ERCIM "ALAIN BENSOUSSAN" FELLOWSHIP PROGRAMME

Scientific Report

First name / Family name	Michaël / Marcozzi
Nationality	Belgium
Name of the *Host Organisation*	INRIA (Saclay – Ile de France)
First Name / family name of the *Scientific Coordinator*	Claude / Marché
Period of the fellowship	01/11/2014 to 31/10/2015

I – SCIENTIFIC ACTIVITY DURING YOUR FELLOWSHIP

My scientific activity during the fellowship has followed three main research axes:

Axis 1. A relational symbolic execution for testing simple data-oriented code units
Symbolic execution [A] is a technique that can be used to enable the automatic generation of test inputs exercising a set of execution paths within a code unit to be tested. If the paths cover a sufficient part of the code under test, the test data offer a representative view of the actual behaviour of this code. This notably enables detecting errors and correcting faults.
Relational databases are ubiquitous in software, but symbolic execution of code units that manipulate them remains a non-trivial problem, particularly because of the complex structure of such databases and the complex behaviour of SQL statements. Finding errors in such code units is yet critical, as it can avoid corrupting important data.
In the previous years [B,C], I have defined a symbolic execution translating base SQL primitives directly into constraints in the SMT-Lib language [D] and I have integrated it with a more traditional symbolic execution of normal program code. During the fellowship, these previous results have been further extended and published in an international scientific journal [1].

Axis 2. AWhy3-based symbolic execution engine for SQL2
The relational symbolic execution mechanism that I have developed in the previous years (see axis 1) exhibits several limitations. Notably, it handles only a small subset of SQL and has only been developed in the context of software testing. The main purpose of my work during the fellowship has been to extend this mechanism to a representative subset of SQL and to adapt it to the context of software formal verification. With Pr. Véronique Benzaken [E] and Dr. Evelyne Contejean [F], we have been working to define a symbolic execution of a representative part of the SQL2 standard into constraints in the Why3 language [G]. The Why3 tool, developed in my host organisation, enables one to target many constraint solvers and theorem provers with a single constraint language. By using Why3 as the target language for our symbolic execution mechanism, we make it usable both in a software testing and software verification context.
The work has been following the forthcoming steps:

1. Selection of a representative subset of the SQL2 standard.
2. State of the art in the domain of query processing and relational algebras.

3. Formalisation of a compilation algorithm from our SQL2 subset into a suitable relational algebra.
4. Acquisition of knowledge in the use of the Why3 language.
5. Development of symbolic execution rules for the relational algebra operators into Why3.

We intend to publish soon a scientific article presenting our results.

Axis 3. Automatic software analysis for hyper-properties verification and testing
In automated software analysis, tested and verified properties are typically single-trace, i.e. determined by the program behaviour over a single execution. However, there exists important program properties to test and verify which are not single trace. These properties are said to be hyper-properties [H].
Automated testing and verification of such hyper-properties, based on symbolic execution, is a new and promising research direction, of particular interest for the Laboratoire de Sûreté du Logiciel (LSL) of the CEA-LIST [I], a research lab that frequently collaborates with my host organisation.
At the end of the fellowship, I have fruitfully investigated the opportunities to collaborate with the LSL on this particular research question, and I will work as a postdoc in this lab for the next two years.

II – PUBLICATION(S) DURING YOUR FELLOWSHIP

[1] **Relational Symbolic Execution of SQL Code for Unit Testing of Database Programs**
Marcozzi et al., Science of Computer Programming, 1 July 2015,
Volume 105 Pages 44-72 Elsevier, ISSN 0167-6423

III – ATTENDED SEMINARS, WORKHOPS, CONFERENCES

Journées Francophones des Langages Applicatifs (JFLA) 2015
Val d'Ajol, France
January 7, 2015 up to January 10, 2015

IV – RESEARCH EXCHANGE PROGRAMME (REP)

Universitat Politècnica de València, Valence, Spain
Professor Tanja Vos - June 1, 2015 up to June 11, 2015

Search-based testing [J] is an alternate approach to the constraint-based approach (symbolic execution) which I have applied to data-oriented programs in the previous years (see axis 1). As constraint-based testing of data-oriented code may face scalability issues in some circumstances, using a search-based (heuristic) approach could remove or at least alleviate this problem. Professor Vos [K] has a many years' first-class research experience in the European search-based testing network. During my stay in Professor Vos' lab, I have:

- Gained a better understanding of Professor Vos' work and network.
- Presented my work about constraint-based testing of data-oriented apps.
- Discussed about the future opportunities to develop a search-based testing of data-oriented apps.

REFERENCES

[A] **Symbolic Execution And Program Testing**
J.C. King, Commun. ACM 19 (7) (1976) 385–394.
[B] **Test input generation for database programs using relational constraints**
M. Marcozzi, W. Vanhoof, J.-L. Hainaut Proceedings of the Fifth International Workshop on Testing Database Systems, DBTest ’12, ACM, New York, NY, USA, 2012, pp. 6:1–6:6.
[C] **A relational symbolic execution algorithm for constraint-based testing of database programs**
M. Marcozzi, W. Vanhoof, J.-L. Hainaut, IEEE 13th International Working Conference on Source Code Analysis and Manipulation, SCAM, ACM, 2013, pp. 179–188.
[D] **The SMT-LIB standard: version 2.0**
C. Barrett, A. Stump, C. Tinelli, Proceedings of the 8th International Workshop on Satisfiability Modulo Theories, A. Gupta, D. Kroening (Eds.), Edinburgh, UK, 2010.
[E] **www.lri.fr/~benzaken/**
[F] **www.lri.fr/~contejea/**
[G] **Why3: Shepherd your herd of provers**
Bobot, F., Filliâtre, J. C., Marché, C., & Paskevich, A. Boogie 2011: First International Workshop on Intermediate Verification Languages (pp. 53-64).
[H] **Hypertesting: The Case for Automated Testing of Hyperproperties**
Johannes Kinder, HOTSPOT 2015
[I] **www-list.cea.fr**
[J] **Search-based software test data generation: A survey.**
Phil McMinn. Software Testing, Verification and Reliability, 14(2):105–156, June 2004**.**
[K] **www.tanjavos.com**