Constraint-based testing

Testing is a common approach used to detect defects in software. Automation of test data selection is important for an efficient testing process.

Constraint-based testing (CBT) is an approach to automate the selection of code inputs covering a given set of execution paths in an unit of code to test. CBT tools execute symbolically each path in the code, producing an SMT instance constraining the inputs to trigger the execution of the path.

Database programs

Software typically contains units of code reading and writing into a large relational database, through complex SQL statements, able to violate the database integrity constraints, expressed in first-order logic.

A symbolic execution for SQL code is necessary to allow CBT tools to select test data (including database states) for such database programs.

Relational symbolic execution

We propose a relational symbolic execution algorithm for SQL into SMT:

1. SQL tables are relational variables
2. Relational variables are uninterpreted predicates over uninterpreted sorts
3. SQL code and integrity check are constraints over these predicates

The algorithm has been evaluated over small-scale apps written in a core Java/SQL.

Going full-scale: handling full SQL, dynamically-crafted SQL, integration with CBT tools...

References

Towards Testing of Full-Scale SQL Applications using Relational Symbolic Execution, CSTVA’14, ACM
Testing Database Programs using Relational Symbolic Execution, Working paper 2014, University of Namur
A Relational Symbolic Execution Algorithm for Constraint-Based Testing of Database Programs, SCAM’13, IEEE
Test Input Generation for Database Programs using Relational Constraints, DBTest’12, ACM