UML Diagram for Graph Transformation Systems

Gabriele Taentzer, Michael Matz
TU Berlin
{gabi,matzmich}@cs.tu-berlin.de

November 2, 2000

At the APPLIGRAPH subgroup meeting in Paderborn (September 5-6, 2000) we discussed a common logical model for graphs and graph transformation systems. In the meantime, an updated version of the logical model for graphs has been provided by A. Schürr, S. E. Sim, R. Holt and A. Winter in [1]. They also proposed a new DTD based on this logical model there.

In this document an updated version of the logical model for graph transformation systems is presented which builds up on the new model for graphs in [1]. Furthermore, the main changes wrt. the original model are discussed.

Classes "Type", "Graph", AttributedElement", "TypeLabel" and "GraphElement" are described in [1].

Major changes wrt. the original model developed in Paderborn:

- Instead of two different classes "lhsgraph" and "rhsgraph" there is now one class "RuleGraph" which comprises not only a graph but also attribute and graph conditions. Each rule has two rule graphs such that also pre- and post-conditions can be expressed. A rule graph does not contain attribute assignments, but this should be possible already for graphs, i.e. each attribute can have a value being an expression.
- A condition independent of being an attribute condition or graph condition has a name and a kind. Furthermore, other conditions like the gluing condition are possible.

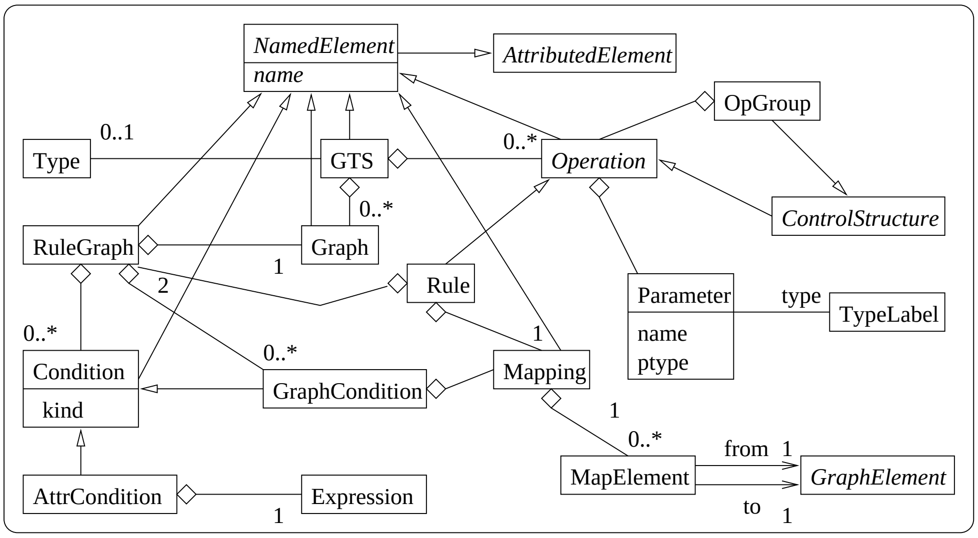

Figure 1: Updated version of the logical model for graph transformation

- A graph condition consists of a rule graph and a mapping. Since a rule graph can contain again graph conditions, also more complex conditions like implications can be stored.
- Similarly to attribute assignments also attribute conditions contain expressions, here Boolean expressions. Up to now, the format of an expression is left open.
- A parameter has a type now, i.e. it points to a type label.
- A mapping is not a map element anymore. Since we do not describe hierarchical graphs by refined graph elements anymore, also hierarchical mappings are not needed anymore.
- In the previous version it was possible to assign an operation to a schema element. In the newest graph model there is no distinction betweeen graph and schema anymore. So, the "belongsto" association is left out here and it has to be discussed in which way this relation can be included again.
- Already in the logical model for graphs there should be an abstract class *NamedElement* being an "AttributedElement" and having the attribute name already. Having such an abstract class available classes "GTS",

"Graph", "Condition", "Operation" and "Mapping" can inherit from that.

Based on this model the following tags should occur in a corresponding DTD:

```
<!ELEMENT GTS (Attr*,Graph*,Rule*,OpGroup*,Type?)>
<!-- Attr* could be: author, comments, version, etc. -->

<!ATTLIST GTS
          name NMTOKEN #REQUIRED>

<!ELEMENT OpGroup EMPTY>

<!ELEMENT Rule (RuleGraph, RuleGraph, Mapping, Parameter*, Attr*)>
               <!-- Attr* could be: embedding descriptions, etc. -->

<!ATTLIST Rule
          id    ID      #REQUIRED
          name  NMTOKEN #IMPLIED>

<!ELEMENT Mapping (Attr*, MapElem*)>
                   <!-- Attr* could be: mapping kind, etc. -->

<!ATTLIST Mapping
          name     NMTOKEN  #IMPLIED>

<!ELEMENT MapElem EMPTY>

<!ATTLIST MapElem
          from   IDREF #REQUIRED  <!-- Node | Edge | Relation -->
          to     IDREF #REQUIRED> <!-- Node | Edge | Relation -->

<!ELEMENT RuleGraph (Graph, AttrCondition*, GraphCondition*)>

<!ATTLIST RuleGraph
          id     ID    #REQUIRED>
```

```
<!ELEMENT GraphCondition (Attr*, RuleGraph, Mapping)>

<!ATTLIST GraphCondition
          name   NMTOKEN  #IMPLIED
          kind   NMTOKEN  #IMPLIED>

<!ELEMENT AttrCondition (Expression)>

<!ATTLIST AttrCondition
          name   NMTOKEN  #IMPLIED
          kind   NMTOKEN  #IMPLIED>

<!ELEMENT Expression (#PCDATA)>

<!ELEMENT Parameter (EMPTY)>

<!ATTLIST Parameter
          name   NMTOKEN         #REQUIRED
          type   IDREF           #REQUIRED  <!-- TypeLabel -->
          ptype  (in|out|inout) #REQUIRED>
```

References

[1] A. Schürr, S. E. Sim, R. Holt and A. Winter. The GXL Graph Exchange Language. Available at: http://www.gupro.de/GXL.