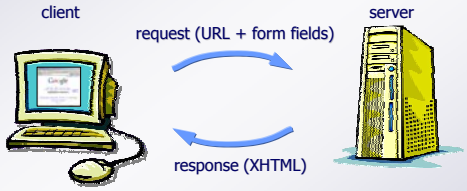
Static Analysis for Java Servlets and JSP

Christian Kirkegaard & Anders Møller

BRICS, University of Aarhus

Java Servlets and JSP

- A powerful framework for **Web application** development
- Communication using **HTTP**:



- JSP pages are compiled into servlets...

The Servlet API

- A **Web application** contains a collection of **servlets**
- A configuration file maps URL requests to servlet names
- Each **servlet** receives **user input** (form fields) and produces **XHTML** output

- *The output is generated by* ***printing characters to a stream!***

Example

Source code (1/3)

```
public class Entry extends javax.servlet.http.HttpServlet {
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession();
    String url =
      response.encodeURL(request.getContextPath()+"/show");
    session.setAttribute("timestamp", new Date());
    response.setContentType("application/xhtml+xml");
    PrintWriter out = response.getWriter();
    Wrapper.printHeader(out, "Enter name", session);
    out.print("<form action=\""+url+"\" method=\"POST\">"+
              "<input type=\"text\" name=\"NAME\"/>"+
              "<input type=\"submit\" value=\"lookup\"/>"+
              "</form>");
    Wrapper.printFooter(out);
} }
```


Source code (2/3)

```
public class Show extends javax.servlet.http.HttpServlet {
  protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
    throws ServletException, IOException {
    Directory directory =
      new Directory("ldap://ldap.widgets.org");
    String name =
      misc.encodeXML(request.getParameter("NAME"));
    response.setContentType("application/xhtml+xml");
    PrintWriter out = response.getWriter();
    Wrapper.printHeader(out, name, request.getSession());
    out.print("<b>Phone:</b> "+directory.phone(name));
    Wrapper.printFooter(out);
  }
}
```

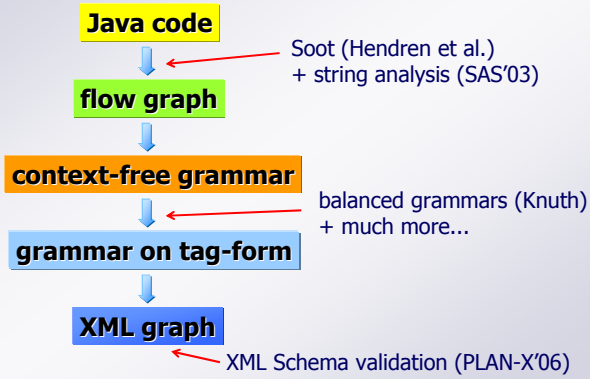
Source code (3/3)

```
public class Wrapper {
  static void printHeader(PrintWriter pw, String title,
                          HttpSession session) {
    pw.print("<html xmlns=\"http://www.w3.org/1999/xhtml\">"+
             "<head><title>"+title+"</title></head><body>"+
             "<hr size=\"1\"/>"+
             "<div align=\"right\"><small>"+
             "Session initiated ["+
             session.getAttribute("timestamp")+"]"+
             "</small></div><hr size=\"1\"/>"+
             "<h3>"+title+"</h3>");
  }
  static void printFooter(PrintWriter pw) {
    pw.print("<hr size=\"1\"/></body></html>");
  }
}
```


Two challenges for Web app developers

1. *Is the response always **well-formed** and **valid**?*
2. *Do **forms and fields** being generated always match the code that receives the input?*

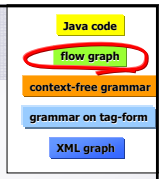
Overview of the analysis



Our results

- Construction of **context-free grammars** that approximate the possible output on **output streams in Java**
- Checks for XML **well-formedness** and XML Schema **validity** on **context-free grammars**
- **Inter-servlet control flow** for Web apps

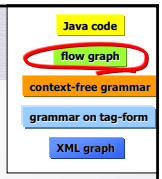
Flow graphs

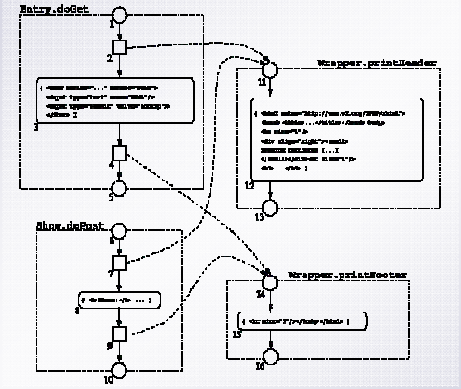


- *Nodes:*
  - `append[`*regular language*`]`
  - `invoke[`*possible targets*`]`
  - `nop`
  - `return`
- *Edges* represent control flow

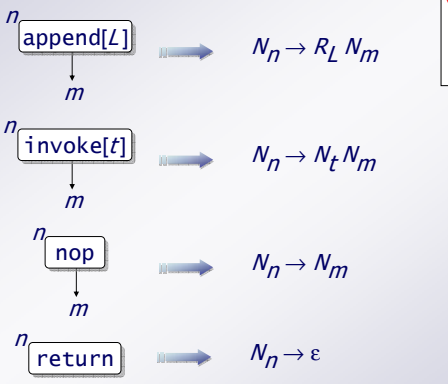
– **Soot** (Hendren et al.) gives us Jimple code, control flow analysis, and alias analysis
– **String analysis** (SAS'03) gives us a regular language for each string expression

Flow graph for the example





Flow graphs → context-free grammars



Grammar for the example

$$
\begin{array}{llll}
N_1 \to N_2 & N_6 \to N_7 & N_{11} \to N_{12} & N_{14} \to N_{15} \\
N_2 \to N_{11} N_3 & N_7 \to N_{11} N_8 & N_{12} \to R_{12} N_{13} & N_{15} \to R_{15} N_{16} \\
N_3 \to R_3 N_4 & N_8 \to R_8 N_9 & N_{13} \to \epsilon & N_{16} \to \epsilon \\
N_4 \to N_{14} N_5 & N_9 \to N_{14} N_{10} & & \\
N_5 \to \epsilon & N_{10} \to \epsilon & &
\end{array}
$$

$R_3 \to$ [<form action="..." method="POST">...</form>]
$R_6 \to$ [Phone:..]
$R_{12} \to$ [<html xmlns="http://www.w3.org/1999/xhtml"><head><title>...</title></head><body>..]
$R_{15} \to$ [<hr size="1"/></body></html>]

Tag-form

- $C \to$ < T A > C </ T > (element)
- $C \to X$ (text)
- $C \to C\, C$ (content sequence)
- $A \to W\, T$ = " V " (attribute)
- $A \to A\, A$ (attribute sequence)
- $A \to \varepsilon$ (empty attr. seq.)

– if the grammar can't be brought on this form, its language is not **well-formed XML**

– on tag-form, we can easily check the remaining properties for well-formedness...

Transforming to tag-form

1. make </ a single symbol
2. obtain **balanced grammar** with respect to < and </
   - *Knuth '67*
   - *Mohri & Nederhof '01*
3. obtain unique contexts (tag / element content / attribute value) – if possible
4. apply standard grammar transformations (inlining, expansion)...

Tag-form grammar for the example

$C_1 \to$ < html A_1 > C_2 C_4 </ html >
$C_2 \to$ < head > C_3 </ head >
$C_3 \to$ < title > X_1 </ title >
$C_4 \to$ < body > C_{10} C_{11} C_{12} C_5 C_9 C_{10} </ body >
$C_5 \to$ < form A_4 A_5 > C_6 C_7 </ form >
$C_6 \to$ < input A_6 A_7 > </ input >
$C_7 \to$ < input A_8 A_9 > </ input >

$A_1 \to$ _xmlns="http://www.w3.org/1999/xhtml"
$A_2 \to$ _size="1"
$A_3 \to$ _align="right"
$A_4 \to$ _action=" V_1 "
$A_5 \to$ _method="POST"

$X_1 \to$ Enter_name | L_{CDATA}
$X_2 \to$ Session_initiated_[L_{DATE}]
$X_3 \to L_{\text{Name}}$

$C_8 \to$ < h3 > X
$C_9 \to C_8 \mid C_{12}\, X_3$
$C_{10} \to$ < hr A_2 ></ hr >
$C_{11} \to$ < div A_3 > C_{12} </ div >
$C_{12} \to$ < small > X_2 </ small >
$C_{13} \to$ < b > Phone: </ b >

$A_6 \to$ _type="text"
$A_7 \to$ _name="NAME"
$A_8 \to$ _type="submit"
$A_9 \to$ _value="lookup"

$V_1 \to$ contextpath/show

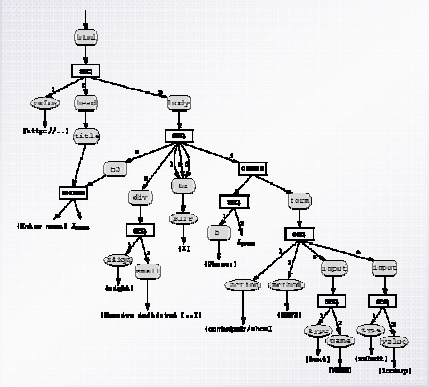
Checking validity with XML graphs

- An **XML graph** is like an XML tree but with
  - *choices* and *loops*
  - attribute/element names and text described with *regular string languages*

  (so one XML graph represents a *set* of concrete XML trees)

- Converting a context-free grammar on tag form into an XML graph is straightforward
- We have a tool for **validating** XML graphs relative to schemas (using **XML Schema**)!

XML graph for the example



Java code
flow graph
context-free grammar
grammar on tag-form
XML graph

Conclusion

- Construction of **context-free grammars** that approximate the possible output on **output streams in Java**
- *Sound and complete* checks for XML **well-formedness** and XML Schema **validity** on **context-free grammars**
- **Inter-servlet control flow** for Servlets/JSP Web apps
- ...and the implementation is on the way – reeeal soon now ☺

Java code
⇩
flow graph
⇩
context-free grammar
⇩
grammar on tag-form
⇩
XML graph