2018-00510 - Fault Localization and Explanation for Concurrent Programs
CONVECS - PHD Campaign - Campagne Doctorants Grenoble Rhône-Alpes

Contract type: Public service fixed-term contract
Level of qualifications required: Graduate degree or equivalent
Function: PhD Position

About the research centre or Inria department
Inria, the French national institute for research in computer science and control, is dedicated to fundamental and applied research in information and communication science and technology (ICST). Inria has a workforce of 3,800 people working throughout its eight research centers established in seven regions of France.

Grenoble is the capital city of the French Alpes. Combining the urban life-style of southern France with a unique mountain setting, it is ideally situated for outdoor activities. The Grenoble area is today an important centre of industry and science (second largest in France). Dedicated to an ambitious policy in the arts, the city is host to numerous cultural institutions. With 60,000 students (including 6,000 foreign students), Grenoble is the third largest student area in France.

Context
CONVECS research activities focus on the formal modeling and verification of asynchronous concurrent systems, which are instantiated in various domains (communication protocols, distributed algorithms, GALS, etc.). To this aim, CONVECS proposes new formal languages for specifying the behaviour and the properties of concurrent Systems, and devises efficient verification algorithms and tools running on sequential and massively parallel machines.

Web: http://convecs.inria.fr

Radu Mateescu is the CONVECS team leader.

The SPADES project-team aims at mastering the complexity and dependability of networked embedded computing systems by focusing on three key questions:
- How to build networked embedded systems as adaptive modular structures?
- How to program systems with resource and behavioral constraints on multicore architectures?
- How to program reliable and fault-tolerant embedded systems with different levels of criticality?

Web: https://team.inria.fr/spades/

Gregor Goessler is the SPADES team leader.

Assignment
Recent computing trends promote the development of hardware and software applications that are intrinsically parallel, distributed, and concurrent. Designing and developing such systems has always been a tedious and error-prone task, and the ever increasing system complexity is making matters even worse. Although we are still far from proposing techniques and tools avoiding the existence of bugs in a system under development, we know how to automatically chase and find bugs that would be very difficult, if not impossible, to detect manually.

Model checking [1] is an established technique for automatically verifying that a model, e.g., a Labelled Transition System (LTS), obtained from higher-level program languages such as process algebra satisfies a given temporal property, e.g., the absence of deadlocks. When the model violates the property, the model checker returns a counterexample, which is a sequence of actions leading to a state where the property is not satisfied. Understanding this counterexample for debugging the specification is a complicated task for several reasons: (i) the counterexample can contain hundreds (even thousands) of actions, (ii) the debugging task is mostly
achieved manually (satisfactory automatic debugging techniques do not yet exist), (iii) the counterexample does not explicitly highlight the source of the bug that is hidden in the model, and (iv) the counterexample describes only one occurrence of the bug and does not give a global view of the problem with all its occurrences.

**Main activities**

The objective of this PhD Thesis is to propose and develop new solutions for understanding and summarizing the origin of bugs detected by model checking techniques. To do so, we would like first to find adequate models for representing not only the semantics of the corresponding concurrent program (LTS) but also the structure of the program. This latter information is particularly useful once the source of the bug has been identified in the model and we want to show to the developer where it occurs in the corresponding program.

Second, we plan to provide analysis techniques combining model checking and data mining techniques in order to identify what parts of the model caused the bug. We will start from the notion of choices proposed in [2] where the model moves from a correct to an incorrect behaviour. These choices turned out to be very relevant from a debugging perspective because they identify portions of the code that may be the source of the bug. To favour usability of our approach, we will revisit graph mining and visualization to propose new visualization techniques by highlighting these choices in the models.

Finally, we will use the information computed in the aforementioned steps for building an abstracted and summarized explanation of the origin of the bug. To do so, we plan to reuse notions such as the concepts of sufficient and necessary behaviour, the distance to the bug or the probability of the bug occurrence from a specific position in the program.

Two complementary ideas will be study during this PhD Thesis. First, we will try to propose debugging techniques for counterexamples produced in the context of equivalence checking techniques where two programs are compared with respect to an equivalence notion, and counterexamples show all differences between these two programs. Second, we will see whether our techniques could go one step further by not only detecting the source of the bug but also by proposing a solution to automatically repair the bug and obtain a correct program.

**Complementary information:**

- **Bibliography**
  


- **Supervisors**
  
  Gwen Salaun is full professor at Université Grenoble Alpes and member of the CONVECS team.

  Gregor Goessler is Inria researcher and head of the SPADES team.

  Vincent Leroy is associate professor at Université Grenoble Alpes.

- **Keywords**

  Debugging, Model Checking, Temporal Logic, Counterexample, Bug Localization, Concurrent Programs, Behavioural Models, Data Visualization, Equivalence Checking, Code Repair.

- **Skills**

  - Knowledge of formal methods (concurrency theory) and verification

  - Knowledge of data mining and data visualization is a plus

  - Candidates who enjoy programming would be appreciated, as the work will include software development

  - Education: MSc/Master 2 research in Computer Science
Benefits package

- Subsidised catering service
- Partially-reimbursed public transport
- Social security
- Paid leave
- Flexible working hours
- Sports facilities

Remuneration


Monthly salary after taxes: around 1596,05€ for 1st and 2nd year. 1678,99€ for 3rd year. (medical insurance included).