Saint-Petersburg
State University ITMO

Automata-based Algorithms Visualization Framework

Georgiy Korneev

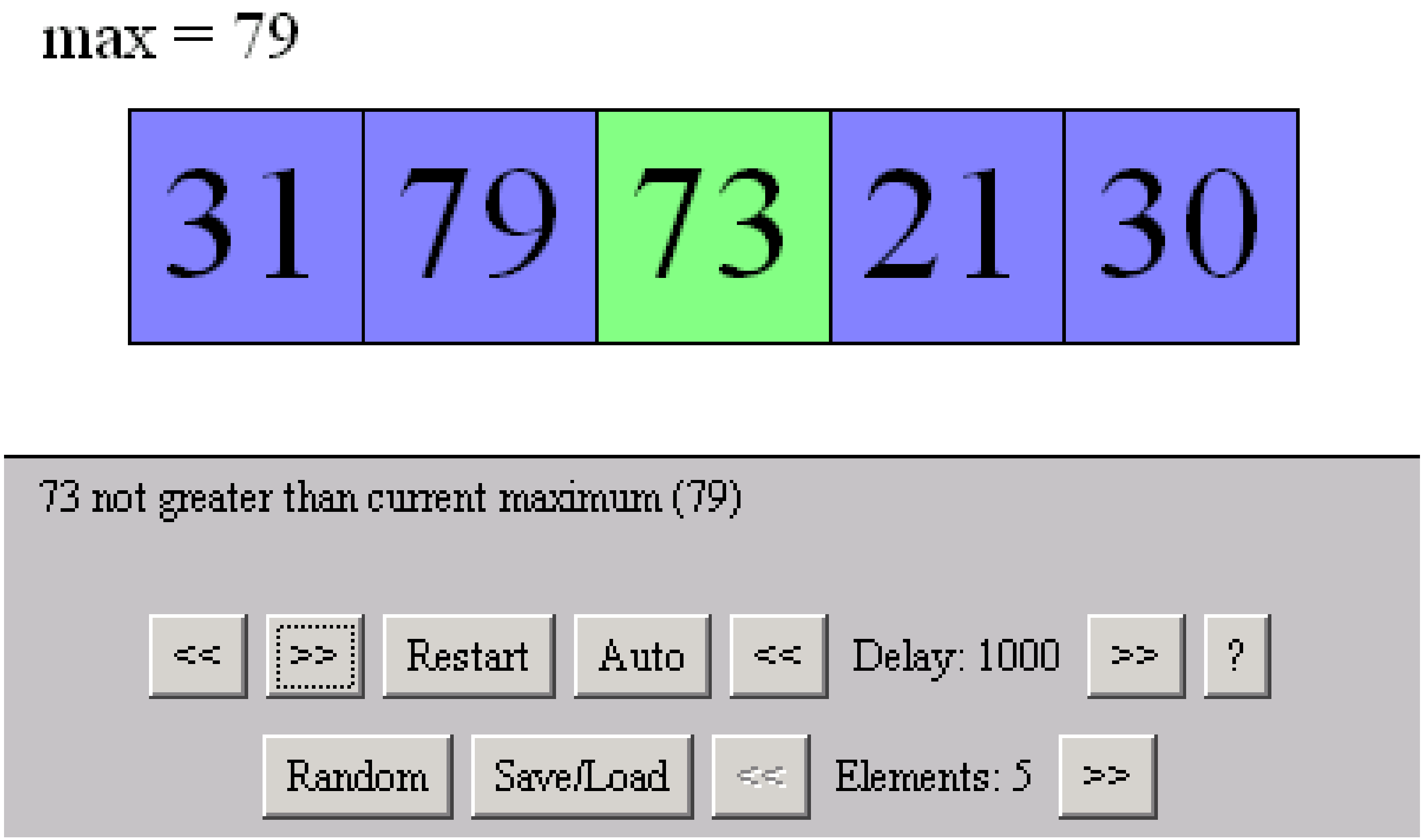
kgeorgiy@rain.ifmo.ru

Computer Technology Department,

Saint-Petersburg State University of
Information Technology, Mechanics and Optics,

Russia

What is visualizer?



Visualizers advantages

- Better algorithms understandability
- Dynamic content
- User can choose input data
- Forward and backward tracing
- Detailed comments

Saint-Petersburg
State University ITMO

Visualizer requirements

- Interface simplicity
- Show all algorithm stages and special cases
- Clear comments for all stages
- Hints for every visualizer element
- Big and small steps
- Automated execution

Saint-Petersburg
State University ITMO

Visualizers history in ITMO

- Sergey Stolyar
  - Initial stage (1998-1999)
- Matvey Kazakov
  - Transitional stage (1999-2000)
- Georgiy Korneev
  - Evolution stage (2000-2002)
  - Industrial stage (2002-2004)

Saint-Petersburg
State University ITMO

Initial stage

- First attempts
- Visualizers in Borland Delphi
- Handmade user interface
- Handmade visualizers logic

- Less than ten visualizers were made
- Very slow development
- Lots of errors

Transitional stage

- Visualizers in Java (applets)
- Similar user interfaces
- First attempts in reversing

- About 15 visualizers
- Slow development
- Less errors

Evolution stage

- Common user interfaces (BaseApplet library)
- Common configuration
- Template-based logic
- All visualizers are reversible

- More than 50 visualizers
- Fast GUI development
- Difficulties in reversing

Saint-Petersburg
State University ITMO

Industrial stage

- Standard user interface
- Automated code generation
- Automated program reverse

- More than 200 visualizers
- Fast logic development
- Automated reverse
- Quick development

Vizi visualization technology

- Visual representation library
- Logic definition language
- Visualizer programming methodology
- Standard project execution flow
- Documentation infrastructure

- Vizi visualization framework binds it together

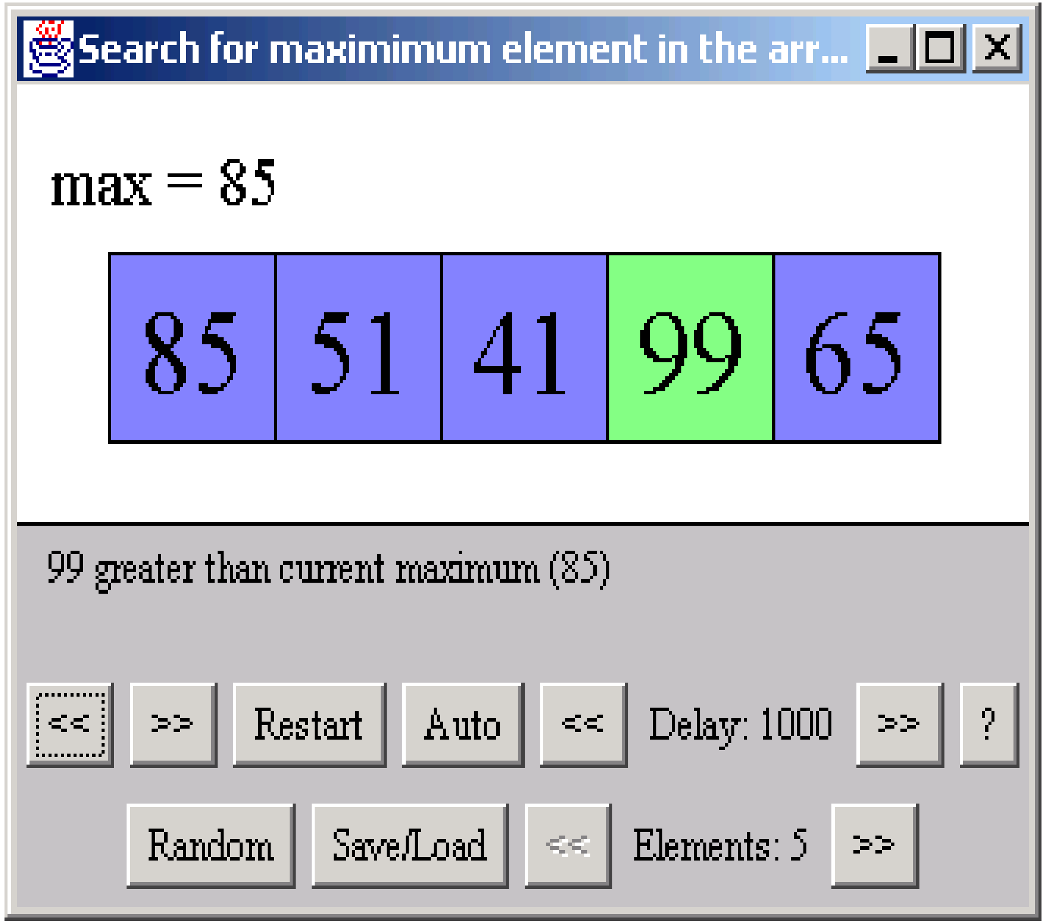
Saint-Petersburg
State University ITMO

Standard interface (1)

Drawing area

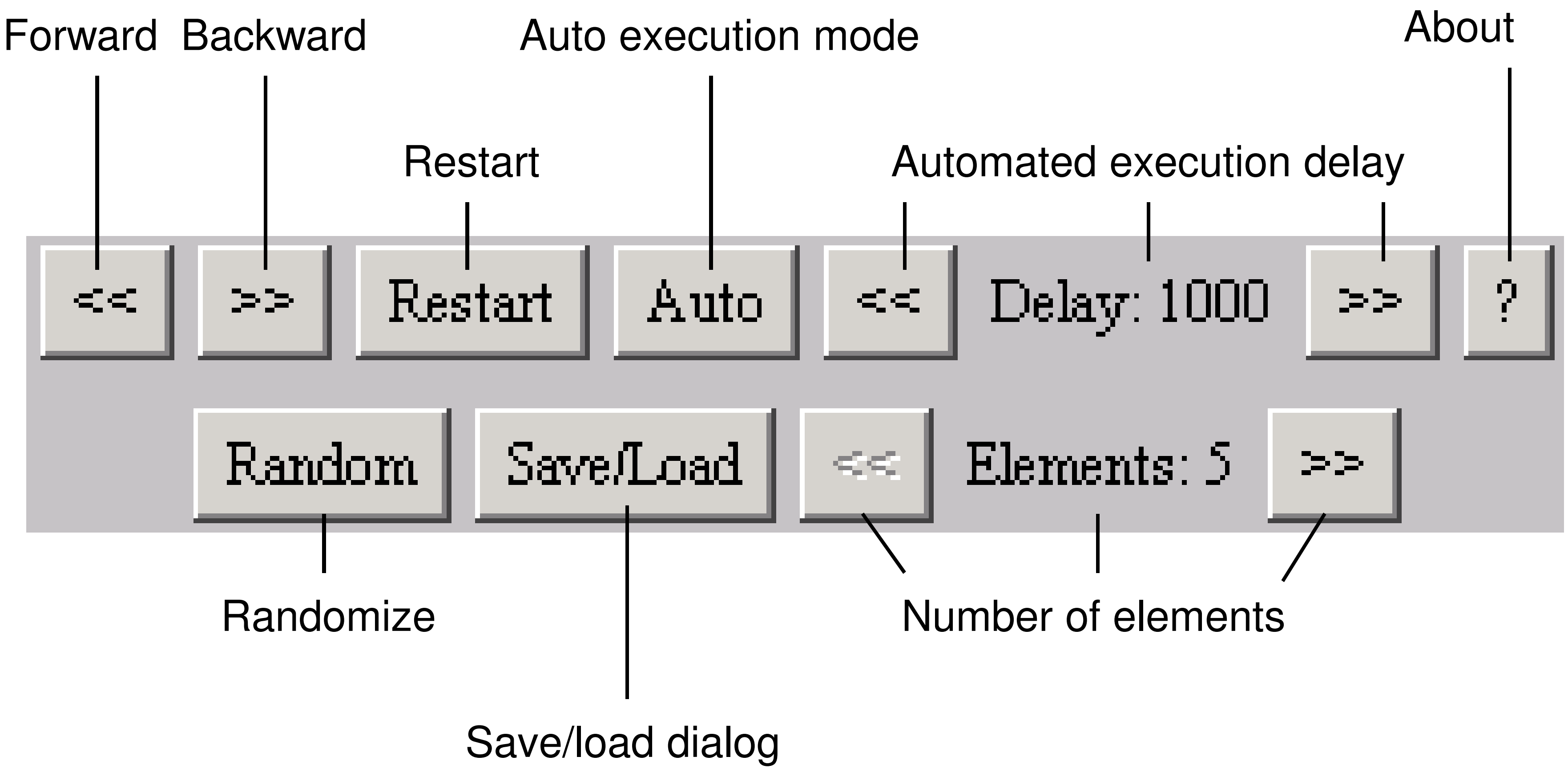
Comment area

Controls area



Saint-Petersburg
State University ITMO

Standard interface (2)



Logic definition language

- XML-based
- Program-like
- Auto-commented
- Auto documentation generation

Saint-Petersburg
State University ITMO

Visualizer programming Methodology

- Based on Switch-technology
- Formal program transformation
- Automated source code generation
- Model-View-Controller paradigm

Switch-technology in visualization

- Proposed by Anatoly Shalyto (1991)
- First application for visualizers
  Matvey Kazakov (2002)
  - Implementation
  - Block diagrams
  - Automata transition diagram
  - Automata-based visualizer programming
- Visualization and reversing technology
  Georgiy Korneev (2003)

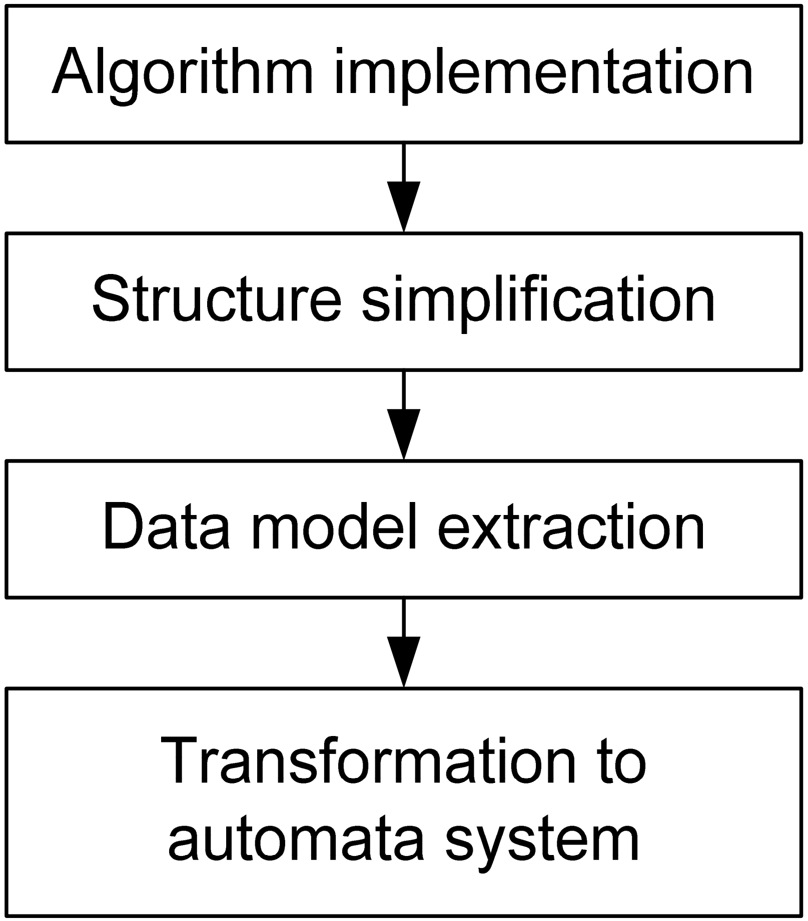
Saint-Petersburg
State University ITMO

Switch-technology in Vizi

- Correlated automata systems
- Automata pair for each procedure
- Automata pair
  - Forward transition graph
  - Backward transition graph
  - Shared automata states

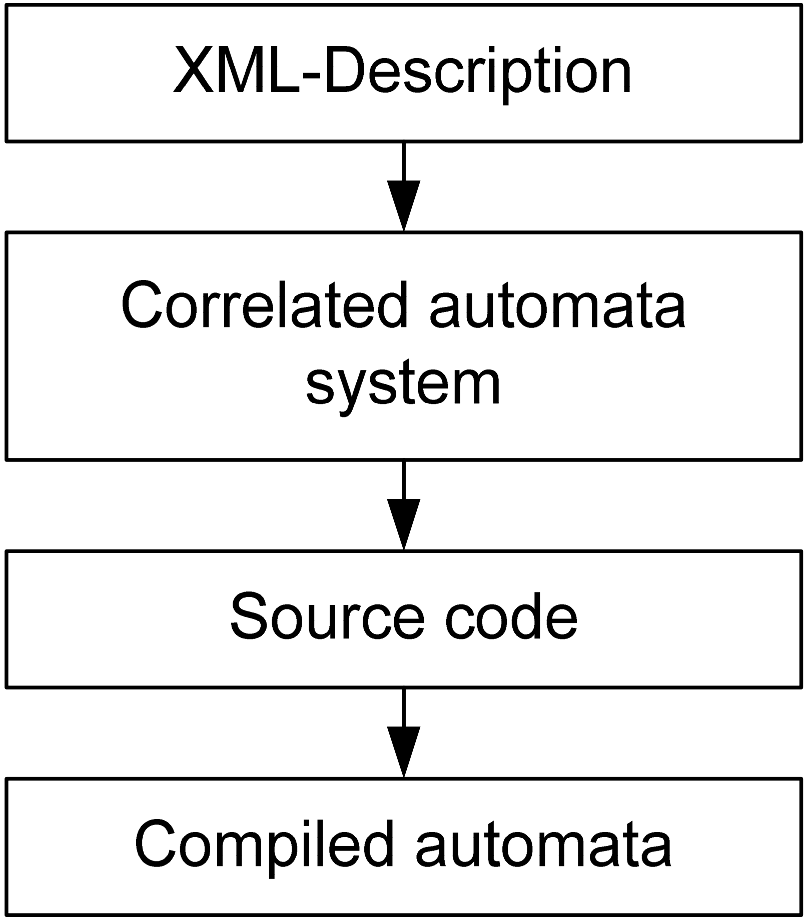
Saint-Petersburg
State University ITMO

Formal program transformations



Saint-Petersburg
State University ITMO

Automated source code generation



Model-View-Controller paradigm

- Model
  - Auto-generated correlated automata system
- View
  - User interface based on Vizi library
- Controller
  - Vizi library

Algorithm reversing problem

- Backward tracing is very useful for education
- Repeatable forward-backward navigation

- How get one step back?

Algorithm reversing problem solutions

- Per-step save/load technique
  - High memory requirements
- Program re-execution
  - Low execution speed
- Program reverse
  - Average memory requirements
  - High execution speed

Saint-Petersburg
State University ITMO

Program reversing difficulties

- Two versions of program
  - for forward tracing
  - for backward tracing
- Algorithm development for backward tracing
- Programs must be synchronized
- Complex modification
- Hard debugging

Automated program reversing

- One version of program (XML-source)
- Simple modification
- One-way debugging

Project execution flow

- Algorithm implementation
- Program structure simplification
- Data model extraction
- Transformation to correlated automata system
- Code generation

Algorithm implementation

- Algorithm is implemented using structural language
- Implementation source code is an input for other stages

Saint-Petersburg
State University ITMO

Program structure simplification

- Most of programming languages are too complicated for automated transformations
- Program should use only simple programming structures
- Structuring theorem

- Formal structure simplification for Java

Allowed structures

- Statements sequence
- Assignment statement
- Short conditional statement
- Full conditional statement
- “While” loop
- Procedure call

Data model extraction

- Data model contains all variables
- Data model fully represents algorithm state
- All data flows going through data model

- Formal data model extraction

Transformation into correlated automata system

- Correlated automata systems
- Pairwise automatas

- Formal and automated program transformation

Saint-Petersburg
State University ITMO

Project documentation (1)

- Annotation
  - Short project description
- Introduction
  - Brief algorithm description
  - Algorithm applications
  - Usage examples
- Chapter 1. Literature analysis
  - References and comments

Saint-Petersburg
State University ITMO

Project documentation (2)

- Chapter 2. Algorithm description
  - Complete algorithm descriptions
  - Special cases analysis
- Chapter 3. Algorithm implementation
  - Implementation comments
  - Decision ground
- Chapter 4. Implementation simplification
  - Simplified implementation

Project documentation (3)

- Chapter 5. Data model definition
  - Descriptions of data model variables
- Chapter 6. “Interesting” stages definition
  - Implementation is divided into “interesting” stages
  - Comments for “interesting” stages
- Chapter 7. User Interface Description
  - Pictures for each “interesting” stage
  - Controls descriptions

Project documentation (4)

- Chapter 8. Configuration Description
  - Complete configuration description
  - Initial configuration
- Conclusions
  - Visualizer properties
- References

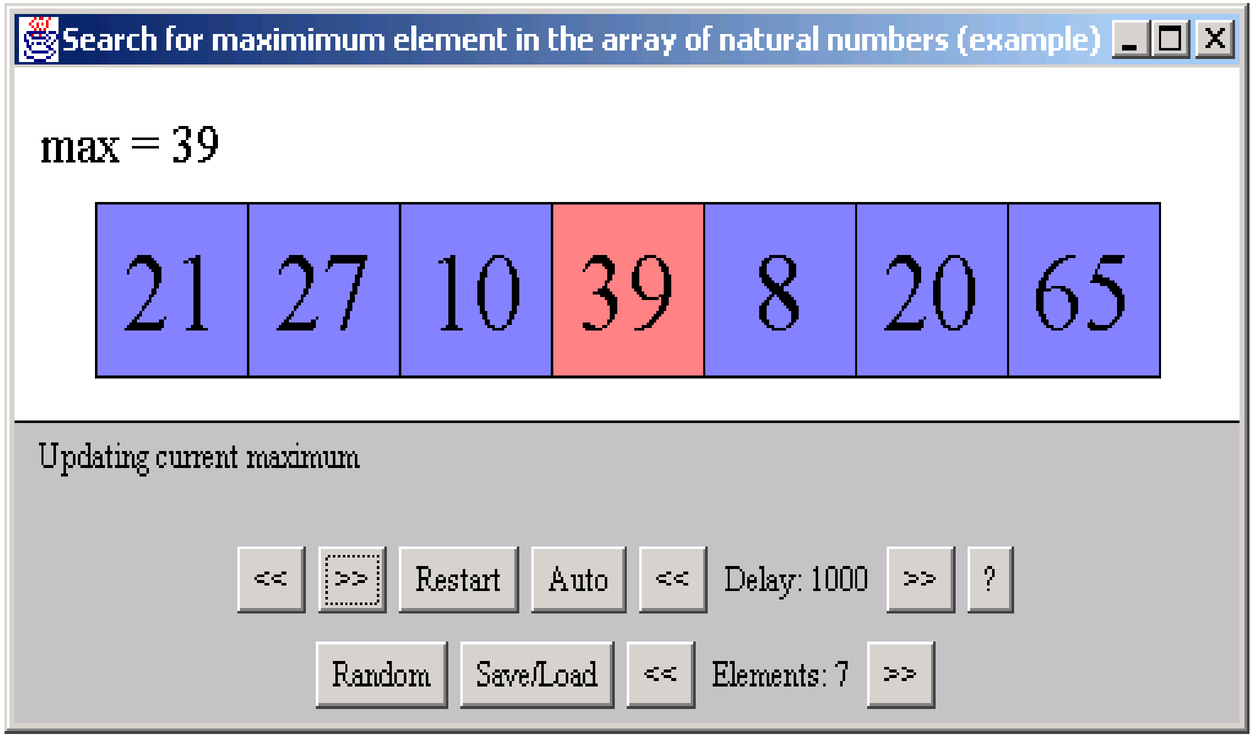
Project documentation (5)

- Appendixes
  - Algorithm Implementations Source Code
  - Transformed Implementation
  - Visualizer XML-description
  - Generated Source Codes
  - User Interfaces Source Codes

Saint-Petersburg
State University ITMO

Simple algorithm example

"FindMaximum" algorithm



FindMaximum
Algorithm implementation

```
int max = 0;
for (int i = 0; i < a.length; i++) {
    if (max < a[i]) {
        max = a[i];
    }
}
```

Saint-Petersburg
State University ITMO

FindMaximum

Simplified implementation

```
int max = 0;
int i = 0;
while (i < a.length) {
    if (max < a[i]) {
        max = a[i];
    }
    i++;
}
```

FindMaximum

Data model

```
public final static class Data {
    public int max;
    public int a[];
    public int Main_i;
}
```

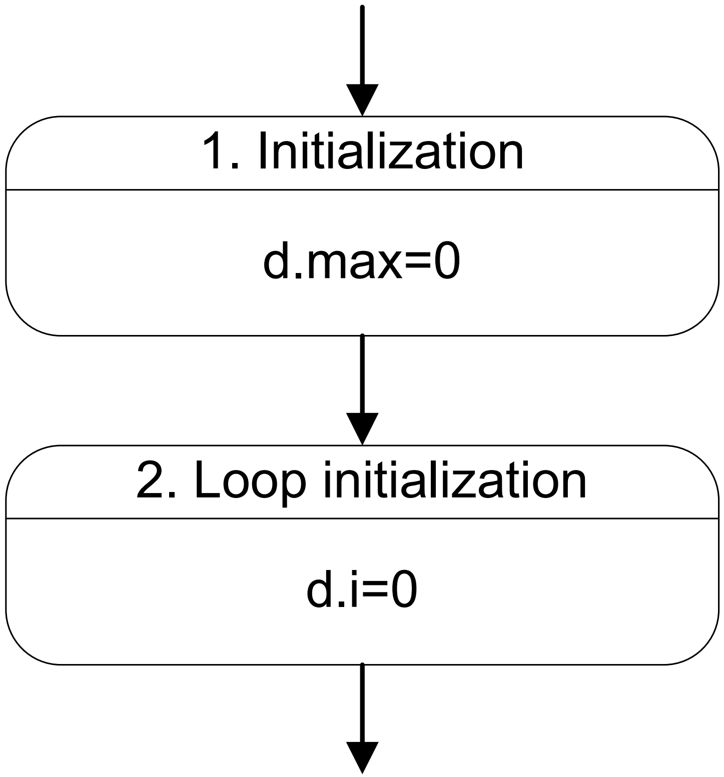
FindMaximum

Transformed implementation

```
d.max = 0;
d.i = 0;
while (d.i < d.a.length) {
    if (d.max < d.a[d.i]) {
        d.max = d.a[d.i];
    }
    d.i++;
}
```

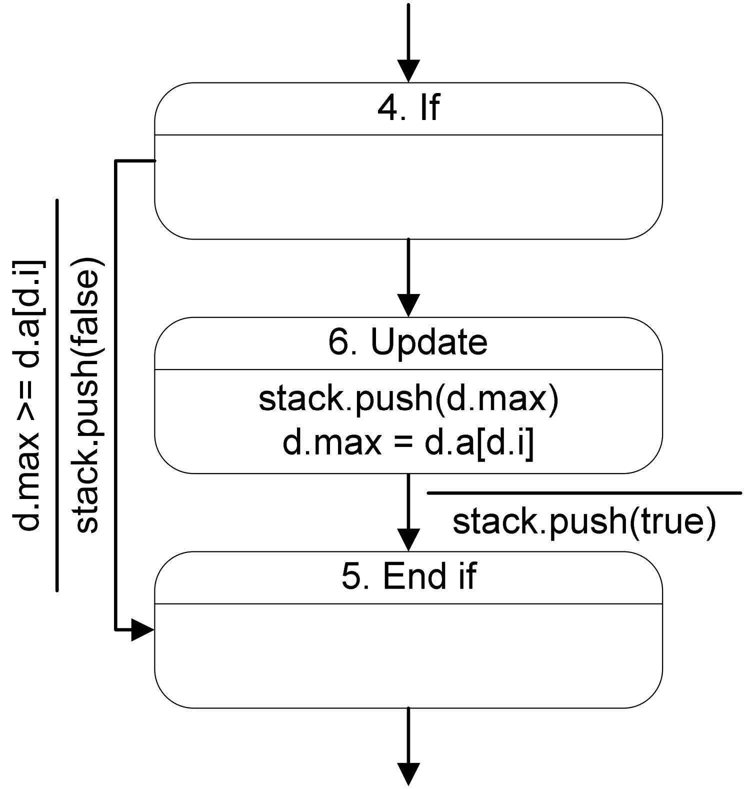
FindMaximum
Transformation to automata (1)

- Initialization



FindMaximum
Transformation to automata (2)

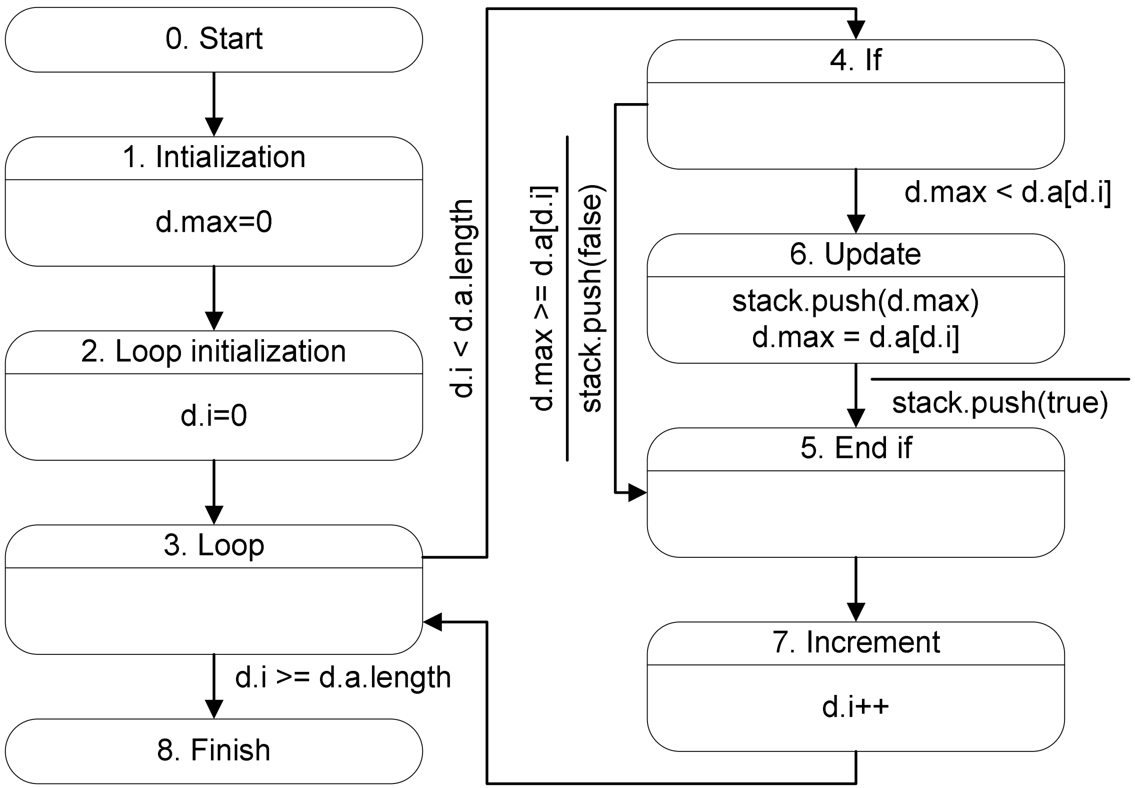
- If statement



Saint-Petersburg State University ITMO

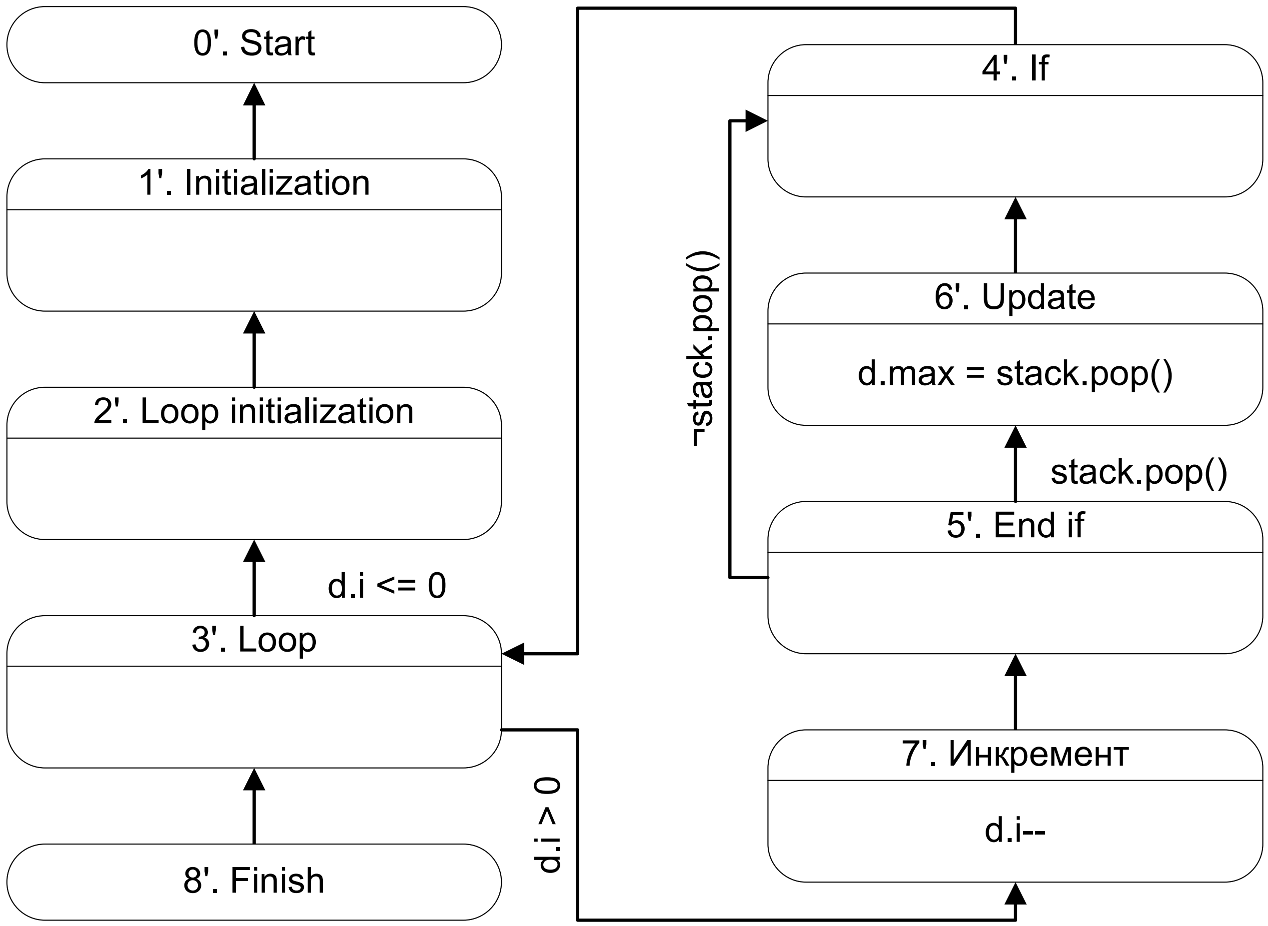
FindMaximum
Transformation to automata (3)

- Full automata



Saint-Petersburg
State University ITMO

FindMaximum
Reversed automata



FindMaximum

XML-representation

```
<step>@max @= 0;</step>
<step>@i @= 0;</step>
<while test="@i < @a.length">
  <if test="@max < @a[@i]">
     <then>
          <step>@max @= @a[@i];</step>
     </then>
  </if>
  <step>@i @= @i + 1;</step>
</while>
```

Saint-Petersburg State University ITMO

FindMaximum
Generated sources extract

```
switch (state) {
    case 0: { // Start state
        state = 1; // Initialization
        break;
    } case 1: { // Initialization
        d.max = 0;
        state = 2; // Loop init
        break;
    } case 2: { // Loop init
        d.i = 0;
        state = 3; // Loop
        break;
    } case 3: { // Loop
        if (d.Main_i < d.a.length)
            state = 4; // If
        else state = END_STATE;
        break;
    }
    case 4: { // if
        if (d.max < d.a[d.Main_i])
            state = 6; // Update
        else state = 5; // End if
        break;
    } case 5: { // End if
        state = 7; // Increment
        break;
    } case 6: { // Update
        d.max = d.a[d.Main_i];
        state = 5; // End if
        break;
    } case 7: { // Increment
        d.Main_i++;
        state = 3; // Loop
        break;
    }
```

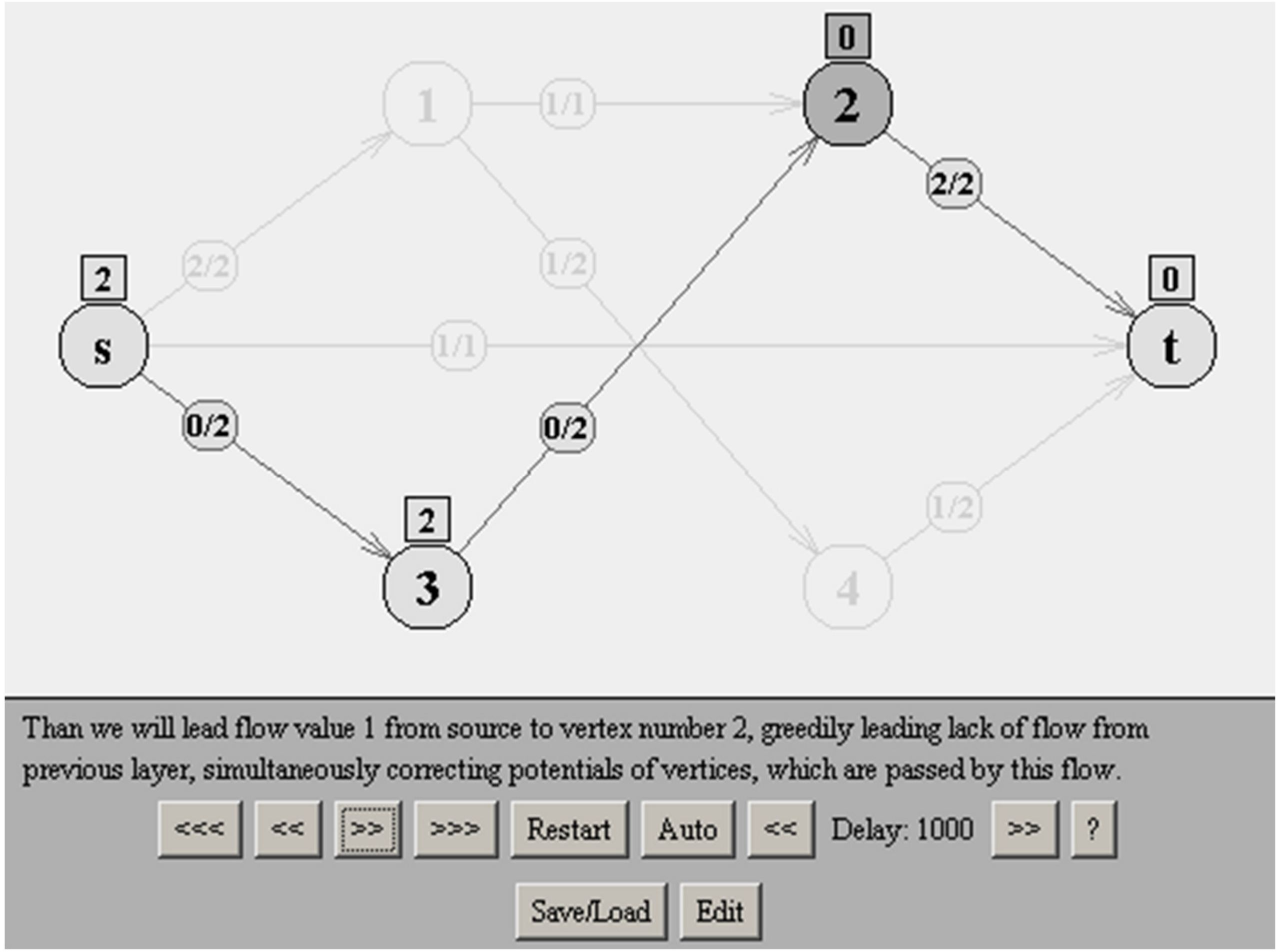
Find maximum
Statistics

Automata	2
States	9
Transitions	22
XML-source (with comments)	87 lines
Automata implementation	326 lines

Saint-Petersburg
State University ITMO

Complex algorithm example

Malhotra, Kumar, Maheshwari maximal flow algorithm



Malhotra, Kumar, Maheshwari Statistics

Automata	18
States	107
Transitions	228
XML-source (with comments)	516 lines
Automata implementation	4069 lines

Saint-Petersburg
State University ITMO

More examples

- Dinic network flow algorithm
- Hopcroft-Karp Bipartite Matching algorithm
- Chu-Liu shortest arborescence of a directed graph
- Algorithms on 2-3 threes
- Bitonic salesman problem
- Ukkonen suffix tree construction algorithm
- Prim minimum spanning tree algorithm
- Simple strings and de Bruin cycles construction algorithms
- …

Saint-Petersburg
State University ITMO

Links

- Vizi project homepage
  - http://ctddev.ifmo.ru/vizi
- Old visualizers examples
  - http://ips.ifmo.ru:8888/ru/visualizers/
- Visualizer-related switch-technology information
  - http://is.ifmo.ru/?i0=vis