Formal Semantics of Dynamic Constraints and Derivation Rules in ORM

Herman Balsters, University of Groningen, Groningen, The Netherlands
Terry Halpin, INTI International University, Nilai, Malaysia

ABSTRACT

This paper provides formal semantics for an extension of the Object-Role Modeling approach to support declaration of dynamic rules. Dynamic rules differ from static rules by involving state transitions, rather than simply individual states. This paper restricts application of dynamic rules to single-step transactions, with a previous state (input to the transaction) and a new state (the result of that transaction). These dynamic rules specify an elementary transaction type by indicating which kinds of objects or facts (being added, deleted or updated) are involved. Dynamic rules may declare pre-conditions relevant to the transaction, and a post-condition stating the properties of the new state. In this paper the authors provide such dynamic rules with a formal semantics based on sorted, first-order predicate logic. The key idea underlying their solution is the formalization of dynamic constraints in terms of static constraints on the database transaction history.

KEYWORDS
Data Modeling, Dynamic Constraints, Formal Semantics, Object-Role Modeling, Temporal Rules

1. INTRODUCTION

Object-Role Modeling (ORM) is a fact-oriented approach for modeling, transforming, and querying information in terms of the underlying facts of interest, where facts and rules are verbalized in language understandable by nontechnical users of the business domain. In contrast to attribute-based modeling approaches such as Entity Relationship (ER) modeling (Chen, 1976) and class diagramming in the Unified Modeling Language (UML) (Object Management Group, 2013), ORM models are attribute-free, treating all facts as relationships (unary, binary, ternary etc.). For example, instead of the attributes Person.isSmoker and Person.birthdate, ORM uses the fact types “Person smokes” and “Person was born on Date”.

Other fact-oriented approaches closely related to ORM include CogNIAM (www.pna-group.com), Fully-Communication Oriented Information Modeling (FCO-IM) (Bakema et al., 2000), and the Semantics of Business Vocabulary and Business Rules (SBVR) specification (Object Management Group, 2013b). Introductions to ORM may be found in (Halpin, 2006b; Halpin, 2010; Halpin, 2011). For detailed coverage of ORM, see (Halpin & Morgan, 2008; Halpin, 2015; Halpin, 2016). The version of ORM discussed in this paper is ORM 2 (Halpin, 2005), as supported by the NORMA tool (Curland & Halpin, 2010).

Business rules include constraints and derivation rules. Static rules (also known as state rules) apply to each state of the information system that models the business domain, and may be checked by examining each state individually (e.g. each moon orbits at most one planet). Dynamic rules reference at least two states, which may be either successive (e.g. no employee may be demoted in...
rank—this kind of dynamic rule is known as a transition constraint) or separated by some period (e.g. invoices ought to be paid within 30 days of being issued). ORM is richer than ER or UML in its ability to depict static constraints graphically, but unlike UML it currently has no graphic notation (e.g. activity diagrams) to specify business processes. To capture dynamic rules, UML supplements its graphical notations with formulae in the Object Constraint Language (OCL) (Object Management Group, 2014; Warmer & Kleppe, 2003), but the OCL syntax is often too difficult for validation by nontechnical users.

Since the 1980s, many extensions to fact-oriented approaches have been proposed to model temporal aspects and processes, for example (Bruza & van der Weide, 1989; Falkenberg & van der Weide, 1988; Halpin, 2008; Halpin & Morgan, 2008; Halpin & Wagner, 2003). For a brief review of such work see (Balsters et al. 2006; Balsters & Halpin, 2008), which introduced to ORM a purely declarative means to formulate dynamic constraints on single-step transactions, with an old state (the input of the transaction) and a new state (resulting from that transaction). Such dynamic rules specify an elementary transaction type indicating which kind of object or fact is being added, deleted or updated, and (optionally) pre-conditions and post-conditions. These dynamic rules are formulated in a syntax designed to be easily validated by nontechnical domain experts. In this paper, we focus on providing a formal semantics for the basic rule patterns for dynamic rules found in (Balsters et al., 2006). Such formalization supports further understanding of dynamic rules, and also provides a step to further tool support. Substantial research has been carried out to provide logical formalizations of dynamic rules, typically using temporal logics, e.g. (Girle, 2003, ch. 8; Pesci et al., 2010), or Event-Condition-Action (ECA) formalisms, e.g. (de Brock, 2000; (Lipeck, 1990); Chomicki, 1992; Paton & Díaz, 1999; Snodgrass, 1994).

We also mention The Situation Calculus (Reiter, 2001) as a formalism (based on first-order logic) for representing and reasoning about dynamic domains. Situation calculus is not state based, but instead aims at formalizing the concept of action, as well as orderings between actions. As such, Situation Calculus (and its implementation language GOLOG) differs significantly from our approach which models dynamic constraints in terms of database semantics.

Our approach differs from previous work by treating a dynamic rule as a special kind of static rule on the transaction history, and by explicitly modeling time to capture temporal aspects in modeling (Halpin, 2008). We define the semantics of a dynamic rule by making explicit the log of all previous transaction instances pertaining to that specific rule. Snapshot data are maintained in the user database, whereas a full history is maintained in the log database. The user database allows retraction of asserted facts, and hence is non-monotonic. Although the log database allows deletion of derived facts, it never deletes asserted facts, so is effectively monotonic increasing. This logging semantics allows us to formalize dynamic constraints and error corrections in a static way, using first-order predicate logic.

While this paper builds on our earlier work (Balsters et al., 2006; Balsters & Halpin, 2008), it makes several new contributions. Unlike our earlier papers, we fully formalize dynamic constraints in ORM, and effectively treat the user database as a view on the log database, which provides a complete ledger of all the transactions that have ever been made on the user database. In a sense, the log database therefore resembles a blockchain of transactions on the user database (https://en.wikipedia.org/wiki/Block_chain_(database)). Furthermore, the current paper introduces fact types to distinguish between a transaction being entered, becoming active, and being marked as erroneous. Among other things, this allows us to formally deal with facts that have erroneously been entered into a user database; and with transactions to later correct such errors (wrongly entered facts). Furthermore, the current paper describes how to delete facts from the user database (while the corresponding log database remains monotonic). The current paper also discusses implementation issues, and offers an implementation example of logging semantics in terms of a deductive database language.

ORM models may be rigorously formalized, as first described by Halpin (1989) using unsorted predicate logic, and later refined and extended by Halpin (2012). For readability, we now use sorted
Related Content

Are Socio-Economic, Health Infrastructure, and Demographic Factors Associated with Infant Mortality in Russia?

Ranking and Risk Factor Scheme for Malicious applications detection and Classifications
www.igi-global.com/article/ranking-and-risk-factor-scheme-for-malicious-applications-detection-and-classifications/218172?camid=4v1a
Implementing Internal Software Process Assessment: An Experience at a Mid-Size IT Company
www.igi-global.com/chapter/implementing-internal-software-process-assessment/77759?camid=4v1a

Validation of IS Security Policies Featuring Authorisation Constraints
www.igi-global.com/article/validation-of-is-security-policies-featuring-authorisation-constraints/123606?camid=4v1a