Science of Computer Programming 96 (2014) 527–551

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Specification of temporal properties with OCL ☆, ☆☆

Bilal Kanso, Safouan Taha *

SUPELEC Systems Sciences (E3S) – Computer Science Department, 3 rue Joliot-Curie, F-91192 Gif-sur-Yvette cedex, France

ARTICLE INFO

Article history:
Received 15 March 2013
Received in revised form 12 September 2013
Accepted 26 February 2014
Available online 12 March 2014

Keywords:
OCL
Temporal patterns
Eclipse/MDT
Model-driven engineering
Formal methods

ABSTRACT

The Object Constraint Language (OCL) is widely used to express static constraints on models and object-oriented systems. However, the notion of dynamic constraints, controlling the system behavior over time, has not been natively supported. Such dynamic constraints are necessary to handle temporal and real-time properties of systems.
In this paper, we first add a temporal layer to the OCL language, based syntactically on Dwyer et al.'s specification patterns. We enrich it with formal scenario-based semantics and integrate it into the current Eclipse OCL plug-in. Second, we translate, with a compositional approach, OCL temporal properties into finite-state automata and we connect our framework to automatic test generators. This way, we create a bridge linking model driven engineering and usual formal methods.

1. Introduction

Formal testing and verification techniques are increasingly used in the industrial areas to enforce hardware and software dependability. These techniques rely on the specification of the system properties that are commonly formalized using temporal logics such as the Linear Temporal Logic (LTL) [4]. However, a major problem, frequently encountered in practice, consists in difficulties in most cases when expressing the system properties in such formalisms. This is of course due to the mathematical nature of temporal logics that require big technical skills to be manipulated. Even more, the correctness of the formal specification towards the intended property, that we want to express, happens to be impossible to ensure. Today, what is needed is a property specification framework that is user-friendly and accessible to system engineers to avoid the potential errors introduced during property formalization.

☆ This work is a revised and extended version of [1]. It consists in adding the following contributions:

- The formal definition of a compositional approach to translate Temporal OCL expressions into automata (see Section 9).
- The development of a tool, available as an Eclipse plug-in, to translate Temporal OCL expressions into automata (see Section 7).
- The support for specification of real-time properties to enlarge the scope of Temporal OCL (see Section 8).
- The definition of the formal semantics of the variants that we added to Dwyer et al.'s patterns (see Section 6).
- Finally, adding systematically more examples and explanations to illustrate all the notions introduced in this paper.

☆☆ This work was funded by the French ANR TASCCC project (ANR-09-SEGI-014) [2] and the Motorola Solutions Foundation [3].

* Corresponding author.
E-mail addresses: bilal.kanso@supelec.fr (B. Kanso), safouan.taha@supelec.fr (S. Taha).
URL: http://wwwdi.supelec.fr/taha/ (S. Taha).

http://dx.doi.org/10.1016/j.scico.2014.02.029
0167-6423/

In this work, we are interested in formal methods applied to object-oriented systems. The Object Constraint Language (OCL) is widely used to express precise and unambiguous constraints in the context of object-oriented systems, even if it is mostly used within UML models [5]. OCL is formally equivalent to a first-order predicate logic over objects,[1] but it offers a user-friendly notation similar to programming languages. The OCL constraints may be invariants that rule each single system state, or preconditions and postconditions that control a one-step transition from a pre-state to a post-state upon the call of some operation. Thus, it is not possible to express temporal and real-time constraints that involve different states of the model that are not necessarily connected by a one-step transition upon an operation call. This is essentially due to the weakness of constructs for time and events in OCL. Adding a temporal layer to the OCL language forms a primordial step towards supporting the specification of temporal and real-time properties of object-oriented systems.

Our contribution to this paper is threefold. We first propose a temporal extension of OCL by relying on Dwyer et al.'s patterns [6]. A temporal constraint consists of a pattern combined with a scope. A pattern specifies the behavior that one wants to exhibit/avoid, while a scope defines the piece of execution trace to which a given pattern applies. This allows us to write temporal OCL constraints without any technical knowledge of formalisms such as LTL or CTL logics. Furthermore, our OCL temporal extension comes with new generic event constructs and a supplementary quantification freedom. We will also show how our extension provides a total support for specifying common real-time patterns.

Second, we enrich our OCL temporal extension with formal scenario-based semantics and we propose an automata generation approach to transform the temporal properties written in our OCL extension, into finite-state automata (FSA). The main idea consists in associating to each pattern an FSA and to each scope a special FSA that has a particular composition state cs. The resulting automaton corresponding to the property is given by substituting the composite state cs of the scope automaton by the pattern automaton. This compositional approach is both extensible and homogeneous. Indeed, besides the natural semantics of the n patterns and m scopes, previous efforts provide formal semantics by mapping each of the $n \times m$ pattern/scope combinations into many formalisms such as temporal logics, quantified regular expressions (QRE), etc. Being compositional our approach only requires considering $n + m$ automata and scales better when extending the language with new patterns and new scopes. Furthermore, our approach is faithful when the existing translational semantics arises a *homogeneity* limitation. Dwyer et al. build their language on a clear separation between pattern and scope notions. From a user point of view, a pattern (resp. scope) has a unique natural semantics regardless of the scope (resp. pattern) with which it is combined. Translational semantics flattens this key separation and translates each pattern and each scope many times into different expressions corresponding to the different possible combinations. Hence, the same pattern (resp. scope) may have different and potentially non-homogeneous interpretations according to the m scope (resp. n pattern) with which it is combined.

Third, we integrated our framework into the official Eclipse/MDT OCL plug-in in order to be in conformance with the standard OCL [5]. Then, we connected our framework to test generation techniques where test purposes are given as automata [7,8]. Test purposes are commonly used to focus on testing particular aspects of models, avoiding other irrelevant ones. As regards practical applications, our framework is used in the context of the TASCCC project [2]. It provides a means to express security properties (specified by Gemalto[2]) on the UML/OCL model of the GlobalPlatform, the latest generation smart card operating system.[3] In this work, the test purposes are obtained from the OCL temporal properties and then transformed into test scenarios. These are then animated using the CertifyIt tool, provided by the Smartesting company to generate test cases.[4]

This paper is organized as follows. Section 2 presents the OCL language while Section 3 discusses its limitations on the temporal aspects. Section 4 recalls related works and existing OCL temporal extensions. Section 5 describes our proposal for extending OCL to support time and events. Section 6 provides the formal scenario-based semantics of our language. Section 7 describes the implementation of the proposed extension in the Eclipse/MDT OCL plug-in. Section 8 describes a proposal to support real-time properties using our OCL temporal extension. Section 9 presents our compositional approach to translate OCL temporal properties into automata. Section 10 presents the use of our proposal as a test purpose framework within the TASCCC project. Finally, Section 11 concludes and presents the future works.

2. Object Constraint Language (OCL)

OCL is a formal assertion language, easy to use, with precise and unambiguous semantics [5]. It allows the annotation of any object-oriented model, even if it is most used within UML diagrams. OCL is very rich, it includes fairly complete support for:

[1] See the OCL OMG document, Annex A [5].
[2] *Gemalto* is a public company providing digital security solutions. www.gemalto.com.
[3] www.globalplatform.org.
[4] www.smartesting.com.