Abstract Interpretation of Hibernate Query Language

Angshuman Jana, Raju Halder and Agostino Cortesi
{ajana.pcs, halder}@iitp.ac.in
Indian Institute of Technology Patna
cortesi@unive.it
Università Ca' Foscari Venezia, Italy

Objectives

• We extend the Abstract Interpretation framework to Hibernate Query Language(HQL) as a way to support semantics-based sound approximation techniques.

• We define concrete and abstract semantics of Hibernate Query Language.

• It serves as a formal verification method for behavioral properties of persistent objects, rather than transient objects, which have permanent representations in the underlying databases.

Hibernate Query Language

• Hibernate Query language (HQL) [1] is an Object-Relational Mapping (ORM) language which remedies the paradigm mismatch between object-oriented languages and relational database models.

• It provides a unified platform for programmers to develop object-oriented application where high-level variables interact with underlying database attributes.

• Therefore application programmers can develop objection-oriented applications without knowing much detail about the underlying database.

• It is treated as an object-oriented variant of SQL.

• Various methods in "Session" are used to propagate object's states from memory to the database (or vice versa) and to synchronize both states when a change is made to persistent objects.

Abstract Interpretation

• Abstract Interpretation [4] is a semantics-based static analysis framework.

• It provides a sound approximation of program semantics focusing on a particular property.

• The intuition is to lift the concrete semantics to an abstract domain, by replacing concrete values by suitable properties of interest and simulating the operations in the abstract domain w.r.t. their concrete counterparts, in order to ensure the soundness.

State-of-art and Motivation

• F. Logozzo [2] introduced an Abstract Interpretation-based framework of Object-Oriented Programming (OOP) languages, aiming at verifying whether the programs respect the specifications correctly.

• It can be used for optimization of the code at class-level.

• The existing work on abstract interpretation of query languages [3] did not consider an access to the database operations through a high-level object oriented language.

• Our objective is fill up the gap in between these two theories, by extending the Abstract Interpretation theory to the case of HQL.

• As an application, the proposed framework can formally and automatically verify enterprise-policies in relational/non-relational abstract domains.

Abstract Syntax of HQL

• Like OOP, the abstract semantic of a program p in HQL is defined as $p = \langle\, c_{main}, \mathrm{L} \,\rangle$ where c_{main} is the main class, L are the other classes present in p.

• Similarly, a class c is defined as a triplet $c = \langle\, \mathrm{init}, \mathrm{F}, \mathrm{M} \,\rangle$ where init is the constructor, F is the set of fields, and M is the set of member methods in c.

Set of Classes

$c \in \mathtt{Class}$
$c ::= \langle \mathtt{init}, \mathtt{F}, \mathtt{M} \rangle$
where `init` is the constructor, $\mathtt{F} \subseteq \mathtt{Var}$ is the set of fields, and `M` is the set of methods.

`Session` methods

$m_{ses} \in \mathtt{M}_{ses}$
$m_{ses} ::= \langle \mathtt{C}, \phi, \mathtt{OP} \rangle$
where $\mathtt{C} \subseteq \mathtt{Class}$ and ϕ represents 'WHERE' clause

$\mathtt{OP} ::= \mathtt{SEL}\big(f(e\vec{x}p'), r(\vec{h}(\vec{x})), \phi, g(e\vec{x}p)\big)$
$\mid \mathtt{UPD}(\vec{v}, e\vec{x}p)$
$\mid \mathtt{SAVE}(\mathtt{obj})$
$\mid \mathtt{DEL}()$
where ϕ represents 'HAVING' clause and `obj` denotes an instance of a class.

HQL Programs

$p \in \mathbb{P}$
$p ::= \langle c_{main}, \mathtt{L} \rangle$
where $c_{main} \in \mathtt{Class}$ is the main class and $\mathtt{L} \subset \mathtt{Class}$

Concrete Semantics of HQL

• We define the concrete semantics of HQL by specifying how the methods are executed on (e, s, ρ_d) where $e \in$ Env is an environment, $s \in$ Store is a store, and $\rho_d \in \mathfrak{D}$ is a database environment, resulting into new state (e', s', ρ_d).

• The semantic definitions are expressed in terms of the semantics of database statements SELECT, INSERT, UPDATE, DELETE [3].

Abstract Semantics

• The concrete semantics can be lifted to an abstract semantics by simply making correspondence of concrete objects (variables values, object instances, stores, states, traces, etc.) into abstract ones representing partial information on them.

• The abstract version of session methods are:

$\mathtt{OP}^\sharp ::= \mathtt{SEL}^\sharp\big(f^\sharp(e\vec{x}p'^\sharp), r^\sharp(\vec{h}^\sharp(\vec{x}^\sharp)), \phi^\sharp, g^\sharp(e\vec{x}p^\sharp)\big)$
$\mid \mathtt{UPD}^\sharp(\vec{v}^\sharp, e\vec{x}p^\sharp)$
$\mid \mathtt{SAVE}^\sharp(\mathtt{obj}^\sharp)$
$\mid \mathtt{DEL}^\sharp()$

• The abstract semantics of $m^\sharp_{ses}$ is defined in terms of the abstract semantic of $\mathtt{INSERT}^\sharp$, $\mathtt{UPDATE}^\sharp$, $\mathtt{DELETE}^\sharp$, $\mathtt{SELECT}^\sharp$.

Future Plan

• Extension to the language-based information-flow security analysis.

• Abstract slicing with respect to the properties of persistent objects.

• To build a static analyzer tool for HQL.

References

[1] Bauer, C., King, G.: Hibernate in Action. Manning Publications Co. (2004)

[2] Logozzo, F.: Class invariants as abstract interpretation of trace semantics. Computer Languages, Systems & Structures 35, 100 - 142 (2009)

[3] Halder, R., Cortesi, A.: Abstract interpretation of database query languages. Computer Languages, Systems & Structures 38, 123 - 157 (2012)

[4] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proc. of the POPL'77, pages 238 - 252, Los Angeles, CA, USA, 1977. ACM Press.