CHAPTER 9

CONCLUSIONS AND FURTHER RESEARCH

We have come a long way to examine the subject of orthogonality and its opposite: inorthogonality. This chapter summarizes all the previous eight chapters of those topics, with conclusions and suggestions for further research. Sec 9.1 presents the conclusions, and sec 9.2 offers further research.

9.1 Conclusions

Throughout this research, we have established well-defined four basic concepts and two supplementary concepts of programming. Together with the BRC (Basic Record Collection) categorization this research has shown the sufficiency of those concepts to explain the source-code in various programming-languages. Thus, this research has filled its first purpose.

This research sets the criteria for highly orthogonal system: the provision of twenty-four (24) facilities. The twenty-four facilities come from the cartesian product of definer (system-defined versus user-defined), basic concepts (value, operator, type, object) and BRC categories (basic, record, collection). The 24 facilities are used throughout several chapters, fulfilling the second purpose of this research.

This research proves that it is possible to create a formal model in which the encapsulation, inheritance, and polymorphism are orthogonal; by unbundling the operators from the record-type. Indeed, the unbundling of operators from the record-types are the first and foremost requirement to achieve orthogonal encapsulation, orthogonal inheritance, and orthogonal polymorphism. Thus, this research has proven the first hypothesis.

A highly orthogonal programming language based on the proposed theory reduces the amount of exceptions. NUSA programming-language is based on the highly orthogonal programming theory presented in this dissertation. We have shown the presence of seven items of inorthogonality in C# and Java, and proven
that NUSA has none of the inorthogonality items. NUSA is a general-purpose programming-language that implements orthogonal encapsulation, orthogonal inheritance, and orthogonal polymorphism. Thus, this research has proven the second hypothesis and fulfilled its third purpose.

9.2 Further research

The applications and proofs for the proposed theory in this dissertation are limited to imperative programming-languages. Further research can be conducted with proofs from declarative languages (such as SQL), even functional and logic programming languages – see Appendix A and B. If successful, the theory will be proven successful universally, for both imperative and declarative programming.

This research is not focused toward the optimization of algorithms. Further research can be carried out to focus on optimizing some algorithms, particularly the ones related to polytype-operators.

Finally, the research can also be directed toward the improvement of NUSA and its code-translators. NUSA is still at its infancy. Further research and development to make NUSA becoming a programming-tool in the university and industry will worth the effort.