Snapshot Generation in a Constructive Object-Oriented Modeling Language

Mauro Ferrari¹, Camillo Fiorentini², Alberto Momigliano², and Mario Ornaghi²

¹ Dipartimento di Informatica e Comunicazione, Università degli Studi dell’Insubria, Italy
mauro.ferrari@uninsubria.it

² Dipartimento di Scienze dell’Informazione, Università degli Studi di Milano, Italy
{fiorenti,momiglia,ornaghi}@dsi.unimi.it

Abstract. CooML is an object-oriented modeling language where specifications are theories in a constructive logic designed to handle incomplete information. In this logic we view snapshots as a formal counterpart of object populations, which are associated with specifications via the constructive interpretation of logical connectives. In this paper, we introduce the “snapshot semantics” of CooML and we describe a snapshot generation (SG) algorithm, which can be applied to validate specifications in the spirit of OCL-like constraints over UML models. Differently from the latter and from the standard BHK semantics, the logic allows us to exploit a notion of partial validation that is appropriate to encodings characterised by incomplete information. SG is akin to model generation in answer set programming. We show that the algorithm is sound and complete so that its successful termination implies consistency of the system.

1 Introduction

We are developing the constructive object-oriented modeling language CooML [19] (http://cooml.dsi.unimi.it), a specification language for OO systems. Similarly to UML/OCL [23], CooML provides a framework for the design of system specifications in the early stages of the development process. The language allows the user to distinguish between internally defined elements and the problem domain, which may involve loosely or incompletely defined components. This encourages the selection of the appropriate level of abstraction w.r.t. specifications.

CooML follows the spirit of lightweight formal methods [10]: it does not focus on full formalization, nor on whole system correctness, but emphasizes partiality in analysis and specification. In particular, in the context of OO modeling, both the validation of a specification and the check of its consistency can be achieved via the notion of snapshot, i.e. a population of objects in a given system state that satisfies the specification. Previous work has used snapshots for validation of UML/OCL models [8] and specifications in JML [4].

The novelty of CooML’s approach resides in its semantics, which is related to the constructive explanation of logical connectives (a.k.a. the BHK interpretation [22]). Specifically, the truth of a CooML proposition in a given interpretation is explained by a mathematical object that we call an information term. For the time being, the latter can be visualized as a sort of proof term inhabiting a type/formula. The underlying logic is
characterized by how classical and constructive information co-exists, the main “entry” point being the different way in which an atomic formula \( A \) is given evidence (for more details we refer the kind reader to the original formulation of the logic in \cite{15}). If we call the pair \( I : P \) a piece of information, where \( P \) is a formula and \( I \) is its information term, then \( I : P \) may be true or false in a classical interpretation \( w \), called a world. Thus, we have a notion of a model of a piece of information based on classical logic. In particular, we use \( T \{ F \} \) to indicate the truth of \( F \); in fact, \( T \) does not contain evidence for \( F \), but it yields a trivial piece of information true in all the models of \( F \). This introduces a novel and flexible way to handle incomplete information, a notorious difficulty in information systems such as relational databases.

Crucially, the constructive side of the logic allows the identification of snapshots with information terms, thus providing a formal counterpart to the intuitive notion of object populations. We argue that CooML’s proof-theoretic snapshot generation may be advantageous in comparison to a model-theoretic one, especially in cases where not all the information required to define a model is even present. The possibility of treating information in this less committed way means that we can select only the relevant information; this may have a cascade of benefits in terms of conciseness of the representation.

The contributions of this paper are twofold. First, we extend the semantics developed in purely logical terms in \cite{15} to object oriented modeling languages. We regard an OO system specification as a CooML theory \( T \), the system snapshots as the pieces of information \( I : T \), and the related information content as a suitable set of formulae. We show that the latter can be seen as the minimum information needed to give evidence to snapshots and we relate that to snapshot consistency. Secondly, we describe (and implement) a snapshot generation algorithm (SGA), taking as inputs: (i) a CooML theory \( T \), axiomatizing a set of classes in a problem domain PD; (ii) the user’s generation requirements \( G \) – they serve an analogous purpose of domain predicates in the grounding phase of ASP’s \cite{17}. As snapshots should be consistent with respect to PD and \( G \), we prove that consistency checking is sound and that snapshot generation is complete, i.e. if a consistent snapshot satisfying the generation requirements exists, it will be generated. This is loosely connected to adequacy results in the theory of CLP’s \cite{7}.

\section{CooML Specifications}

In this section we informally present the language via an example (adapted from \cite{3}), while we defer the formal exposition to Section 2.1. The problem domain concerns a small coach company. Each coach has a specified number of seats and can be used for regular or private trips. In a regular trip, each passenger has its own ticket and seat number. In a private trip, the whole coach is rented and there may be a guide. The corresponding CooML specification is contained in the package coachCompany (Fig. 1). To explain our example we need to introduce CooML types system. We distinguish among data types (in our example, Integer and Boolean), PD types (Person), and object types (Coach, Trip, Passenger). They all inherit from the top type Value the identity relation and the string representation. Data types are “statically” defined, i.e., their values do not depend on the current state. CooML assumes the