Universität Koblenz-Landau
Institut für Softwaretechnik
Re-Group

Graph Exchange Language

Andreas Winter

joint work with:
Ric Holt
Andy Schürr
Susan Sim

Contents

- Motivation and Idea
- Definition of GXL
- Exchanging graphs with GXL
- Exchanging schemas with GXL
- Conclusion

GXL Objective

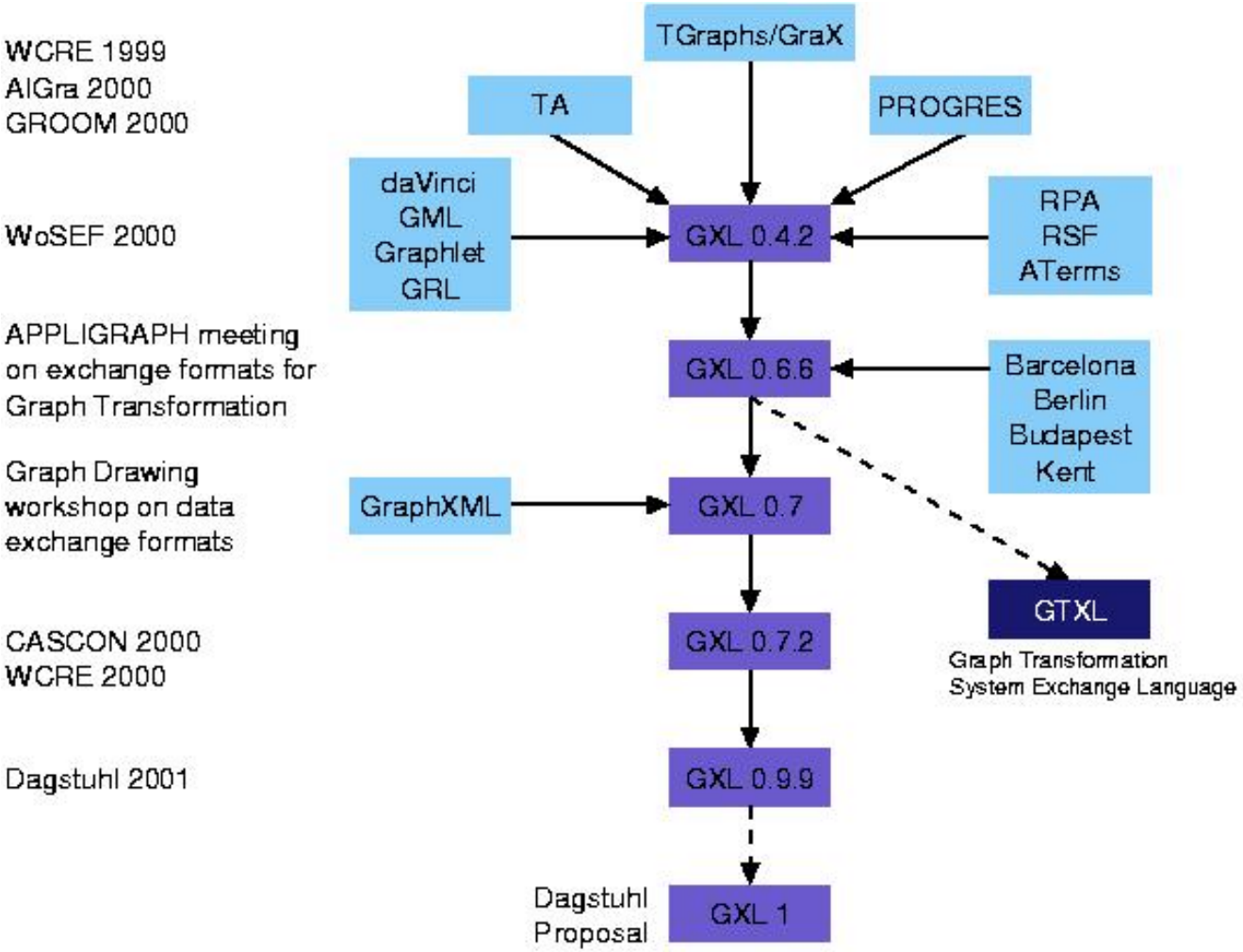
standard exchange language
- for interchanging data between reengineering tools

mathematical model
- typed, attributed, directed graphs

notation
- eXtensible Markup Language (XML)
- Unified Modeling Language (UML)

application to other areas in software engineering
- graph transformation
- graph drawing

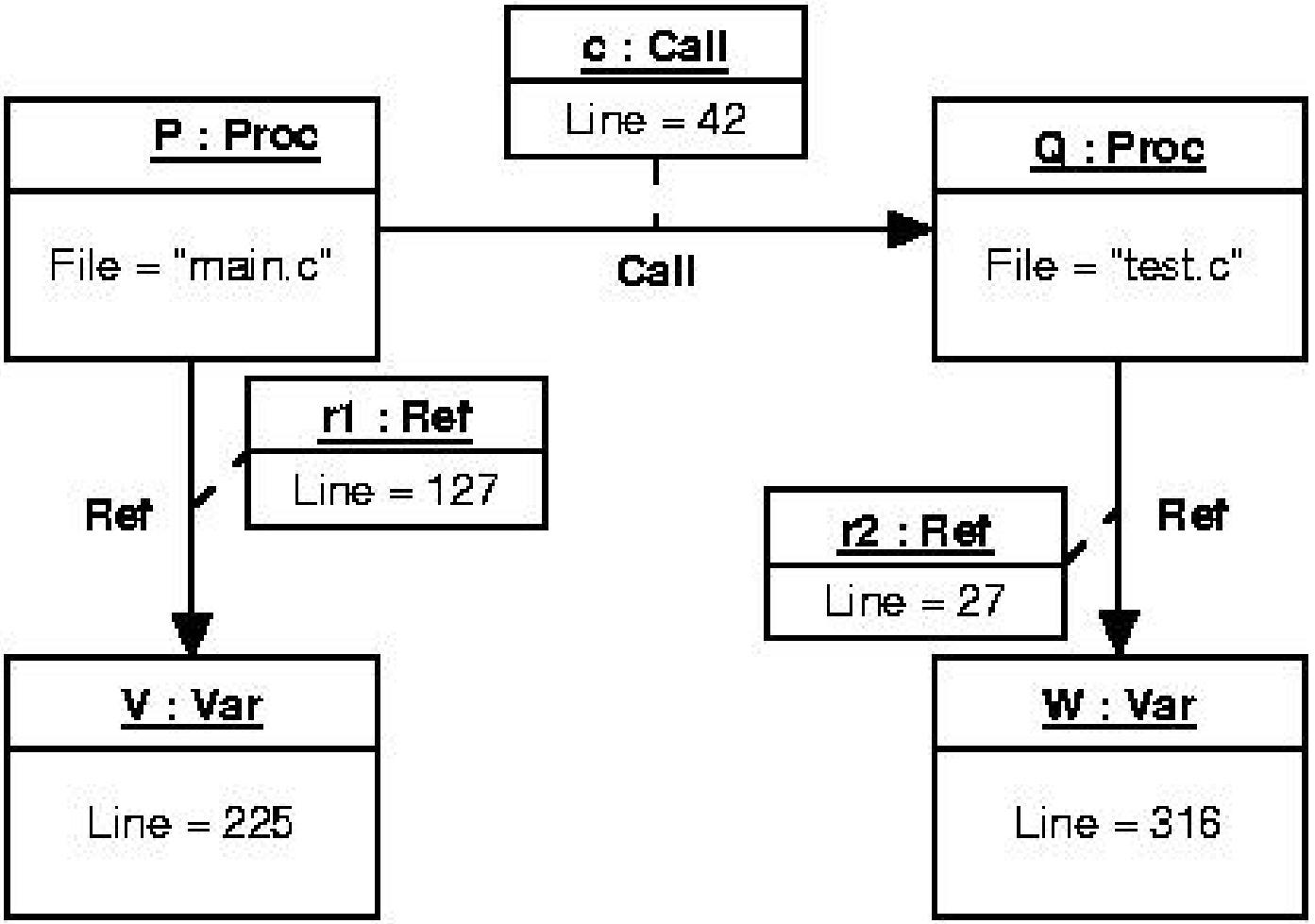
History


WCRE 1999
AIGra 2000
GROOM 2000
WoSEF 2000
APPLIGRAPH meeting on exchange formats for Graph Transformation
Graph Drawing workshop on data exchange formats
CASCON 2000
WCRE 2000
Dagstuhl 2001
Dagstuhl Proposal
TGraphs/GraX
TA
PROGRES
daVinci
GML
Graphlet
GRL
RPA
RSF
ATerms
GXL 0.4.2
GXL 0.6.6
Barcelona
Berlin
Budapest
Kent
GraphXML
GXL 0.7
GTXL
Graph Transformation System Exchange Language
GXL 0.7.2
GXL 0.9.9
GXL 1

GXL Partners

Bell Canada (Datrix Group), Canada
IBM Centre for Advanced Studies, Canada
Mahindra British Telecom, India
Nokia Research Center (Software Technology Laboratory), Finland
Philips Research (Software Architecture Group), The Netherlands
RWTH Aachen (Department of Computer Science III), Germany
TU Berlin (Theoretical CS/Formal Specification Group), Berlin
University of Berne (Software Composition Group), Switzerland
University Bw München (Institute for Software Technology), Germany
University of Koblenz (IST, GUPRO), Germany
University of Oregon (Department of Computer Science), U.S.A.
University of Paderborn (AG Softwaretechnik), Germany
University of Stuttgart (BAUHAUS Group), Germany
University of Victoria (RIGI Group), Canada
Universities of Waterloo and Toronto (SWAG), Canada

GXL Example


c : Call
Line = 42
P : Proc
File = "main.c"
Call
Q : Proc
File = "test.c"
r1 : Ref
Line = 127
Ref
r2 : Ref
Line = 27
Ref
V : Var
Line = 225
W : Var
Line = 316

GXL Example

```
<gxl>
 <graph>
  <node id = "P" >
   <attr name = "File">
    <string> main.c </string> </attr>
  </node>
  <node id = "Q" >
   <attr name = "File">
    <string> test.c </string> </attr>
  </node>
   <node id = "V" >
   <attr name = "Line">
    <int> 225 </int> </attr>
  </node>
  <node id = "W" >
   <attr name = "Line">
    <int> 316 </int> </attr>
  </node>
  <edge id = "r1"
        from = "P" to = "V">
   <attr name = "Line">
    <int> 127 </int> </attr>
  </edge>
  <edge id = "r2"
        from = "Q" to = "W">
   <attr name = "Line">
    <int> 27 </int> </attr>
  </edge>
  <edge id = "c"
    from = "P" to = "Q">
   <attr name = "Line">
     <int> 316 </int></attr>
  </edge>
 </graph>
</gxl>
```

Dimensions of Reengineering Data

Programming Languages

- single Languages (Ada, C, C++, Cobol, Java)
- multi-language systems

Level of Abstraction

- AST-level
- Architectural level

Relational Aspects

- Dataflow, Controlflow, ...
- Includes, Calls, Uses, ...

Definition of GXL

Requirements for Exchange Formats

- independent from
  - specific reengineering dimensions
  - specific reengineering applications
  - specific reengineering tools
- concrete enough to be interpreted by different reengineering tools

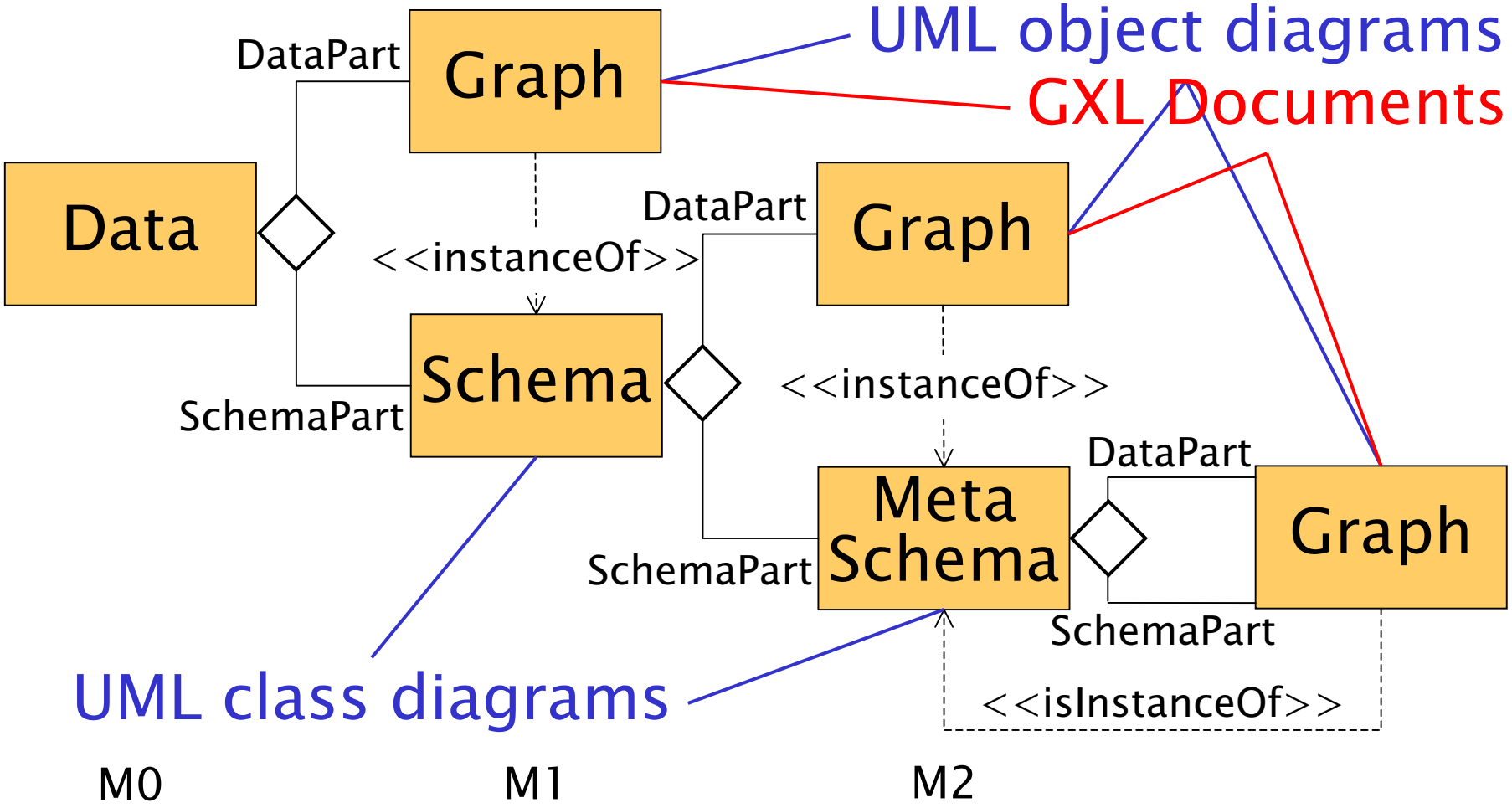
GXL Approach

- exchanging *instance data* and *schema data*

GXL First Directive

Everything is a typed, attributed, directed graph

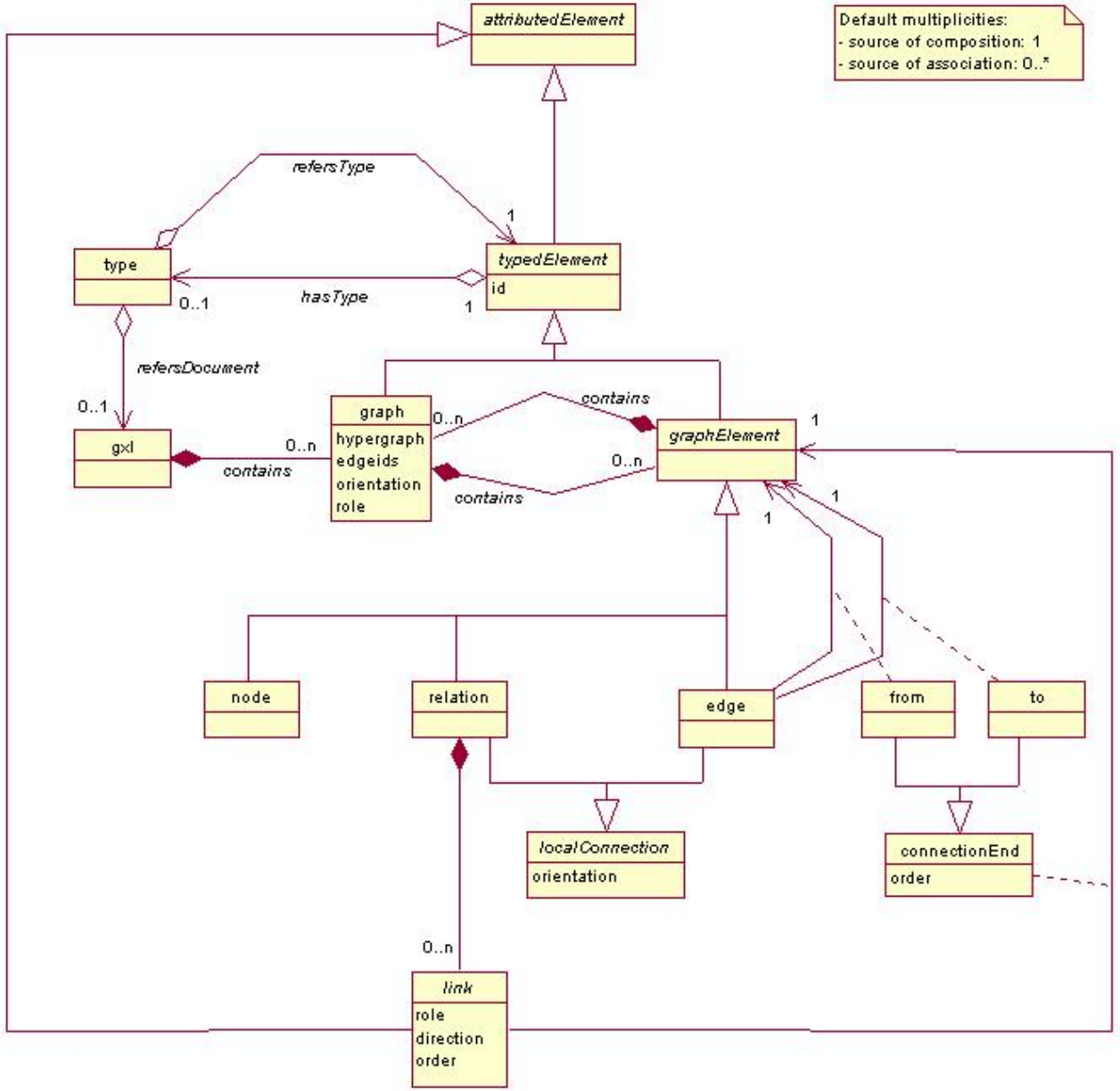
GXL Idea


UML object diagrams
GXL Documents
DataPart
Graph
Data
<<instanceOf>>
DataPart
Graph
Schema
SchemaPart
<<instanceOf>>
DataPart
Meta Schema
Graph
SchemaPart
SchemaPart
<<isInstanceOf>>
UML class diagrams
M0
M1
M2

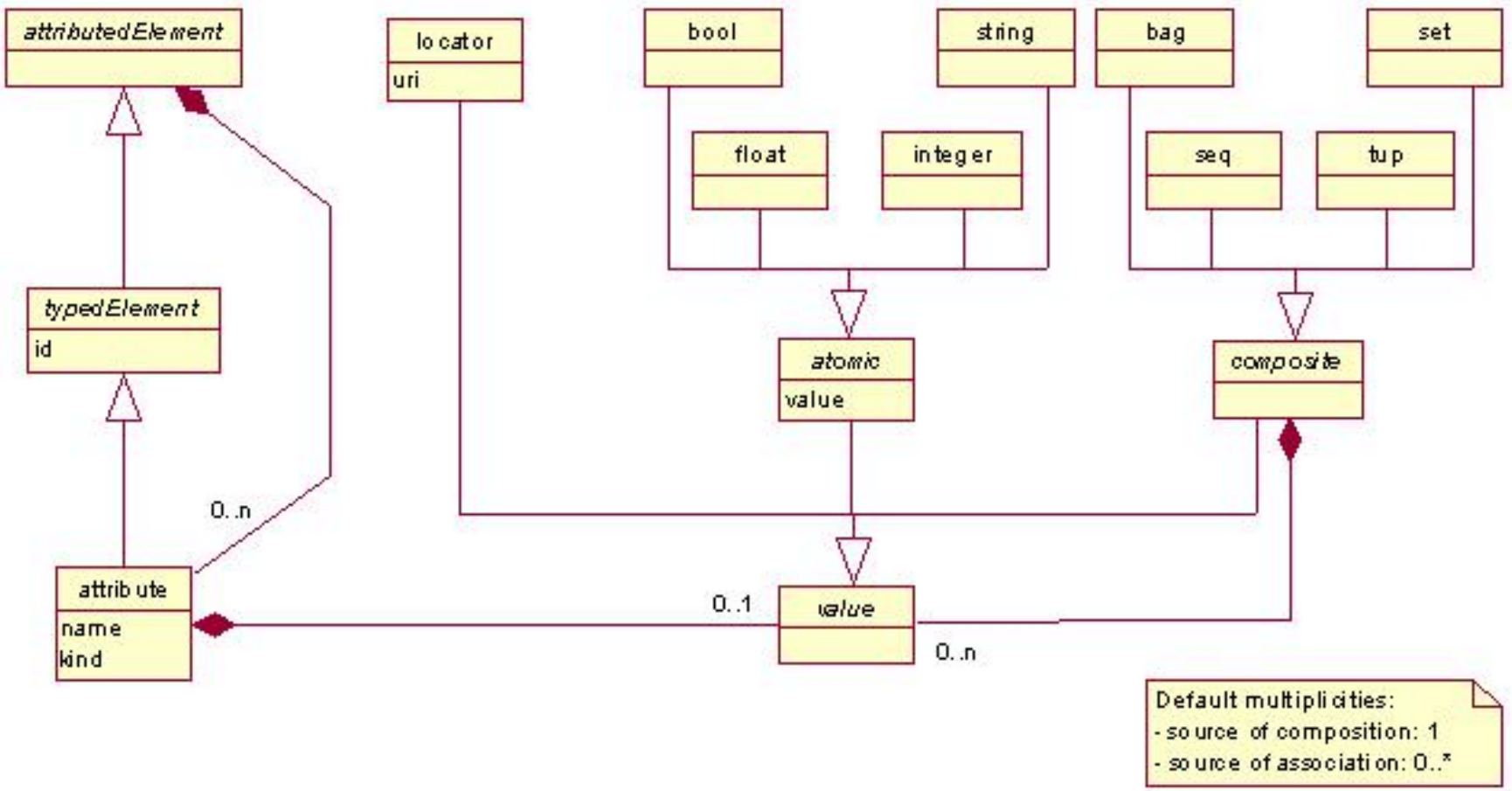
Definition of GXL

- simple and compact representation of *directed and undirected graphs*
- suitable for a *broad spectrum of graph models*,
  - including support for hypergraphs and
  - support for hierarchical graphs
- support for representation of *complex attribute* values
- *extensibility* of graph elements and attribute types
- *uniform representation* for graphs and graph schemas

GXL Graph Model



GXL Attribute Model



GXL Document Type Definition

automatic generation of GXL DTD

- ☺ follows MOF/XMI standard
- ☹ entity types and attributes are not distinguished
- ☹ blows up the number of XML elements (GXL 0.7.2: 136/74 elements)

manual definition of GXL DTD

- ☹ manual work to do
- ☺ distinction between entity types and attributes reflects design decisions
- ☺ small and simple DTD (23 elements)

GXL Document Type Definition (0.99)

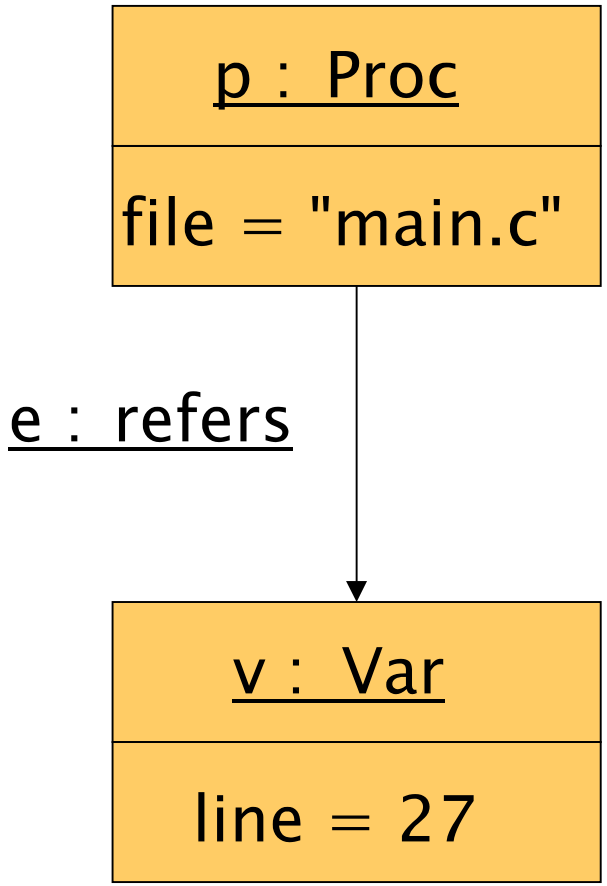
```
<!ENTITY % *-extension  "" >
<!ELEMENT gxl  (%gxl-extension; graph*) >
<!ATTLIST  gxl xmlns:xlink      CDATA   #FIXED
                "www.w3.org/1999/xlink">
<!ELEMENT type EMPTY>
<!ATTLIST  type xlink:type  (simple)      #FIXED "simple"
                xlink:href      CDATA    #REQUIRED>
<!ELEMENT graph (%graph-extension;
                type? , attr* , ( node | edge | rel )*) >
<!ATTLIST  graph id             ID          #REQUIRED
                 role           NMTOKEN  #IMPLIED
                 edgeids        ( true | false ) "false"
                 hypergraph  ( true | false ) "false"
           orientation ( directed | undirected ) "directed">
<!ELEMENT node (%node-extension; type? , attr*, graph*) >
<!ATTLIST  node  id             ID        #REQUIRED>
<!ELEMENT edge (%edge-extension; type?, attr*, graph*) >
<!ATTLIST  edge  id             ID          #IMPLIED
                 from           IDREF     #REQUIRED
                 to             IDREF     #REQUIRED
                 fromorder      CDATA    #IMPLIED
                 toorder        CDATA    #IMPLIED
           orientation ( directed | undirected ) #IMPLIED>
<!ELEMENT rel (%rel-extension;
           type? , attr*, graph*, link* ) >
<!ATTLIST  rel     id           ID            #IMPLIED
           orientation ( directed | undirected ) #IMPLIED>
<!ELEMENT link (%link-extension; attr*) >
<!ATTLIST  link    ref          IDREF       #REQUIRED
                   role         NMTOKEN   #IMPLIED
                   direction    ( in | out | none )  #IMPLIED
                   startorder   CDATA     #IMPLIED
                   endorder     CDATA     #IMPLIED >
<!ELEMENT attr (type?, attr*, (%val;)) >
<!ATTLIST  attr    id           IDREF       #IMPLIED
                   name         NMTOKEN   #REQUIRED
                   kind         NMTOKEN   #IMPLIED >
<!ENTITY % val " %value-extension; locator | bool |int |
     float | string | seq | set | bag |tup ">
<!ELEMENT locator  EMPTY >
<!ATTLIST  locator xlink:type  (simple)    #FIXED  "simple"
                   xlink:href  CDATA       #IMPLIED >
<!ELEMENT bool |int | float | string   (#PCDATA) >
<!ELEMENT seq | set | bag | tup  (%val;)* >
```

Exchanging Graphs with GXL

- Attributed, typed, directed Graphs
- Undirected Graphs
- Ordered Graphs
- Hypergraphs
- Hierarchical Graphs

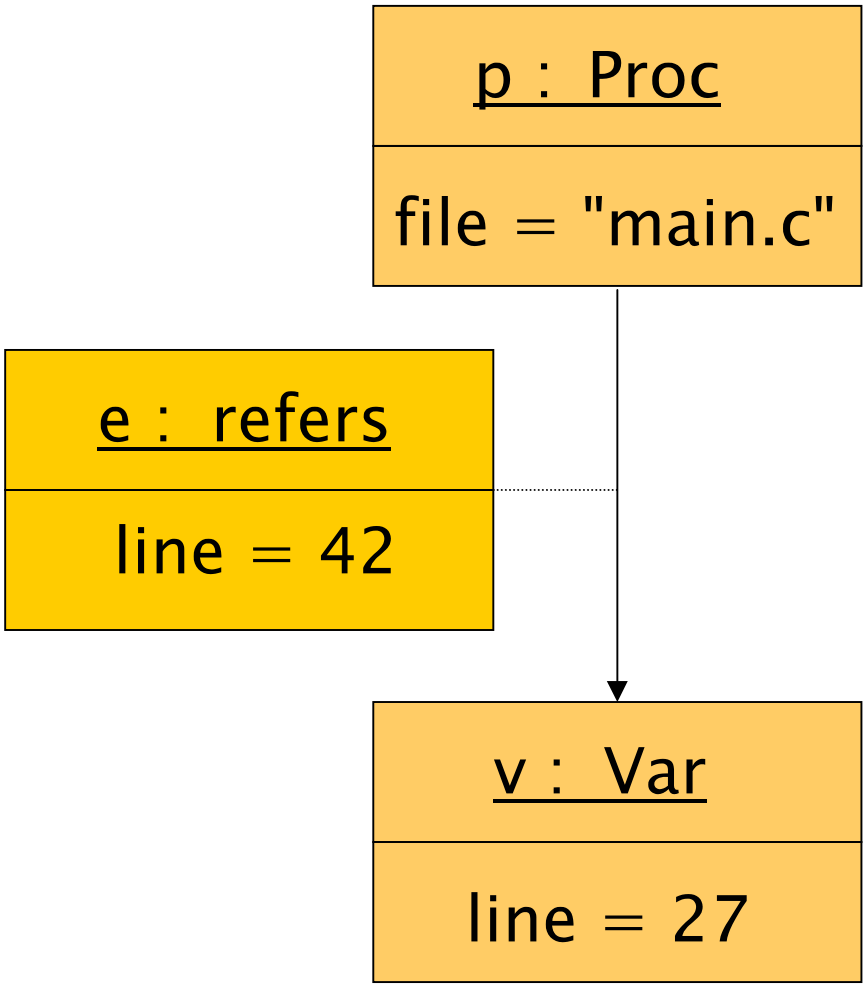
Typed, Attributed, Directed Graphs

typed attributed graph



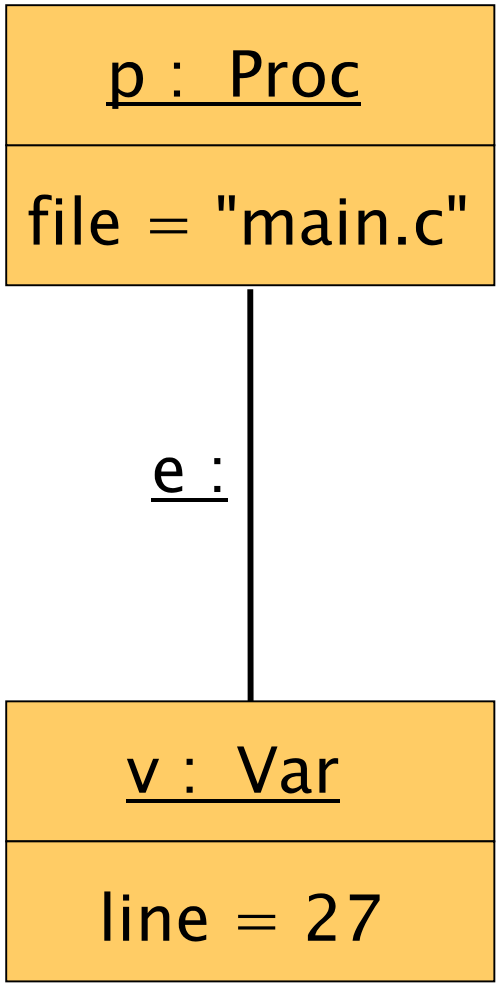

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "gxl.dtd">
<gxl><graph>
<node id = "p">
  <type xlink:href =
        "schema.gxl#Proc"/>
  <attr name = "file">
    <string>main.c</string></attr>
</node>
<node id = "v">
  <type xlink:href =
        "schema.gxl#Var"/>
  <attr name = "line">
    <int>27</int></attr>
</node>
<edge id = "e"
  from = "p" to = "v">
  <type xlink:href =
        "schema.gxl#refers"/>
</edge>
<graph></gxl>
```

Attributed Edges




```
<edge id = "e"
  from = "p" to = "v">
  <type xlink:href =
      "schema.gxl#refers"/>
  <attr name = "line">
    <int>42</int>
  </attr>
</edge>
```

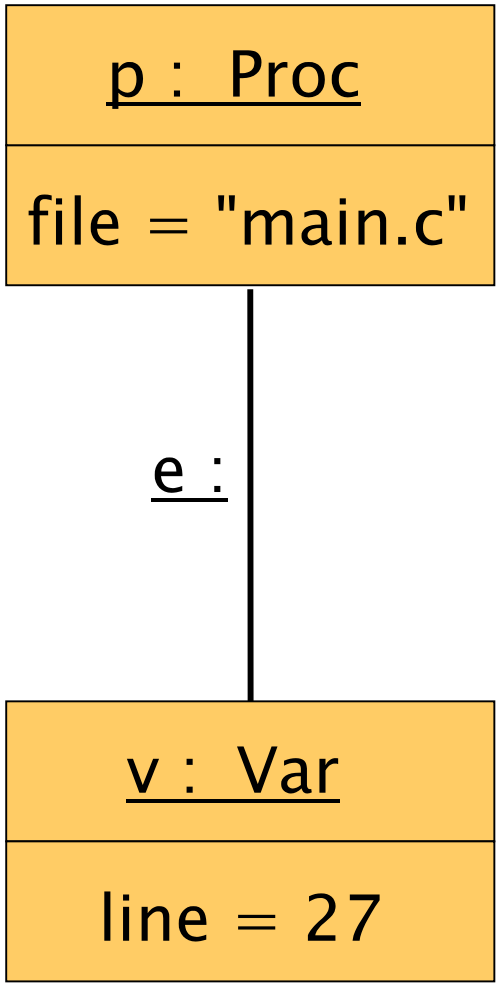
Undirected Graphs



edges are always noted as directed
but can be interpreted as undirected

```
<graph
      orientation = "undirected" >
...
<edge id = "e"
      from = "p" to = "v">
</edge>
...
</graph>
```

Undirected Graphs



edges are always noted as directed
but can be interpreted as undirected

undirected edge
in directed graph

```
<graph
      orientation = "directed" >
...
<edge id = "e"
      orientation = "undirected"
      from = "p" to = "v">
</edge>
...
</graph>
```

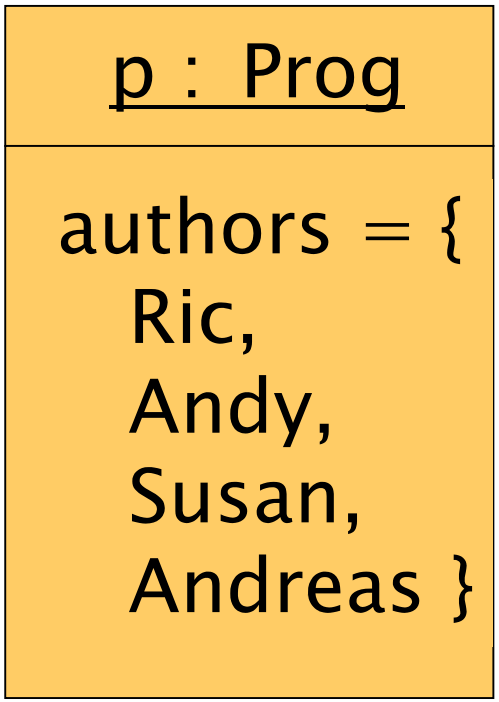
GXL Documents (DTD)

```
<!ELEMENT gxl  (%gxl-extension; graph*) >
<!ATTLIST  gxl  xmlns:xlink CDATA          #FIXED
                    "www.w3.org/1999/xlink">

<!ELEMENT graph (%graph-extension;
                    type? , attr* , ( node | edge | rel )*) >
<!ATTLIST  graph
      id             ID             #REQUIRED
      role           NMTOKEN    #REQUIRED
      edgeids        ( true | false ) "false"
      hypergraph ( true | false ) "false"
      direction      ( directed | undirected ) "directed" >
```

Nodes and Edges (DTD)

```
<!ELEMENT node (%node-extension;
                type? , attr*, graph*) >
<!ATTLIST  node
      id            ID              #REQUIRED >
<!ELEMENT edge (%edge-extension;
                type? , attr*, graph*)>
<!ATTLIST  edge
      id            ID              #IMPLIED
      from          IDREF           #REQUIRED
      to            IDREF           #REQUIRED
      fromorder     CDATA           #IMPLIED
      toorder       CDATA           #IMPLIED
      orientation   (directed|undirected ) #IMPLIED>
```

Attribute Types




```
<node id = "p">
  <type xlink:href =
      "schema.gxl#prog"/>
  <attr name = "authors">
    <set>
      <string>Ric</string>
      <string>Andy</string>
      <string>Susan</string>
      <string>Andreas</string>
    </set>
  </attr>
</node>
```

Attribute Types

GXL supports

- set,
- sequence,
- bag,
- tuple

of

- bool,
- int,
- float,
- string,
- references (URI),
- complex values

```
<node id = "p">
  <type xlink:href =
      "schema.gxl#prog"/>
  <attr name = "authors">
    <set>
      <string>Ric</string>
      <string>Andy</string>
      <string>Susan</string>
      <string>Andreas</string>
    </set>
  </attr>
</node>
```

Attributes (DTD)

```
<!ENTITY % val "  %value-extension;
      locator |
      bool | int | float | string |
      seq | set | bag | tup ">

<!ELEMENT attr (type?,  attr*, (%val;)) >
<!ATTLIST  attr
      ID          IDREF          #IMPLIED
      name        NMTOKEN        #REQUIRED
      kind        NMTOKEN        #IMPLIED >
```

Attribute Values (DTD)

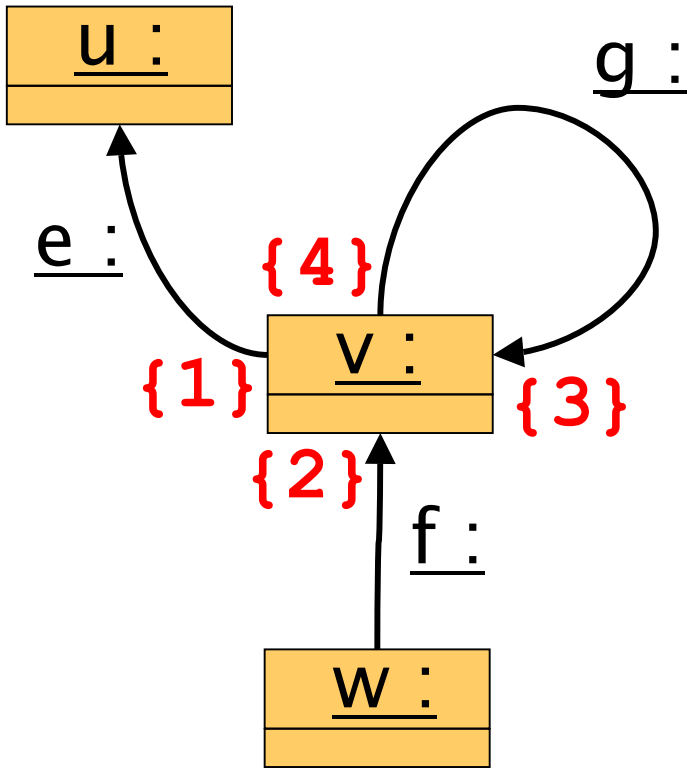
```
<!ELEMENT locator  EMPTY >
<!ATTLIST locator
   xlink:type  (simple)   #FIXED  "simple"
   xlink:href  CDATA      #IMPLIED >

<!ELEMENT     bool    (#PCDATA) >
<!ELEMENT     int     (#PCDATA) >
<!ELEMENT     float   (#PCDATA) >
<!ELEMENT     string  (#PCDATA) >

<!ELEMENT     seq     (%val;)* >
<!ELEMENT     set     (%val;)* >
<!ELEMENT     bag     (%val;)* >
<!ELEMENT     tup     (%val;)* >
```

links to external objects

simple attributes

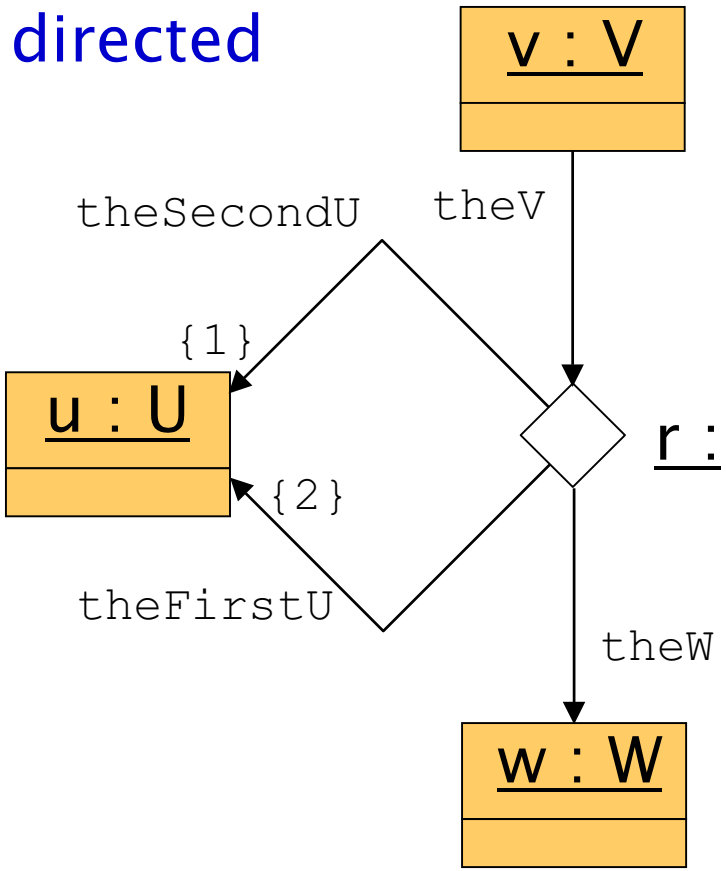
complex attributes

Ordering of Incidences



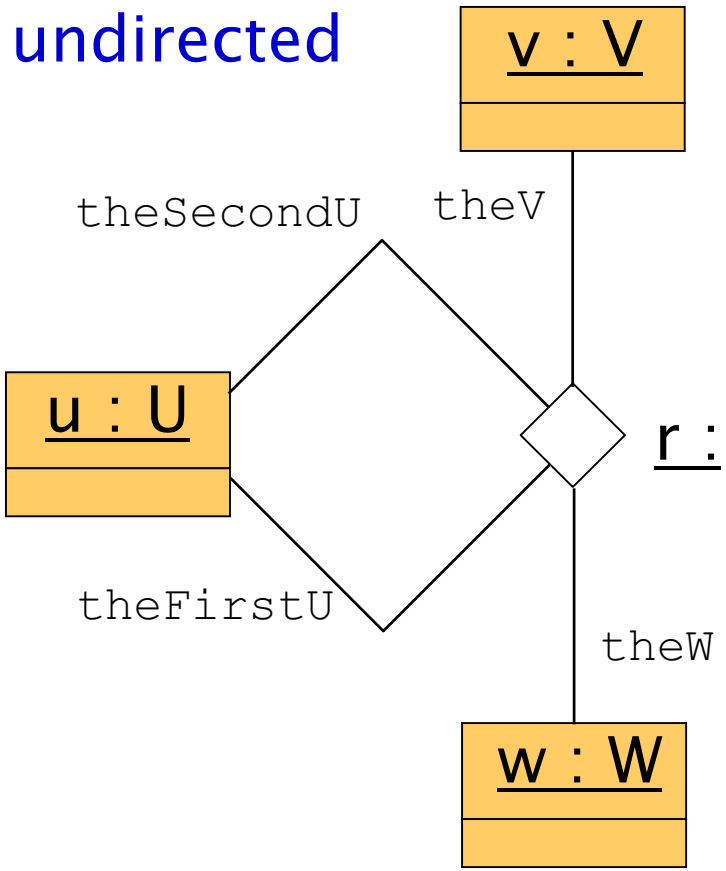

```
<node id = "u" />
<node id = "v" />
<node id = "w" />
<edge id = "e"
    from = "v" to = "u"
    fromorder = "1"      />
<edge id = "f"
    from = "w" to = v"
    toorder = "2"        />
<edge id = "g"
    from = "v" to = "v"
    toorder = "3"
    fromorder = "4"   />
```

Hypergraphs and n-ary Relations




```
<rel id = "r"
     orientation = "directed">
  <link ref  = "u"
        role = "theFirstU"
        direction = "out"
        endorder = "2"    />
  <link ref  = "u"
        role = "theSecondU"
        direction = "out"
        endorder = "1"    />
  <link ref  = "v"
        role = "theV"
        direction = "in"  />
  <link ref  = "w"
        role = "theW"
        direction = "none"/>
</rel>
```

Hypergraphs and n-ary Relations

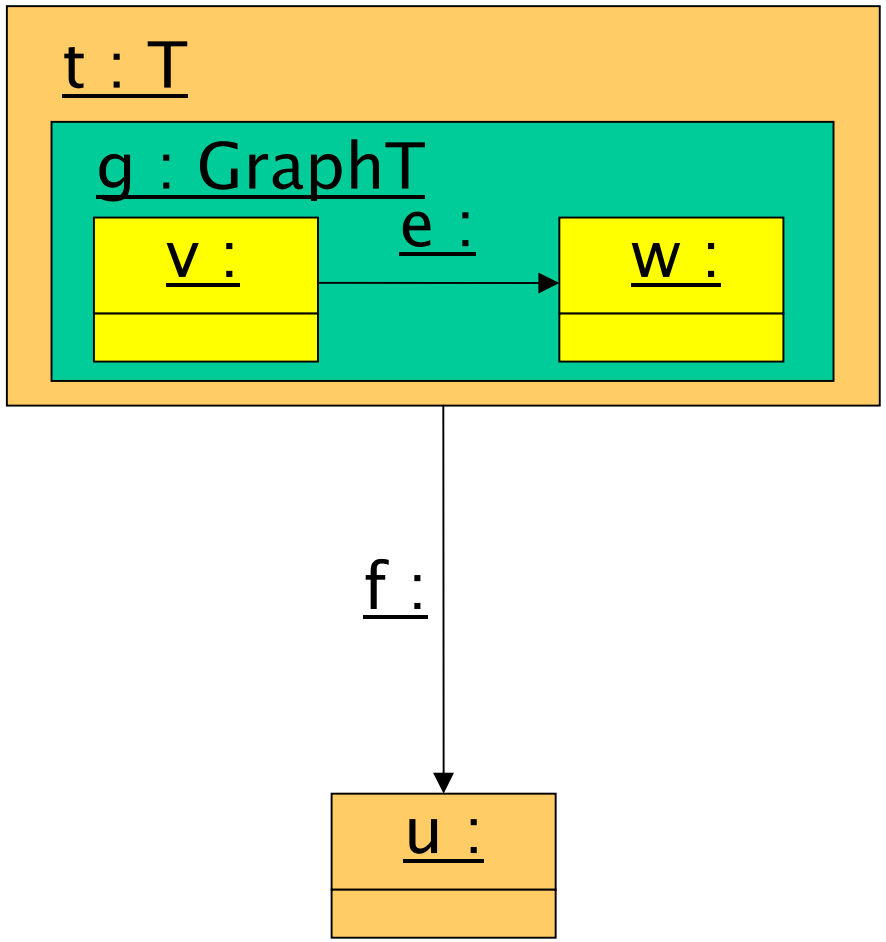



```
<rel id = "r"
     orientation = "undirected">
  <link ref  = "u"
        role = "theFirstU"  />
  <link ref  = "u"
        role = "theSecondU" />
  <link ref  = "v"
        role = "theV"       />
  <link ref  = "w"
        role = "theW"       />
</rel>
```

Relations (DTD)

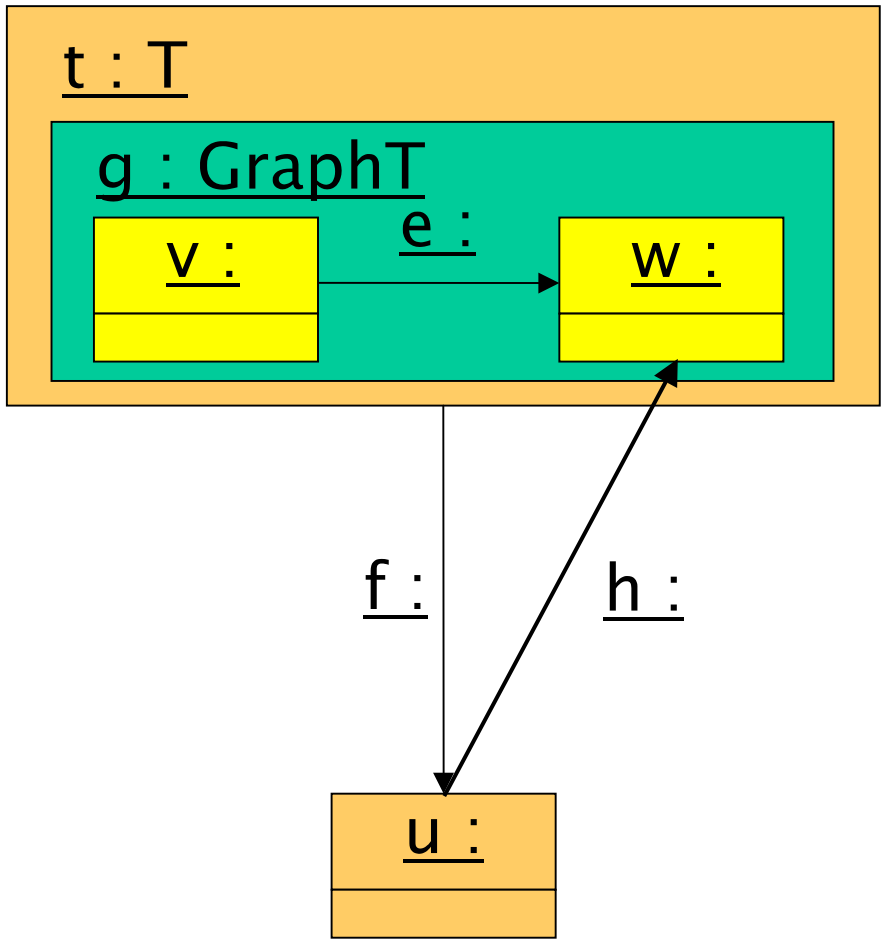
```
<!ELEMENT rel (%rel-extension;
                    type? , attr*, graph*, link* ) >
<!ATTLIST  rel
      id              ID                          #IMPLIED
      orientation (directed | undirected)    #IMPLIED>
<!ELEMENT link (%link-extension; attr*) >
<!ATTLIST  link
      ref             IDREF                       #REQUIRED
      role            NMTOKEN                     #IMPLIED
      direction       ( in | out | none)          #IMPLIED
      startorder      CDATA                       #IMPLIED
      endorder        CDATA                       #IMPLIED >
```

Support for Hierarchical Graphs




```
<node  id = "t">
  <type xlink:href =
        "schema.gxl#T" />
  <graph id = "g">
    <type xlink:href =
          "schema.gxl#GraphT"/>

    <node  id = "v"/>
    <node  id = "w"/>
    <edge  id = "e"
           from = "v" to = "w"/>
  </graph>
</node>
<node  id = "u"/>
<edge  id = "f"
       from = "t" to = "u"/>
```

Support for Hierarchical Graphs




```
<node  id = "t">
  <type xlink:href =
        "schema.gxl#T" />
  <graph id = "g">
    <type xlink:href =
          "schema.gxl#GraphT"/>
    <node  id = "v"/>
    <node  id = "w"/>
    <edge  id = "e"
           from = "v" to = "w"/>
   </graph>
</node>
<node  id = "u"/>
<edge  id = "f"
       from = "t" to = "u"/>
```

```
<edge id = "h"
      from = "u" to = "w"/>
```

GXL Extension

- by redefining entities
- by including GXL DTD (external entities)
- GXL offers extending
  - GXL documents
  - graphs
  - nodes
  - edges
  - hyperedges (rel)
  - links
  - values

```
<!ENTITY % gxl-extension
  "FOO ," >
<!ELEMENT FOO (#PCDATA)>

<!ENTITY % basegxl SYSTEM
  "gxl.dtd">

%basegxl;
```

GXL Extension (DTD)

```
<!ENTITY % gxl-extension      "" >
<!ENTITY % graph-extension    "" >
<!ENTITY % node-extension     "" >
<!ENTITY % edge-extension     "" >
<!ENTITY % rel-extension      "" >
<!ENTITY % link-extension     "" >
<!ENTITY % value-extension    "" >
```

Summary: Exchanging Graphs

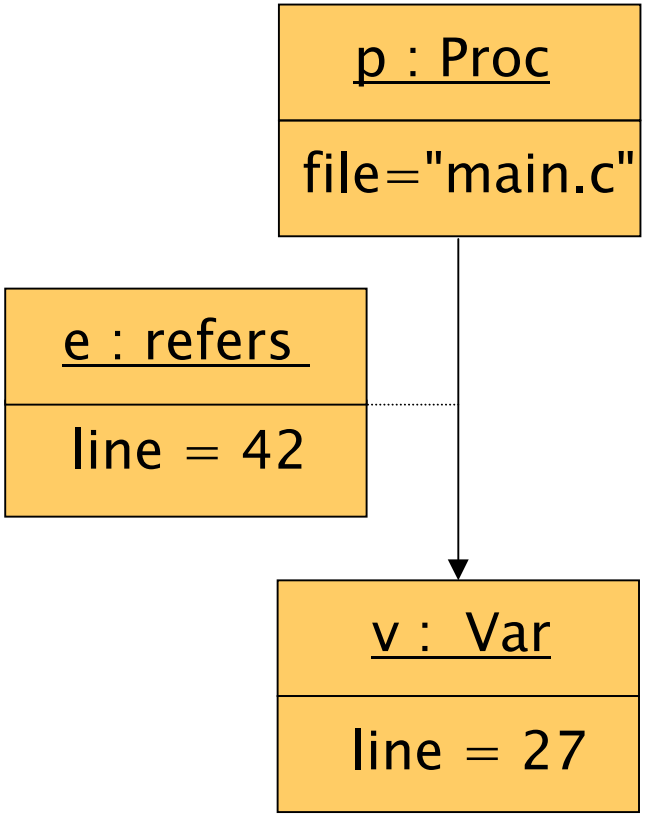
- Attributed, typed, directed Graphs
- Undirected Graphs
- Ordered Graphs
- Hypergraphs
- Hierarchical Graphs

Exchanging Schemas with GXL

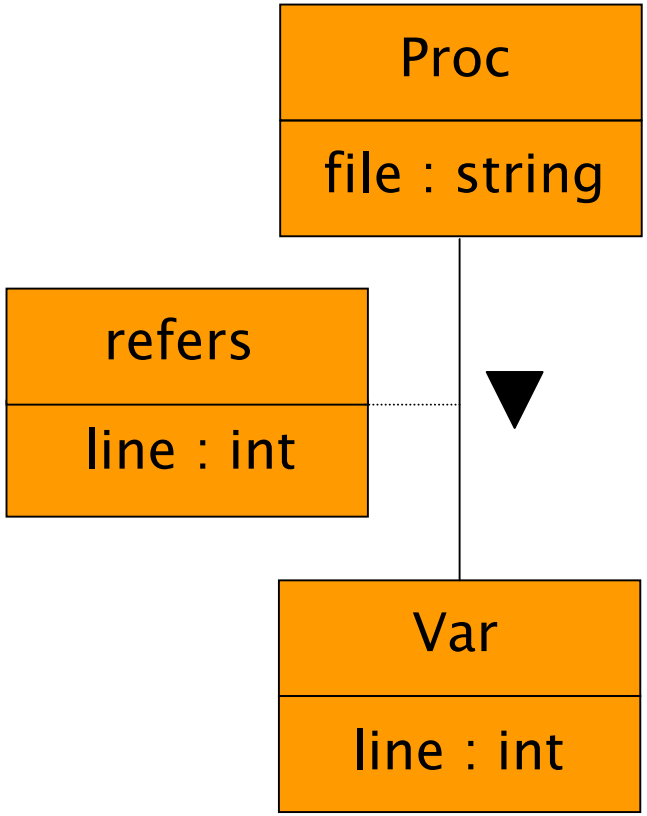
- Graph Schema Definition with UML Class Diagrams
  - attributed, typed, directed graphs
  - hypergraphs
  - hierarchical graphs
- GXL Representation of class diagrams
- GXL Metaschema

Graph Schema – Graphs

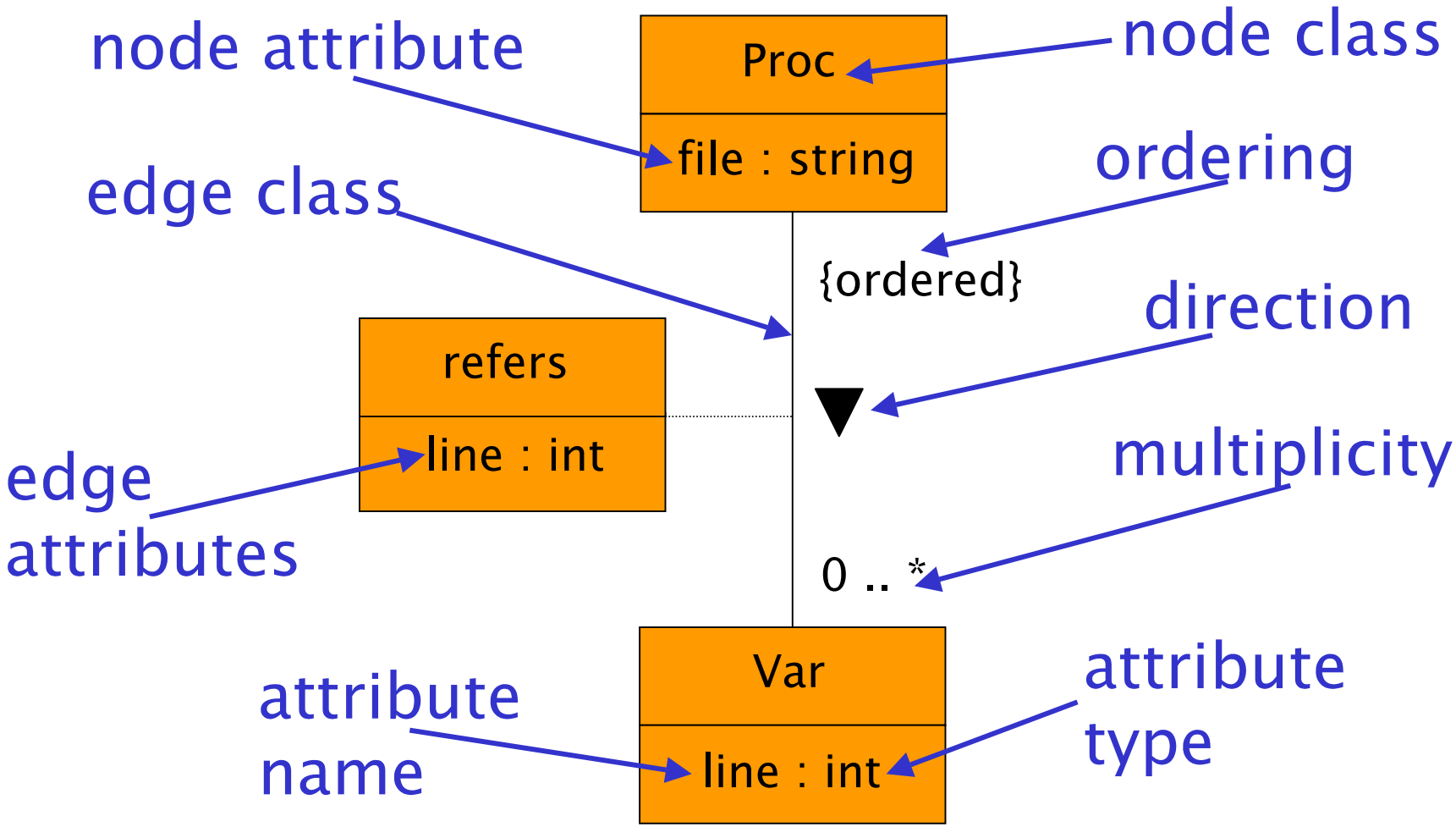
UML object diagram



UML class diagram

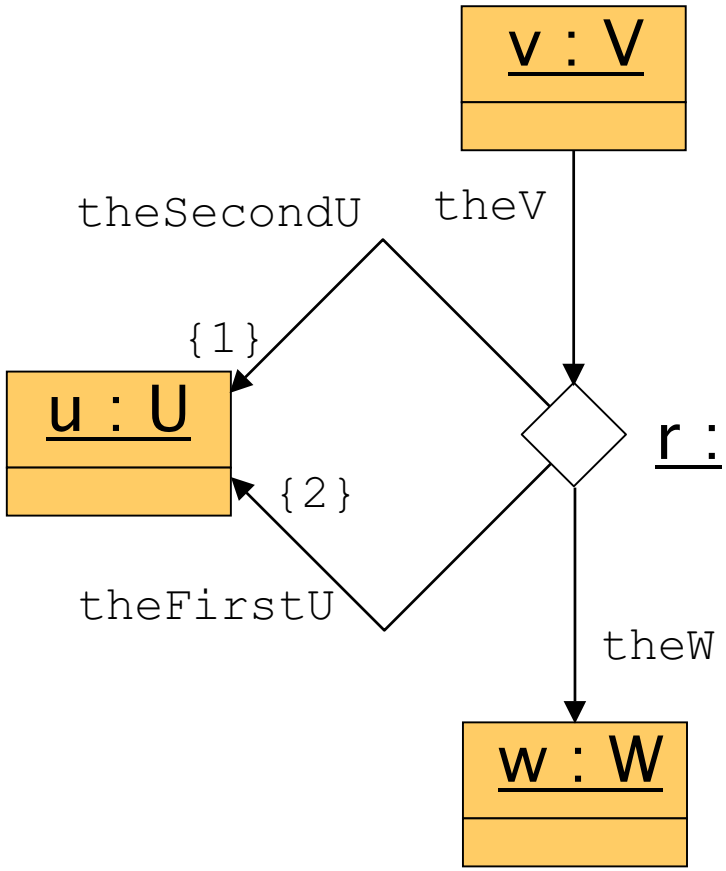


Graph Schema – Notation

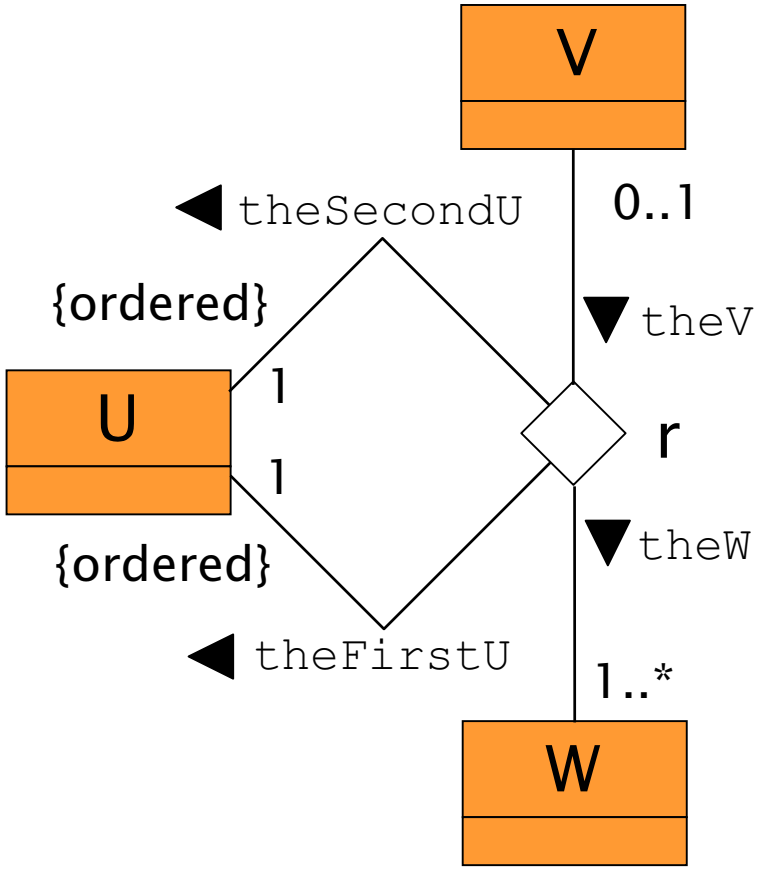


Graph Schema – Hypergraphs

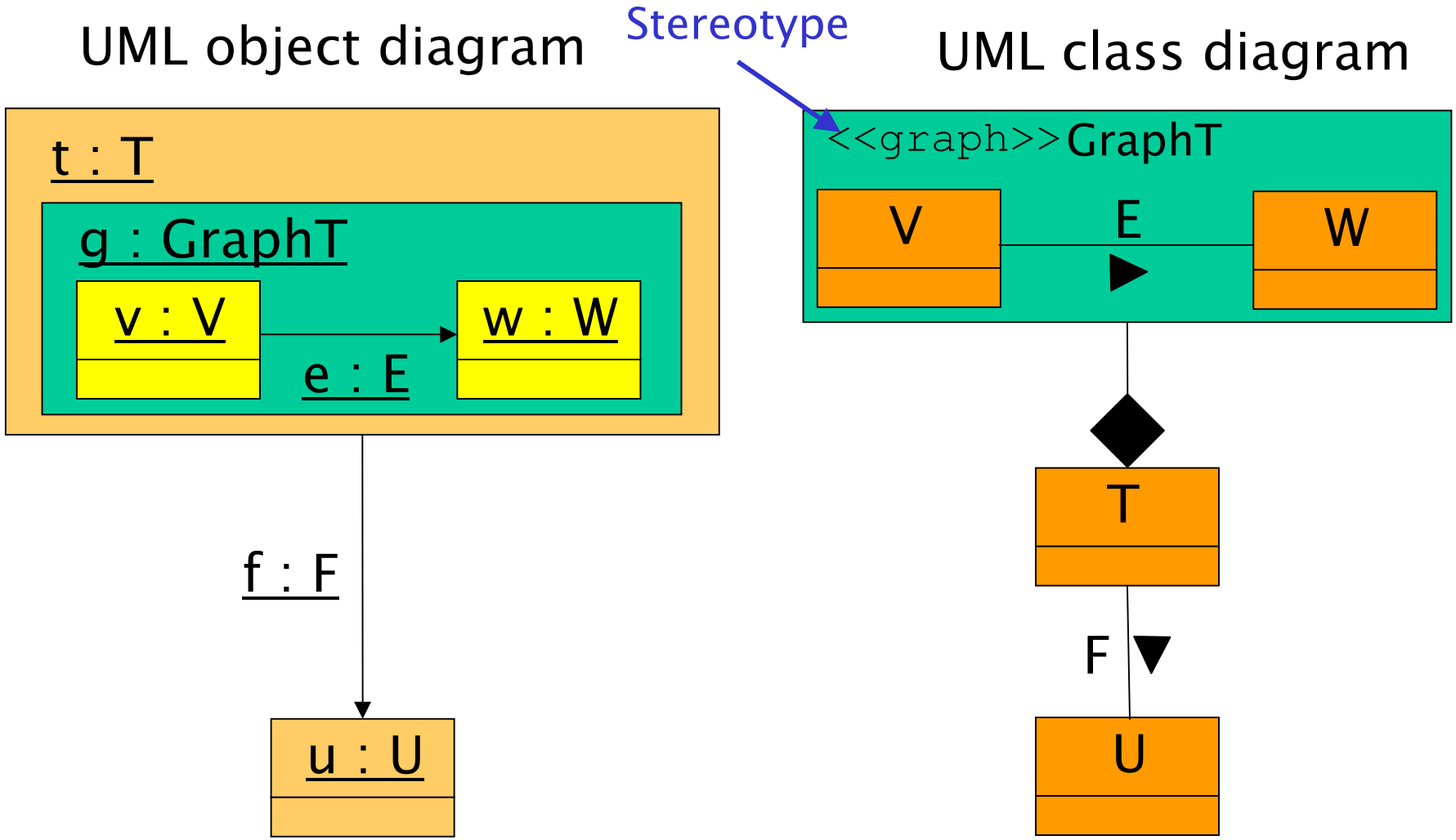
UML object diagram



UML class diagram

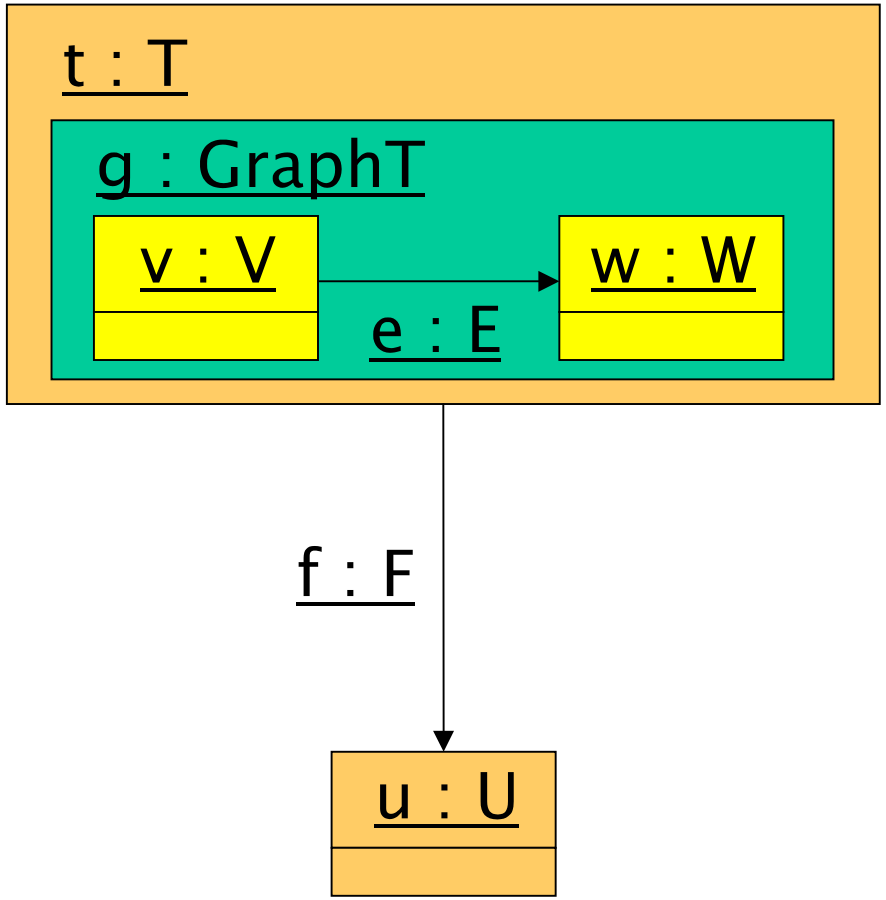


Graph Schema – Hierarchical Graphs

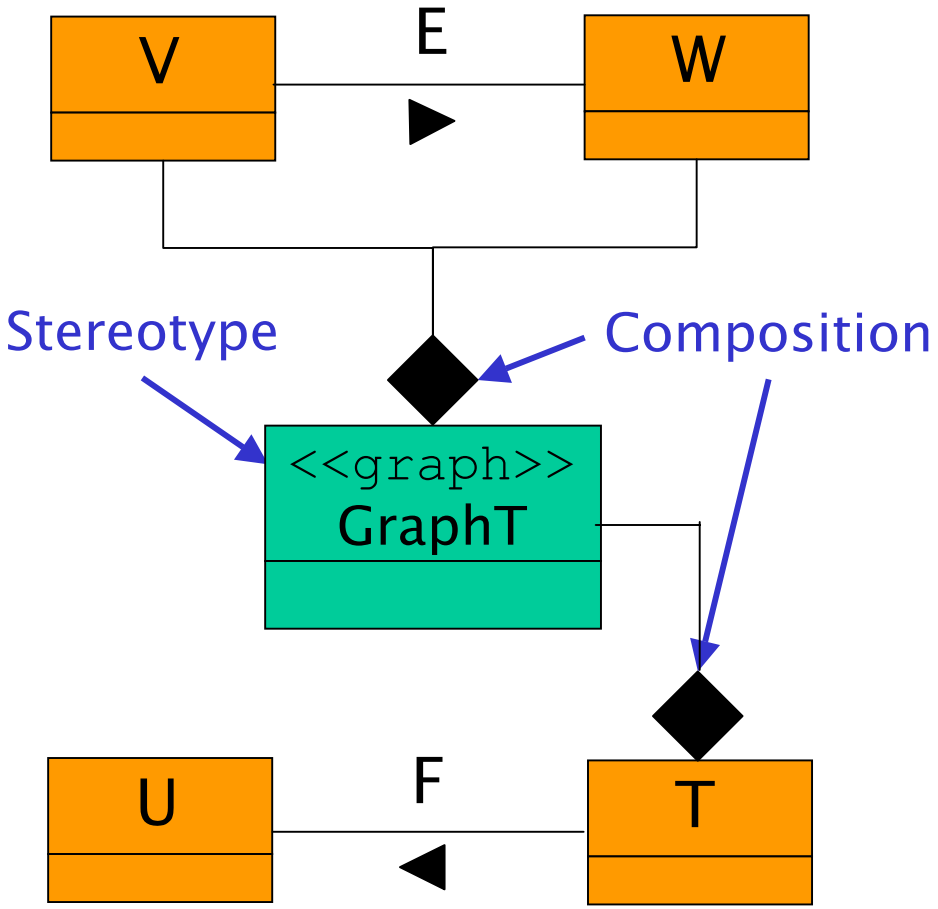


Graph Schema – Hierarchical Graphs

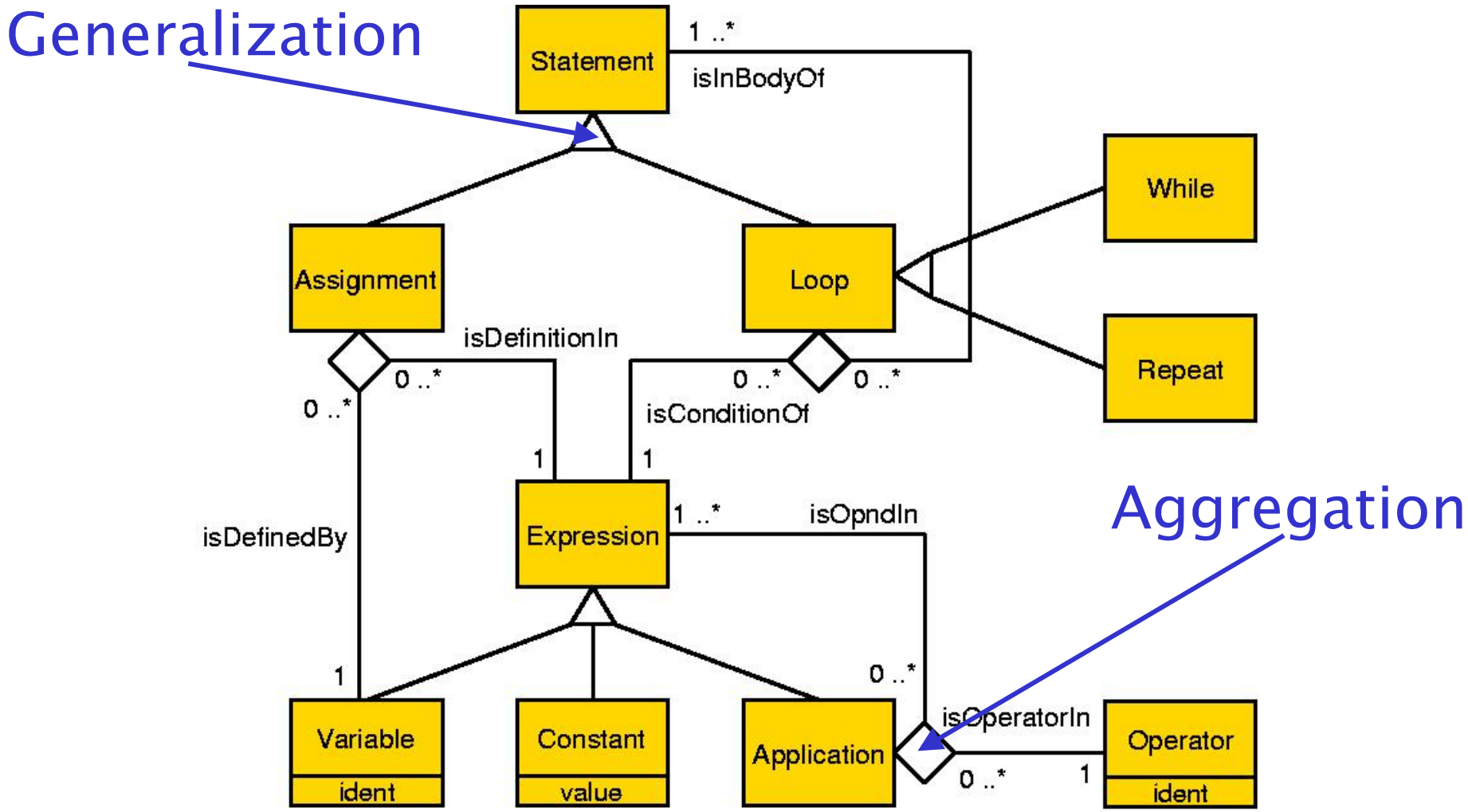
UML object diagram



UML class diagram



Graph Schema – Higher Constructs



Summary: Exchanging Schemas

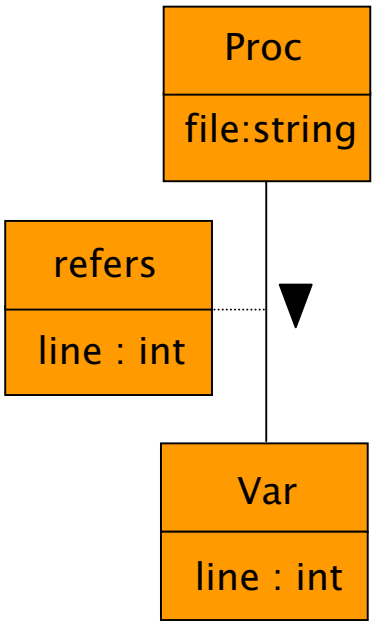
- Graph Schema Definition with UML Class Diagrams
  - attributed, typed, directed graphs
  - hypergraphs
  - hierarchical graphs
- GXL Representation of class diagrams
- GXL Metaschema

Representing Schemas

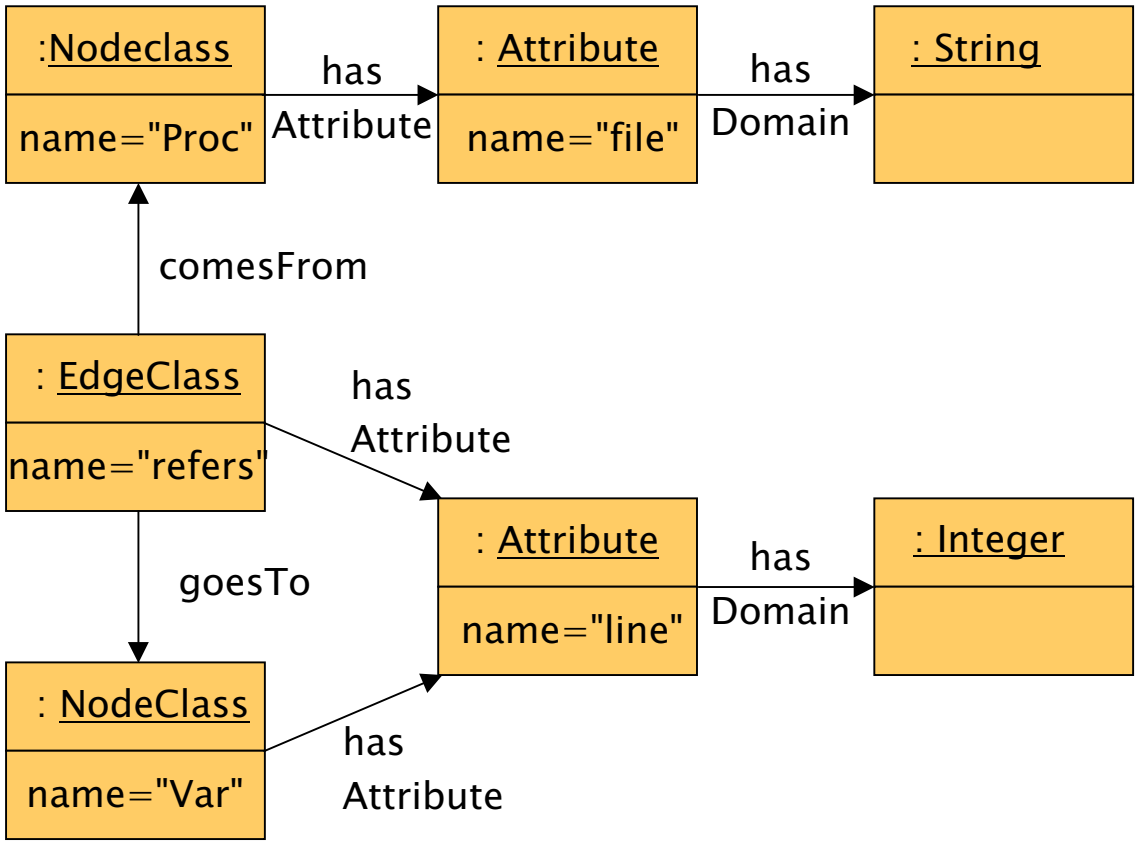
- Schemas (graph classes) are represented as graphs as well
- Schemas are exchanged as GXL documents suiting a metaschema for graph classes
- only one common and simple DTD for exchanging
  - graphs matching different graph schemas
  - graph classes matching a metaschema

GXL Schema Representation

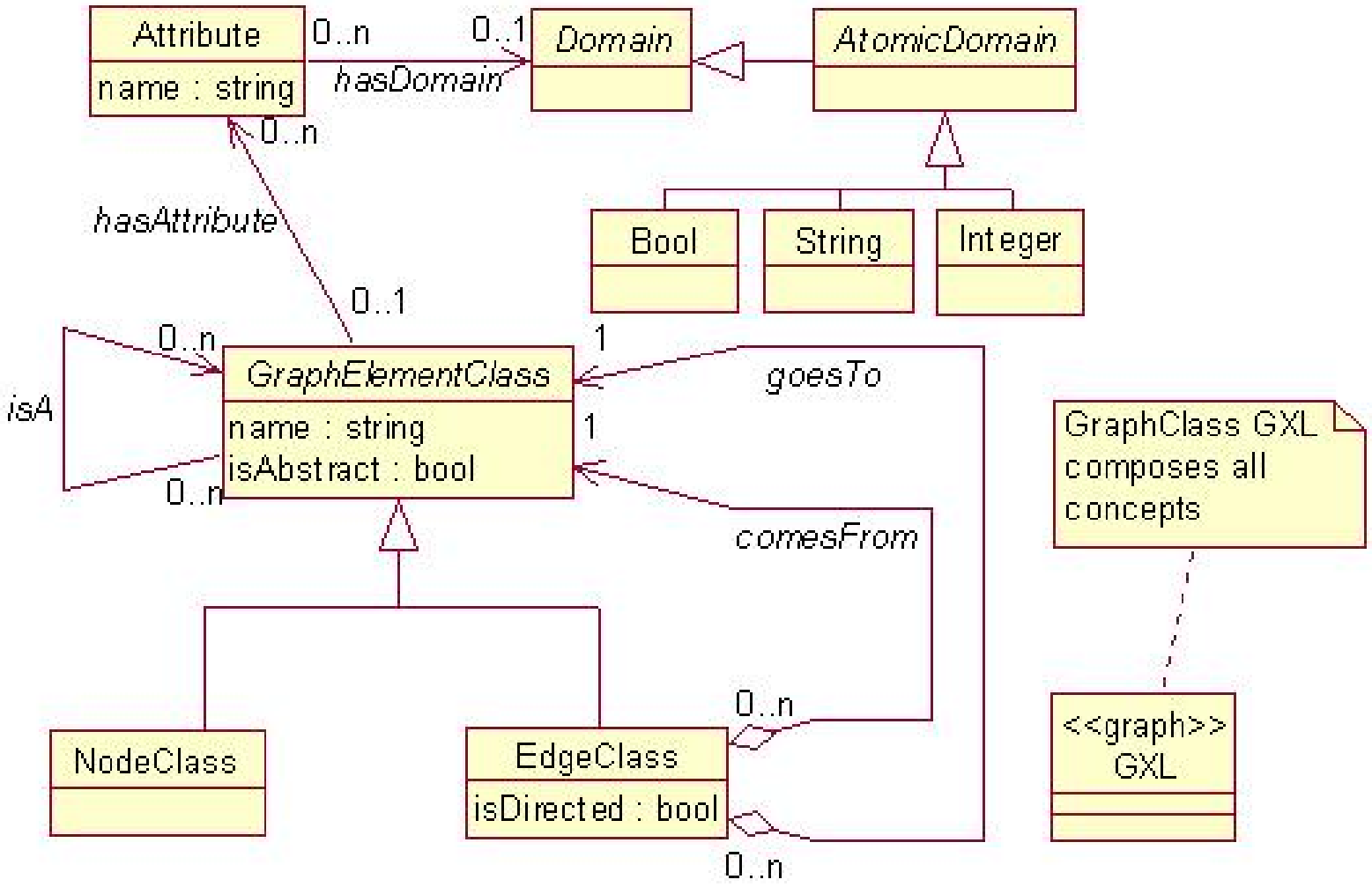
UML class diagram



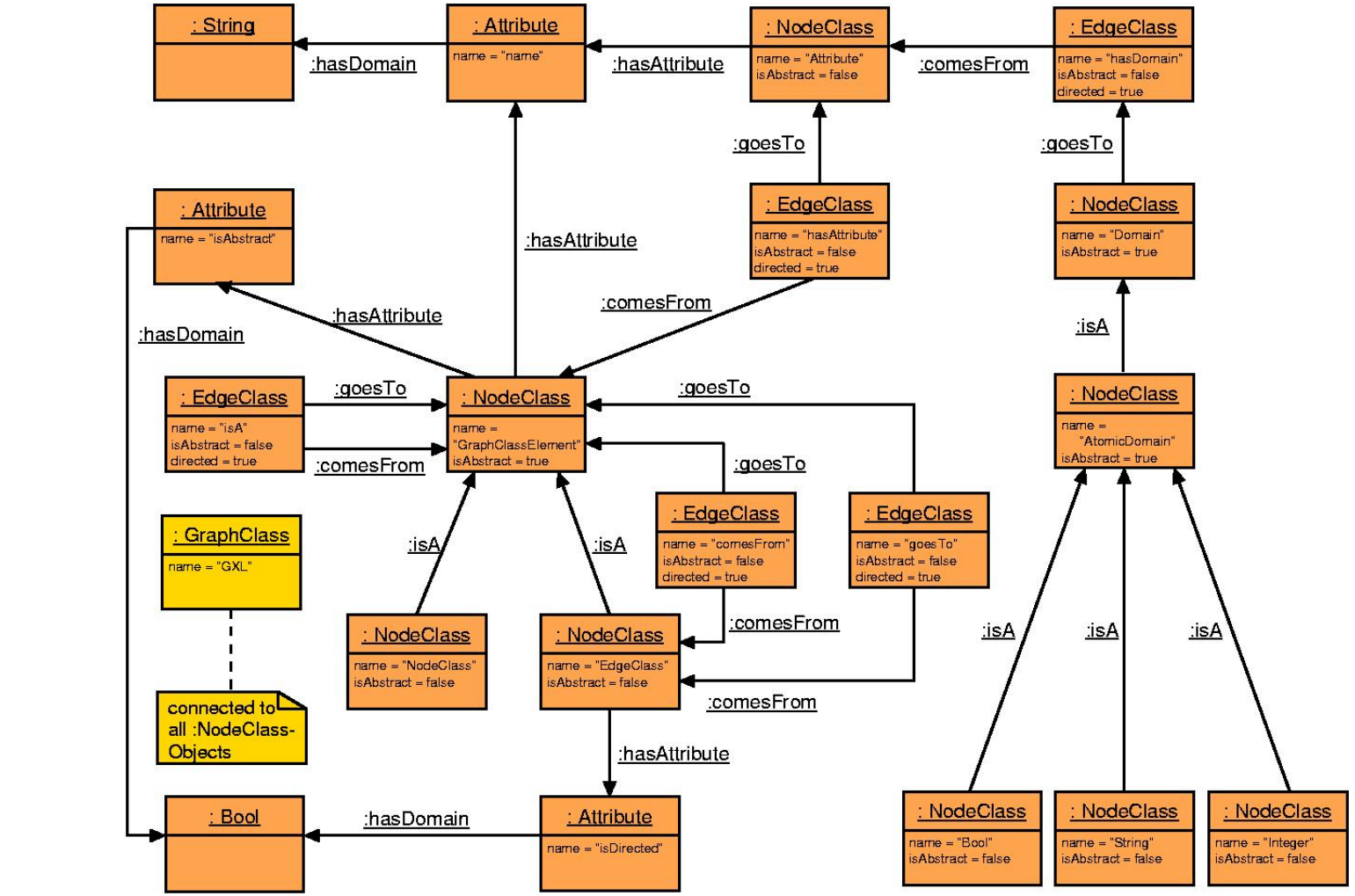
schema graph



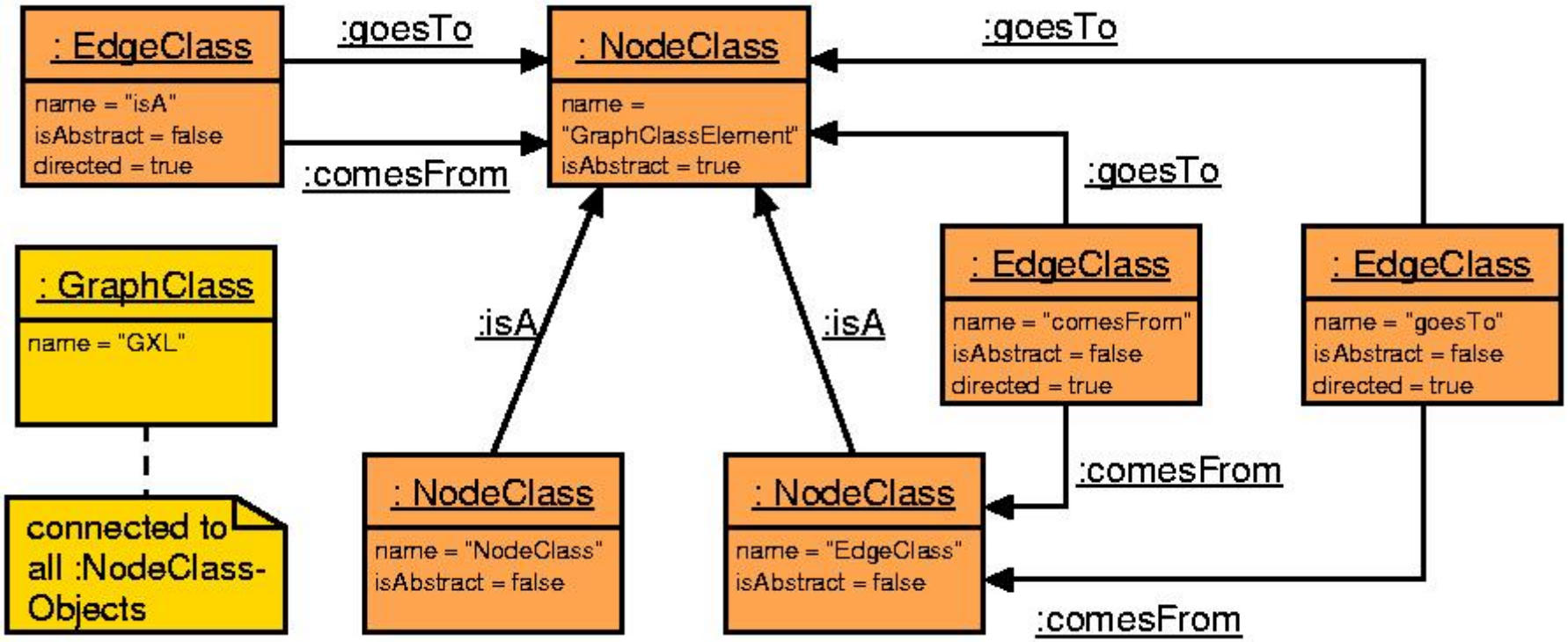
GXL Metaschema (Extract)



GXL Metaschema – Graph (Extract)



GXL Metaschema – Graph (Extract)



GXL Metaschema – GXL Document

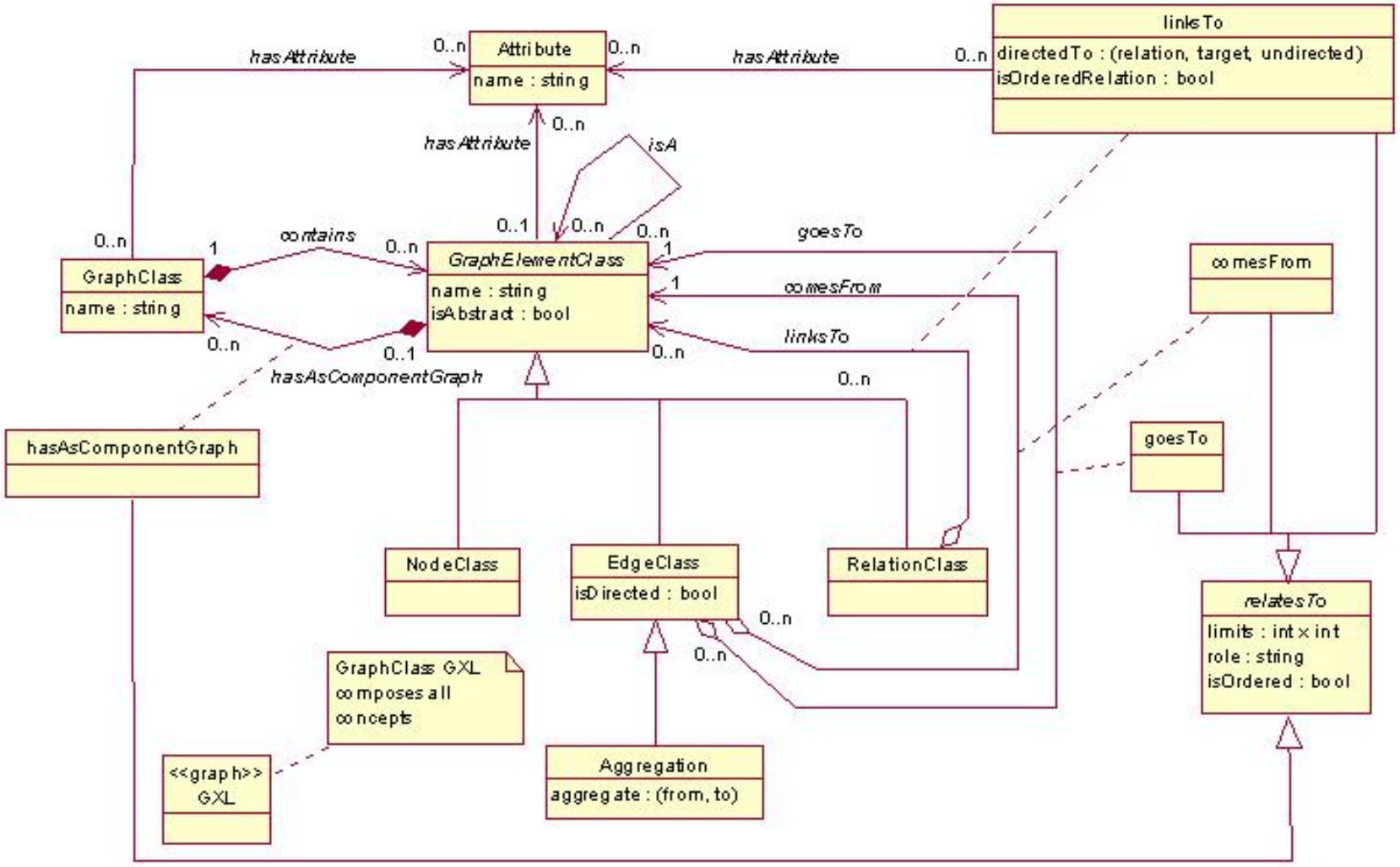
```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "gxl.dtd">
<gxl>
  <graph id = "gxl">
  <type xlink:href =
        "gxl.gxl#gxl"/>

<node id = "NodeClass">
  <type xlink:href =
        "gxl.gxl#NodeClass"/>
  <attr name = "name">
    <string>NodeClass</string>
  </attr>
  <attr name = "isAbstract">
    <bool>false</bool>
  </attr>
</node>

...
```

```
<node id = "GraphClassElement">
  <type xlink:href =
        "gxl.gxl#NodeClass"/>
  <attr name = "name">
    <string>
      GraphClassElement
    </string>
  </attr>
  <attr name = "isAbstract">
    <bool>true</bool>
  </attr>
</node>
...
<edge
  from = "NodeClass"
  to   = "GraphClassElement">
  <type xlink:href =
        "gxl.gxl#isA"/>
</edge>
```

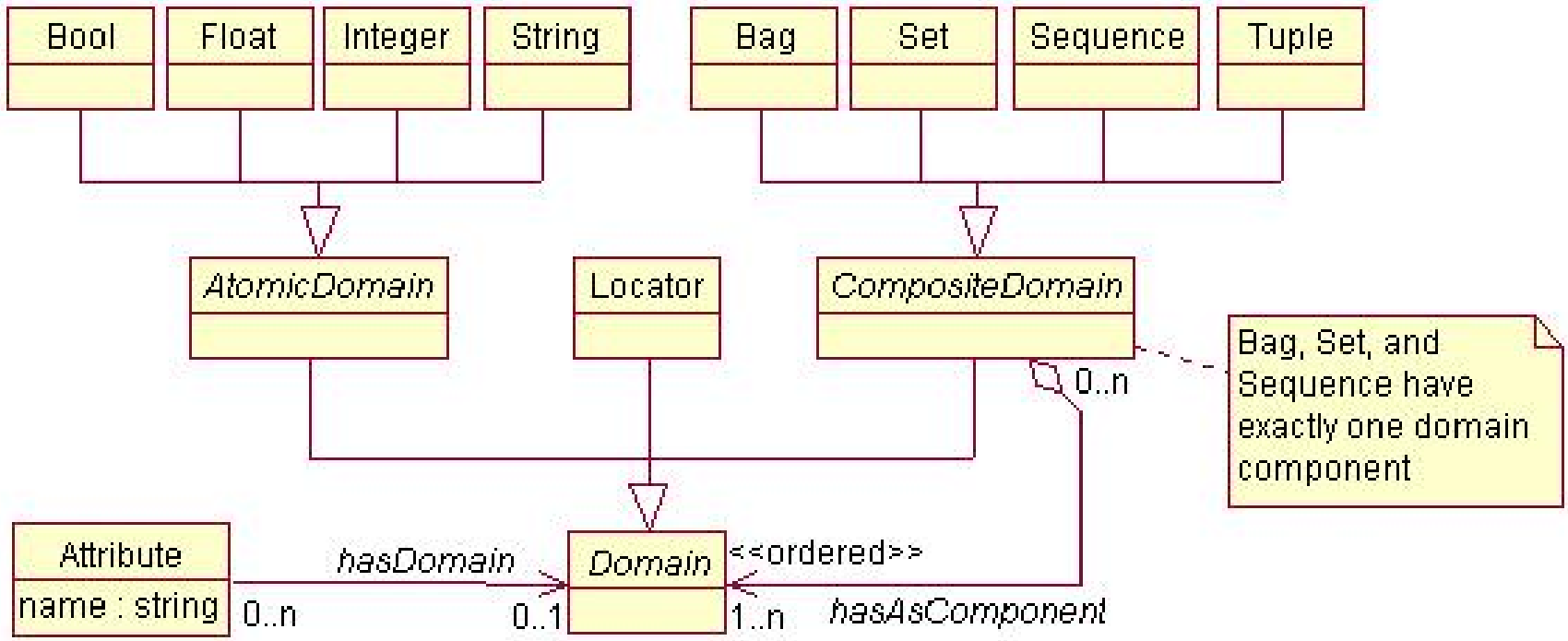
Concepts in GXL Class Diagrams

- Definition of *NodeClasses*, *EdgeClasses*, and *RelationClasses.*
- Definition of (structured) *Attributes.*
- Definition of *Orientation*, *Multiplicities*, *Roles*, and *Ordering*.
- Definition of *Graph Hierarchy*.
- Definition of *Graphclasses*.
- Definition of *Generalization* and *Aggregation*.

GXL Metaschema (Graph Part)



GXL Metaschema (Attribute Part)



Conclusion

GXL offers

- a language for describing graphs
  - directed and undirected, typed, attributed graphs
  - hypergraphs and hierarchical graphs
- a language for defining graph classes
- a language for exchanging graphs
  - instance graphs
  - schema graphs

more information

- http://www.gupro.de/GXL