A Theory of Tagged Objects (Artifact)

Joseph Lee[1], Jonathan Aldrich[1], Troy Shaw[2], Alex Potanin[*2], and Benjamin Chung[†1]

1 **Carnegie Mellon University**
Pittsburgh, PA, USA
josephle@andrew.cmu.edu, aldrich@cs.cmu.edu, bwchung@andrew.cmu.edu

2 **Victoria University of Wellington**
New Zealand
troyshw@gmail.com, alex@ecs.vuw.ac.nz

Abstract

A compiler and interpreter for Wyvern programming language written in Java and hosted on `http://github.com/wyvernlang/wyvern` and some sample programs (.wyv) including the main example from the paper in `borderedwindow.wyv`. We also include an extract of all the unit tests of which a large number may be designed to fail – therefore they are best run using JUnit which can be done by checking out the source tree from the GitHub project link above.

1998 ACM Subject Classification D.3.3 Language Constructs and Features

Keywords and phrases objects, classes, tags, nominal and structural types

Digital Object Identifier 10.4230/DARTS.1.1.3

Related Article Joseph Lee, Jonathan Aldrich, Troy Shaw, Alex Potanin and Benjamin Chung, "A Theory of Tagged Objects", in Proceedings of the 29th European Conference on Object-Oriented Programming (ECOOP 2015), LIPIcs, Vol. 37, pp. 174–197, 2015.
`http://dx.doi.org/10.4230/LIPIcs.ECOOP.2015.174`

Related Conference 29th European Conference on Object-Oriented Programming (ECOOP 2015), July 5–10, 2015, Prague, Czech Republic

1 Scope

The artifact is a prototype of the Wyvern language compiler that supports the tags as described in the paper.

2 Content

The artifact package includes:
- wyvern.jar which contains the compiler,
- three examples (.wyv),
- readme.txt and a selection of JUnit tests.

What is included is a Wyvern compiler. It takes a program written in Wyvern Programming Language (e.g. borderedwindow.wyv) and then compiles and executes (interprets) the result printing the final value of the evaluation to the standard output.

We included 3 sample programs written in Wyvern with the implementation inside wyvern.jar. Here is an example of running it on Linux:

[*] Core artifact developer.
[†] Core artifact developer.

```
cuba: [WyvernECOOP2015Artifact] % java -version
java version ''1.8.0_25''
Java(TM) SE Runtime Environment (build 1.8.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode)
cuba: [WyvernECOOP2015Artifact] % java -jar wyvern.jar borderedwindow.wyv
''big''
cuba: [WyvernECOOP2015Artifact] % java -jar wyvern.jar json.wyv
15
cuba: [WyvernECOOP2015Artifact] % java -jar wyvern.jar testmethods.wyv
64
```

The reader can try to modify the programs or write their own following either the description in the paper or using the samples included in the "tests" folder or by looking at the Wyvern language web site with some specifications and papers or GitHub repository. The main example in the paper is `borderedwindow.wyv` file that shows the core of the paper's contribution (dynamic tags) working – compiling and executing.

If a reader wishes to rebuild the program, say in Eclipse, here is what one needs to do. There are two parts to the compiler implementation: (1) is the parser generator Copper that converts the grammar in Wyvern.x into Wyvern.java that is then used inside the Wyvern implementation (2). The ant script in the tools folder is to run Copper to generate the Wyvern.java from Wyvern.x.

The instructions are as follows:

1. Clone GitHub repository.
2. Run ant inside wyvern/tools that will generate Wyvern.java from Wyvern.x.
3. Open Eclipse (or IntelliJ or NetBeans or similar) and use "import existing project into the workspace" and point it to "wyvern/tools" folder.
4. We do not include .classpath as it is platform specific but we include all libraries in wyvern/tools/lib/ folder: `CopperCompiler.jar`, `asm-debug-all-5.0.1.jar`, `hamcrest-core-1.3.jar`, `javatuples-1.2.jar`.
5. After setting the Java Build Path in Eclipse to use Java 8, one needs to (1) add library which is JUnit 4 and (2) add external jars which would be: CopperCompiler.jar, asm-debug-all-5.0.1.jar, hamcrest-core-1.3.jar, javatuples-1.2.jar from wyvern/tools/lib.
6. The above will allow Eclipse to build the project with no errors. At that point one can execute the tests (which are work in progress) by right clicjing on the entire source tree and selecting "Run As. . . " → "JUnit tests" or more specifically by selecting the tests in wyvern/tools/tests if one prefers . . .

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The latest version of our code is available on GitHub at the following URL: `http://github.com/wyvernlang/wyvern` in particular under the TaggedTypes branch but also in the master branch of the Wyvern compiler.

4 Tested platforms

The artifact is known to work on Windows, Linux, and Mac OS X running Oracle Java 8.

5 License

GPL v2

6 MD5 sum of the artifact

d8ab0c1d7e5e0e679459cb705a0e10e0

7 Size of the artifact

2.1 MB

Acknowledgements. The authors wish to thank Benjamin Chung – the primary maintainer of the Wyvern Compiler, as well as Troy Shaw – the implementor of the initial version of the tags as described in this paper.