Modelling features of object-oriented languages in second order functional languages with subtypes

Giorgio Ghelli

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy
e-mail: ghelli@dipisa.di.unipi.it

ABSTRACT: Object oriented languages are an important tool to achieve software reusability in any kind of application and can increase dramatically software productivity in some special fields; they are also considered a logical step in the evolution of object oriented languages. But these languages lack a formal foundation, which is needed both to develop tools and as a basis for the future evolution of these languages; they lack also a strong type system, which would be essential to assure that level of reliability which is required to large evolving systems. Recently some researches have tried to insulate the basic mechanisms of object oriented languages and to embed them in strongly typed functional languages, giving to these mechanism a mathematical semantics and a set of strong type rules. This is a very promising approach which also allows a converging evolution of both the typed functional and the object oriented programming paradigms, making it possible a technology transfer in both directions. Most works in this field are very technical and deal just with one aspect of object oriented programming; many of them use a very similar framework. In this work we describe and exemplify that common framework and we survey some of the more recent and promising works on more specific features, using the framework introduced. We describe the results achieved and point out some problems which are still open, especially those arising from the interactions between different mechanisms.

1 This work was carried on with the partial support of E.E.C., Esprit Basic Research Action 3070 FIDE and of Italian C.N.R., P.F.I. “Sistemi informatici e calcolo parallelo”.
1 Introduction

The notion of “object oriented language” groups a bunch of different features found in rather different languages, notably Simula67 as the first one and SmallTalk as the most devoted to the approach [Goldberg 83]. A language is “object oriented” when it offers a set of features, described below, called “encapsulation” “inheritance” and “object identity”; these features help following the “object oriented programming methodology”, i.e. building a software system as a set of “communicating objects” grouped in “classes”.

The interest about object oriented programming methodologies and languages is growing. Among the reasons of this interest, we can list the following ones:

• The object oriented approach is well suited to develop some specific kinds of applications, especially applications driven by a graphical interface and simulation programs

• The encapsulation mechanisms of object oriented languages support a high degree of reusability of code, and the inheritance mechanisms still increase the ability of reusing code

• The SmallTalk programming environment proves that a good object-oriented programming environment allows rapid prototyping by combining the code reusability features of the language with a rich standard library in an interactive environment

• In the database field the object oriented approach is an ideal support for the modelization mechanism introduced by the semantic data models to overcome the limitations of the traditional and relational data models.

The problem of object oriented languages is that they lack a solid formal foundation. Another problem is that up to now object oriented languages lack a solid, static and strong type system, which would be badly needed to develop complex applications with a high grade of reliability. These two problems are currently faced by trying to insulate the basic components which compose the object-oriented paradigm and by embedding them in a functional language with a strong but flexible type system. This is an approach whose origins can be traced back to the researches about the Galileo language, described in [Albano Cardelli Orsini 85] and in [Cardelli 84]; the basis of most of the current researches is the type system described in [Cardelli Wegner 85], which is an extension of second order λ-calculus ([Girard 72, [Reynolds 74]) with subtypes.

In this paper we review some of the results of this research and point out some of the open problems. We focus mainly on a gentle introduction to the common features of most current researches; will we hint at some if the more important open problems.

The paper is structured as follows. In Section 2 we describe the features of an object oriented language. In Section 3 we describe the functional language Fun, which is a functional language with subtyping and explicit polymorphism well suited to host object oriented operators. In Section 4 we detail existential types (ET’s), which are a construct used to model abstract data types, which has a basic rôle in modeling some features of object-oriented languages. In Section 5 we show how record types can be used to model message passing. In Section 6 we discuss state encapsulation and inheritance, which is the subject of many current researches. Section 7 contains the conclusions.