Overloading

Elizabeth Labrada Deniz *[1]

[1] *Computer Science Department (DCC), University of Chile, Chile*

Abstract

1 Introduction

- Que es Overloading?
- Para que sirve, nombrando algunos lenguajes conocidos que lo tienen.
- Explicar que es considerado un tipo de polimorfismo.
- Explicar brevemente distintos tipos de polimorfismo y las diferencias con overloading [1].
- Explicar que la mayoria de los lenguajes que poseen polimorfismo resuelven el overloading en presencia de otras clases de polimorfismo, por ejemplo type classes y Java. Queremos iniciar con un lenguaje que solo contenga overlaoding, para estudiar esta caracteristica en si.[2, 3, 4, 5]
- Explicar brevemente como se resuelve overloading en la mayoria de los lenguajes, que es en tiempo de compilacion, o al menos se utiliza informacion en tiempo de compilacion, tal como FeatherWeight Java con multi-method. Nombrar Common Lisp como un lenguaje diferente.
- Decir que el lenguaje que queremos diseñar es para posteriormente estudiarlo con tipos graduales, y asi explicar que nos conllevo al diseo de la semantica del lenguaje.

*Funded by grant CONICYT, CONICYT-PCHA/Doctorado Nacional/2015-63140148

$$Polyphormism = \begin{cases} universal & \begin{cases} parametric \\ inclusion \end{cases} \\ ad\ hoc & \begin{cases} overloading \\ coercion \end{cases} \end{cases}$$

Figure 1: Varieties of polyphormism.

2 Background

- Substitucion explicita, explicar brevemnte que es y porque la vamos a utilizar.
- Ambiguedad
- Local Type Inference, si se utiliza en el sistema de tipo
- Otros

3 Calculus

- Explicar que agrega cada semantica con respecto a la anterior, por ejemplo deteccion de errores.
- Explicar relaciones entre las semanticas,

3.1 Semantic 1

3.2 Semantic 2

3.3 Semantic 3

3.4 Semantic 4

3.5 Type System

3.6 Proofs

4 Conclusions

- Conclusiones
- Trabajo Futuro

References

[1] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computing Surveys*, 17(4):471–522, 1986.

[2] T. Nipkow and C. Prehofer. Type checking type classes. In *Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '93, pages 409–418, New York, NY, USA, 1993. ACM.

[3] M. Odersky, P. Wadler, and M. Wehr. A second look at overloading. In *Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture*, FPCA '95, pages 135–146, New York, NY, USA, 1995. ACM.

[4] R. Ribeiro and C. Camarão. Ambiguity and context-dependent overloading. *Journal of the Brazilian Computer Society*, 19(3):313–324, 2013.

[5] P. Wadler and S. Blott. How to make ad-hoc polymorphism less ad hoc. In *Proceedings of the 16th ACM Symposium on Principles of Programming Languages (POPL 89)*, pages 60–76, Austin, TX, USA, Jan. 1989.