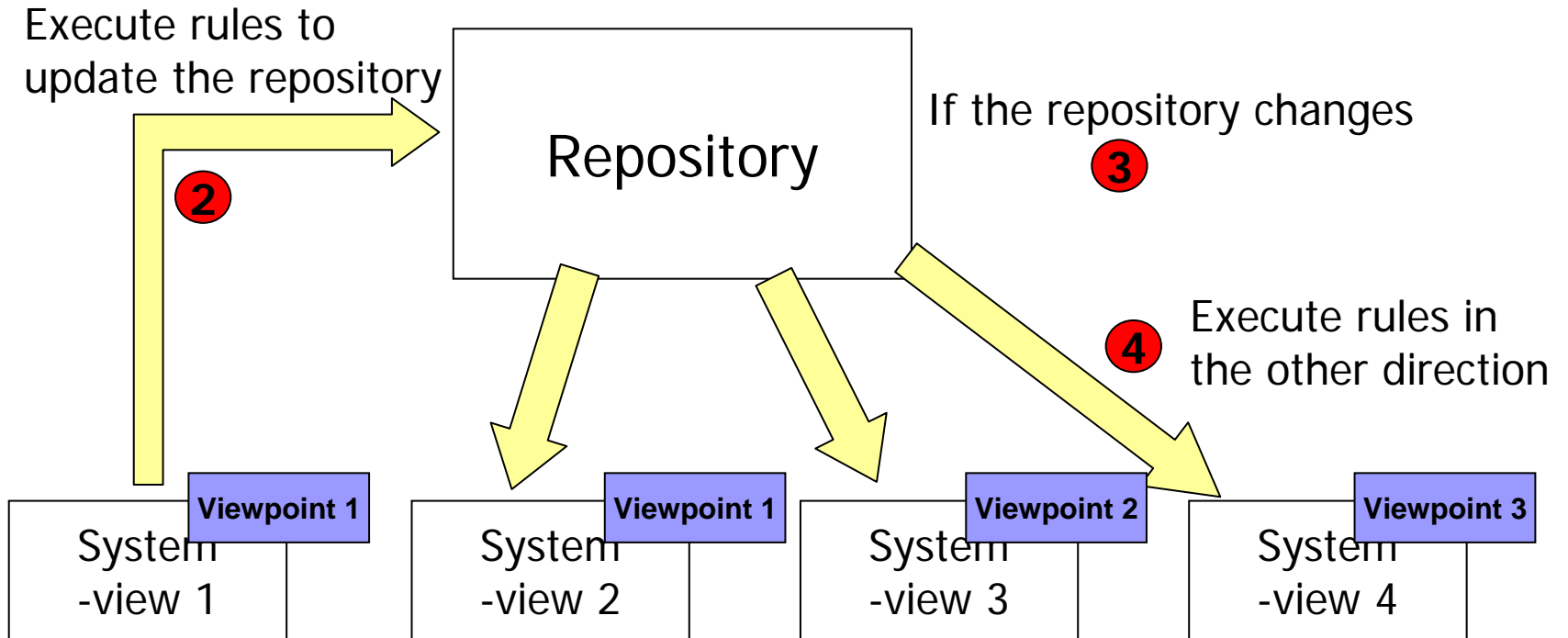
Integration with meta-modelling



Agenda

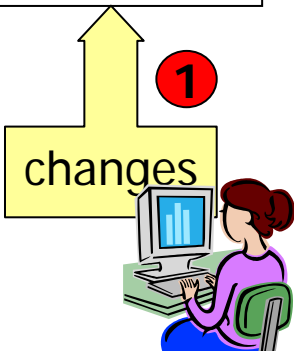
- Defining Multi-view visual languages.
 - **Background: Triple Graph Grammars**
 - **Syntactic consistency. Behavioural patterns.**
- Derived and Audience-oriented views.
- Semantic views.
- Example.
- Conclusions and Future work.

Reminder...



- Two kinds of grammars:

- To update the repository from a change in a view.
- To update a view from a change in the repository.



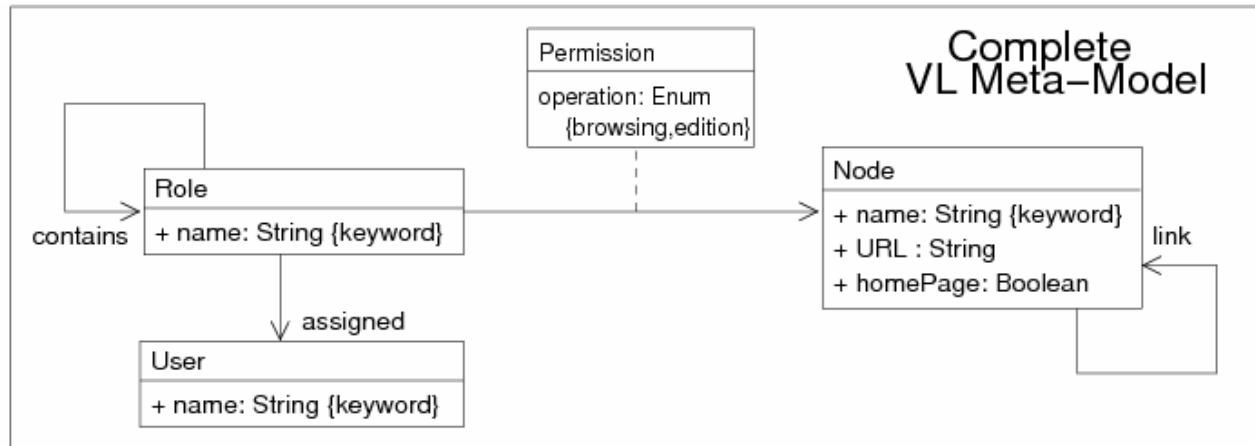
Multi-view languages.

Syntactic.

- Rules are created for:
 - Creation of nodes.
 - Modification of node (edges) attributes (both directions).
 - Deletion of nodes.
 - Connection of nodes (creation of edges).
 - Disconnection of nodes (deletion of edges).
- For each element type in each view.
- Pullback objects indicate which rules should be tried.

Example.

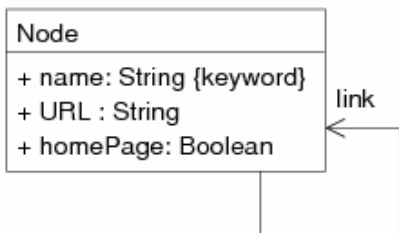
RBAC meta-model for web systems (simplified).



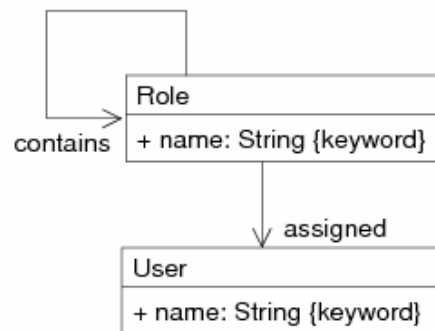
Viewpoint

Viewpoint

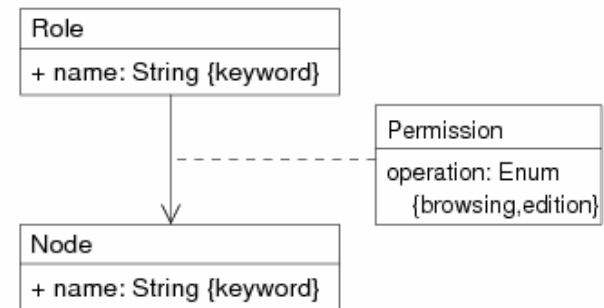
Viewpoint



Navigation Diagram Meta-Model



User Diagram Meta-Model

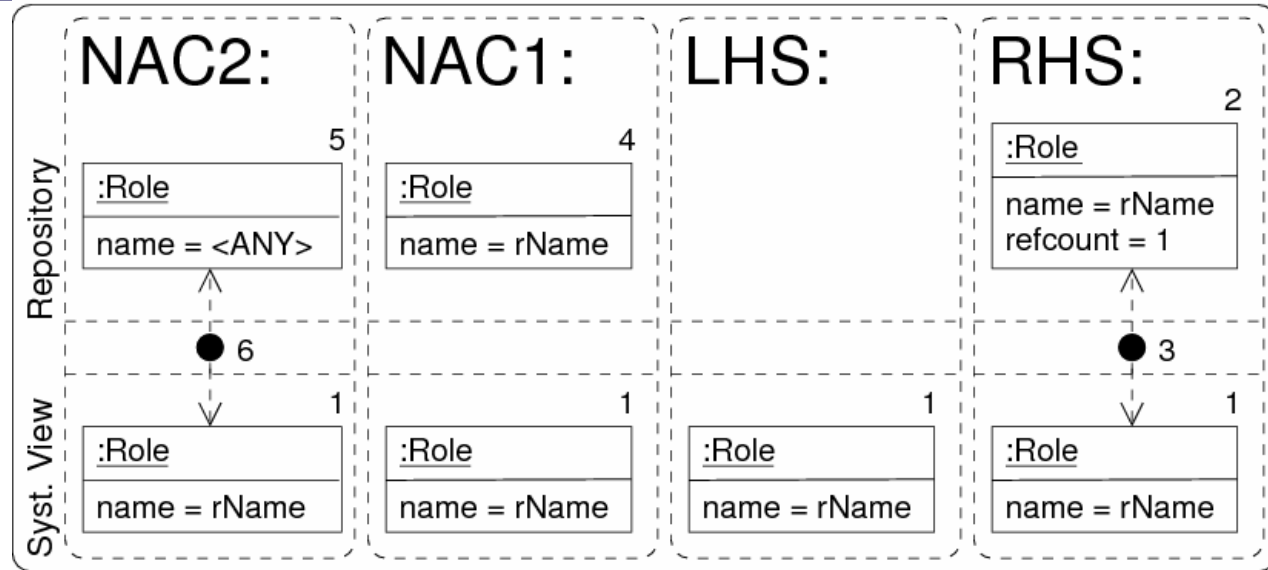


Permission Diagram Meta-Model

Multi-view languages.

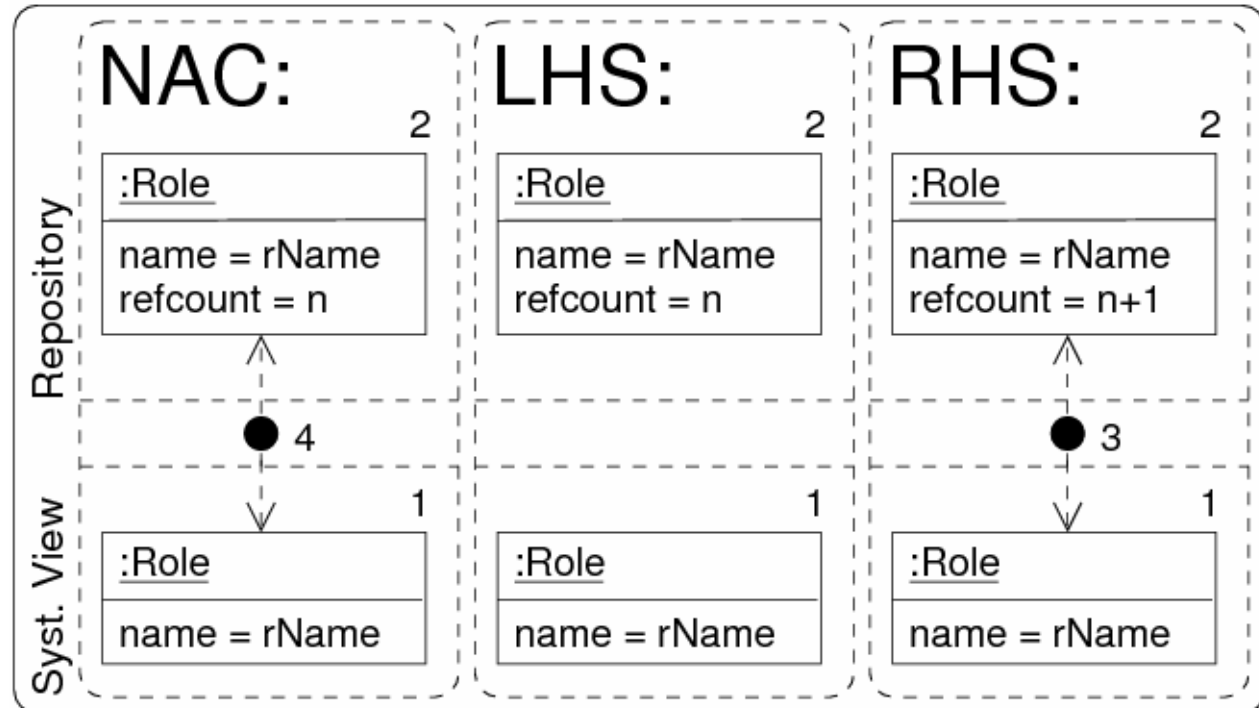
Example.

Creation Rules:



- Repository update Grammar.

Creation Rules



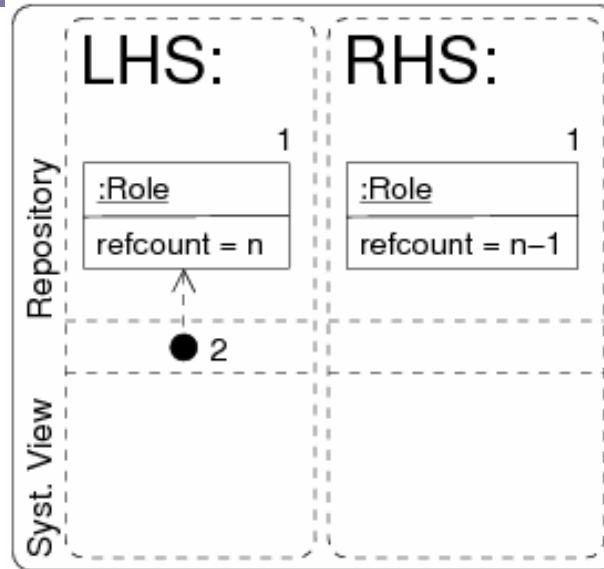
Multi-view languages.

Example.

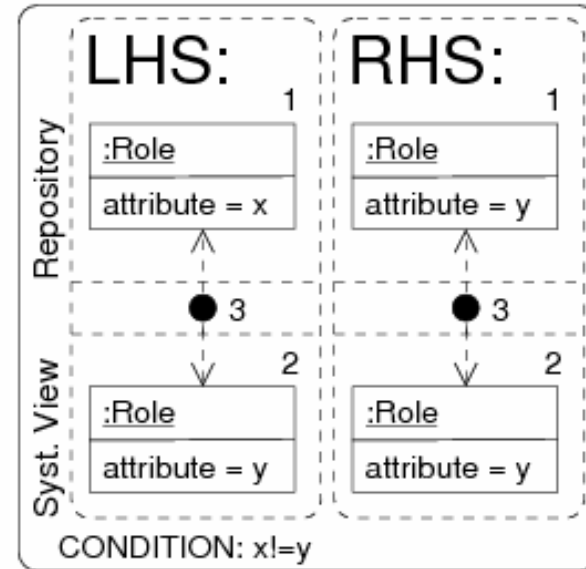
- Repository update Grammar.

Deletion and Edition rules.

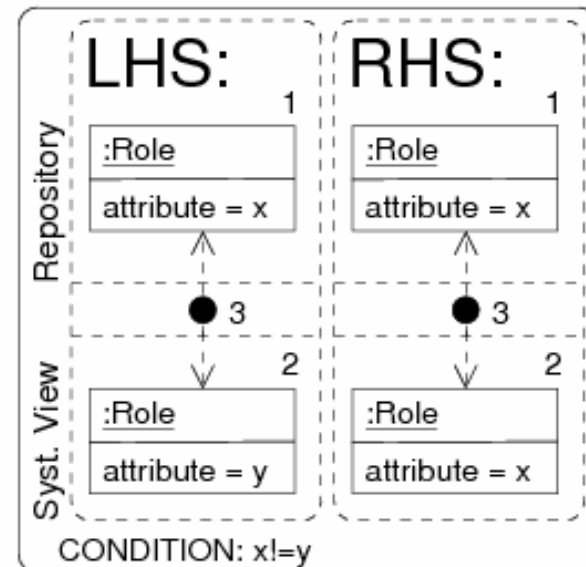
Deletion Rules:



Edition Rule:



Propagation Rule:

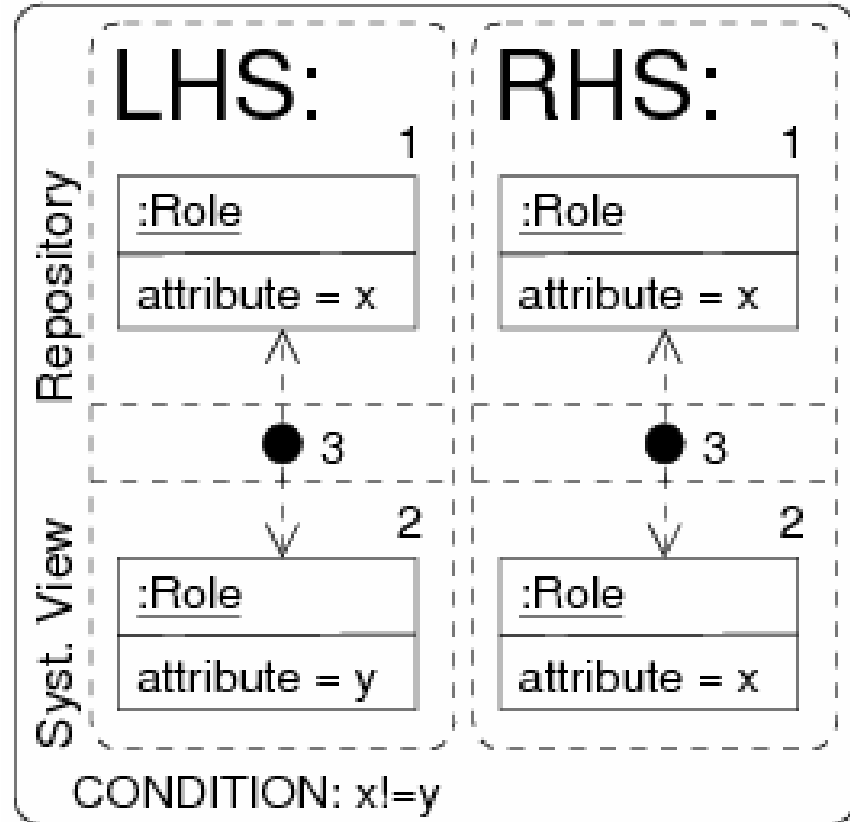


Multi-view languages.

Example.

- Change propagation Grammar.

Propagation Rule:



Multi-View Visual Languages.

Configurable behaviour patterns.

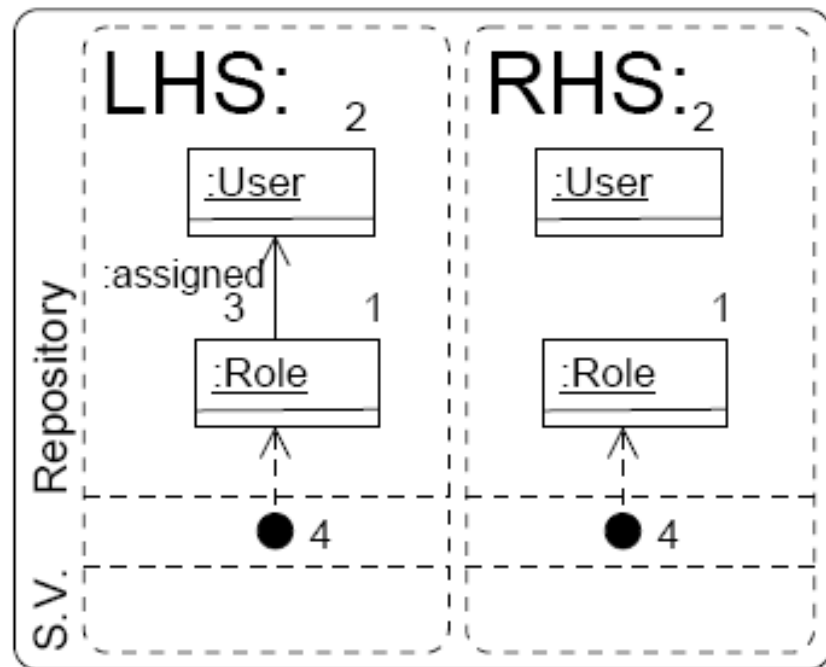
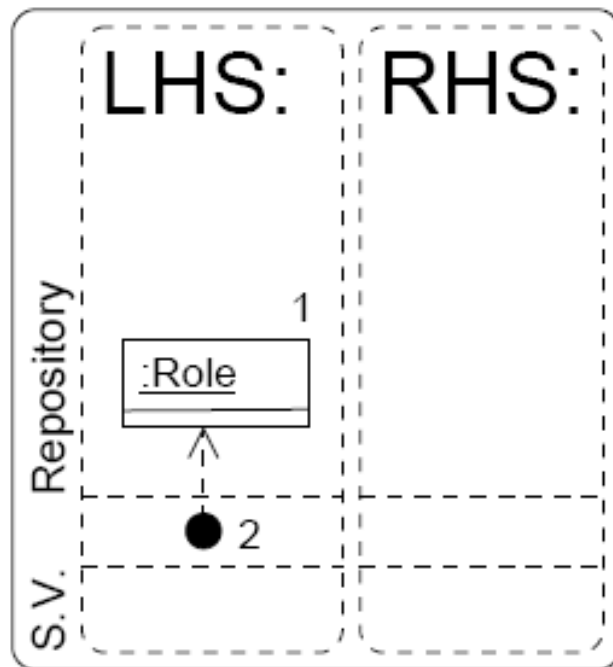
- The previous rules define the behaviour for a view management.
- Different MVVLs need different ways of handling creation, deletion and edition of elements in the different views.
- Different behaviours, depending on the DSVL:
 - Cascading deletion, vs. Conservative Deletion.
 - Copying attributes on creation.
 - Changing the identifier of an object in a view.
 - ...
- Can be selected by the MVVL designer.
- Can be modelled with different rule sets.

Multi-View Visual Languages.

Configurable behaviour patterns.

Cascading Deletion Rules:

Triple Rules from the Views to the Repository:

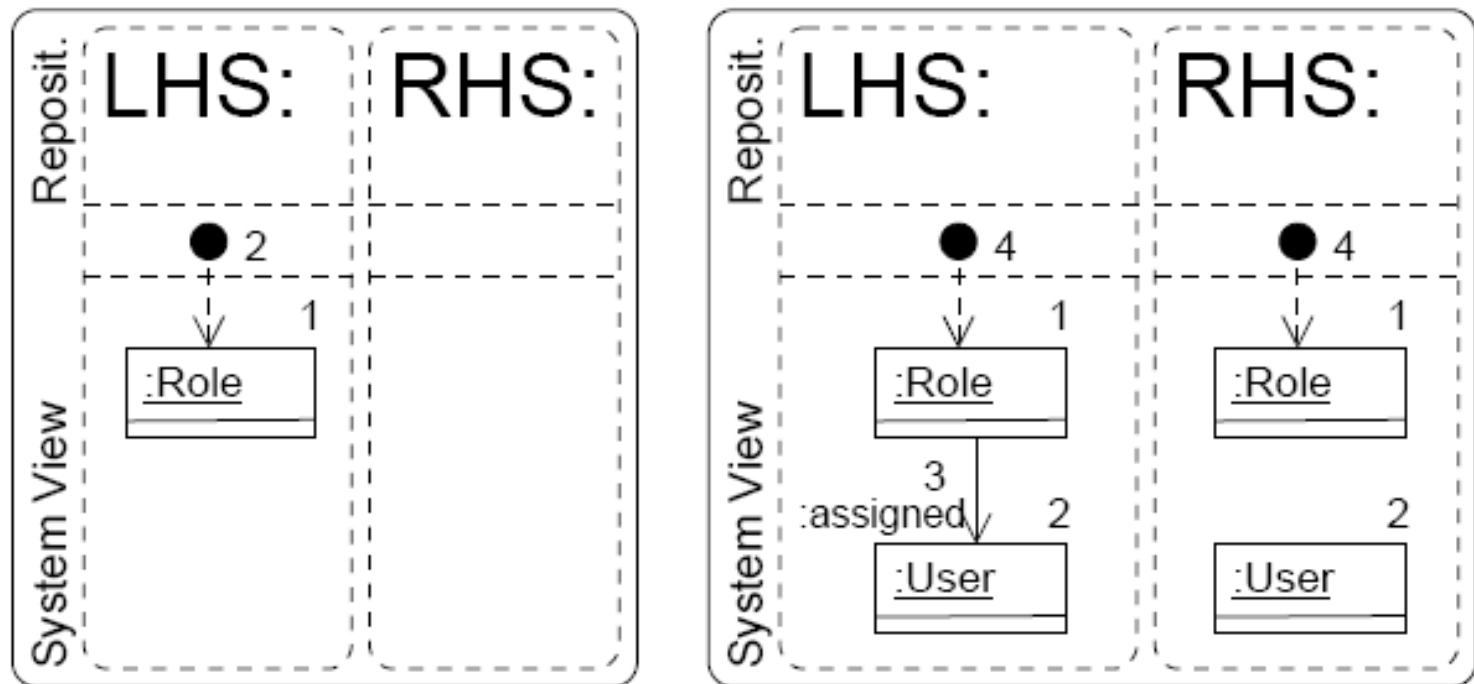


Multi-View Visual Languages.

Configurable behaviour patterns.

Cascading deletion (ii)

Triple Rules from the Repository to the Views:



Agenda

- Defining Multi-view visual languages.



- **Derived and Audience-oriented views.**

- Semantic views.
- Example.
- Conclusions and Future work.

Multi-view languages.

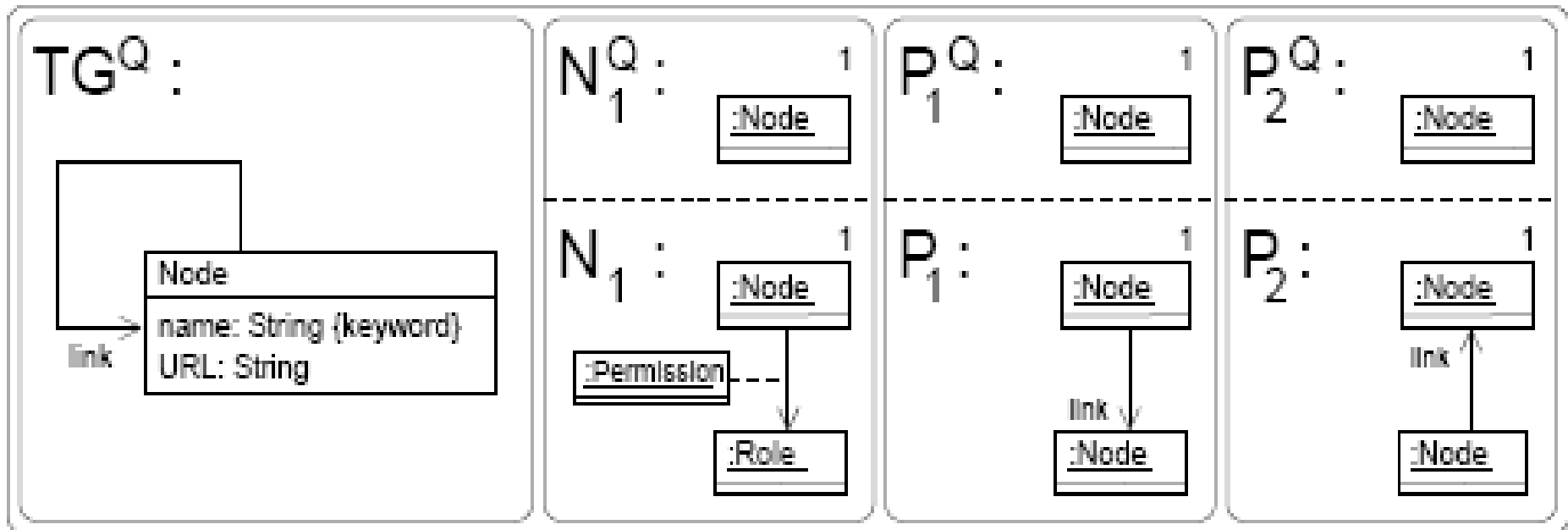
Derived Views.

- Extract information from the system.
- The information does not need to be conformant to any system view meta-model.
- It can be extracted from the repository or any other view.
- Graphical approach: graph query patterns.

Multi-view languages.

Derived Views.

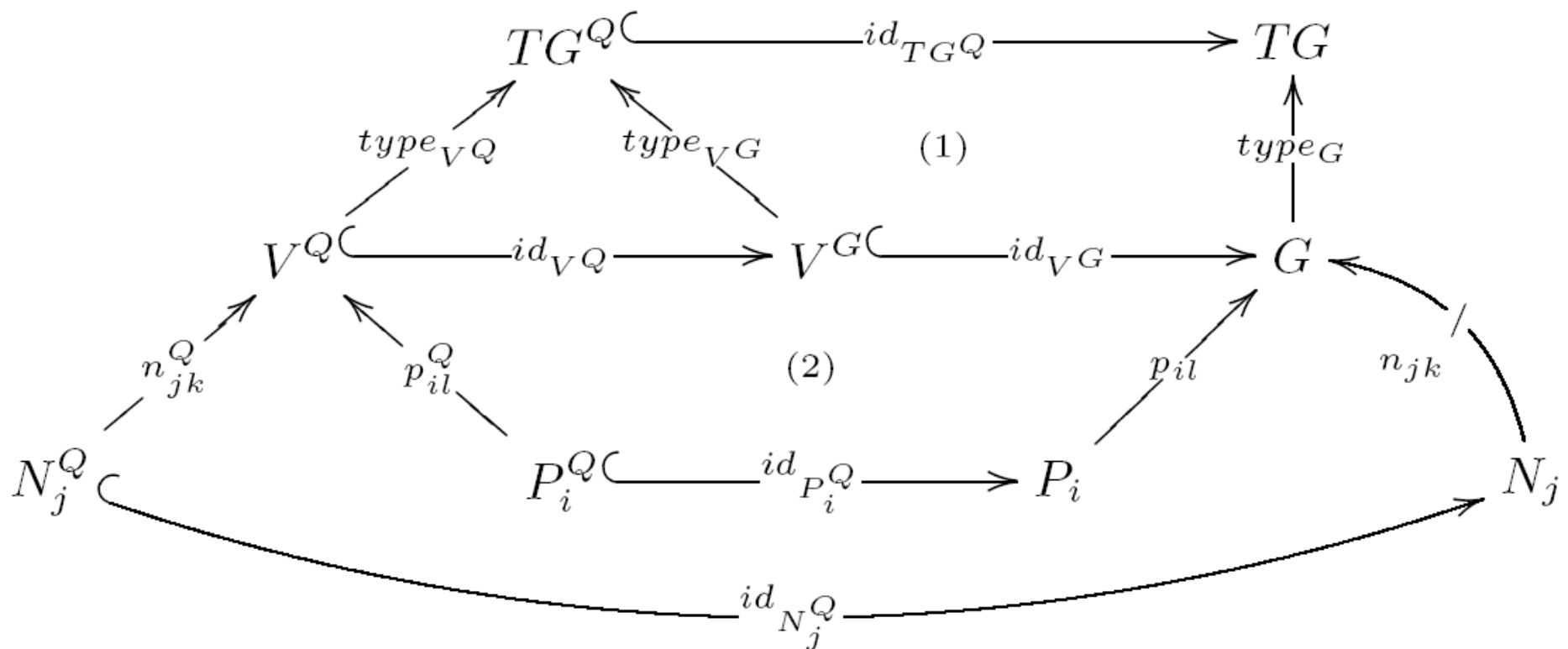
- Extract nodes (and links) such that a) no role has permissions on it and b) the node is source or target of a link.



Multi-view languages.

Derived Views.

- Graph query pattern: $Q = (TG^Q, \{P_i^Q, P_i\}_{i \in I}, \{N_j^Q, N_j\}_{j \in J})$



Multi-view languages.

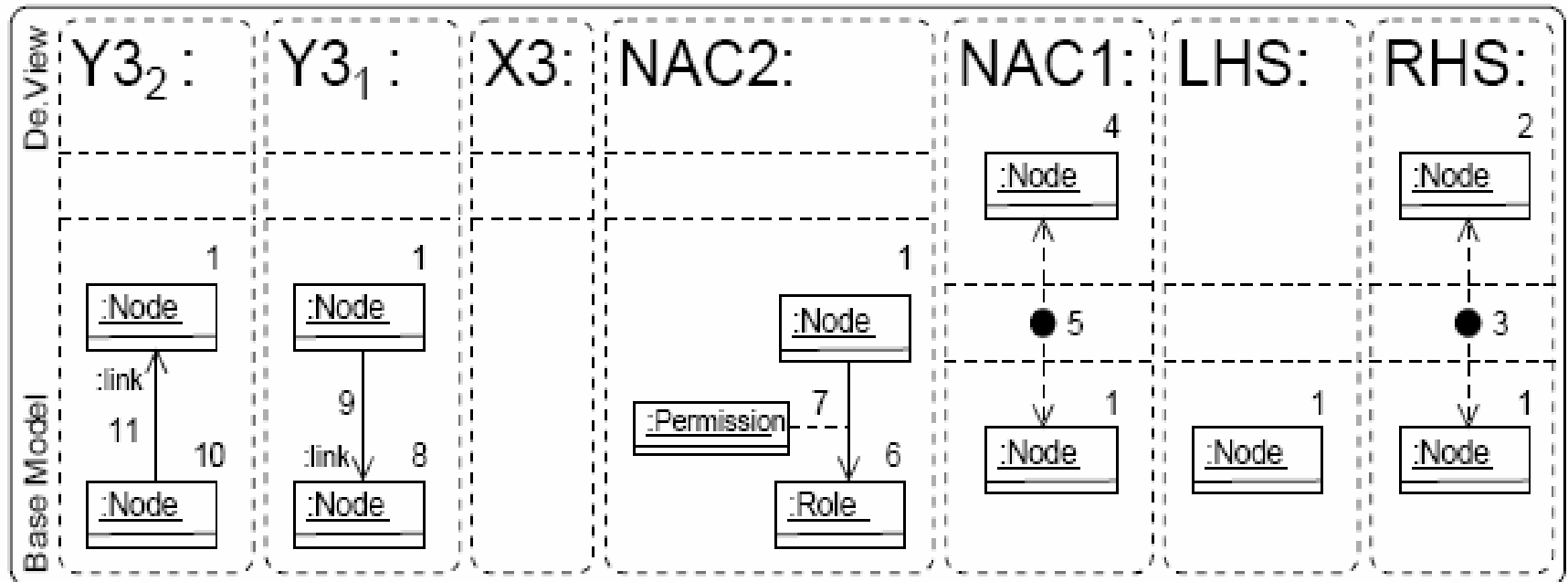
Derived Views.

- From the patterns, triple graph grammars are generated that build the derived view and keep a mapping to the base model.
- The grammars keep consistent the base model and the derived view.
- Similar to the previous rules, but with application conditions derived from the patterns.

Multi-view languages.

Derived Views.

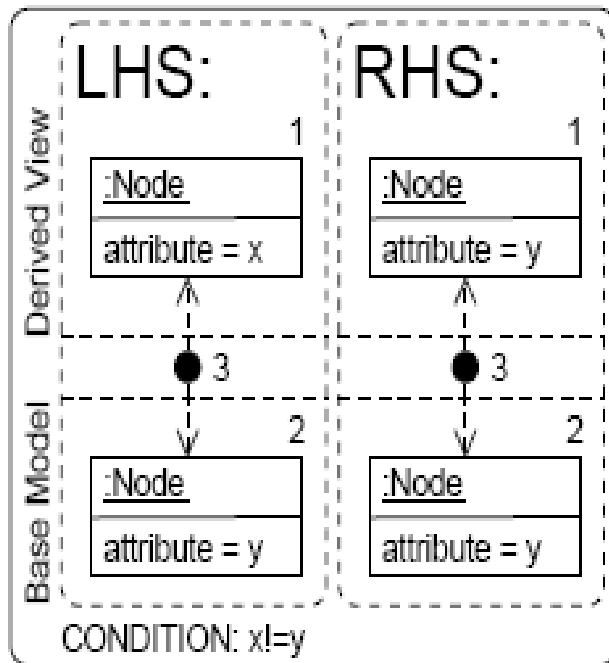
Creation Rule:



Multi-view languages.

Derived Views.

Edition Rule:

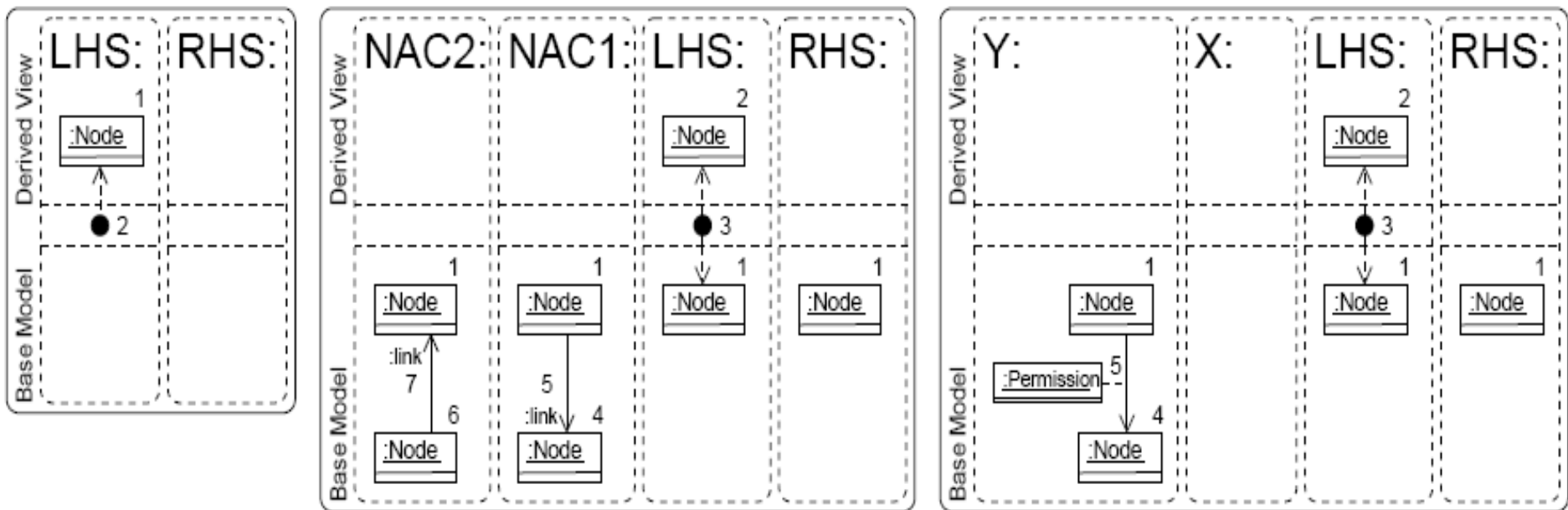


- These two kind of rules (for each node and edge type) are able to generate the derived view.
- Additional rule for updating the derived view in case something should be deleted.

Multi-view languages.

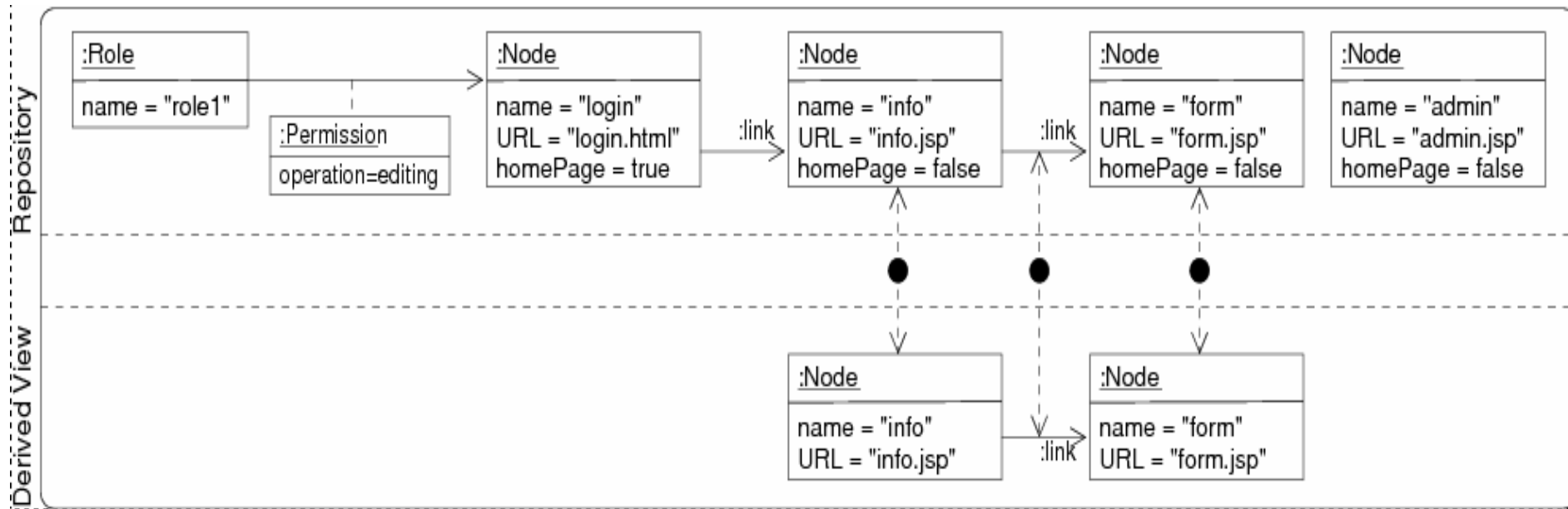
Derived Views.

Deletion Rules:



Multi-view languages.

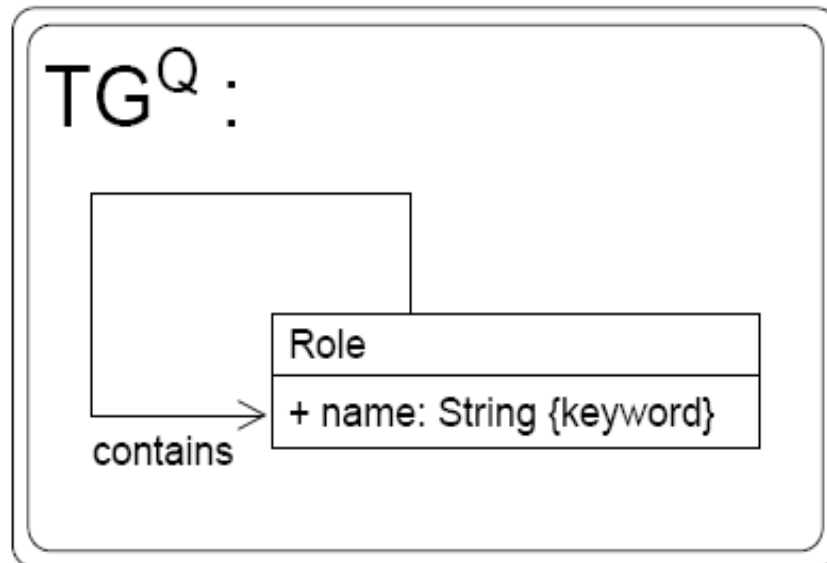
Derived Views.



Multi-view languages.

Audience-Oriented Views.

- Pre-defined queries by the VL designer.
- Oriented to a certain kind of users.



Agenda

- Defining Multi-view visual languages.
- Derived and Audience-oriented views.



- **Semantic views.**
- Example.
- Conclusions and Future work.

Multi-view languages.

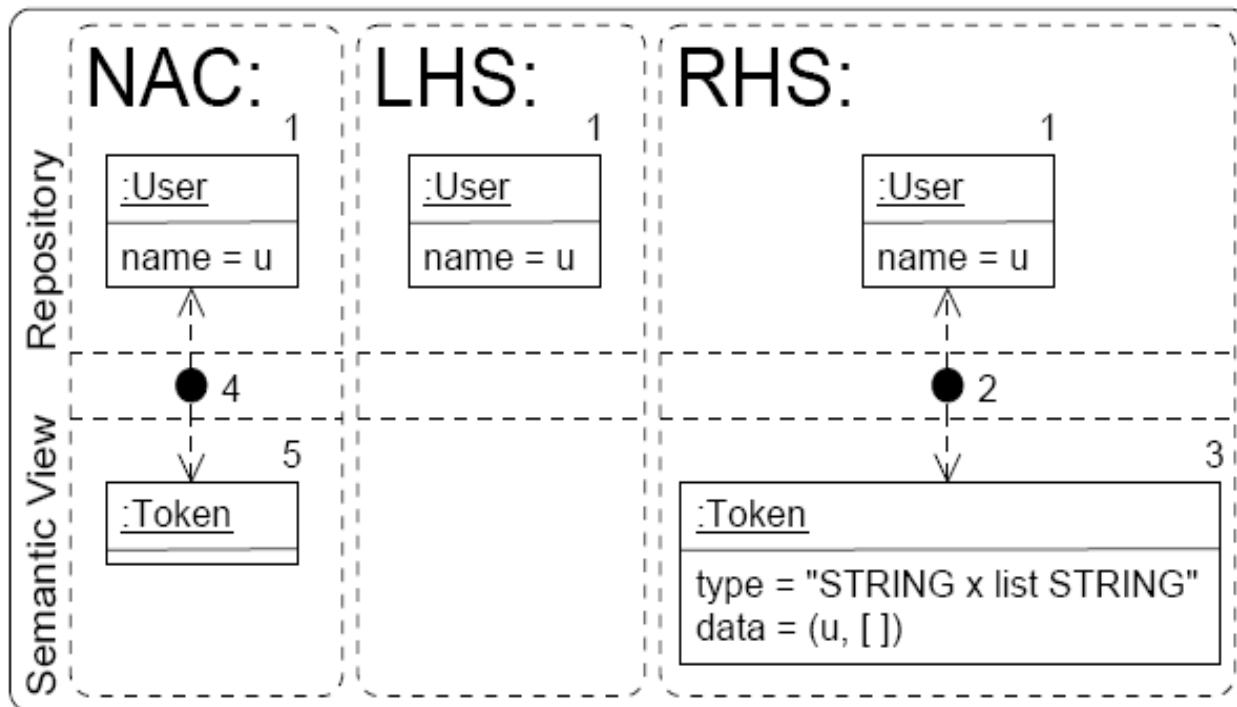
Semantic Views.

- Result of the translation into a semantic domain.
- Usually with the purposes of analysis or simulation.
- With triple graph grammars, to keep a mapping to the original model.
- Back-annotation when the analysis is performed.

Multi-view languages.

Coloured Petri Nets Semantic View

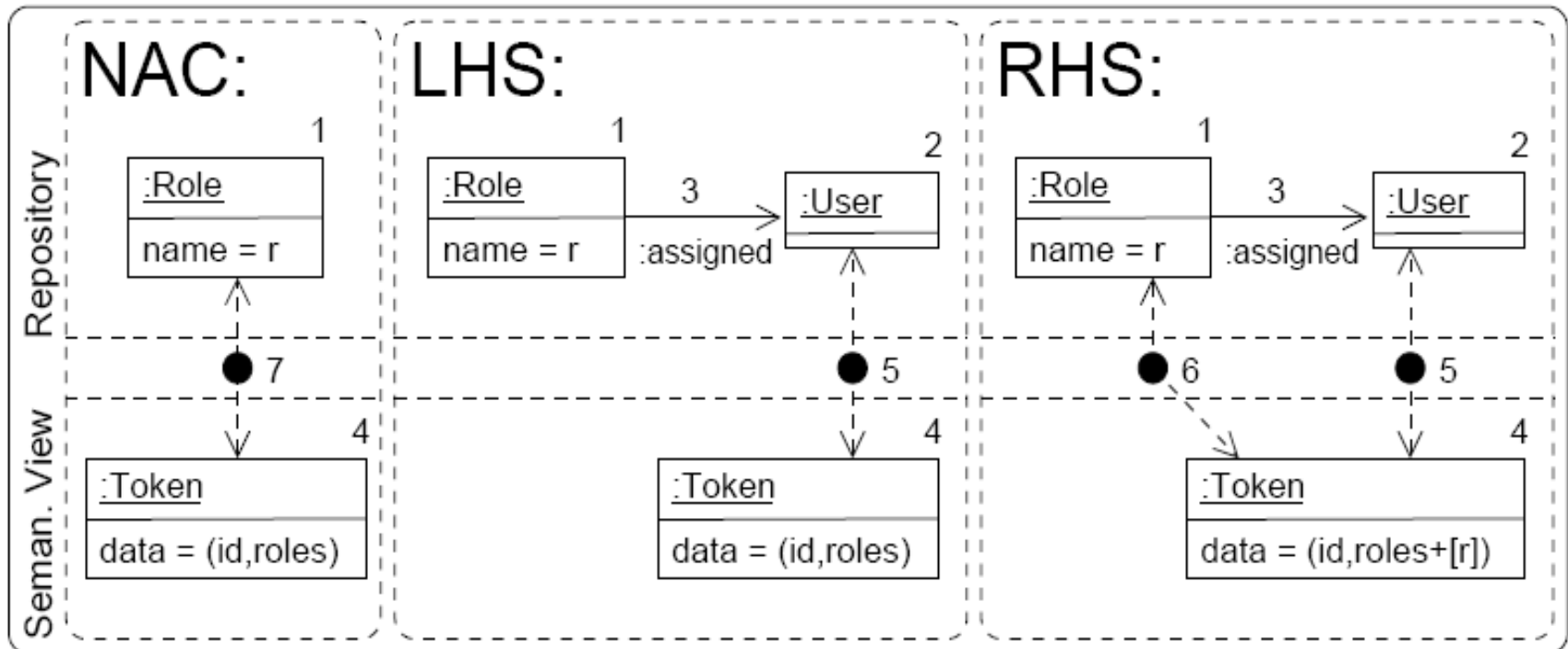
Users2Tokens:



Multi-view languages.

Coloured Petri Nets Semantic View

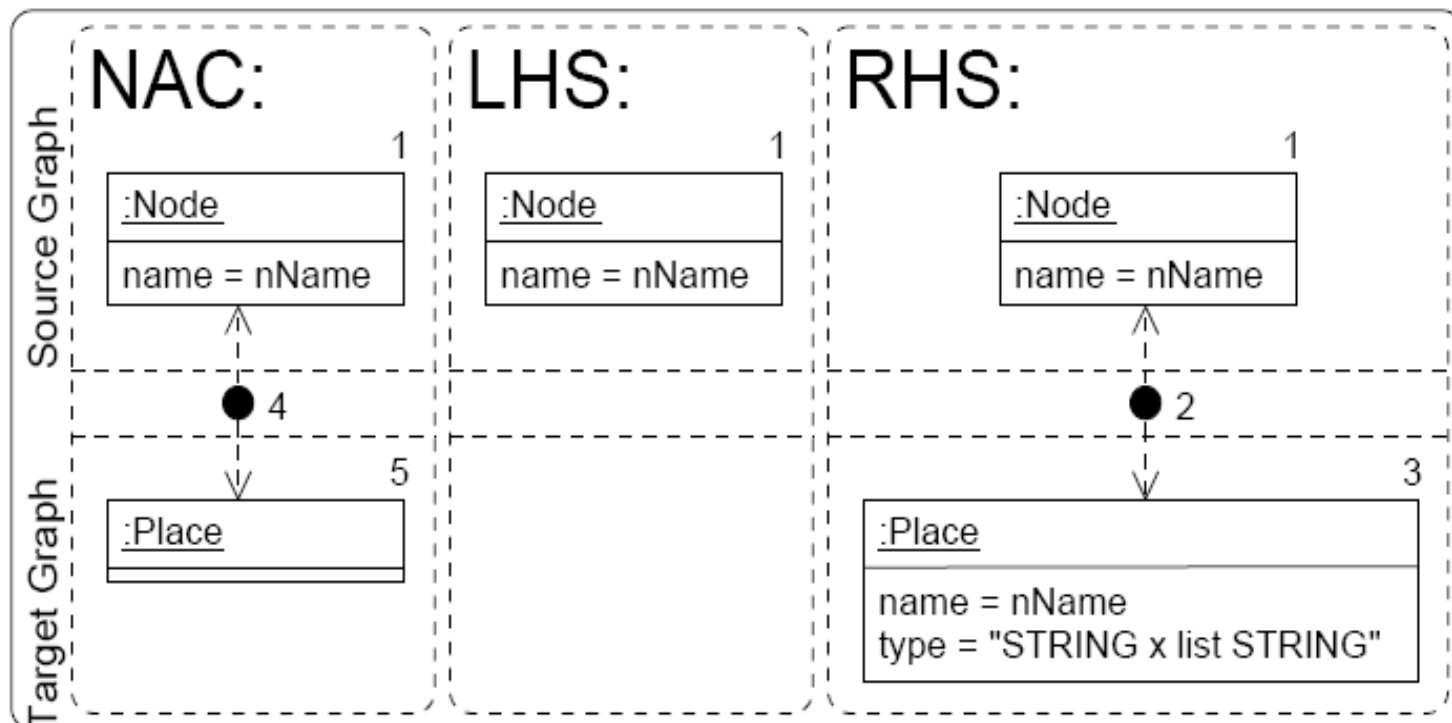
Assignments2DataValues:



Multi-view languages.

Coloured Petri Nets Semantic View

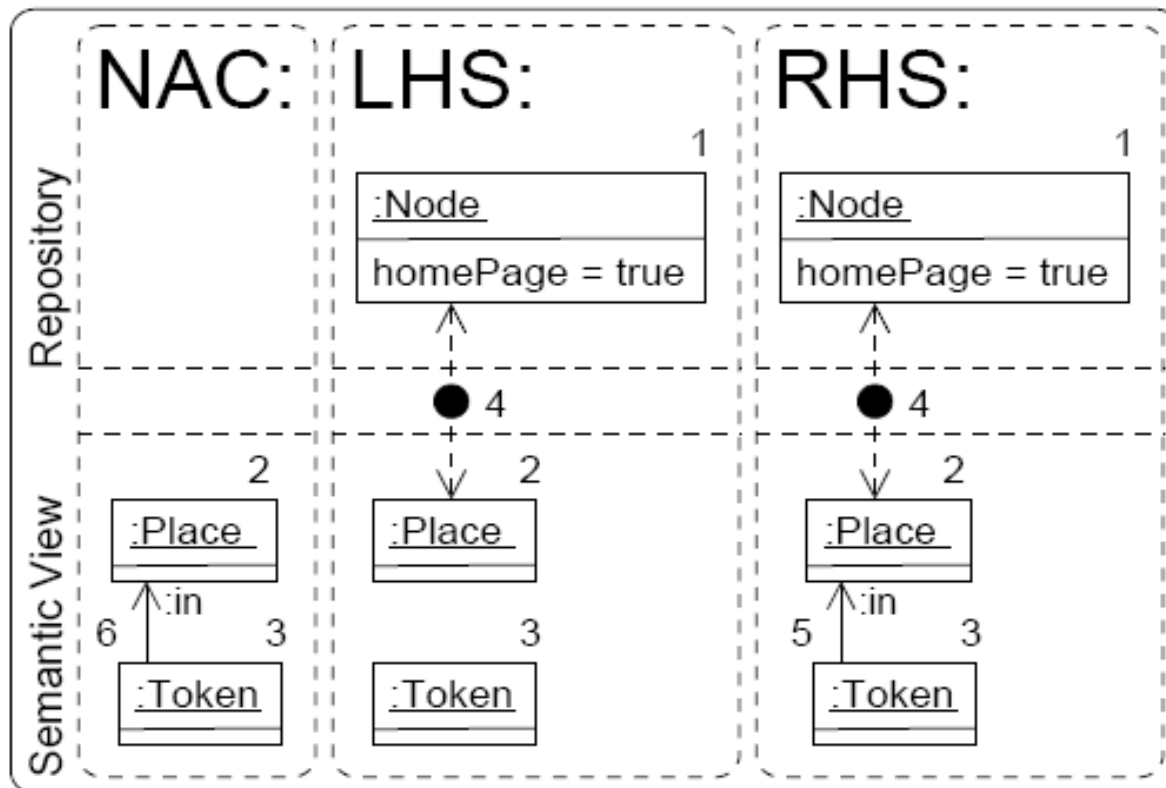
Nodes2Places:



Multi-view languages.

Coloured Petri Nets Semantic View

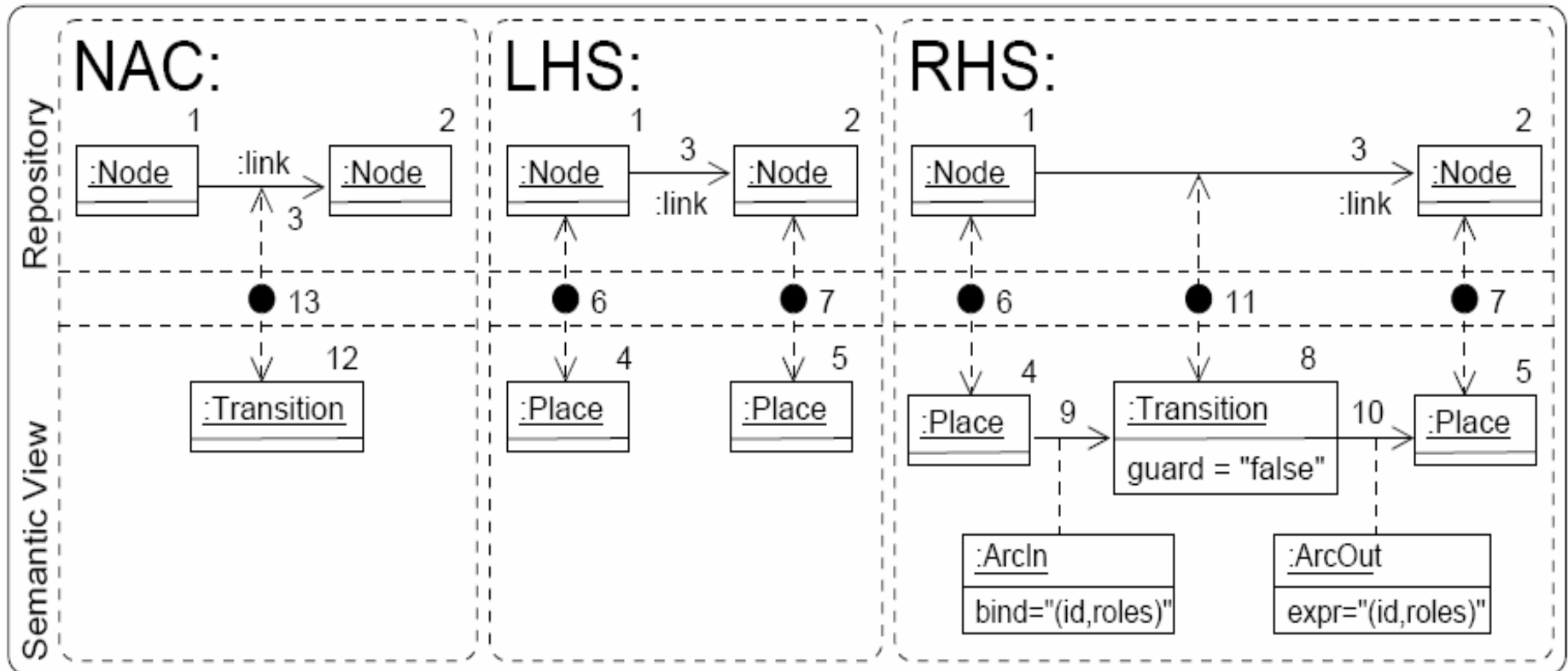
UsersAtHome:



Multi-view languages.

Coloured Petri Nets Semantic View

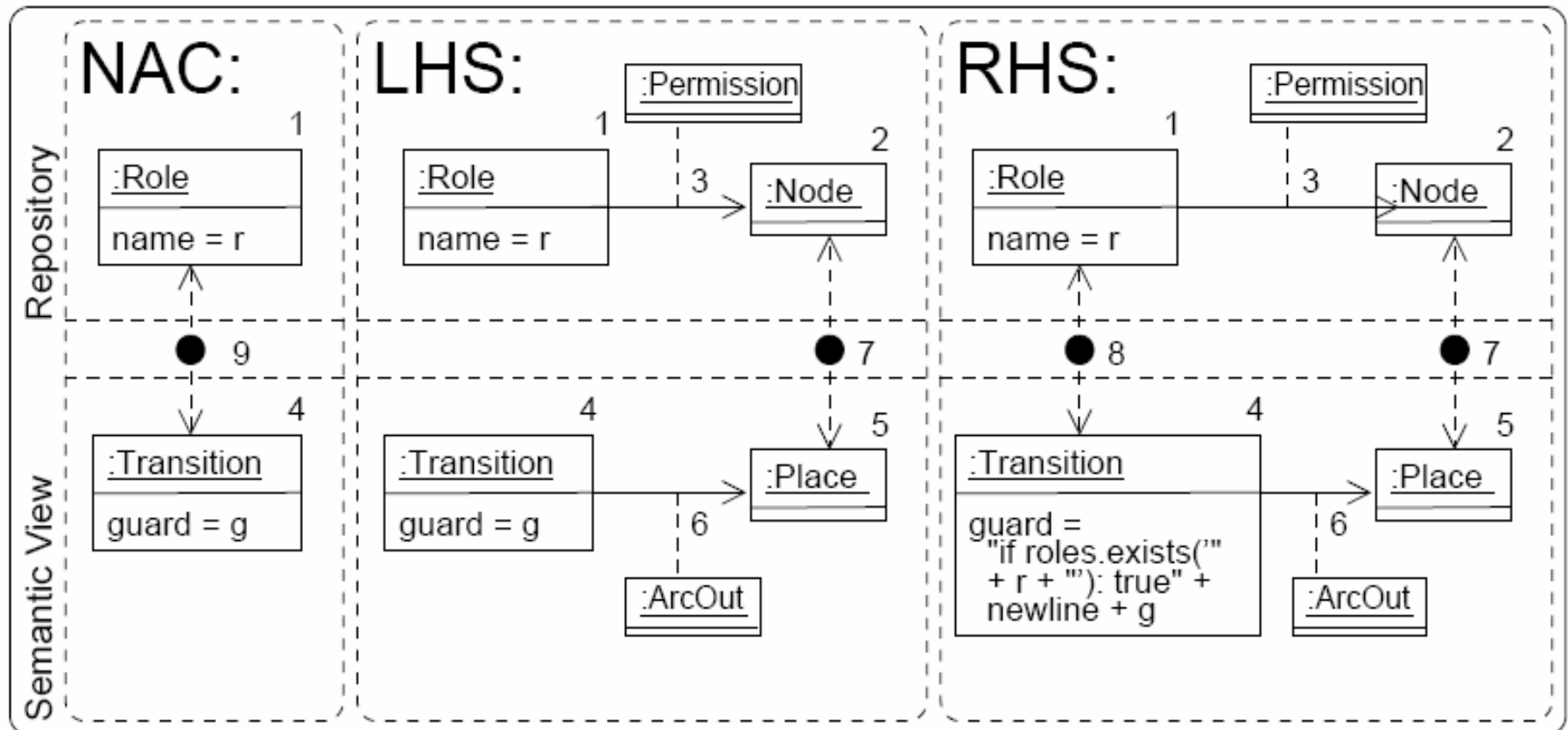
Links2Transitions:



Multi-view languages.

Coloured Petri Nets Semantic View

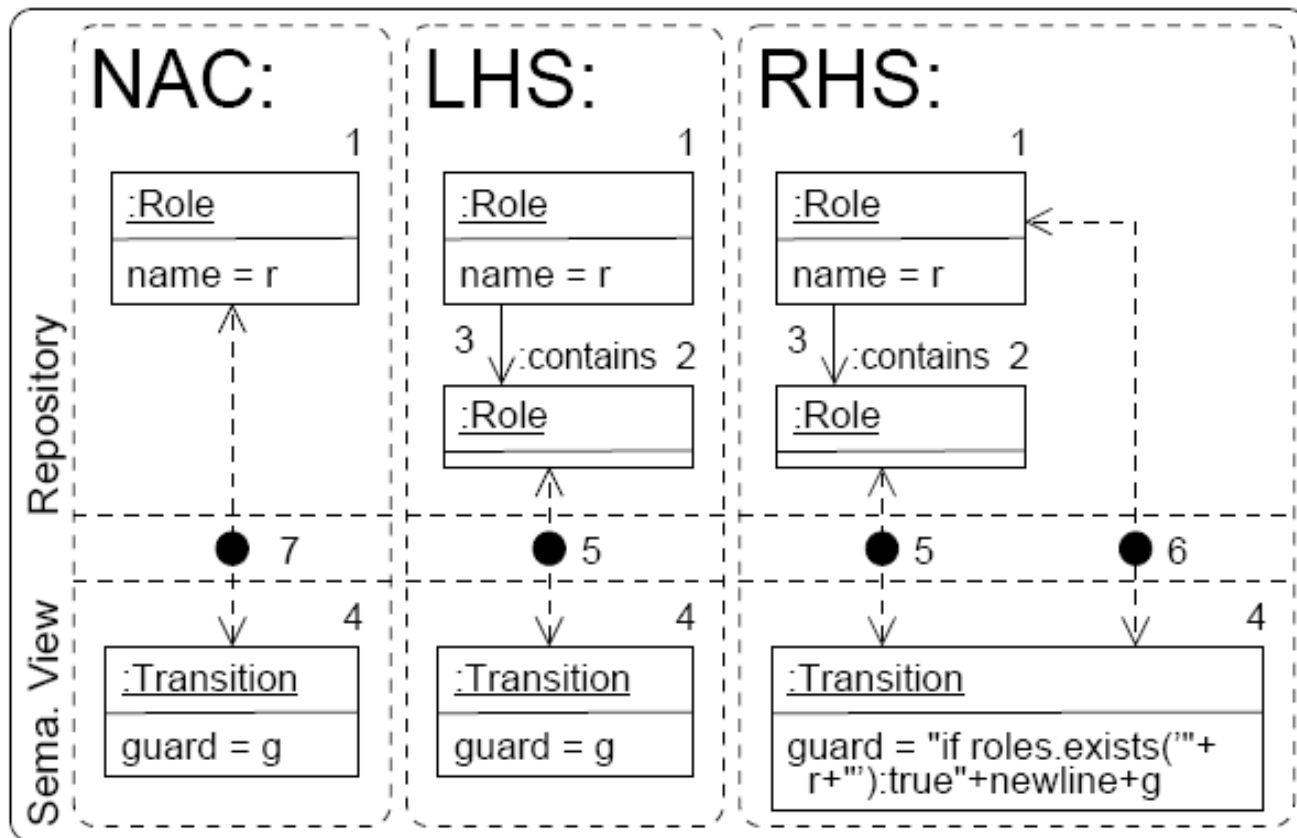
Permissions2Guards:



Multi-view languages.

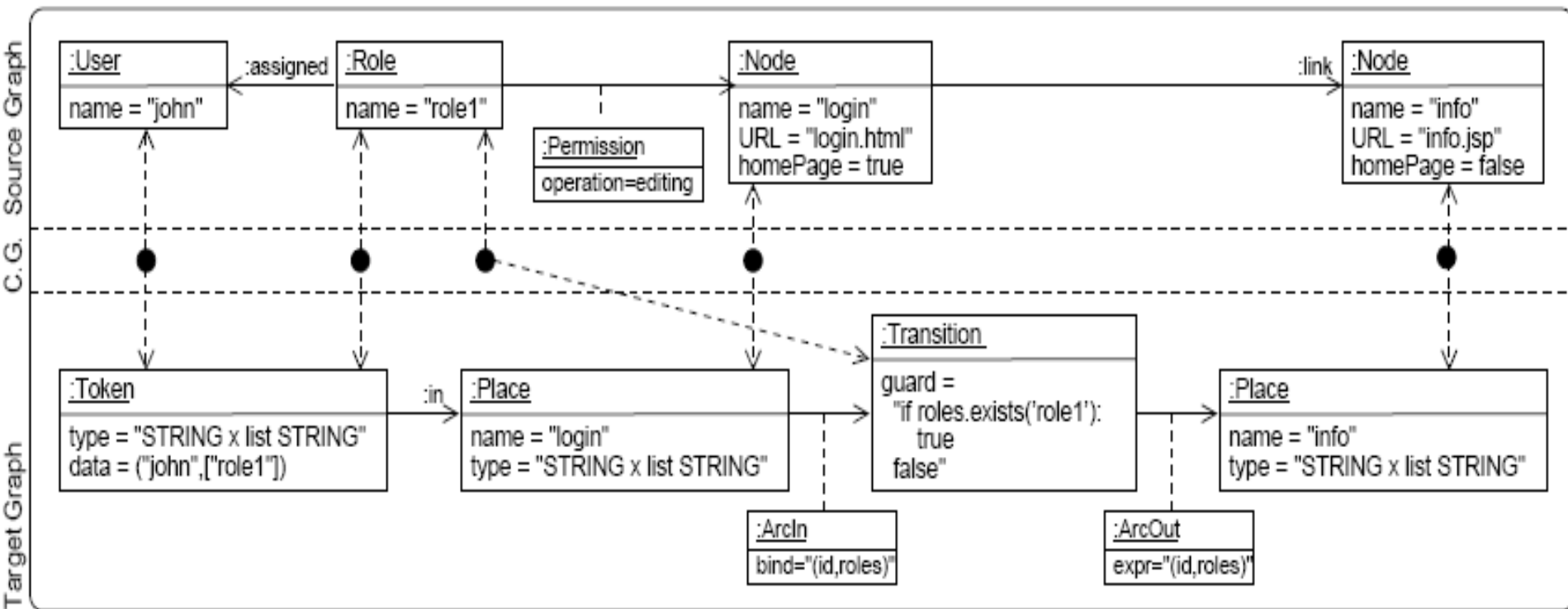
Coloured Petri Nets Semantic View

InheritedPermissions2Guards:



Multi-view languages.

Coloured Petri Nets Semantic View

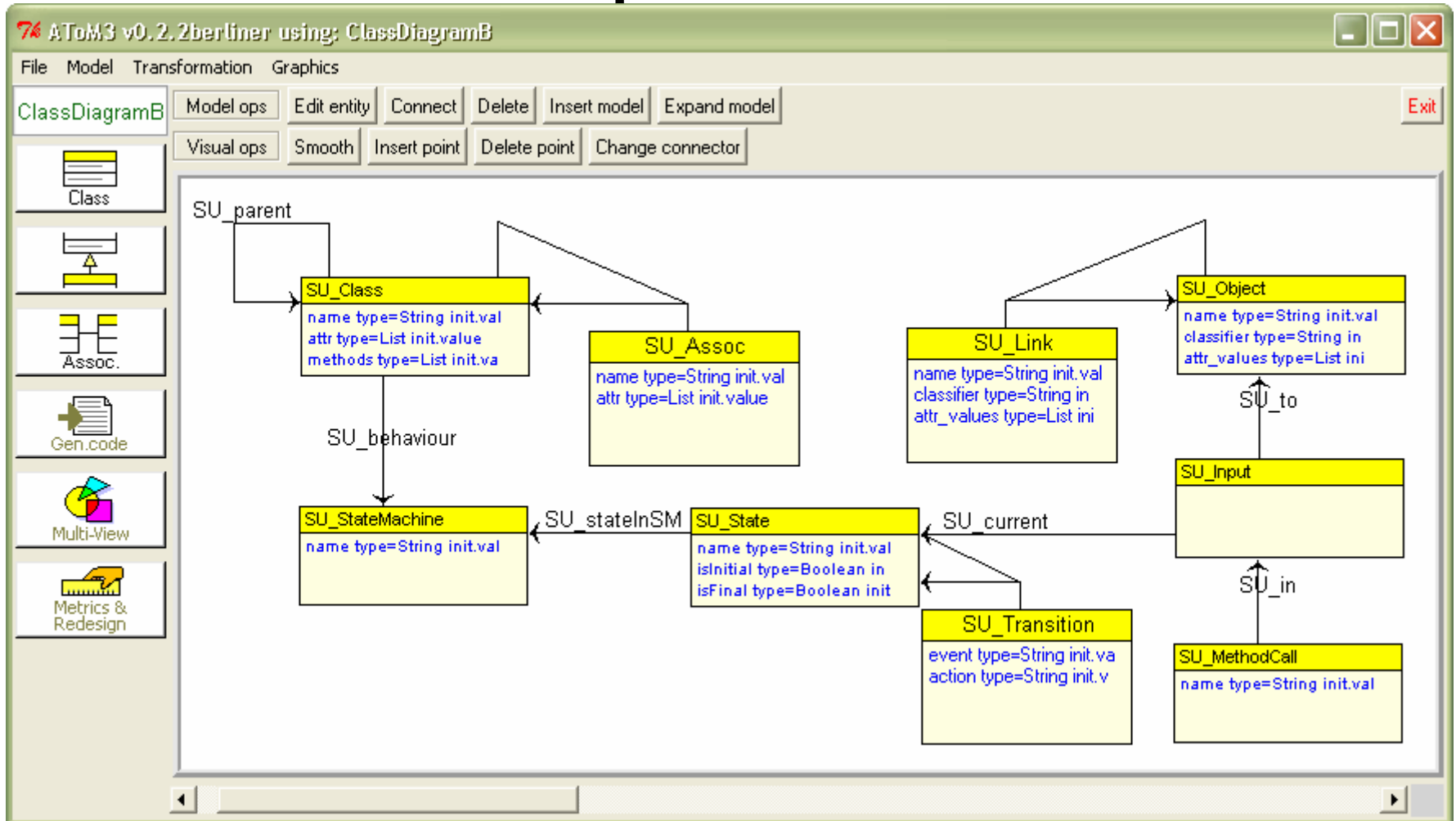


Agenda

- Defining Multi-view visual languages.
- Derived and Audience-oriented views.
- Semantic views.
- **Example.**
- Conclusions and Future work.

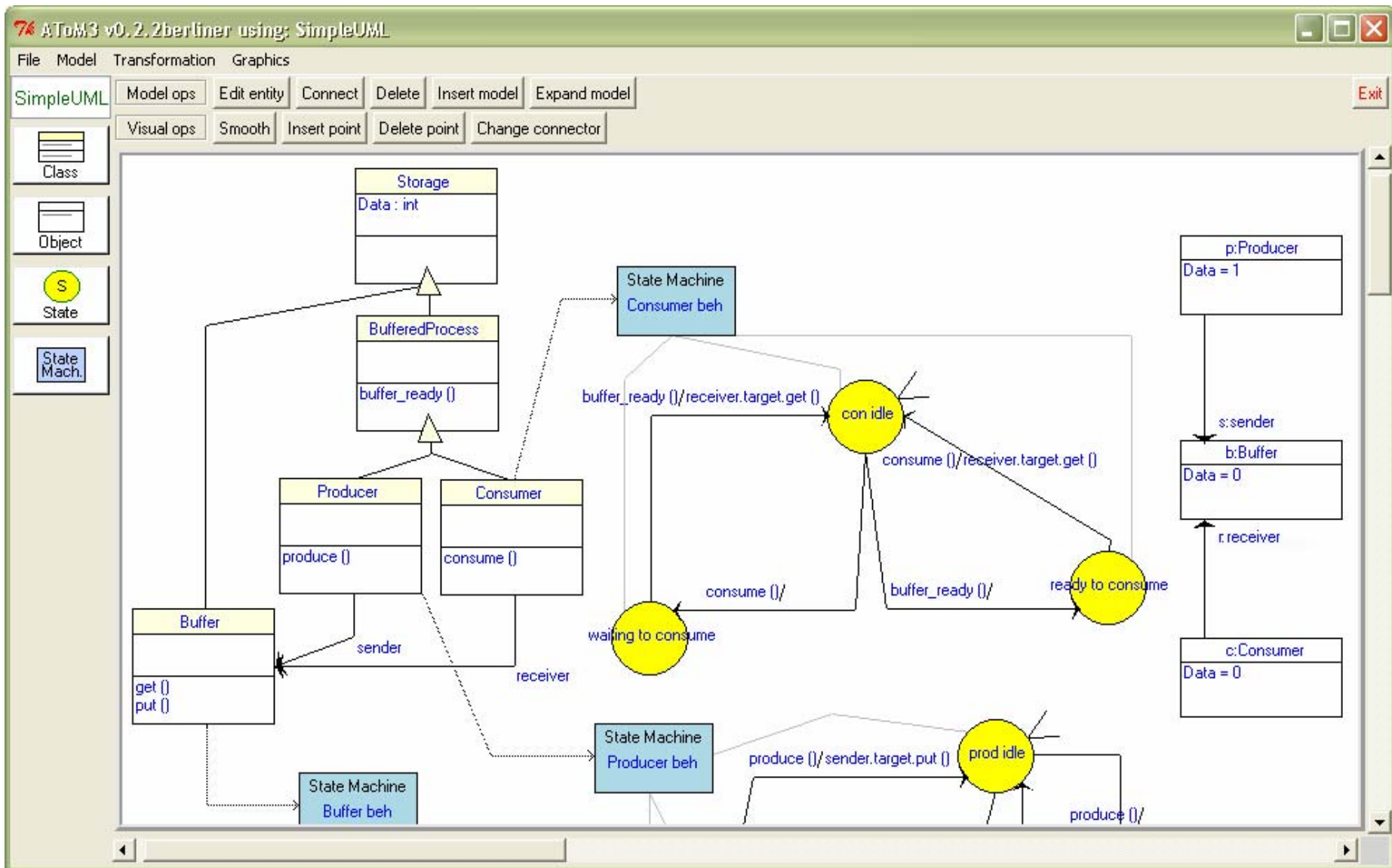


AToM³ example



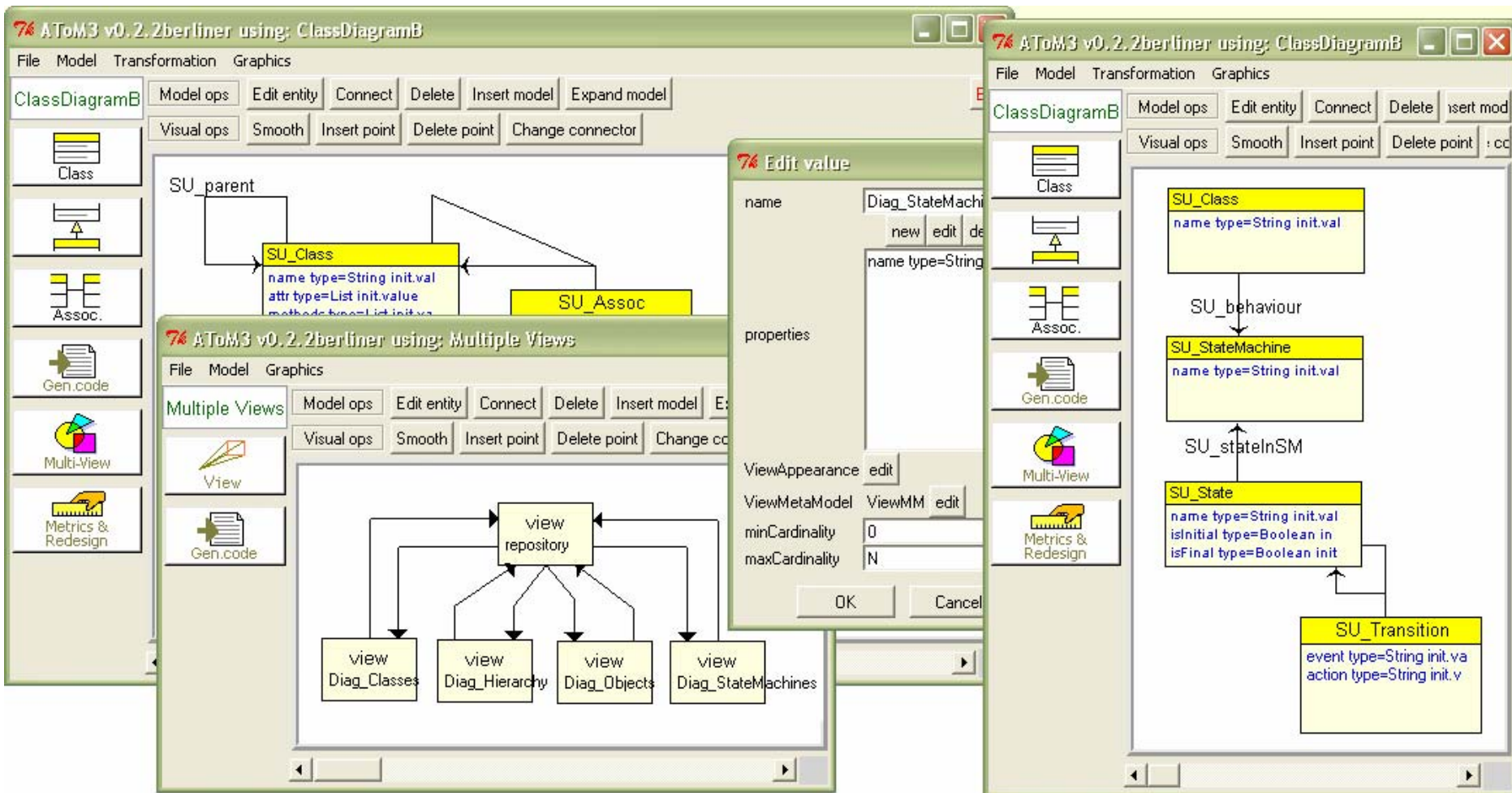
Single view meta-model

AToM³ example



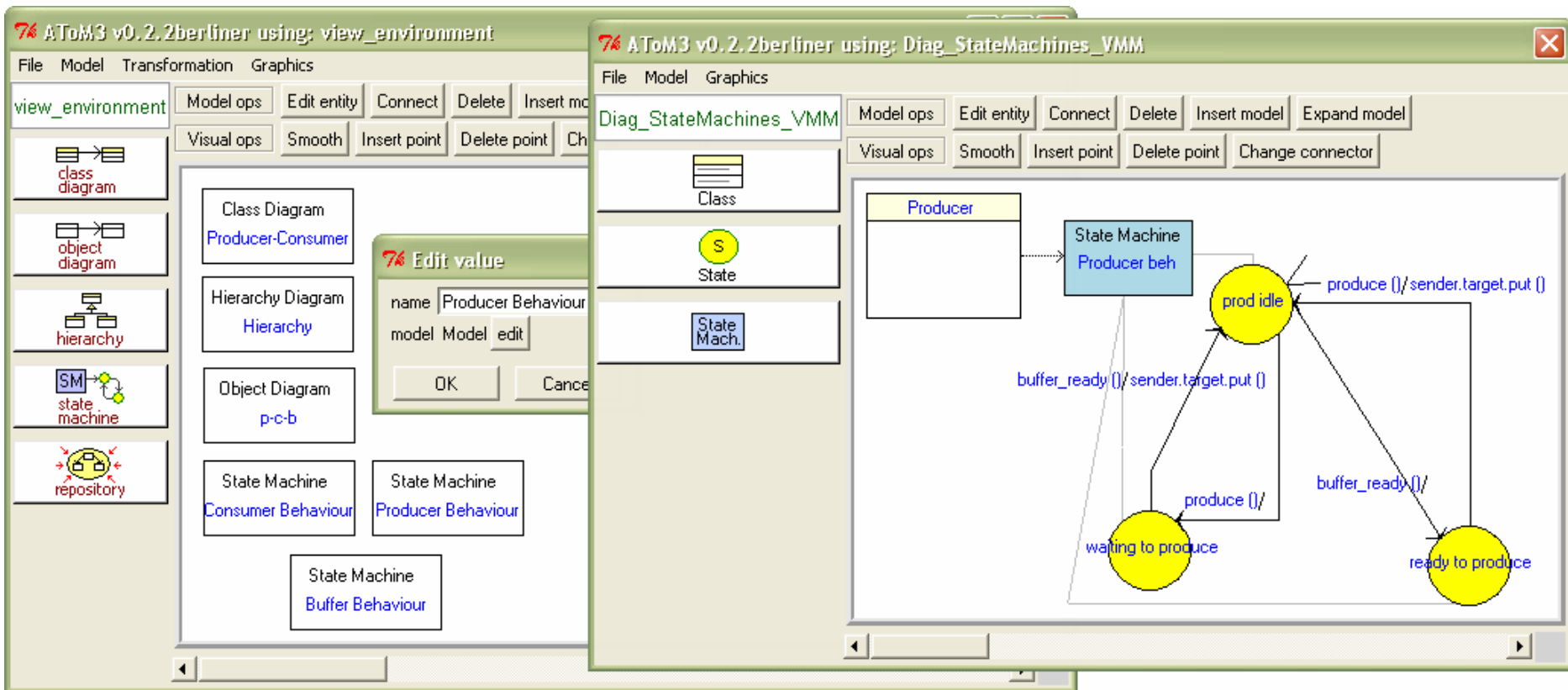
Generated environment

AToM³ example



Multi-View meta-model

AToM³ example



Generated multi-view environment

Agenda

- Defining Multi-view visual languages.
- Derived and Audience-oriented views.
- Semantic views.
- Example.



- **Conclusions and Future work.**

Conclusions

- Handling of Multi-view visual languages.
 - Based on meta-modelling and triple graph grammars.
- Different kinds of views:
 - System views.
 - Derived and audience oriented views.
 - Semantic views.
- Support for consistency.

Future work

- Implement graph query patterns in AToM³.
- Improve expressivity of graph query patterns.
- Improve efficiency (rule loading).
- Back annotation mechanisms.
- Generalization to multi- formalism modelling?
- Viewpoints which are not restrictions of the complete MVVL meta model.