CPSC 490 Proposal

Mike Mester

1 February 2010

Introduction

Object-oriented languages play an indisputably important role in modern software development, and the years of collected wisdom and experience captured in object oriented design patterns contribute to the usefulness of object oriented programming generally. Given this importance, the lack of a suitable formalism for describing object oriented design is glaring. Descriptions of design patterns are still largely heuristic, informal, and imprecise. While there is substantial utility in this approach to object oriented design, a formal foundation could distill the essential elements of object oriented design from the distracting artifacts of informality. A rigorously mathematical system for describing object oriented design would therefore augment the considerable informal literature on the topic that already exists. In this project, we hope to examine a few existing proposals for such a formal system and propose a new system that hopes to better capture the fundamental aspects of object oriented designs. To that end, there are four roughly separate components to the project, each of which is discussed in more detail below:

1. Evaluation of existing formalizations of object oriented design
2. Proposal for a new formal system for object oriented design
3. Comparison of major object oriented programming languages (e.g. C++ and Java) using the system from (2)
4. Application of the system from (2) to some common design patterns

As used here, the term “object oriented design” is meant to evince the defining features of the object oriented approach to programming as distinct from the “traditional” approach. It is on these defining features that we will focus. Thus, the goal of this project is decidedly not to come up with a complete formal semantics of object oriented programming generally. Instead, we hope to apply some formalism to the discussion of object oriented design generally.

1 Evaluation of existing proposals

Although there is a reasonably large body of literature on the formal semantics of object oriented languages (see e.g. [1]), surprisingly little work exists on the formal specification of object oriented designs. Fortunately, some of the formal semantics work appears applicable to our project here, particularly the discussions of the object oriented type system (which underlies much of what distinguishes object oriented languages from their older counterparts). Nevertheless, the focus of formal semantics is at once too broad and too narrow for our present goals. The work is too broad because it attempts to formalize all the necessary minutiae of a modern object oriented language, much of which is largely irrelevant to high level design of programs. Similarly, the work is too narrow because it tends to concentrate at the level of single classes, thereby largely ignoring the relationships between them. Thus, while we will examine the formal semantics literature for guidance, this project is ultimately asking a different question.
Comparatively little work has been done on the formalization of object oriented design. Some early attempts have been made to formalize UML (e.g. [8, 5]), which is similar to the goal of this project. Indeed, UML itself is a system for describing object oriented design, though its semantics lack the mathematical rigor sought here. More general formal systems have also been proposed, ranging from algebraic structures (e.g. [6]) to various logics (e.g. [7, 4, 3]). We hope to examine these proposals with respect to their simplicity, descriptive power, and formal rigor. The multiplicity of these systems suggests that no proposal has been entirely satisfactory so far. This prior work provides important guidance, however.

2 Formal system for object oriented design

Building off the discussion in Section 1, we then hope to propose a new system for formally specifying object oriented designs. The inspiration for this proposal is the relational database model ([2]), which was novel in its combination of simplicity and power.

Our preliminary approach is (like Codd’s relational model) to represent objects as sets of primitives (likely fields and methods) and then to build the essential relations of object oriented design off of that structure. Presumably, the most fundamental of these relations will be the inheritance relation. This approach differs from many of the prior proposals for formalizing object oriented design, in particular [3], in that it looks at the structure of objects themselves rather than treating them as atomic primitives. In this sense, we hope to build something of a bridge between the formal semantics of object oriented languages (which focus extensively on type hierarchies and object structure) and the attempts to formalize design. The attempt to develop this system constitutes the heart of the proposed project.

3 Formal comparison of object oriented languages

With the help of a formal system like the one proposed above in Section 2, it can be useful to evaluate the sorts of designs that different object oriented languages facilitate. The syntax and structure of different languages can have consequences on object oriented design, and a formal way of describing object oriented design could help illuminate those consequences.

For instance, while C++ allows multiple inheritance, Java does not (at least not directly). A formal system for object oriented design might show whether multiple inheritance is simply a feature for convenience or whether Java’s single inheritance meaningfully restrains design options. A similar analysis could be made of C# and Java’s use of a universal base class, which C++ lacks.

4 Formal analysis of design patterns

One of the primary uses of a formal system for describing object oriented design is to inject rigor into the discussion of design patterns. Thus, a natural application of a system like the one described in Section 2 would be to formalize a few common design patterns. This exercise would serve at least two purposes.

First, applying a formal system to design patterns would facilitate a practical evaluation of that system. A formalism that is too unwieldy to conveniently apply to real designs would be of little use. Thus, a few examples of our proposed formalism would go along way towards revealing its actual value. Of course, such examples can also help to illustrate the system itself.

Second and perhaps more interestingly, formal descriptions of design patterns provide a framework for proving relationships among the patterns. When design patterns remain at the level of informal heuristics, it is difficult to establish any rigorous organization among them. Differences in the linguistic descriptions of different patterns can obscure more fundamental relationships. A formal system for describing object oriented design, however, might show that some patterns are more general while others are special cases. Since many design patterns seem intuitively similar to each other, it might even be possible to prove in a rigorous way that two patterns are equivalent. If such a proof were possible, it would be an example of a formal system revealing something important about the patterns that was otherwise hidden.
Conclusion

The project proposed here is admittedly very broad. Nevertheless, the core of the project lies in the evaluation of existing formal systems for describing object oriented design and the proposal of such a system. This is where most attention will be focused. This is certainly no trivial task, but even a negative result would be informative. Failures of formal systems reveal complexity in the structure of object oriented design that might not otherwise be apparent. Although the hope of this project is to propose a formalization of object oriented design, then, the project can be instructive even if that attempt is unsuccessful.

Deliverable

The deliverable for this project is a written report detailing the analysis described above.

References


