Ubiquitous Object Orientation to Foster the Advancement of Programming Languages

Darya Kurilova
Carnegie Mellon University
darya@cs.cmu.edu

There are myriads of improvements that a programming language can offer: from enhancements aimed at minimizing the amount of code programmers must write to various analyses that help programmers better understand and control program behavior. As the number and complexity of programming language features increases, it becomes harder to account for their interactions. To solve this problem, Wyvern—a pure, statically typed, object-oriented programming language—provides an easy-to-reason-about framework that supports addition of programming language features. Wyvern represents core programming language constructs as objects, which creates a level playing field for a diverse set of features and analyses, such as sandboxing, information flow analysis, and security analysis, that can be performed both statically and dynamically.

The idea of representing "everything" as an object was tried in other languages, such as Self and JavaScript. Wyvern differs from those languages in that it is statically typed and guarantees type safety. In some respects, Scala treats modules and packages as objects, but those objects are a special type of Scala objects, whereas objects representing modules in Wyvern are not differentiated and are generally uniform. Therefore Wyvern provides a different approach in the design space of representing programming language constructs as objects.

The first step towards a unified representation of programming language constructs in Wyvern was proving that classes are not an essential construct of a programming language and can be translated into objects [1]. We extend that work by translating modules into objects. To give an insight into how it is done, consider the following code snippet:

```
1  module SecretNum : sigSecretInt
2    var secret : Int = 42
3
4  module SecretRevealer : sigRevealer
5    import SecretNum as SN
6    def reveal(Unit) : Int = SN.secret
```

This is a simple example of how a programmer would write a module that imports another module in Wyvern. (Module signatures are omitted for conciseness.) Internally, however, this code is translated to:

```
1  val SecretNum : sigSecretInt
2    bind in
3      new(SecretNum =>
4        var secret : Int = 42)
5
6  val SecretRevealer : sigRevealer
7    bind
8      val SN = SecretNum
9    in
10     new(SecretRevealer =>
11       def reveal(Unit) : Int = SN.secret)
```

Here the two modules from the first code snippet are values, and the module import is done via a **bind** construct, which is similar to a **let**.

The translation of classes and modules into objects is straightforward, yet powerful. One benefit comes directly form the nature of objects: Since objects are first-class values, it is possible to dynamically configure a program at runtime by providing different classes and modules as parameters. In addition, the uniform representation of programming language constructs streamlines analyses across different types of constructs by avoiding the necessity to separately analyze objects and modules. Finally, having only one type of entity, i.e. objects, to consider makes reasoning about program properties easier and simplifies the implementation of various programming language features.

We are using this approach in an ongoing work on Wyvern, in which we leverage the uniformity of presentation to perform a capability-based security analysis and control module access, e.g. to restrict access to modules such as the foreign function interface (FFI) module.

At the workshop, we would like to present Wyvern's approach of the unified translation of programming constructs to objects and exemplify how the translation supports the development of new programming language features. Furthermore, we would like to discuss how the approach can simplify implementation of current programming language features and analyses as well as how it can foster the development of desired, but unexplored, features.

References

[1] L. Nistor, D. Kurilova, S. Balzer, B. Chung, A. Potanin, and J. Aldrich. Wyvern: A Simple, Typed, and Pure Object-Oriented Language. In *MASPEGHI*, 2013.