2007 State of the Universe Address

Werner Dietl
ETH Zurich, Switzerland
Werner.Dietl@inf.ethz.ch
http://www.sct.inf.ethz.ch/

Peter Müller
Microsoft Research, USA
mueller@microsoft.com

Abstract

This position paper summarizes recent developments related to the Universe type system and suggests directions for future work.

1. Universe Type System

The Universe type system is an ownership type system that enforces the owner-as-modifier discipline. In this section, we summarize recent developments and suggest future work to improve the expressiveness and formal foundation.

1.1 Expressiveness

The Gang-of-Four design patterns are common design idioms for object-oriented programs. In [16], we compare how Ownership Types, Ownership Domains, and Universe Types handle these patterns. Based on this experience, we extended the Universe type system to support generics and ownership transfer.

Recent Developments. Generic Universe Types [4] extend Universe Types to generic types. Like Universe Types, Generic Universe Types enforce the owner-as-modifier discipline which does not restrict aliasing, but requires modifications of an object to be initiated by its owner.

Universe Types with Transfer [15] is an extension of Universe Types that supports ownership transfer. UTT combines ownership type checking with a modular static analysis to control references to transferable objects. UTT is very flexible because it permits temporary aliases, even across certain method calls. Nevertheless, it guarantees statically that a cluster of objects is externally-unique when it is transferred and, thus, that ownership transfer is type safe. UTT provides the same encapsulation as Universe Types and requires only negligible annotation overhead.

Future Work. Generic Universe Types reduce the number of necessary ownership casts in a program. Currently, we investigate Path-dependent Universe Types [18] to express additional relationships between objects and thereby further reduce the number of ownership casts.

Universe Types provide a very limited support for static fields and methods. The main problem is that global data enables a form of re-entrant method calls that is otherwise prevented by the type system. This form of re-entrancy causes problems for the verification of object invariants. We will formally integrate the Universe Type System with the Boogie methodology for the verification of object invariants [11], which can handle arbitrary forms of re-entrancy.

For the verification of object invariants, one has to control aliasing between fields of one object that are declared in different classes [13, 11]. We are currently extending Universe Types to enforce an ownership structure where each object has a context for each superclass of its dynamic type. This will allow us to enforce that the contexts for different superclasses are disjoint.

Another line of work to support program verification is to build an effects system on top of Universe Types. This effects system will be similar to Clarke and Drossopoulou's work [2], but has to handle `any` references, which makes read effects more complex. We plan to use the effects system to check side effects of methods and to support reasoning about pure methods.

1.2 Formal Foundation

Recent Developments. We proved in the theorem prover Isabelle that the Universe type system is sound and that the owner-as-modifier discipline is enforced [9]. We also wrote a detailed type safety proof on paper for Generic Universe Types [3]. A similar, but less comprehensive proof is available for Universe Types with Transfer [14].

Future Work. We aim at extending our Isabelle formalization of Universe Types to Generic Universe Types.

2. Type Inference

One strength of the Universe type system is the low annotation overhead; it is further reduced by appropriate defaults. However, the resulting ownership structure is flat and annotating existing software remains a considerable effort. We work on inferring deep ownership structures using static and dynamic techniques.

2.1 Static Universe Type Inference

Recent Developments. We generate constraints from the Java AST of a program and use a pseudo-boolean solver to find possible ownership modifiers [8, 17, 7]. The weighting function of the solver is used to find a deep ownership structure.

We allow partially annotated programs as input. The programmer can simply annotate some fields and method signatures and can then use the static inference to propagate the ownership modifiers and to achieve complete code coverage.

Future Work. Currently, the inference tools work with Universe Types. We will investigate how to incorporate the inference of Generic Universe Types and Universe Types with Transfer.

Another form of static inference is to infer the types of local variables from field and parameter types. We are pursuing this line of work in the context of Universe Type with Transfer. Here, inference for local variables is particularly interesting because locals can change their type from program point to program point as objects get transferred. We are currently implementing our ideas in the JML compiler.

2.2 Runtime Universe Type Inference

Recent Developments. We analyze the execution of standard Java programs and infer Universe modifiers from the execution traces [5, 12, 1, 7]. The advantage of runtime Universe type inference is that the deepest possible ownership structure is deduced. Good

code coverage is needed for runtime inference. We allow the user to combine multiple program traces as input to the inference in order to achieve good coverage.

Static and runtime Universe type inference can be combined to get a deep ownership structure and ensure sound results. The result of the runtime inference is used as weight for the static inference. The static inference achieves perfect code coverage and can still change ownership modifiers if that improves the overall structure.

Future Work. A major topic for future work is to apply our inference to real applications. This will provide valuable insights in the expressiveness of Universe Types, the power of our inference tools, and especially to ownership structures that can be found in large systems. We expect especially the last result to be important for the ownership community as a whole.

3. Tool Support

3.1 Compiler and Runtime Support

Recent Developments. The type checker for Universe Types is implemented in the JML tool suite [10] since 2004. The JML compiler also produces the code needed for the runtime check of ownership downcasts. It also stores the ownership modifiers in the bytecode, which allows to typecheck programs without having the Java source code. We will commit the extensions for Generic Universe Types soon. The type checker for Universe Types is also implemented in recent version of ESC/Java2.

To make the interaction with the command-line tools easier for programmers we developed a set of Eclipse [6] plug-ins. The JML checker and runtime assertion checking (RAC) compiler can be invoked from within Eclipse and we created comfortable configuration dialogs. Error messages are parsed and displayed in a separate window and code with RAC can be executed from Eclipse. We also provide code templates that make entering ownership modifiers easy. See Fig. 1 for a screen shot.

Future Work. We are finishing the implementation of the Universe Types with Transfer type checker and runtime support. This extension of the JML compiler also supports inference for local variables. We work on integrating Universe Types with Transfer and Generic Universe Types.

3.2 Inference Tools

Recent Developments. Executing the command-line inference tools requires some knowledge to configure the programs correctly. We provide Eclipse plug-ins that allow the configuration through dialogs and that make management of temporary results easy.

The results of static inference are displayed in a comfortable tree view, see left pane in Fig. 2. The user can change ownership modifiers directly in this pane and see what effects a modification has—without parsing the source code again.

For the runtime inference, we provide a visualization of the ownership structure, see right panels in Fig. 2. The programmer can step through the program execution and observe how the ownership structure is built up.

Both inference tools create their results in a special annotation XML format that describes what ownership modifiers need to be added to a program source. We provide a customized editor for this XML format and the annotations can be automatically inserted into the Java source code.

Future Work. We are working on optimizing the inference tools to handle large programs and will then evaluate the tracing overhead and inference time. The visualizer for the inference tools is an interesting playground for visualizing ownership structures.

References

[1] M. Bär. *Practical Runtime Universe Type Inference*. Master's thesis, Department of Computer Science, ETH Zurich, 2006.

[2] D. Clarke and S. Drossopoulou. Ownership, encapsulation and the disjointness of type and effect. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 292–310. ACM Press, 2002.

[3] W. Dietl, S. Drossopoulou, and P. Müller. Formalization of Generic Universe Types. Technical Report 532, ETH Zurich, 2006.

[4] W. Dietl, S. Drossopoulou, and P. Müller. Generic Universe Types. In E. Ernst, editor, *European Conference on Object-Oriented Programming (ECOOP)*, Lecture Notes in Computer Science. Springer-Verlag, 2007. To appear.

[5] W. Dietl and P. Müller. Runtime universe type inference. In *International Workshop on Aliasing, Confinement and Ownership in object-oriented programming (IWACO)*, 2007. To appear.

[6] The Eclipse Foundation. Eclipse — an open development platform. `http://www.eclipse.org/`.

[7] A. Fürer. *Combining Runtime and Static Universe Type Inference*. Master's thesis, Department of Computer Science, ETH Zurich, 2007.

[8] N. Kellenberger. *Static Universe Type Inference*. Master's thesis, Department of Computer Science, ETH Zurich, 2005.

[9] M. Klebermaß. *An Isabelle Formalization of the Universe Type System*. Master's thesis, Department of Computer Science, ETH Zurich, 2007.

[10] G. T. Leavens, E. Poll, C. Clifton, Y. Cheon, C. Ruby, D. Cok, P. Müller, and J. Kiniry. JML reference manual. Department of Computer Science, Iowa State University. Available from `www.jmlspecs.org`, 2006.

[11] K. R. M. Leino and P. Müller. Object invariants in dynamic contexts. In M. Odersky, editor, *European Conference on Object-Oriented Programming (ECOOP)*, volume 3086 of *Lecture Notes in Computer Science*, pages 491–516. Springer-Verlag, 2004.

[12] F. Lyner. *Runtime Universe Type Inference*. Master's thesis, Department of Computer Science, ETH Zurich, 2005.

[13] P. Müller, A. Poetzsch-Heffter, and G. T. Leavens. Modular invariants for layered object structures. *Science of Computer Programming*, 62:253–286, 2006.

[14] P. Müller and A. Rudich. Formalization of ownership transfer in Universe Types. Technical Report 556, ETH Zurich, 2007.

[15] P. Müller and A. Rudich. Ownership Transfer in Universe Types. In *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2007. To appear.

[16] S. Nägeli. *Ownership in Design Patterns*. Master's thesis, Department of Computer Science, ETH Zurich, 2006.

[17] M. Niklaus. *Static Universe Type Inference using a SAT-Solver*. Master's thesis, Department of Computer Science, ETH Zurich, 2006.

[18] D. Schregenberger. *Universe Type System for Scala*. Master's thesis, Department of Computer Science, ETH Zurich, 2007.

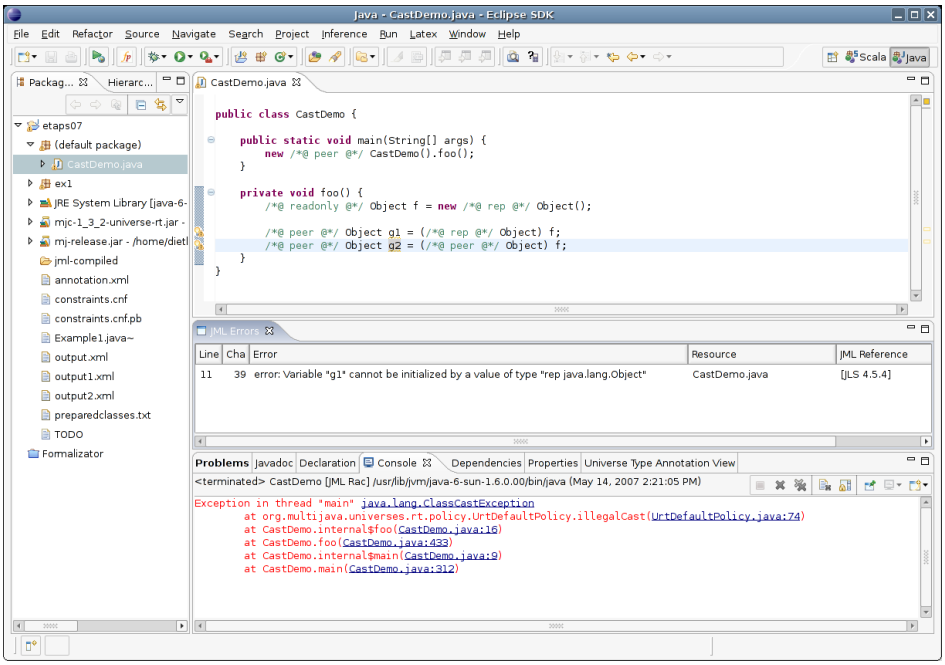

Figure 1. Eclipse Integration of JML tools: JML compiler error message in the middle and JML RAC runtime error at bottom.

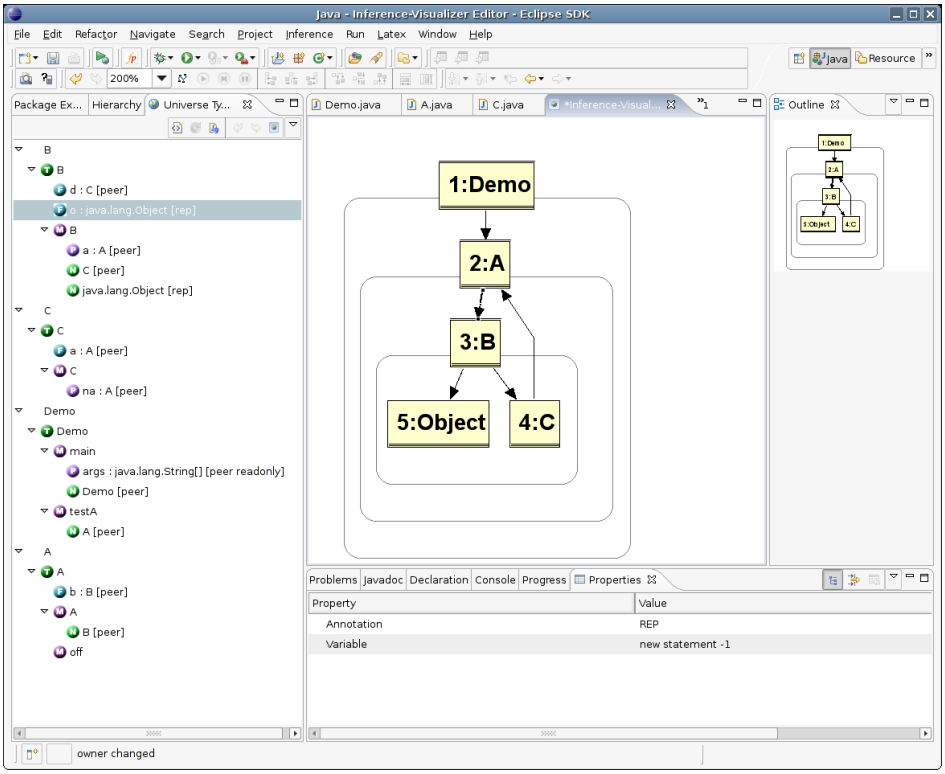

Figure 2. Inference mode. Static inference results on the left. Visualization of runtime inference in the center.