Introduction to Hacking the ABS Compiler & Tools

Radu Muschevici

Software Engineering Group, Department of Computer Science, TU Darmstadt (Germany)
Fakultas Ilmu Komputer, Universitas Indonesia

TECHNISCHE
UNIVERSITÄT
DARMSTADT

FAKULTAS
ILMU
KOMPUTER

FASILKOM RSE Lab Training, 11 January 2016

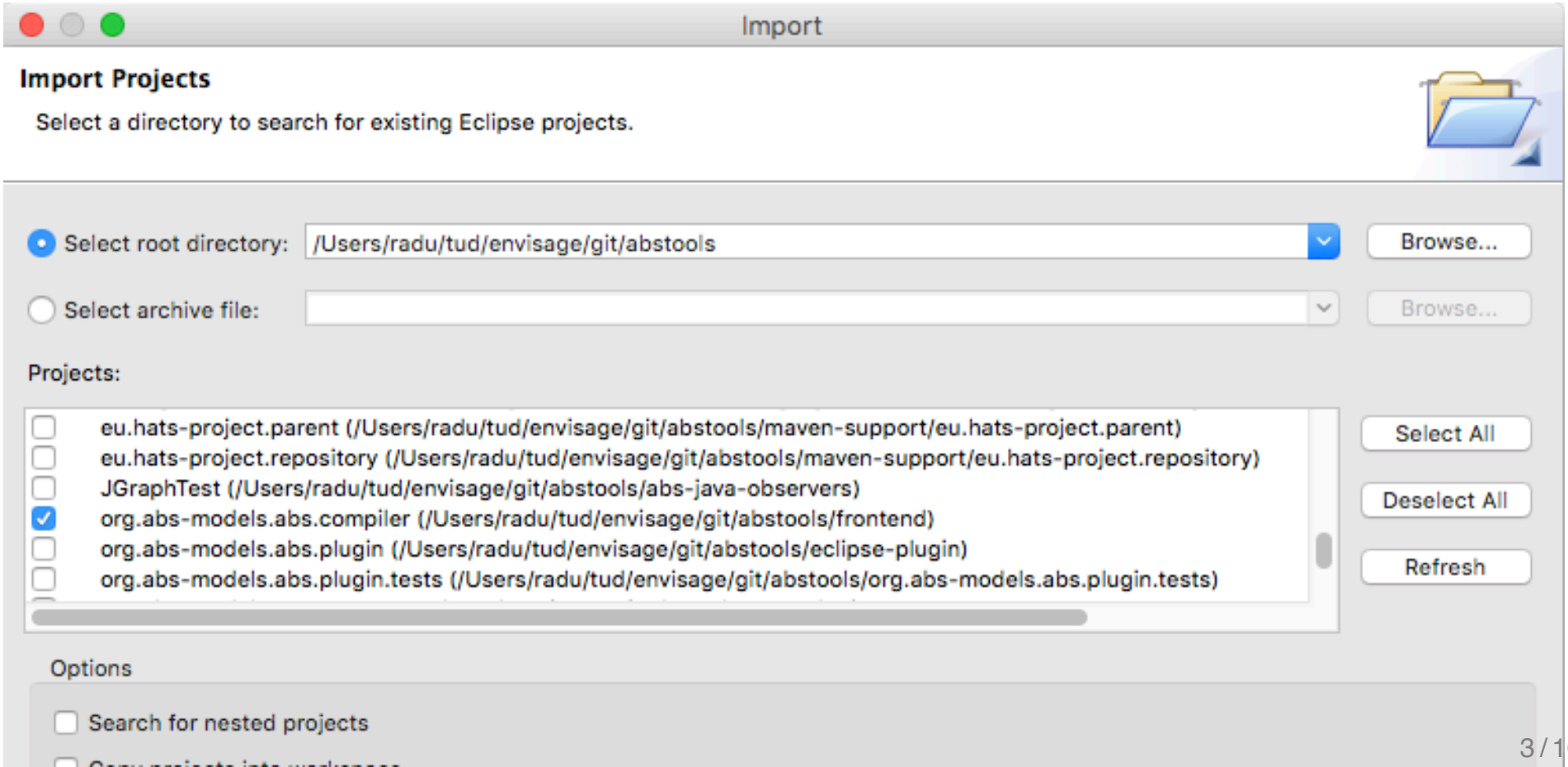
Resources

- Source code on Github
  `https://github.com/abstools/abstools`
- Clone to local folder:
  `git clone https://github.com/abstools/abstools.git`

- ABS language documentation `http://docs.abs-models.org/`
- General ABS home page: `http://abs-models.org/`

Importing ABS project in Eclipse

- Install Java 7 SDK (Java 8 probably works too)
- Install Eclipse (Luna or later version recommended)
- Import ABS compiler Java project (`org.abs-models.abs.compiler`) into workspace:
  File > Import > General > Existing Projects Into Workspace



Configure Eclipse

Apply ABS tools **code formatting conventions**

- Import `.settings/absformatter.xml` file:
  Project > Properties > Java Code Style > Formatter > Import...
- See also: `CODING-CONVENTIONS.txt`

Set up default editors for JastAdd files (.ast, .jadd, .jrag)

- Associate JastAdd files with the Eclipse editors:
  - `.ast` files with TextEditor
  - `.jadd` and `.jrag` files with JavaEditor
- Detailed instructions here:
  `http://jastadd.org/old/manual/running-jastadd-under-eclipse#editors`

Compiler

Folder structure of `org.abs-models.abs.compiler` project

- All source code: `src/`
- JUnit tests: `tests/`
- Generated code: `gen/` (once the compiler has been built)

- ANT builder configuration: `build.xml`
- Compiler front-end: `src/abs/frontend`
- Compiler back-end: `src/abs/backend`

Compiler

Parser

- Implemented using ANTLR (`http://www.antlr.org/`)
- Location: `src/abs/frontend/antlr`
- File `ABS.g4` contains the concrete syntax of ABS

Abstract Syntax Tree

- Implemented using JastAdd (`http://jastadd.org/web/`)
- Location: `src/abs/frontend/ast`
- Files `ABS.ast` and `mTVL.ast` contain abstract syntax of ABS

JastAdd

- For every non-terminal defined in the abstract grammar (`.ast` files), JastAdd generates a corresponding Java class in `gen/abs/frontend/ast`, which represents a type of Abstract Syntax Tree (AST) node. The AST of an ABS program is built from these AST nodes.
- The AST classes can be extended by defining methods and attributes inside **aspect** files (`.jadd/.jrag` extension).
- To understand JastAdd and the concept of Reference Attribute Grammars (RAGs), consult the JastAdd documentation at `http://jastadd.org/web/documentation/concept-overview.php` and follow the JastAdd tutorial: `http://jastadd.org/web/documentation/tutorial.php`

Building & Using the Compiler

Building the compiler

- In Eclipse, select the project `org.abs-models.abs.compiler`
- Project > Build Project
- After some time, “BUILD SUCCESSFUL” should appear in the Console view

Further documentation

- See `README` file (“How to use the ABS Frontend tools”)

Further Reading About ABS

- Hähnle, R. The abstract behavioral specification language: A tutorial introduction. In Formal Methods for Components and Objects, E. Giachino, R. Hähnle, F. Boer, and M. Bonsangue, Eds., vol. 7866 of LNCS. Springer, 2013, pp. 1–37.
- Clarke, D., Diakov, N., Hähnle, R., Johnsen, E., Schaefer, I., Schäfer, J., Schlatte, R., and Wong, P. Modeling spatial and temporal variability with the HATS abstract behavioral modeling language. In Formal Methods for Eternal Networked Software Systems, M. Bernardo and V. Issarny, Eds., vol. 6659 of LNCS. Springer, 2011, pp. 417–457.
- Wong, P. Y., Albert, E., Muschevici, R., Proença, J., Schäfer, J., and Schlatte, R. The ABS tool suite: modelling, executing and analysing distributed adaptable object-oriented systems. Journal on Software Tools for Technology Transfer 14 (2012), 567–588.

Further Reading About ABS (2)

- Johnsen, E. B., Hähnle, R., Schäfer, J., Schlatte, R., and Steffen, M. ABS: A core language for abstract behavioral specification. In International Symposium on Formal Methods for Components and Objects (2011), B. Aichernig, F. S. de Boer, and M. M. Bonsangue, Eds., vol. 6957 of LNCS, Springer, pp. 142–164.
- Clarke, D., Muschevici, R., Proença, J., Schaefer, I., and Schlatte, R. Variability modelling in the ABS language. In International Symposium on Formal Methods for Components and Objects (2011), B. Aichernig, F. S. de Boer, and M. M. Bonsangue, Eds., vol. 6957 of LNCS, Springer, pp. 204– 224.
- Hähnle, R., Helvensteijn, M., Johnsen, E., Lienhardt, M., Sangiorgi, D., Schaefer, I., and Wong, P. Hats abstract behavioral specification: The architectural view. In International Symposium on Formal Methods for Components and Objects, B. Beckert, F. Damiani, F. de Boer, and M. Bonsangue, Eds., vol. 7542 of LNCS. Springer, 2013, pp. 109–132.