Proving code equivalence in database-driven applications and SPARQL queries

Milena Vujošević Janičić

Faculty of Mathematics, University of Belgrade, Serbia
milena @ matf.bg.ac.rs

Key words:
software verification, code equivalence, embedded SQL, SPARQL, SMT solving, code refactoring

Software verification techniques for proving different kinds of code equivalence [19] are essential in the context of code refactoring and optimizations. Code refactoring and optimizations are daily programming practices that are, on the one hand, necessary to maintain software quality, but on the other increase the risk of introducing errors into the code [9, 17]. Therefore, reliable automatic checks of the functional equivalence of the original and modified code are highly desirable.

An important refactoring/optimization context is within database-driven applications. This context includes the synergy of different programming languages and paradigms, namely imperative/object-oriented programming (for example, C/C++) and declarative programming (SQL). Modifying such code may contain simultaneous changes (changes that include both SQL and a host language code) and that together preserve the overall code equivalence (but if considered separately, the modified SQL query itself or the modified imperative code itself are not semantically equivalent to the originals). We propose a first-order logic modeling of SQL queries and link this modeling to C/C++ semantics implemented within the tool LAV [18, 16, 21, 15]. We implement an SQLAV framework that is publicly available and open-source [20, 11]. SQLAV generates equivalence conditions that are efficiently solved by SMT solvers. The framework confirms the equivalence of the modified code or points to potential problems and explains why equivalence cannot be proven [10, 9].

Another context that we consider are queries written in SPARQL [6]. SPARQL is the standard query language and protocol for Linked Open Data [4] and can be used to express queries across diverse data sources. Query equivalence can be reduced to a query containment problem, a problem of deciding if each result of one query is also a result of another query (for any given dataset) [5]. We consider the containment problem in both standard and subsumption forms. We reduce SPARQL query containment problem to the satisfiability problem in first-order logic and formally prove the soundness and completeness of the proposed approach [14]. We implement a tool SpeCS [13, 14] which covers a wide range of the language constructs, e.g. conjunctive queries, filter, union, optional, graph clauses, blank nodes, projections, subqueries, built-in functions, etc. It also supports reasoning under the RDF schema entailment regime [2]. As the query containment problem is reduced to the satisfiability problem in first-order logic, conditions generated by SpeCS can be solved by first-order logic provers (like Vampire [7]) or by SMT solvers (like Z3 [3]). SpeCS is publicly available and open-source [12] and its evaluation on standard benchmarks [8, 1] shows that it is fast, accurate and reliable [13, 14].

References

[1] Melisachew Chekol, Jérôme Euzenat, Pierre Genevès, and Nabil Layaïda. SPARQL Query Containment under Schema. *Journal on Data Semantics*, 7(3):133–154, April 2018.

[2] Melisachew Wudage Chekol, Jérôme Euzenat, Pierre Genevès, and Nabil Layaïda. SPARQL Query Containment under RDFS Entailment Regime. In *Proceedings of the 6th International Joint Conference on Automated Reasoning*, IJCAR'12, page 134–148. Springer, 2012.

[3] L. De Moura and N. Bjorner. Z3: An Efficient SMT Solver. In *TACAS*, pages 337–340, 2008.

[4] John P. McCrae. The Linked Open Data Cloud. Insight Centre for Data Analytics, retrieved February 20th, 2022.

[5] Reinhard Pichler and Sebastian Skritek. Containment and Equivalence of Well-Designed SPARQL. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '14, pages 39–50. ACM, 2014.

[6] Eric Prud'hommeaux and A Seaborne. SPARQL 1.1 Query Language, 2013. https://www.w3.org/TR/sparql11-query/.

[7] A. Riazanov and A. Voronkov. The Design and Implementation of VAMPIRE. *AI Communications*, 15(2-3):91–110, 2002.

[8] Muhammad Saleem, Claus Stadler, Qaiser Mehmood, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Generating SPARQL Query Containment Benchmarks Using the SQCFramework. In *Proceedings of the 17th International Semantic Web Conference (ISWC), 2018 - Posters & Demos*, volume 2180 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.

[9] Mirko Spasić and Milena Vujošević Janičić. Verification supported refactoring of embedded SQL. *Software Quality Journal*, pages 1–37, 2020.

[10] Mirko Spasić and Milena Vujošević Janičić. First steps towards proving functional equivalence of embedded SQL. In *Types for Proofs and Programs (TYPES)*, pages 78–79. Univ. Minho, 2018.

[11] Mirko Spasić and Milena Vujošević Janičić. GitHub repository: SQLC, 2020. https://github.com/mirkospasic/sqlc, retrieved May 15, 2022.

[12] Mirko Spasić and Milena Vujošević Janičić. SPECS Web page, 2022. http://argo.matf.bg.ac.rs/?content=specs, retrieved May 15, 2022.

[13] Mirko Spasić and Milena Vujošević Janičić. SpeCS — SPARQL Query Containment Solver. In *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, pages 31–35, 2020.

[14] Mirko Spasić and Milena Vujošević Janičić. Solving the SPARQL Query Containment Problem with SpeCS, 2022. Submitted to Journal of Web Semantics.

[15] Milena Vujošević Janičić. Concurrent Bug Finding Based on Bounded Model Checking. *International Journal of Software Engineering and Knowledge Engineering*, 30(05):669–694, 2020.

[16] M. Vujošević Janičić, M. Nikolić, D. Tošić, and V. Kuncak. Software verification and graph similarity for automated evaluation of students' assignments. *Information and Software Technology*, 55(6), 2013.

[17] Milena Vujošević Janičić. Maintenance and maintainability within agile software development. *Science of Maintenance Journal*, 1(1):9–19, 2021.

[18] Milena Vujošević Janičić and Viktor Kuncak. Development and Evaluation of LAV: An SMT-Based Error Finding Platform. In *Verified Software, Theories, Tools and Experiments (VSTTE)*, Lecture Notes in Computer Science (LNCS), pages 98–113. Springer, 2012.

[19] Milena Vujošević Janičić and Filip Marić. Regression verification for automated evaluation of students programs. *Computer Science and Information Systems*, 17(1):205–228, 2020.

[20] Milena Vujošević Janičić and Mirko Spasić. Tools LAV and SQLAV, 2020. http://argo.matf.bg.ac.rs/?content=lav, retrieved May 15, 2022.

[21] Branislava Živković and Milena Vujošević Janičić. Parallelization of Software Verification Tool LAV. In *Types for Proofs and Programs (TYPES)*, pages 103–104. Eotvos Loránd Univ., 2017.