Combinatorial Test Set Generation:

Concepts, Implementation, Case Study

Drs. Vadim V. Zaytsev

22 June 2004

Legal stuff

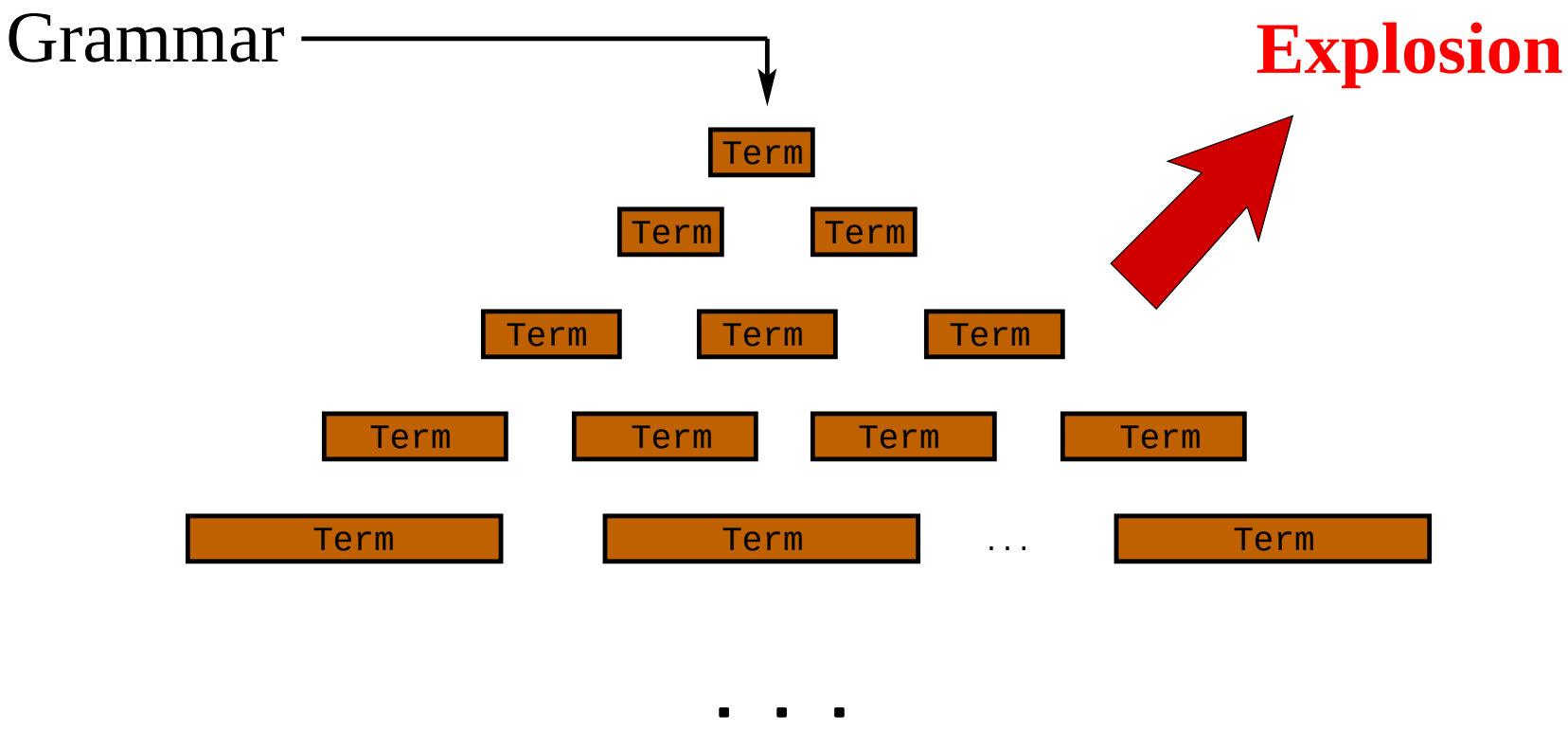
- Supervisor: Prof.dr. Hendrik Brinksma, UT
- Ext.Supervisor: Dr.ing. Ralf Lämmel, VU&CWI
- Hosting organisation: Vrije Universiteit Amsterdam
- Contributes to a collaboration between Dr. Wolfram Schulte from MSR/FSE and Dr.ing. Ralf Lämmel from VU&CWI (**Geno** project).

Contents of the project

- Testing
  - Combinatorial test data generation
  - Differential approach
- *Grammar*ware
  - XML Schema as grammar description formalism
  - XML validators as grammar-based software

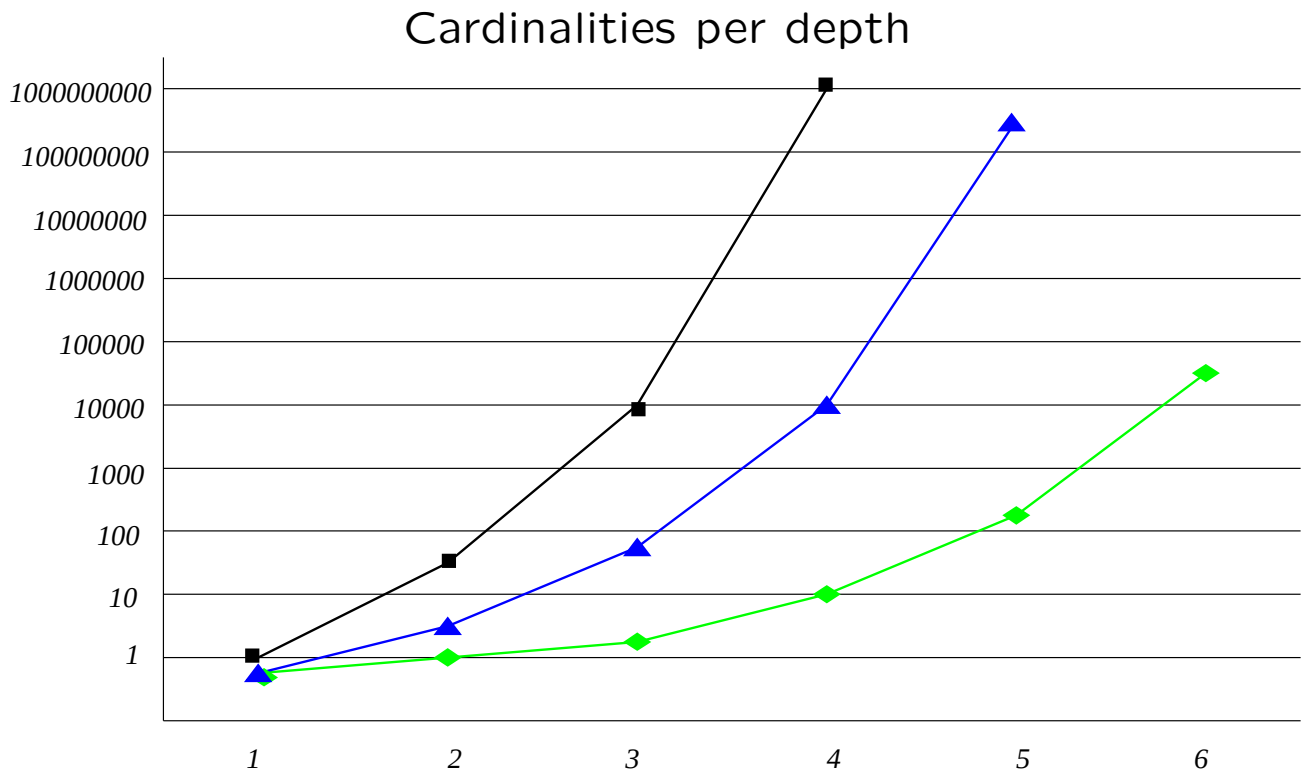
`http://www.cs.vu.nl/grammarware/`

Combinatorial exploration



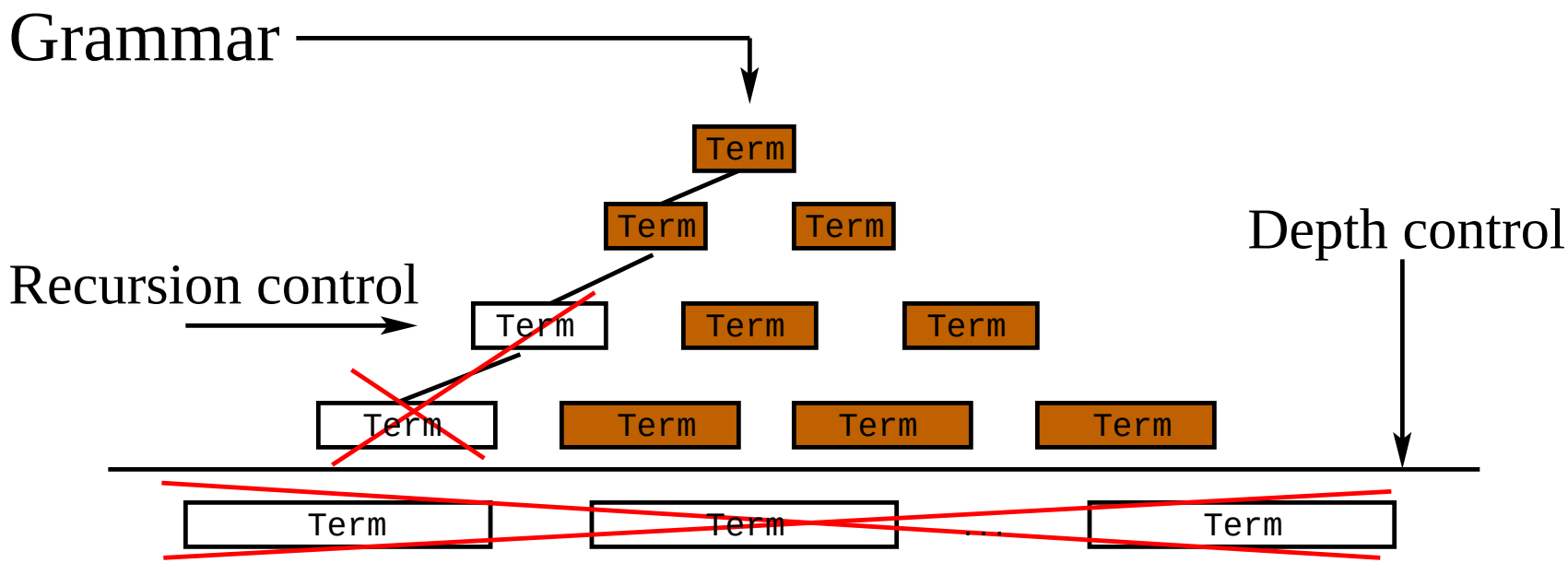
Adversary of stochastic testing

Explosion examples



Number of generated terms grows fast with depth and eventually explodes (becomes greater than 18446744073709551616).

Controlled explosion



. . .

\+ other mechanisms

Control mechanisms

- depth control
  - intuitive definition
- recursion control
  - nested unfolding of sorts
- equivalence control
  - building equivalence classes

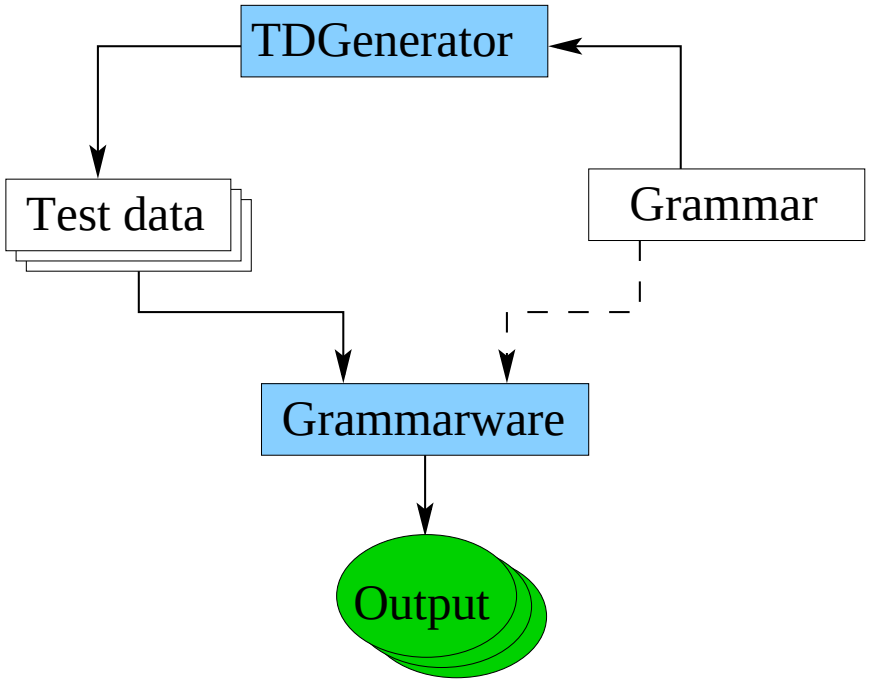
Control mechanisms (contd.)

- balance control
  - limit the preceding levels
- combination control
  - limit Cartesian product
  - pair-wise testing
- context control
  - enforce context conditions

What to test in the XML

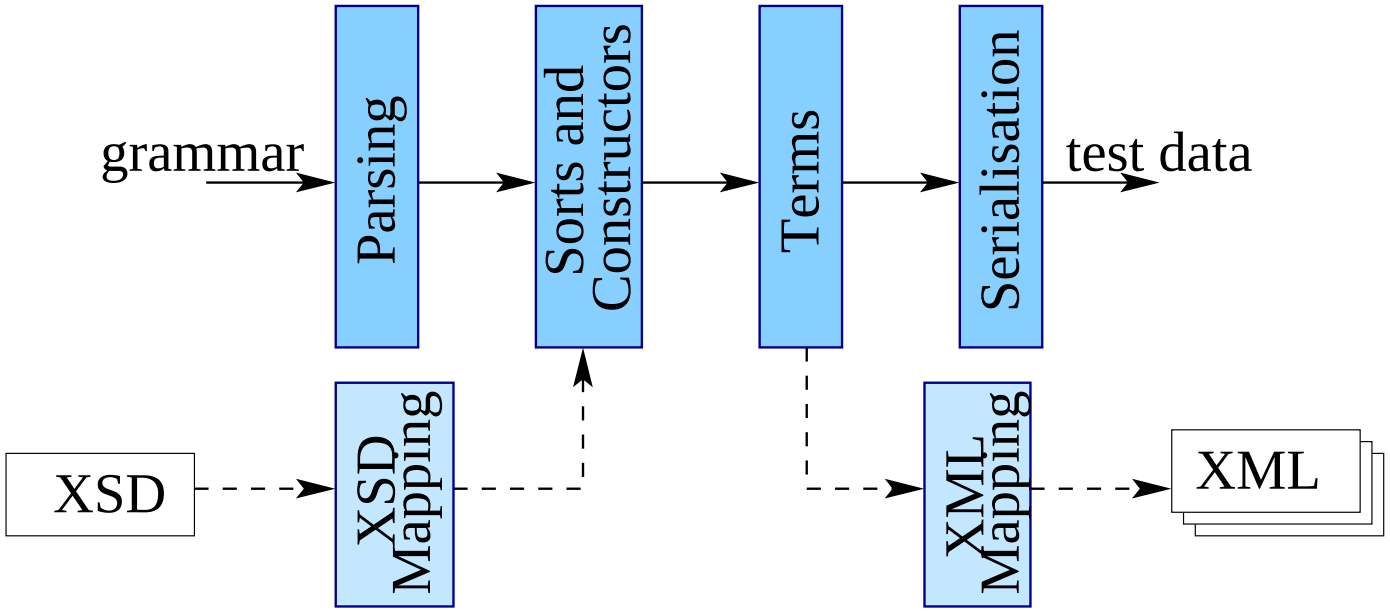
- levels of XML file conformance
- levels of XML processor conformance
- grammar features: attributes, references, . . .
- advanced features: namespaces, schema-related markup, . . .
- secondary features: header, scalability, . . .

Design of **Geno**



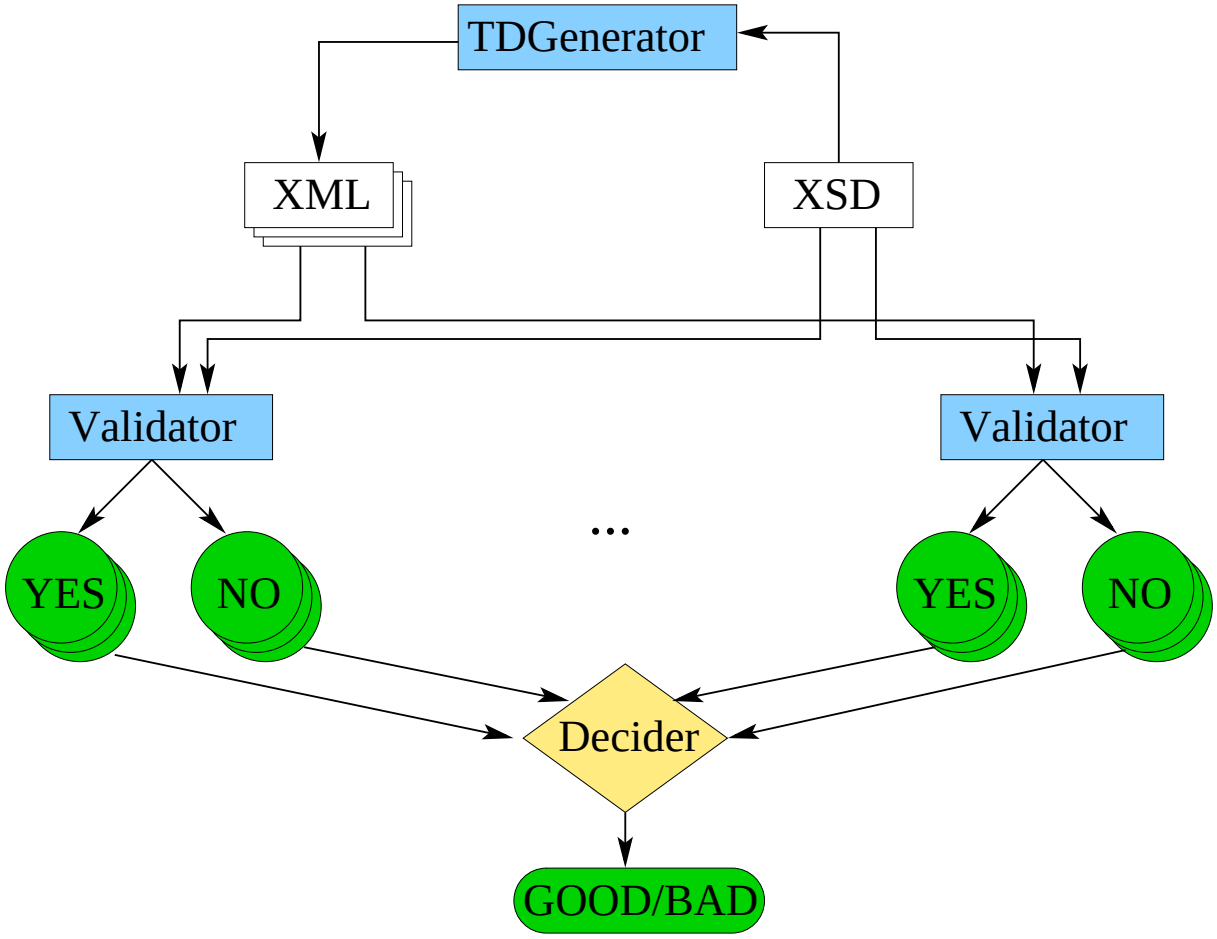
Grammar-based testing tool generates terms in a combinatorial way.

Work on **Geno**



Change the input language: grammar adaptation;
change the output language: serialisation.

Case study with **Geno**



XML validators

- C# .NET-based
  - wrapper written
- Sun Multi-Schema XML Validator 1.2
  - used as is
- Python XSV
  - wrapper written

Results

- The *infrastructure* of the XML-based data generator
- The *case study*: XHTML Strict 1.1

— — —

- Generation process *visualisation*
- Illustration and *rationalisation* of control mechanisms

Scenarios

- Huge valid test data set
- Grammar mutation
- Point-wise stress testing

	Depth reached	Sorts in the signature	Constructors in the signature	Terms total	Terms of the root sort
Valid	8	234	478	9914261	**37240**
Mutation	5	234	684	347339	**64247**
Stress	1000	5	6	1500	**499**

BUGS — better say "differences"

- Third outcome: lax validation, warnings, etc
- Duplicate attribute drives C# and Python APIs mad
- Stress testing

— — —

- `FOR` cycle
- Running in parallel

Thanks for your attention!