Incremental Inheritance Model for an OODBMS

Mohammed Benattou and Lotfi Lakhal

LIMOS
Université Blaise Pascal Clermont-Ferrand II
Complexe Scientifique des Cézeaux
63177 Aubière cedex
France.

Abstract. The semantics of inheritance presented in this paper is based on the incremental modification mechanism that is formalized by the generator associated with class, modification function, and the building inheritance operators. The model is based upon an intuitive explanation of the proper use and purpose of inheritance, and is essentially dedicated to dynamic (i.e. run-time) inheritance of properties, for OODBMS. A simple typing of inheritance is derived by Cook's constraint, that defines the conditions of the validity of generator derivation. We show by using the subtyping relation defined in the O2 OODBMS, the correctness of the proposed model (i.e. the Cook's constraint is respected). And therefore, the integration of our incremental model in O2 is valid.

1 Introduction

Inheritance allows objects of different structures to share properties (methods or attributes) related to their common parts. Several authors have tried to define a semantic of inheritance. Cardelli [10] identifies inheritance with subtype relation; McAllister and Zabith [22] suggest the boolean classes system for knowledge representation; Minsky and Rozenshten [23] use the concept of regulatory laws for message passing to characterize inheritance. In OODBMS [4], the semantics of inheritance is based on subtyping (≤) : the system checks whether the inheritance definition is legal, that is if there is no subtyping violation. Although the subtyping relationship provides static type-checking, it does not allow a formalization of the inheritance mechanism corresponding to the form of inheritance used in object-oriented programming [12]. Inheritance is not subtyping [13] but an intricate mechanism, featuring dynamic binding together with some clever naming conventions (pseudo-variables self and super). Inheritance is a mechanism for incremental modifications [12, 18].

This paper develops a formal model for the management of the inheritance mechanism. This model is built on the incremental modification principle, based upon an intuitive explanation of the proper use and purpose of inheritance, and is essentially dedicated to dynamic inheritance of properties, for OODBMS. The incremental inheritance mechanism allows to construct a new class by modifying one of the existing superclasses. We formalize this mechanism by the specification of the modification function and the definition of the building operators. In
order to formalize multiple inheritance, we propose an extension of the incre-
mental model of single inheritance to multiple inheritance with dynamic conflicts
resolution at run-time.

Type theory defines the condition of the validity of generator derivation. A simple
typing of inheritance is derived by Cook’s constraint [13]. Unfortunately, this
constraint is not respected, when we use the subtyping rule for functional type
defined by L. Cardelli [10] (contravariance [11] \( \alpha \rightarrow \beta \leq \alpha' \rightarrow \beta' \) iff \( \alpha' \leq \alpha \)
and \( \beta \leq \beta' \)). This problem is caused by the interaction between recursion and
specialization” \( \alpha \rightarrow \beta \leq \alpha' \rightarrow \beta' \) iff \( \alpha \leq \alpha' \) and \( \beta \leq \beta' \)) as the subtyping
rule for method types in the O2 DBMS. This approach gives a less restrictive
type system, but is safe [21], more natural, flexible and expressive [11]. We
show that the proposed model respects the Cook’s constraint, when we use the
subtyping relation (\( \leq \)) defined in the O2 DBMS. And therefore, the integration
of denotational method for dynamic message evaluation using the incremental
inheritance mechanism [6, 7] in O2 is valid.

This paper is organized as follows: Section 2 describes the basic concepts of
records, self-reference, and generator used in the fixed-point theory to analyze
self-reference definitions. Section 3 introduces the formal model for single inher-
itance based on W. R. Cook’s works. Section 4 presents the incremental model
for multiple inheritance with conflict resolution. Section 5 shows by formal vali-
dation the correctness of our model, using the O2 type system.

2 Self-reference and generator

Self-reference occurs when a structure is defined in term of itself. This tech-
nique is frequently used in programming languages: recursive procedures, func-
tions and datatypes. In the object-oriented paradigm, self-reference, which is
syntactically standarized by the pseudo-variable self, that enables to refer to a
property (method or attribute) of the object where the message is sent. The
pseudo-variable self represents recursion or self-reference [18]. The fixed-point
semantics provides the mathematical setting for the analysis of recursive pro-
grams [12, 18]. At run-time, we consider an object as a record [13, 18] where
fields (attributes or methods) contain values. The record-based model proposed
by L. Cardelli [10] constitutes the basic model for our framework. We extend it
to object databases, in particular we focus on the values taken by the fields of a
record at run time.

Definition 1. A record \( S \) is a finite mapping from a set of labels to a set
of values. A record with labels \( x_1, x_2, ..., x_n \) and values \( v_1, v_2, ..., v_n \) is written
\( [x_1 \mapsto v_1, x_2 \mapsto v_2, ..., x_n \mapsto v_n] \). All labels which are not in the list are
mapped to \( \perp \). The domain of a record \( S \) is defined by \( \text{Dom}(S) = \{ x \setminus S(x) \neq \perp \} \) (\( S(x) \) represents the value of the property \( x \) in \( S \), \( S(x) = \perp \) where \( x \) is
assumed to be mapped to an undefined value in \( S \)).