A .NET-based Test-Data Generator for Combinatorial Grammar- and Schema-based Testing

Vadim Zaytsev

with: Ralf Lämmel (VU), Wolfram Schulte (MSR)

14 April 2004
Grammarware

- grammars

- grammar-dependent software

- In this project:
  - XML Validators
  - W3C XML Schemata as grammars

http://www.cs.vu.nl/grammarware/
Scenarios for grammar-based testing

• testing virtual processors
  – virtual machines
  – just-in-time compilers

• testing front-ends
  – automated software modification & analysis

• testing implementations
  – optimisation of XPath
Combinatorial exploration

Grammar

Explosion

Adversary of stochastic testing
Explosion examples

Cardinalities per depth

- Depth 1
- Depth 2
- Depth 3
- Depth 4
- Depth 5
- Depth 6

0,1,-,+
0,1,2,+,−,*,/
C#
**Controlled explosion**

Grammar

Recursion control

Depth control

\[ \ldots \]

\[ \text{+ other mechanisms} \]
Control mechanisms

- depth control
  - maximum “length” of terms

- recursion control
  - nested constructor application

- equivalence control
  - build equivalence classes
Control mechanisms (contd.)

- balance control
  - limit the preceding levels

- combination control
  - limit Cartesian product
  - pair-wise testing

- context control
  - enforce context conditions
Emphases in this project

- The case study of XSD usage in testing XML Validators
- Implementing and using control mechanisms for test data generation
- Developing a tool to support combinatorial testing
Problem

System Under Test

![Diagram showing XML and XSD feeding into a Validator, which outputs YES or NO.]

Validator

XML -> YES

XSD -> NO
Solution

Stage A:

Oracle

YES

Validator

YES

NO

GOOD/BAD

TDGenerator

XML

XSD
Solution

Stage B:

- TDGenerator
  - XSD
  - XML
- Validator
  - YES
  - NO
- Oracle
  - GOOD/BAD
What kind of Oracle?

- Differential testing
  - run two or more against one another
  - if the outputs are different, something has to be wrong
  - in our case: different XML Validators
    * using Microsoft .NET API
    * Sun Multi-Schema XML Validator (JAXB)
    * Ant Validation Task (JBind)
The Tool we have

Tree = Nil
    | Node(Tree,Tree)
;

Constructors of all sorts

Terms as objects
+Serialisation

Grammar: Tree Nil = Nil Node-1(Nil,Nil) ...

Terms as objects +Serialisation
Solution proposition

The Tool

Grammar in BNF

Terms in text

Grammar

Parsing

Sorts and Constructors

Terms

Serialisation

Test data

XSD Mapping

XML Mapping

XSD

XML
Problems underway

• XSD is not meant to be implemented
  – (as a whole)

• YACCification

• how to deal with XML attributes

• implementing control mechanisms

• …
Conclusion

The buzzwords are:

- Test data generation
- Combinatorial testing
- Controlled explosion
- Differential testing
- The .NET Framework