Rascal: A DSL for SCAM

Jurgen Vinju
Tijs van der Storm
Paul Klint

Hall of fame
Bob Fuhrer
Emilie Balland
Arnold Lankamp
Bas Basten
The complexity of bridging an analysis tool to a transformation tool shadows the complexity of the analyses and transformations themselves [Vinju & Cordy Dagstuhl 2005]
Every SCAM tool requires both “analysis” and “transformation” features
Every SCAM tool requires both “analysis” and “transformation” features
Every SCAM tool requires both “analysis” and “transformation” features
Every SCAM tool requires both “analysis” and “transformation” features
Every SCAM tool requires both “analysis” and “transformation” features.

- Slicing
- Refactoring
- Reverse Engineering
- Compilation
Every SCAM tool requires both “analysis” and “transformation” features.
Every SCAM tool requires both “analysis” and “transformation” features.
Most language parametric SCAM tools are geared towards either “analysis” or “transformation”
Most language parametric SCAM tools are geared towards either “analysis” or “transformation”
Most language parametric SCAM tools are geared towards either “analysis” or “transformation”.

StrategoXT

TXL
Most language parametric SCAM tools are geared towards either "analysis" or "transformation".
Most language parametric SCAM tools are geared towards either “analysis” or “transformation”
Most language parametric SCAM tools are geared towards either “analysis” or “transformation”
Most language parametric SCAM tools are geared towards either “analysis” or “transformation”

- Crocopat
- GROK
- RScript
- ASF+SDF
- StrategoXT
- TXL
Many great SCAM tools are implemented in a GPL
Many great SCAM tools are implemented in a GPL

A GPL allows fine-grained integration between analysis and transformation
Many great SCAM tools are implemented in a GPL

A GPL allows fine-grained integration between analysis and transformation

Tuesday, September 29, 2009
A DSL for SCAM?

- That covers SCAM
- That scales down and scales up
- That is easier
The SCAM domain

- Transformation
- Source Code
- Extraction
- Generation
- Analysis
- Models
- Visualization
- Formalization
- Pictures
- Conversion
The SCAM domain

- Uses common...
  - Data-structures
  - Algorithms
  - Nothing new! But still...
  - Synthesize into a DSL
- Conceptually
- Syntactically
- Semantically

Source Code

Models

Pictures
The SCAM domain

- Uses common...
- Data-structures
- Algorithms

Conceptually
Syntactically
Semantically

Pattern Matching

Source Code
Models
Pictures

Transformation
Extraction
Analysis
Visualization

Generation
Formalization
Conversion

Pattern Matching
Syntactically
Semantically

Source Code
Models
Pictures

Tuesday, September 29, 2009
Rascal has to scale down to make simple tasks short and easy to experiment with
Rascal has to scale down to make simple tasks short and easy to experiment with

Plain old REPL, no static type errors
Rascal has to scale down to make simple tasks short and easy to experiment with.

Plain old REPL, no static type errors

First regexps, then CFG’s
Rascal has to scale down to make simple tasks short and easy to experiment with.

Complete & domain specific, expression language: visit, pattern matching, relational calculus.

Plain old REPL, no static type errors

First regexps, then CFG's
Rascal has to scale down to make simple tasks short and easy to experiment with.

Plain old REPL, no static type errors

First regexps, then CFG's

URI literals: files, projects, SVN, ...

Complete & domain specific, expression language: visit, pattern matching, relational calculus
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability.
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability.
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability

**Modules are statically checked**

**Immutable data**
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability.

- Modules are statically checked
- Immutable data
- Co-variant sub-types
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability.

- Modules are statically checked
- Immutable data
- Co-variant sub-types
- h.o. parametric polymorphism, rank-1 and 2
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability.

- Modules are statically checked
- Immutable data
- Co-variant sub-types
- h.o. parametric polymorphism, rank-1 and 2
- Efficient and fully typed (de)serialization
Rascal has to scale up to complex analysis and transformation algorithms that employ reuse of (library) functionality and allow design for maintainability.

- Modules are statically checked
- Immutable data
- Co-variant sub-types
- Efficient and fully typed (de)serialization
- Function-local type inference
- h.o. parametric polymorphism, rank-1 and 2
Rascal has to scale up to complex analysis and transformation algorithms that employ **reuse** of (library) functionality and allow **design** for maintainability.

- Modules are statically checked
- Immutable data
- Co-variant sub-types
- h.o. parametric polymorphism, rank-1 and 2
- Efficient and fully typed (de)serialization
- Function-local type inference
- lexically scoped backtracking
Rascal has to scale up to complex analysis and transformation algorithms that employ reuse of (library) functionality and allow design for maintainability lexically scoped backtracking.

Modules are statically checked.

Immutable data.

Co-variant sub-types.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.

Efficient and fully typed (de)serialization.

h.o. parametric polymorphism, rank-1 and 2.

IDE support.

Concrete syntax.

Function-local type inference.

(h.o.) parametric polymorphism, rank-1 and 2.
We have implemented Infer generic type arguments [Fuhrer et al. ECOOP2005] on Generic Featherweight Java [Igarashi et al. TOPLAS2002]

The size of the Rascal code is equal to the size of the formal definitions in the papers
Rascal

Analysis and manipulation are happily married. We should think of them as one.
Rascal

Analysis and manipulation are happily married. We should think of them as one.

http://www.meta-environment.org