

XSLT based method for automatic generation of a graphical representation of a decision diagram represented using XML

Stanković Stanislav, Jaakko Astola
Institute of Signal Processing
Tampere University of Technology



TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan osasto



Project


Main project:

Methods for Efficient Representation of
Decision Diagrams



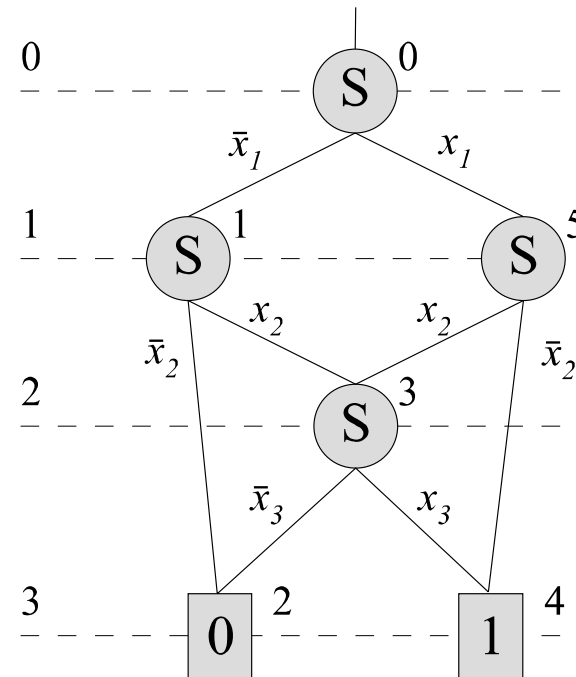


Outline of Presentation

- XML based framework for DDs:
 - Decision Diagrams
 - What is XML
 - Data structures for representation of DDs
 - XML representation of DDs
 - Applications of proposed system
 - Framework in larger context
 - SVG vector graphic language
 - XSLT conversion mechanism
 - Automatic conversion to SVG
 - Examples
 - Conclusions
 - Future work
- 

Decision Diagrams

- Efficient method for representation of discrete functions
(in terms of time and memory)
- Obtained by reduction of a decision tree
- Recursive application of a decomposition rule
(i.e. Shannon, Positive Davio, Negative Davio)
- Canonic representation





What is XML

XML (eXtensible Markup Language) - special data descriptive language

Able to represent various kinds of data with complex internal structure

Specifies **rules** describing the structure,
not the structure itself



XML and DD

Recursive structure of decision diagrams translates well to XML

XML representation flexible enough for different types of decision diagrams

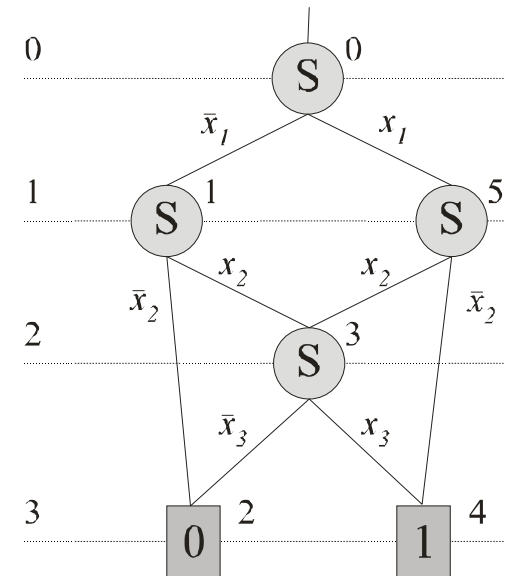
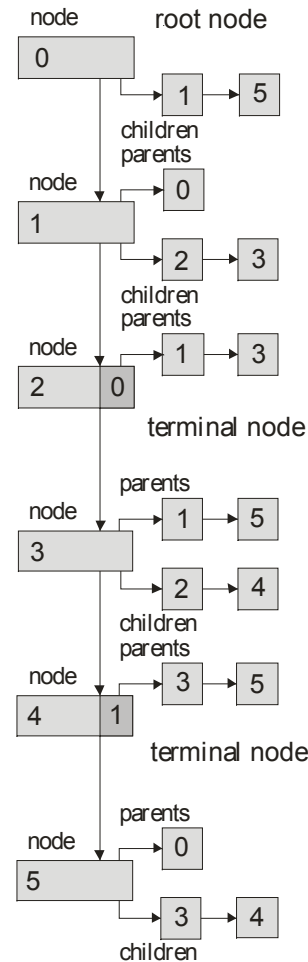
Easy manipulation and conversion to different, application specific, formats



Decision Diagrams Data Structures

Standard approach:

- Decision diagram = directed acyclic graph
- Nodes and Edges (connections)
- Nodes stored in a linked list
- Children and parents stored in linked lists (edges)
- Number of parents and children in general not limited
- Each node may contain additional data (information about decomposition rule)



XML Document

- Structured text
- Hierarchy of elements
(nested elements)
- Tree-like structure
- Recursive structure
(elements can contain other elements of the same type)

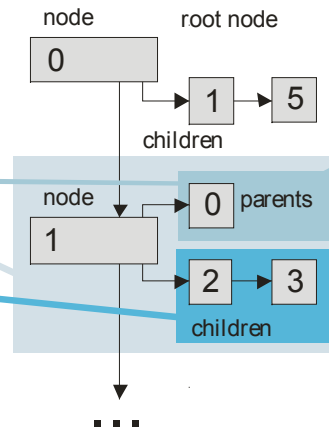
```
<diagram>
  <node>
    <data/>
  </node>
  <node>
    <node>
      <data/>
    </node>
    <node>
      <data/>
    </node>
  </node>
</diagram>
```

XML for Decision Diagrams

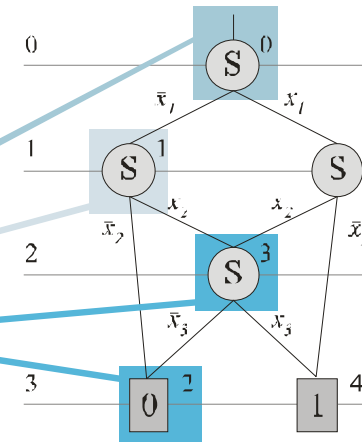
XML code

```
<?xml version="1.0" ?>
<dd:tree type="BDD" num_levels="4">
  <dd:root id="0" level="0" rule="Shannon">
    <dd:next id="1" level="1" rule="Shannon">
      ...
      <dd:parents point="0"/>
      <dd:children point="2">
        <dd:next_child point="3"/>
      </dd:children>
    </dd:next>
    <dd:children point="1">
      <dd:next_child point="5"/>
    </dd:children>
  </dd:root>
</dd:tree>
```

Data structure

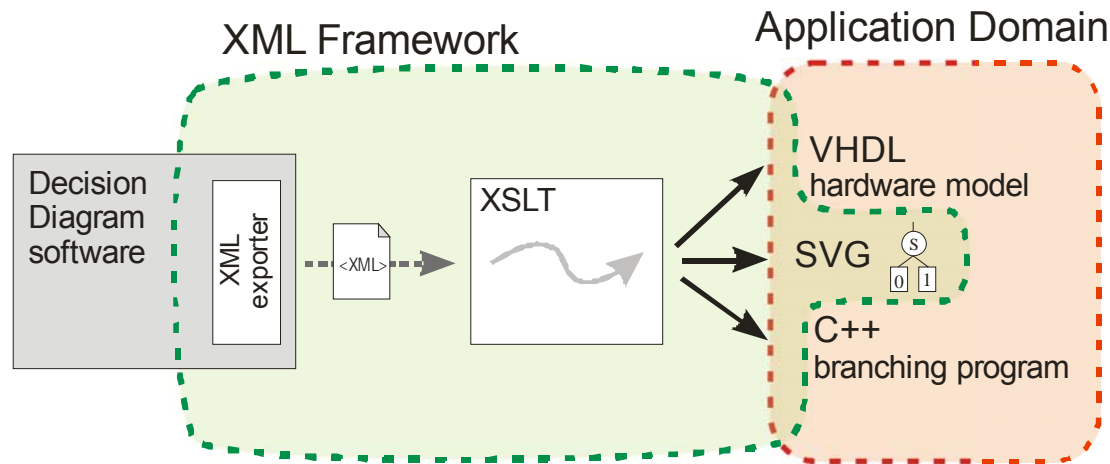


Graph representation



- XML elements correspond to decision diagrams data structures:
 - `<tree>` - top level element in hierarchy, Wrapper for all other elements
 - `<node>` - attributes: rule, ID, type, etc.
 - `<children>` - descendants of the current node
- Addition data stored in form of attributes


Possible applications



- XML based framework as an intermediary between decision diagram software and applications
- Easy conversion to application specific format
- XSLT as a **conversion** mechanism
- One possible application, visualization using SVG



Why SVG?

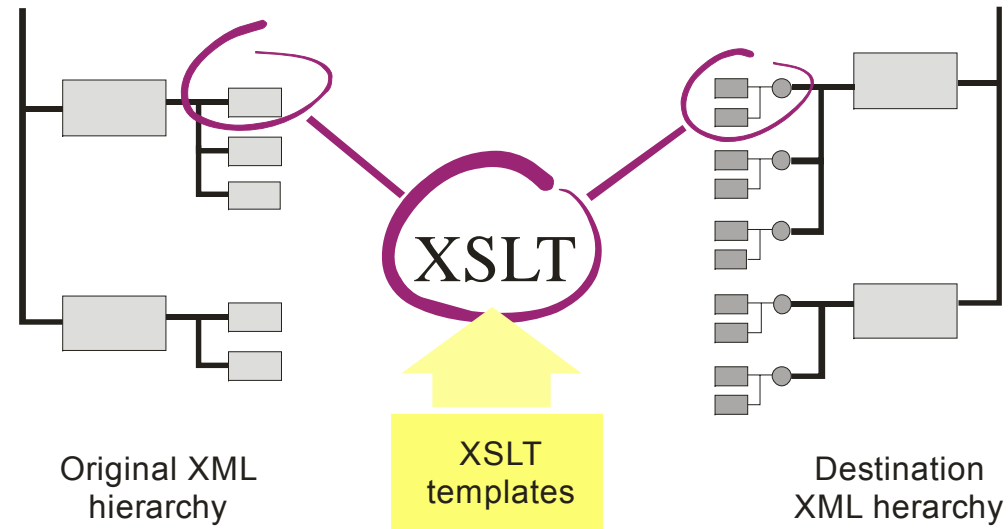
- Illustration of **flexibility** of proposed system
 - Use **XML** to **represent** decision diagrams
 - Use **XSLT** to **convert** this representation in other (application specific) formats
 - Automatic **visualization** using **SVG** one such application
 - SVG is XML based = easy conversion process
- 



Scalable Vector Graphics

- Special purpose mark-up language for 2D vector graphics
- Based on XML – easy manipulation using XSLT
- Suitable for various visualization applications
- Uses a standard limited set of graphical primitives to describe the image
- Image document in form of a structured text
(just like any XML document)

XSLT




- Establish **correspondence** of elements of two XML hierarchies
- Specify **set of rules** how elements are converted
- Rules given in form of **XSLT Templates**
- XSLT Style sheet = set of XSLT templates = XSLT “script”



Conversion using XSLT

Two level process:

- Preprocessing
 - Visualizations
- 




Preprocessing

- XML document may **contain** data **unnecessary** for this process. (i.e. explicit links to parent nodes)
- Recursive nature of decision diagrams/XML documents not well suited for conversion process
- Take in the original XML document, strip away all unnecessary data and produce new, intermediate XML document
- Processing done using a separate XSLT style sheet




Preprocessing II

- Each element of original XML document has an equivalent element in intermediate XML document.
 - Identify the element.
 - Apply the appropriate XSLT template from the style sheet.
 - Template generates a new element in intermediary XML document.
 - Identify next sub-element and repeat the process using appropriate template.
- 



Visualization

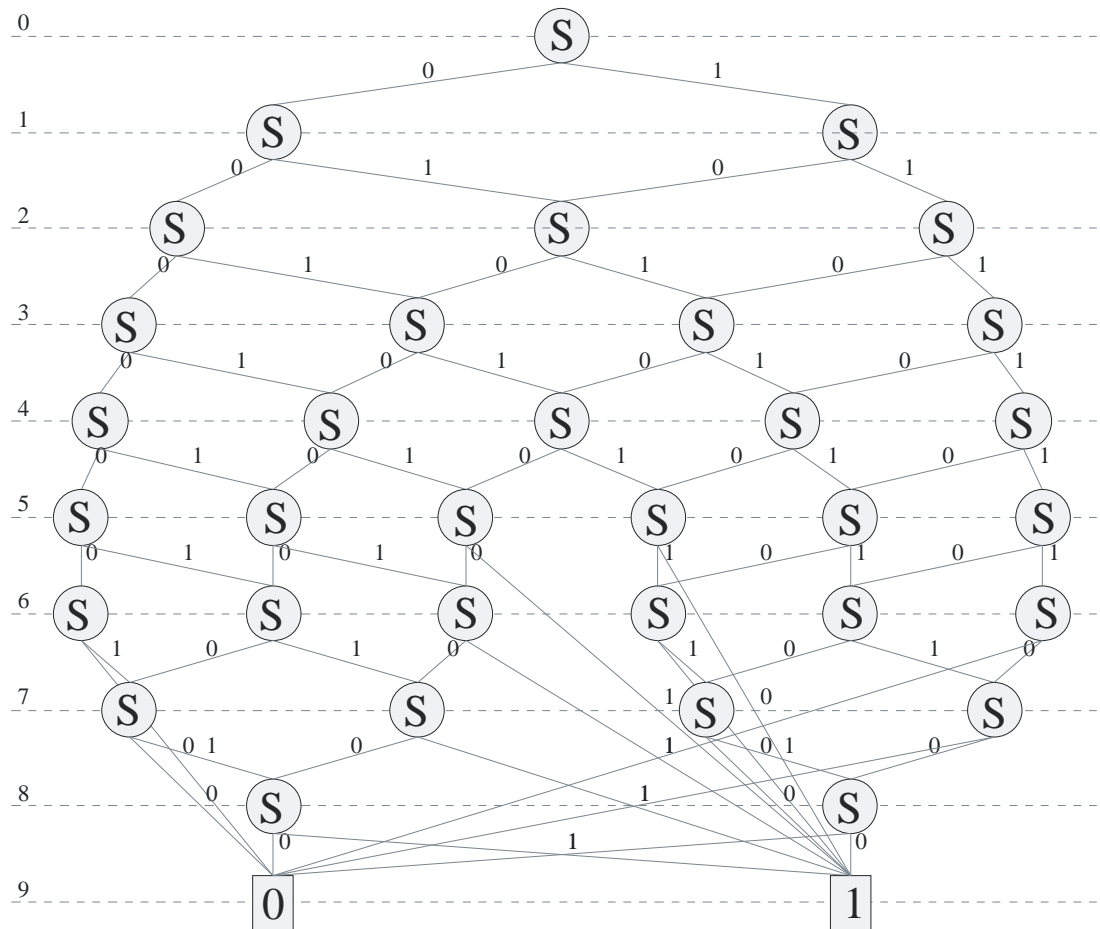
- Same basic idea as preprocessing
 - Produce final **SVG image** using an **intermediary XML document**
 - Conversion using a separate XSLT style sheet
 - Each **XML element** is represented with a **graphic element**
 - Graphic elements expressed in terms of SVG syntax
- 



Visualization II

- `<tree>` = image container
(specifies basic image properties i.e. width and height)
- `<node>` and `<root>` = circle (non-terminal), rectangle (terminal)
- `<children>` and `<next_child>` = lines, diagram edges connecting the nodes
- Appropriate text labels for all graphics elements based on attributes of XML elements


Example



9sym, benchmark function from MCNC set.




Conclusion

- Proof of concept
 - XML based platform flexible and robust enough to represent different types of binary decision diagrams
 - XSLT can be used to automatically convert XML documents to other types of representation
- 



Future work

- Use the same XSLT based methodology for **other applications**
 - Conversion even to **non-XML** based description languages
 - One possible application: automatically generate **hardware models** in **VHDL**
- 



Acknowledgements

Authors are grateful to the Reviewers whose constructive and useful comments improved presentation in this paper.

This work was supported by the Academy of Finland, Finnish Center of Excellence Program, Grant No. 5107491.

