Grailog KS Viz 2.0: Graph-Logic Knowledge Visualization by XML-Based Translation

Leah Bidlake

Master Thesis Defence

June 22, 2016

UNB Fredericton

Outline

- Introduction
- Related Work
- Objectives
- Grailog KS Viz 2.0
  - Architecture
  - Implementation
  - Test Case
- Conclusions
  - Results
  - Future Work

Knowledge Visualization

- Knowledge visualization supports transfer and analysis of knowledge
- Visualization increases the rate and quality of (human-to-human and machine-to-human) knowledge transfer and refinement
- (Semi-)Formal knowledge as used in Data Modeling, the Semantic Web, etc. can be visualized using (generalized) graphs

Graph Inscribed Logic (Grailog) (1)

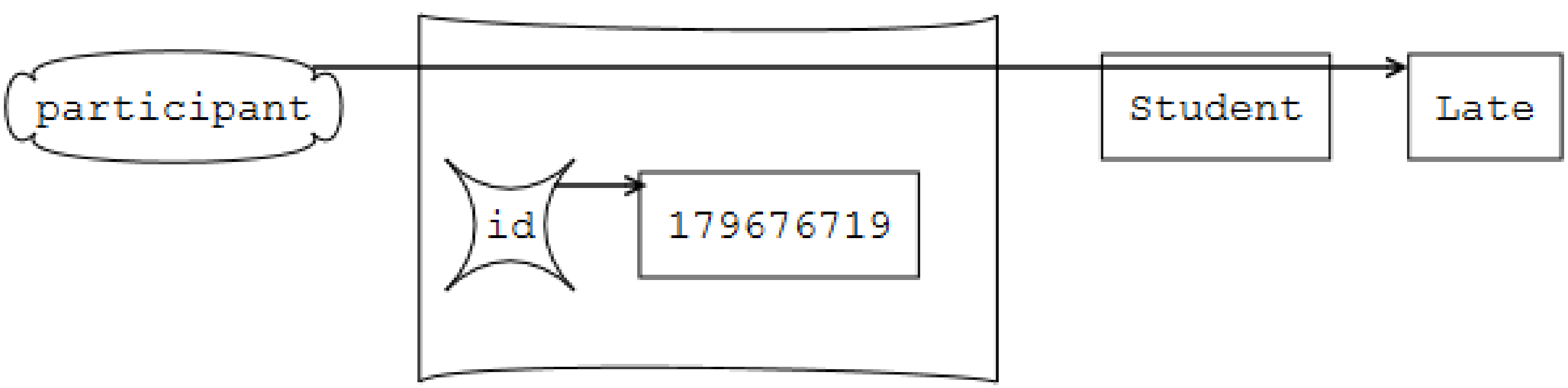
- Grailog is used to present languages of the Rule Markup Language (RuleML) system
- Highly expressive generalized graphs for logical knowledge visualization (in labelnode normal form)
- Contain directed n-ary hyperarcs that begin at a class/relation labelnode, pass through n–1 intermediate argument nodes, and point to the n^{th} argument node

Graph Inscribed Logic (Grailog) (2)

- Conference registration knowledge about participants who can be Student, Late (vs. Early), etc.
- Participant <Rel>ation becomes labelnode starting hyperarc arrow
- Hornlog RuleML:

```
<Atom><Rel>participant</Rel><Expr>                        <Ind>Student</Ind><Ind>Late</Ind></Atom>
                                 <Fun per="copy">id</Fun>
                                 <Ind>179676719</Ind>
                             </Expr>
```

- Grailog:



Related Work

- Fresnel Editor
  - Visualizes Resource Description Framework (RDF) data using simple data modeling
- GrOWL
  - For visualizing and editing Web Ontology Language (OWL) as graphs
  - Provides more descriptive semantics

Objectives

- Proceed from earlier Datalog to computationally complete language on the level of Horn Logic (Hornlog) by visualizing nested terms
- Transformation from Hornlog with Equality to Grailog visualization
- Visualizations in labelnode normal form of Grailog (includes classes as unary relations)
- Remove internal JavaScript from the Grailog/SVG to increase efficiency and security

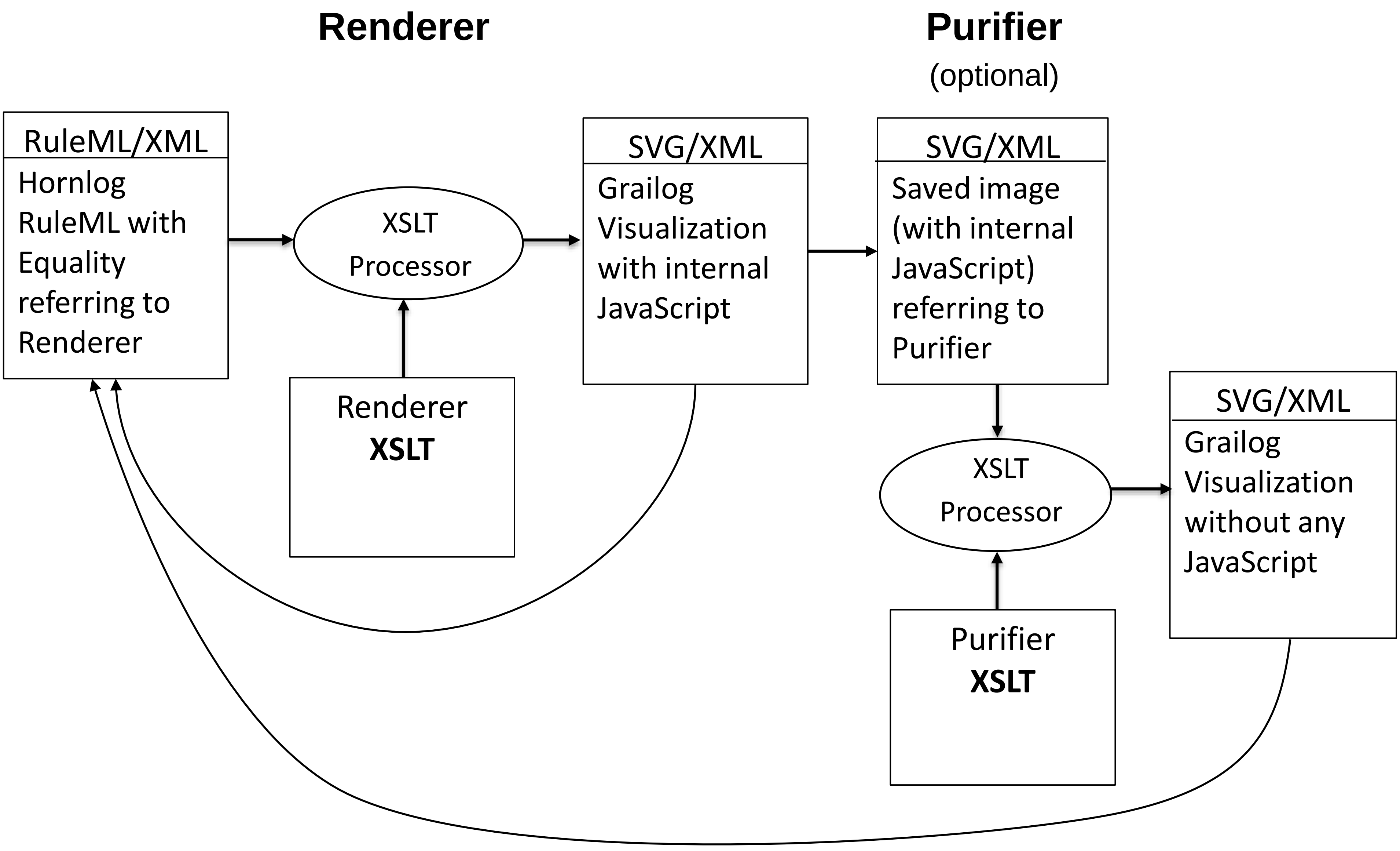
Design

- XSLT translation for end users on common modern Web browsers that support XSLT 2.0
- Source RuleML/XML:
  - Requires stylesheet processing instruction to automate transformation in the browser
  - Cannot contain namespaces
- Target SVG/XML:
  - Node-copy normal form of Grailog used to allow scalability for large KBs and human readability
  - Contains internal JavaScript that will be optionally removed

Grailog KS Viz 2.0: Horn Logic with Equality in SVG (1)

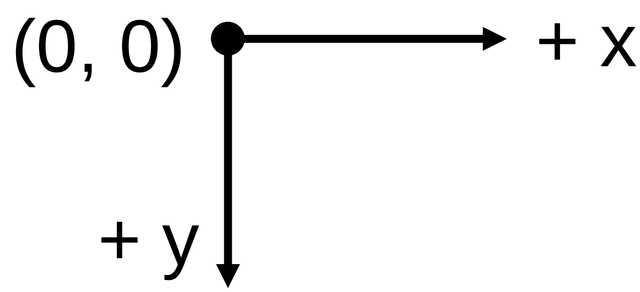
- The **Renderer** transforms XML documents containing HornlogEq RuleML – using an XSLT stylesheet and processor – into Grailog visualizations in SVG format that contain JavaScript
- The **Purifier** removes the JavaScript that is no longer required in the static SVG

Grailog KS Viz 2.0 Workflow: Horn Logic with Equality in SVG (2)



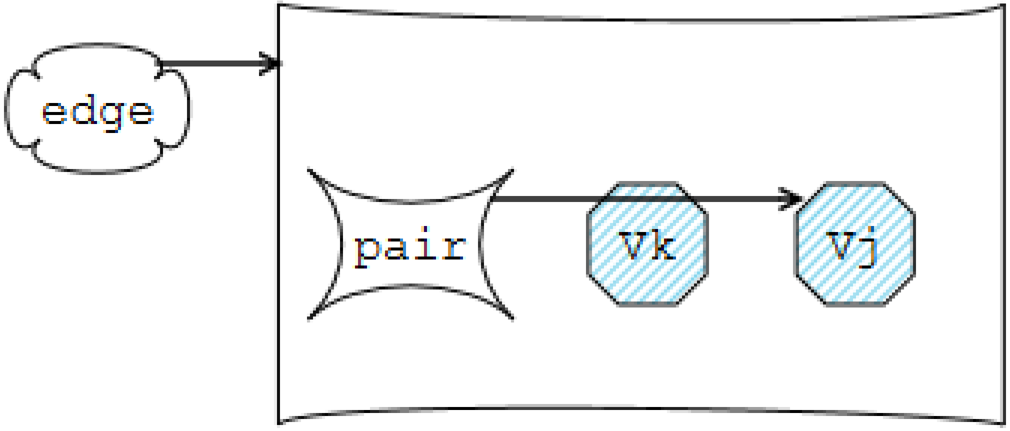
Grailog KS Viz Implementation (1)

- SVG canvas allows for a virtually infinite area for the content to be rendered
- SVG Viewport
  - Finite rectangular subregion of the canvas
  - Originates at the upper-left corner
  - Expands downward and to the right
  - Dimensions are determined by the attributes width and height



Grailog KS Viz Implementation (2)

- SVG
  - Drawings contain text, rectangles, polygons, patterns, straight paths, rounded rectangles, markers
  - Labelnodes and function applications require the use of cubic Bézier curves to draw convex and concave paths



Grailog KS Viz Implementation (3)

- SVG
  - Unique ID attributes, used to identify each element, are created by concatenating strings and numbers.
  - Strings identify the type of SVG element (rect, text, etc.) and Grailog structure (relation, rule, etc.)
  - Numbers refer to the hierarchical position of the node in the XML tree

Grailog KS Viz Implementation (4)

- XPath Expressions
  - Used for addressing parts of an XML document by tracing its hierarchical structure
  - Location paths select a set of nodes relative to the context node
- XPath Expression Limitations
  - Inability to distinguish between the descendants of siblings that have the same path to the parent node
  - No function to determine the level of nesting

Grailog KS Viz Implementation (5)

- Internal JavaScript
  - Calculates, assigns, and accesses the position and size values of the SVG elements
  - Updates the variables used to determine the SVG viewport height and width
  - Accesses the contents of the nodes provided by the user

Grailog KS Viz Implementation (6)

- Purifier removes JavaScript from the static SVG image
  - Requires stylesheet processing instruction in the prolog of the SVG file
  - Assures users that images do not contain malicious scripts
  - Reduces file size of SVG visualization
  - Requires less time to render the SVG visualization

Grailog KS Viz Implementation (7)

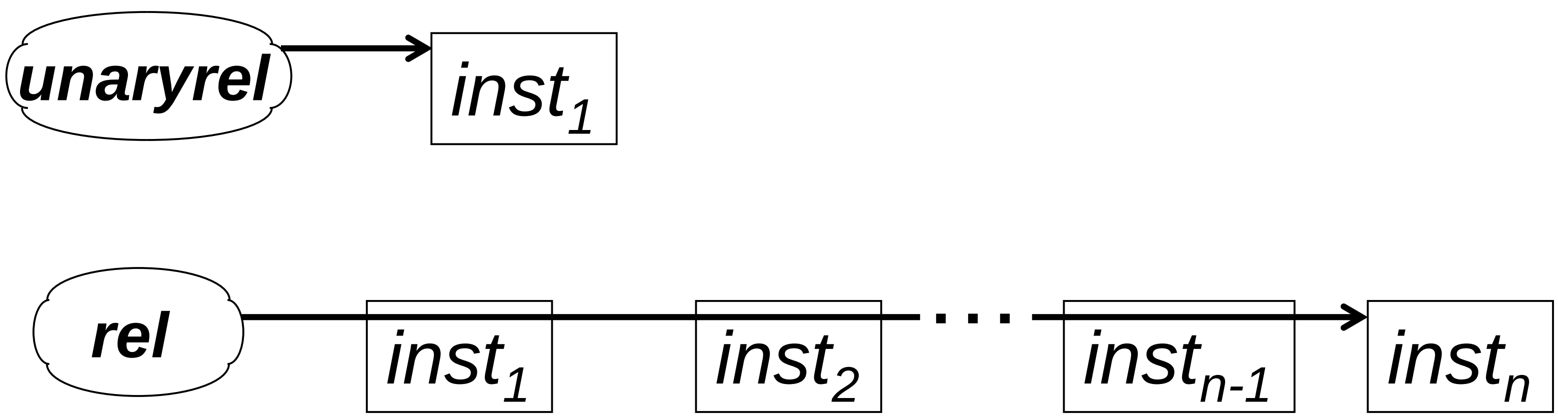
- XSLT Templates
  - Templates given RuleML tag names are applied to nodes with matching pattern
  - Named templates are given descriptive names and are applied when called by name
  - Template parameters specify variables whose values are set when the template is called; this allows the binding of the variables to be updated or changed

Renderer XSLT Implementation

- Set up SVG file with an initial viewport to contain the drawings
- Dimensions of viewport are determined using JavaScript
  - Height is determined by a variable that is updated with the last y-coordinate of each new drawing
  - Width is determined by a variable that stores the greatest x-coordinate of all the drawings

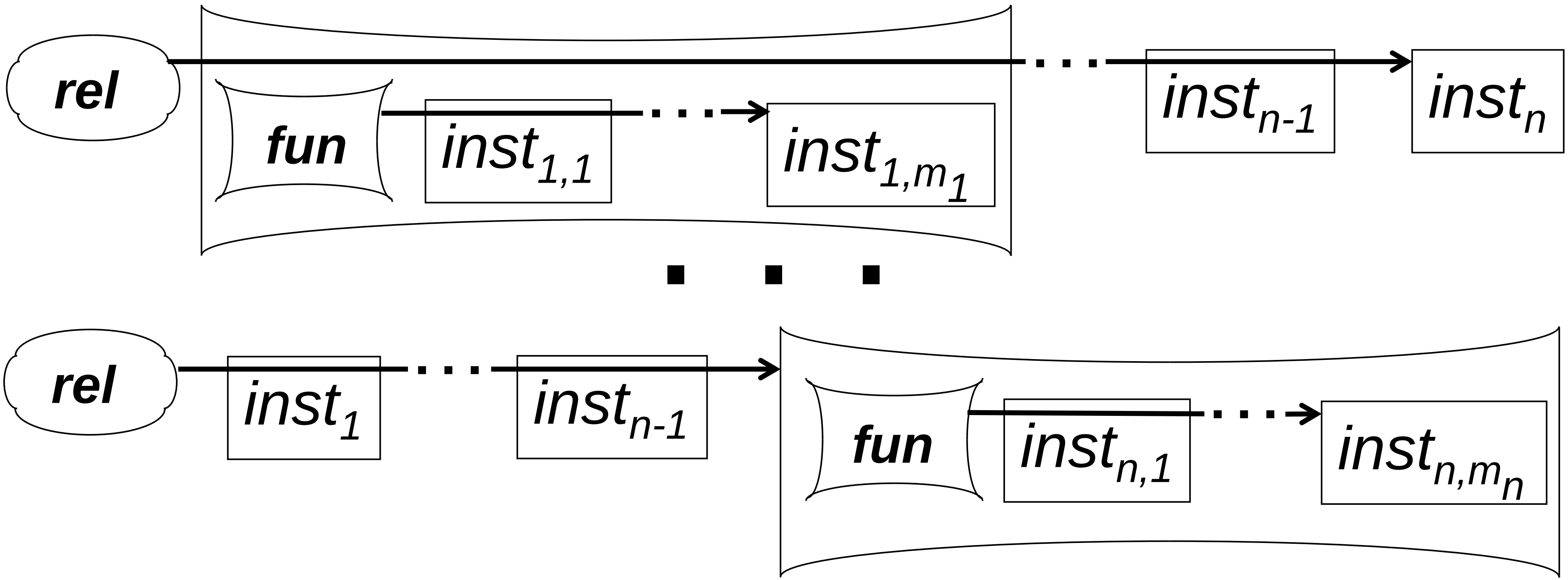
<Atom> Template (1)

- Draws n-ary relation (n ≥ 1) in labelnode normal form as facts, or as the single premise and/or conclusion of a rule
- Draws the relation node found in the first position inside a labelnode



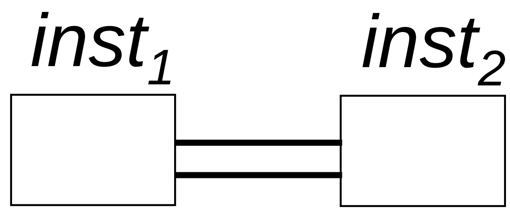
<Atom> Template (2)

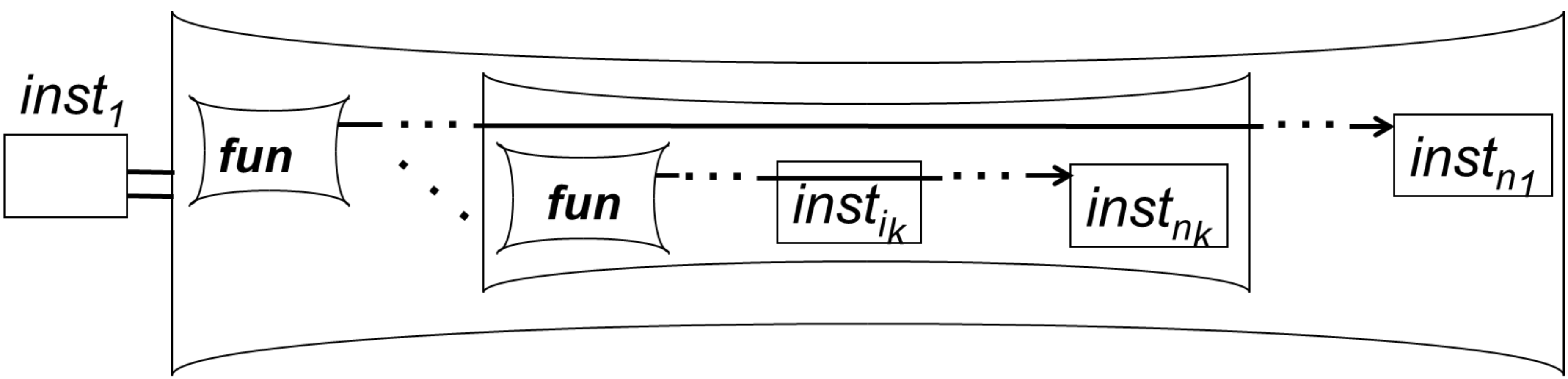
- Invokes <NestedExpr> template to draw relations with arbitrary levels of nested (constructor) function application in any position



<Equal> Template

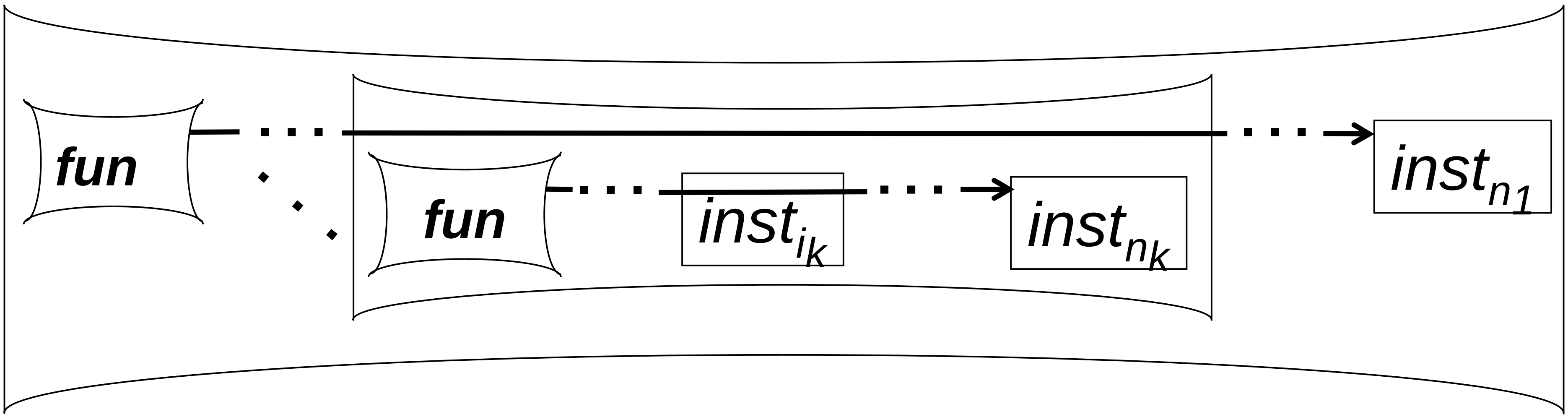
- Draws Datalog+ and Hornlog+ Equality as a special binary atom, or as the single premise and/or conclusion of a rule
- No orientation tags to distinguish placement
- Invokes <NestedExpr> template to draw nested function application





<NestedExpr> Template (1)

- Recursive, named template
- Parameters passed by calling template replace default values and are used to construct unique ID names for elements
- Drawing begins with the **outermost** function node, then draws the siblings and descendants

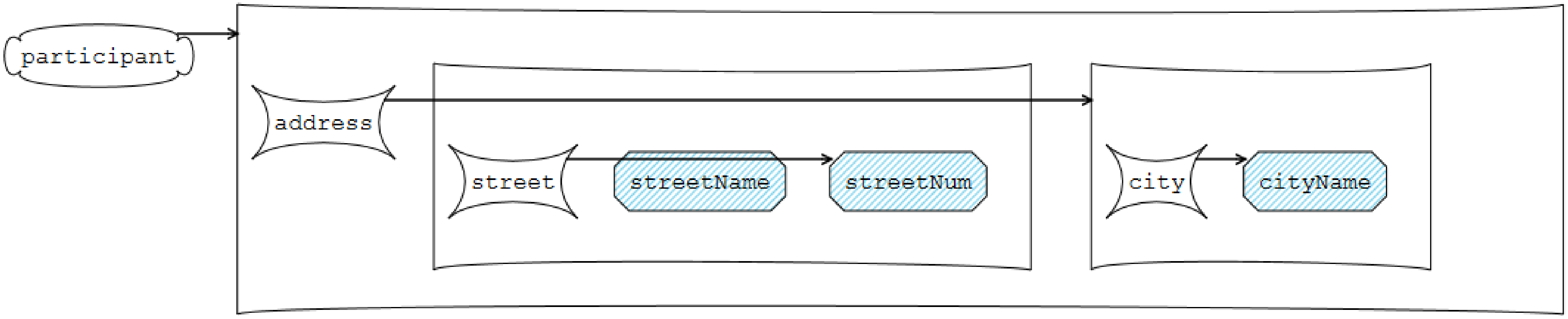


<NestedExpr> Template (2)

- Surrounding boxes of functions:
  - Drawn after the function and argument nodes
  - **Innermost** surrounding box drawn first etc.
  - Required to expand down and to the right to surround any depth of nesting
  - Vertical spacing is dependent on level of nesting
- Height of surrounding box for each function is the product of a constant, and the difference between the function node’s descendants and children (≥ level of nesting)

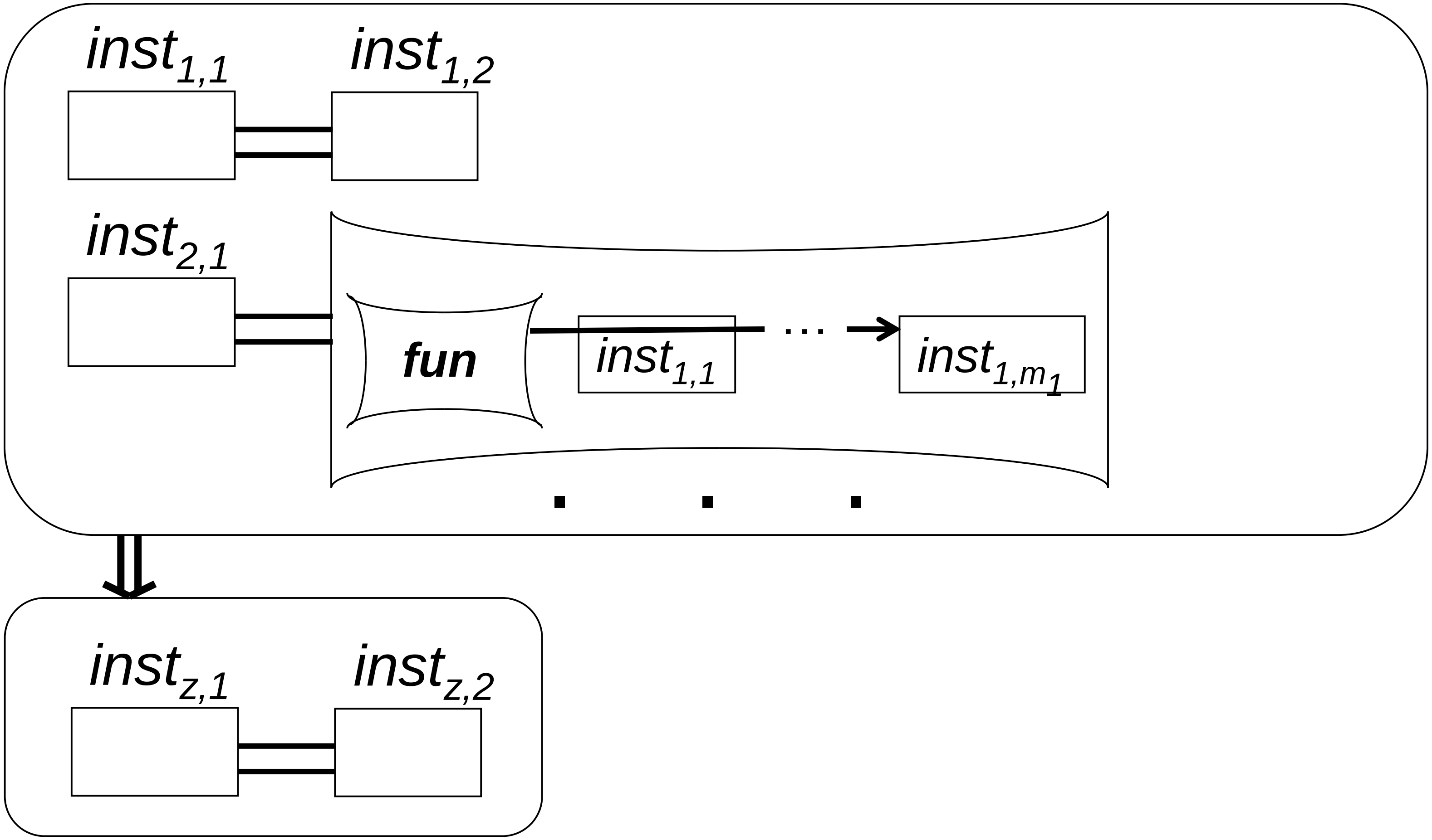
<NestedExpr> Template (3)

- To distinguish between the descendants of siblings that are both nested function applications:
  - The calling template sets a parameter to the number of preceding function siblings
  - The parameter is only updated when the template is called recursively



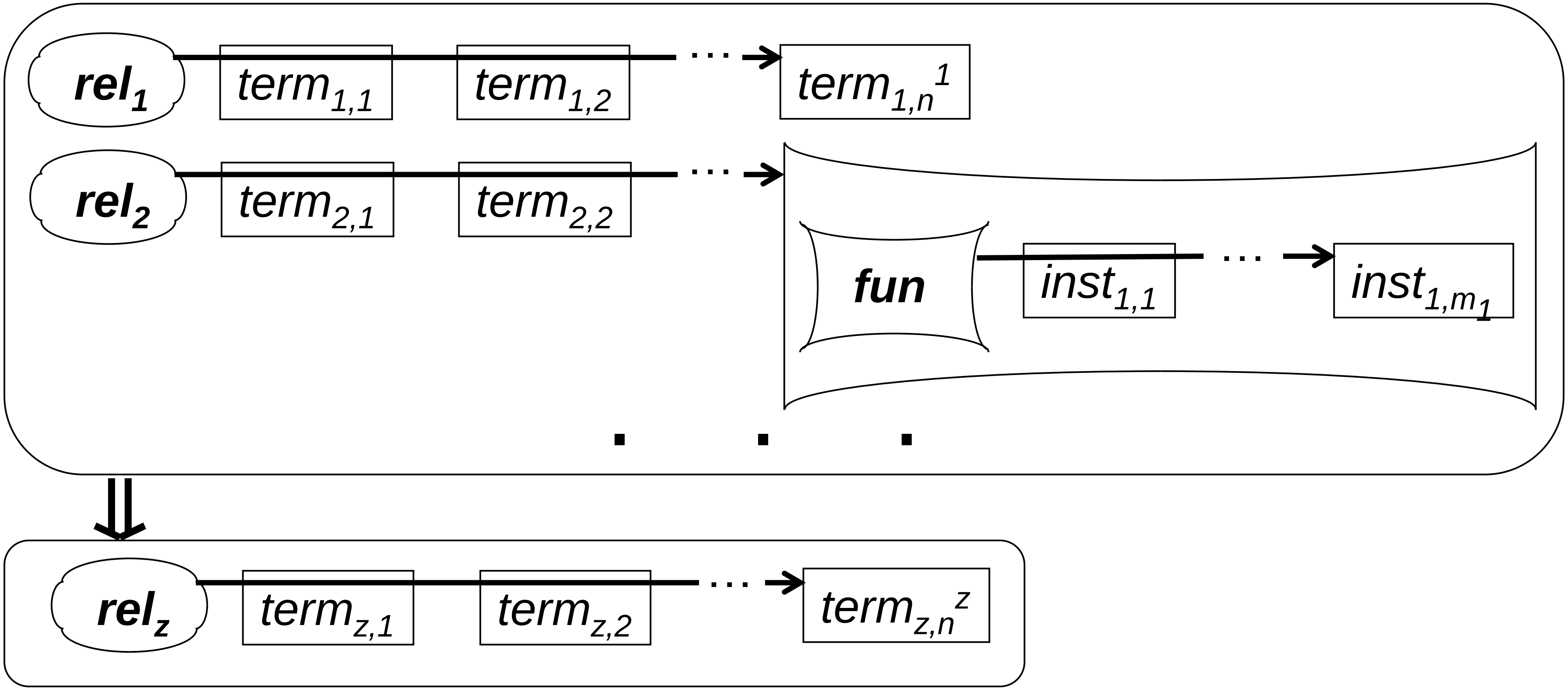
<And> Template

- Draws the premises of a multi-premise rule
- Premises may include relations and equality with arbitrary levels of nested function applications



<Implies> Template

- Draws the surrounding rectangles for the premise(s) and conclusion of single- and multi-premise rules, and the double-shafted Implies arrow between them
- Invokes <And>, <Equal> and/or <Atom> templates to draw contents of the rule



Purifier XSLT Implementation (1)

- XSLT Identity Template
  - Commonly known recursive template
  - Matches all node patterns and recursively copies all nodes and their attributes

Purifier XSLT Implementation (2)

- XSLT Template <svg:script>
  - Template for node with matching pattern
  - Matches a specific node pattern, resulting in a higher priority than the identity template
  - Script nodes are only processed by this template and not by the identity template
  - Empty template results in script nodes not being copied
  - Amounts to omission of all script nodes

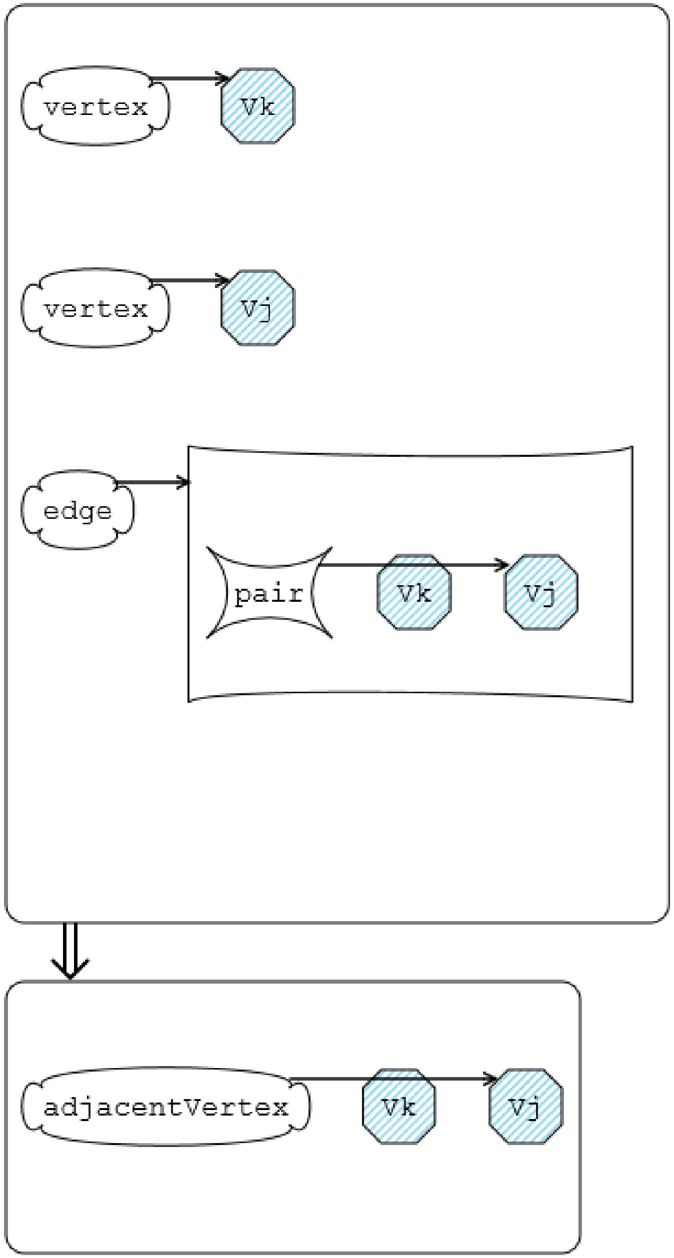
Test Cases in Math Education

- Set of input and output pairs used to evaluate functionality and features of the tool
- Graph theory knowledge visualized in Grailog demonstrates the accurateness of the tool and its ability to visualize complex terms with arbitrary levels of nesting

Hornlog Example: Multi-Premise Rule

"If Vk is a vertex,
Vj is a vertex,
and the pair of
vertices Vk, Vj
is an edge,
then Vk is an
adjacent vertex to Vj"

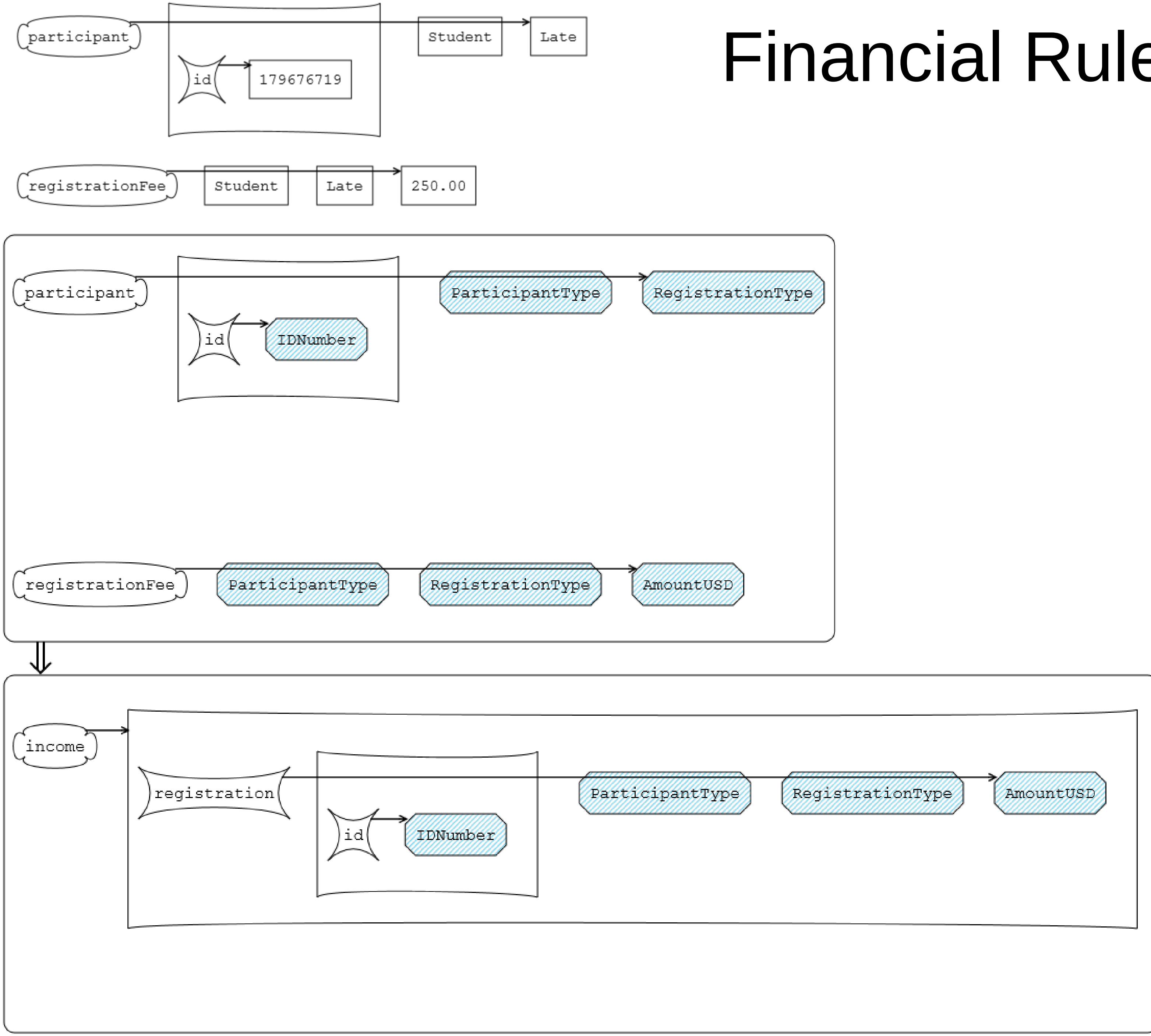
```xml
<Implies>
    <And>
        <Atom>
            <Rel>vertex</Rel>
            <Var>Vk</Var>
        </Atom>
        <Atom>
            <Rel>vertex</Rel>
            <Var>Vj</Var>
        </Atom>
        <Atom>
            <Rel>edge</Rel>
            <Expr>
                <Fun per="copy">pair</Fun>
                <Var>Vk</Var>
                <Var>Vj</Var>
            </Expr>
        </Atom>
    </And>
    <Atom>
        <Rel>adjacentVertex</Rel>
        <Var>Vk</Var>
        <Var>Vj</Var>
    </Atom>
</Implies>
```



Use Case in Financial Math

- Teaches business rules for managing the financial aspect of a non-profit organization
- Financial rules expressed in Hornlog RuleML were transformed to Grailog visualization
- Demonstrates uses of the tool:
  - Corporate memory
  - Knowledge transfer (training new personnel)
  - Knowledge validation

Financial Rules



Results (1)

- Grailog KS Viz has been extended to the labelnode normal form of Grailog with n-ary (including unary) relations
- Visualizes Datalog$^+$ and Hornlog$^+$ Equality
- Visualizes Hornlog's nested function applications, allowing arbitrary levels of nesting
- Tested on common modern Web browsers: IE, Firefox, Chrome, Safari
- Instant rendering of test cases and use case
- Grailog KS Viz 2.0 provides security and efficiency for viewing, sharing, and storing visualizations

Results (2)

- Formal validation of resulting SVG 1.1 by W3C Markup Validation Service
- Use of template parameters demonstrates improved design to increase reusability for future development
- Removal of JavaScript by the Purifier XSLT:
  - Reduces the time to generate the visualizations
  - Results in significantly smaller file sizes
  - Provides assurances of security when sharing the visualizations
- Download: http://www2.unb.ca/~lbidlak1/GrailogKSViz2.0.html

Future Work

- Complement browser-XSLT by online-XSLT-processor use
- Continue to improve software reusability
- Optional merging of labelnode copies
- Inverse translator, parsing Grailog into RuleML
- Extend to visualize more languages of RuleML such as First Order Logic (FOL), Higher-Order, and Modal RuleML