of spurious counter-examples reported by the model checker. This research involves the development of a new technique and of a prototype implementation. As an assignment, one will have to answer the following questions:

**Main activities**

To solve the problem, one will have to answer the following questions:

1. Which metrics should one use in order to bound variable values? Clearly, for an integer variable, one can use the value of the variable. It is less obvious which metric one should use for sets of sets or for functions of TLA*.

2. How to automatically find reasonable bounds on the variables of individual variables, given a few bounds from the user? Given two different integer variables in a TLA* specification, the range of their values in an execution may significantly differ. Thus, using a uniform bound for all the variables may be inefficient.

3. Which SMT encoding would work best for this problem? In practice, performance of an SMT solver may dramatically improve when one carefully chooses an encoding of the problem in SMT. Thus, one has to experiment with various encodings of TLA*.

4. Is the approach able to detect bugs in practical TLA* specifications? We propose to evaluate applicability of the technique by checking safety of fault-tolerant distributed algorithms.

This research involves the development of a new technique and of a prototype implementation. As an extended research beyond the proposed problem, the student can investigate how to leverage the technique to parameterized model checking or to strengthen the invariant when a counter-example is found. The first extension would allow users to check invariants for sets of instances where bounds are not fixed for some variables of the specification. The second extension would reduce the number of spurious counter-examples reported by the model checker.


**Assignment**

In this project, we propose to develop a technique for invariant checking of TLA* specifications that would require the user to give an upper bound on the parameters and variables that constitute a system state, rather than precisely fixing parameter values. For instance, one could bound the values of the integer variables, cardinality of the sets, and cardinality of the function domains and codomains. Given such an upper bound, the technique will produce an SMT encoding of the invariant checking problem, which uses only decidable theories. If an invariant candidateInv violates one of the proof obligationsInit ⇒ Inv or Inv \[\land\] Next ⇒ Inv for some parameters within the provided bounds, the SMT solver should report satisfiability of the problem and give the user a counterexample to the invariance of Inv. Otherwise, the SMT solver should report unsatisfiability, which guarantees system safety within the provided bounds.

**Context**

TLA* is a general-purpose language that has found applications in formal specification of distributed systems. In the last years, the developers of the TLA* tools made significant progress towards automation of the proof system for TLA* [1]. Since writing proofs is an ingenius task, TLA* users debug their specifications with the TLC model checker [2] before investing significant efforts into writing formal and complete proofs. TLC enumerates reachable states and thus rapidly exhausts all available computer resources.

Recently, we have started to develop a symbolic model checker for TLA* [3, 4] that builds upon two successful approaches to verification: (1) bounded model checking (BMC), and (2) satisfiability modulo theories (SMT). In our current approach, similar to TLC, the user has to define the values of all parameters. Hence, although our model checker benefits from using symbolic techniques, verification is restricted to fixed parameter values, and one has to try several instances in order to get some confidence in the safety of the system.

**Assignment**

In this project, we propose to develop a technique for invariant checking of TLA* specifications that would require the user to give an upper bound on the parameters and variables that constitute a system state, rather than precisely fixing parameter values. For instance, one could bound the values of the integer variables, cardinality of the sets, and cardinality of the function domains and codomains. Given such an upper bound, the technique will produce an SMT encoding of the invariant checking problem, which uses only decidable theories. If an invariant candidateInv violates one of the proof obligationsInit ⇒ Inv or Inv \[\land\] Next ⇒ Inv for some parameters within the provided bounds, the SMT solver should report satisfiability of the problem and give the user a counterexample to the invariance of Inv. Otherwise, the SMT solver should report unsatisfiability, which guarantees system safety within the provided bounds.

**Level of qualifications required**

Graduate degree or equivalent

**Function**

PhD Position

**Context**

TLA* is a general-purpose language that has found applications in formal specification of distributed systems. In the last years, the developers of the TLA* tools made significant progress towards automation of the proof system for TLA* [1]. Since writing proofs is an ingenius task, TLA* users debug their specifications with the TLC model checker [2] before investing significant efforts into writing formal and complete proofs. TLC enumerates reachable states and thus rapidly exhausts all available computer resources.

Recently, we have started to develop a symbolic model checker for TLA* [3, 4] that builds upon two successful approaches to verification: (1) bounded model checking (BMC), and (2) satisfiability modulo theories (SMT). In our current approach, similar to TLC, the user has to define the values of all parameters. Hence, although our model checker benefits from using symbolic techniques, verification is restricted to fixed parameter values, and one has to try several instances in order to get some confidence in the safety of the system.

**Assignment**

In this project, we propose to develop a technique for invariant checking of TLA* specifications that would require the user to give an upper bound on the parameters and variables that constitute a system state, rather than precisely fixing parameter values. For instance, one could bound the values of the integer variables, cardinality of the sets, and cardinality of the function domains and codomains. Given such an upper bound, the technique will produce an SMT encoding of the invariant checking problem, which uses only decidable theories. If an invariant candidateInv violates one of the proof obligationsInit ⇒ Inv or Inv \[\land\] Next ⇒ Inv for some parameters within the provided bounds, the SMT solver should report satisfiability of the problem and give the user a counterexample to the invariance of Inv. Otherwise, the SMT solver should report unsatisfiability, which guarantees system safety within the provided bounds.

**Main activities**

To solve the problem, one will have to answer the following questions:

1. Which metrics should one use in order to bound variable values? Clearly, for an integer variable, one can use the value of the variable. It is less obvious which metric one should use for sets of sets or for functions of TLA*.

2. How to automatically find reasonable bounds on the values of individual variables, given a few bounds from the user? Given two different integer variables in a TLA* specification, the range of their values in an execution may significantly differ. Thus, using a uniform bound for all the variables may be inefficient.

3. Which SMT encoding would work best for this problem? In practice, performance of an SMT solver may dramatically improve when one carefully chooses an encoding of the problem in SMT. Thus, one has to experiment with various encodings of TLA*.

4. Is the approach able to detect bugs in practical TLA* specifications? We propose to evaluate applicability of the technique by checking safety of fault-tolerant distributed algorithms.

This research involves the development of a new technique and of a prototype implementation. As an extended research beyond the proposed problem, the student can investigate how to leverage the technique to parameterized model checking or to strengthen the invariant when a counter-example is found. The first extension would allow users to check invariants for sets of instances where bounds are not fixed for some variables of the specification. The second extension would reduce the number of spurious counter-examples reported by the model checker.


**Contexts**

- **Inria Team**: VERIDIS
- **Recruiter**: Konnov Igor / igor.konnov@inria.fr

**The keys to success**

- **Required qualification**: Candidates must have obtained a master degree or equivalent in computer science prior to the start of the PhD.

**Additional information**: The duration of the PhD position is 36 months. The regular starting date is October 1st, 2018, but can be delayed until December 1st, 2018. The salary is 1,982 euros gross per month during the first two years and 2,085 euros per month during the third year, health insurance included. It is possible to augment the salary through complementary activities such as teaching. Inria can help with finding accommodation and with obtaining a residence permit.

**How to apply**: Upload your CV on jobs.inria.fr. This should be a pdf file of at most 2Mo.

In addition, send the following documents to stephan.merz@inria.fr and igor.konnov@inria.fr in a single pdf or ZIP file:

- CV.
- A cover/motivation letter describing your interest in this topic.
- Your degree certificates and transcripts for Bachelor and Master (or the last 5 years).
- Master thesis (or equivalent) if it is already completed, or a description of the work in progress, otherwise.
- Publications or web links, if any (it is not expected that you have any).

In addition, one recommendation letter from the person who supervises (supervised) your Master thesis (or research project or internship) should be sent directly by his/her author to stephan.merz@inria.fr.

Applications are to be sent as soon as possible.

**Conditions for application**

- **PhD position**: 36 months. The regular starting date is October 1st, 2018, but can be delayed until December 1st, 2018. The salary is 1,982 euros gross per month during the first two years and 2,085 euros per month during the third year, health insurance included. It is possible to augment the salary through complementary activities such as teaching. Inria can help with finding accommodation and with obtaining a residence permit.

- **PhD position duration**: 3 years.
- **Starting date**: 2018-10-01
- **Deadline to apply**: 2018-05-01

- **Inria Center**: CRI Nancy - Grand Est
- **Town/city**: Villers-lès-Nancy
Skills
We expect the candidate to be familiar with the following subjects:

- Solid background on formal logic and verification techniques, in particular model checking and/or theorem proving or SMT solving.
- Solid programming experience, preferably in a functional language.
- Prior knowledge of TLA⁺ or a similar set-based specification language is a plus, but is not required.

The candidate should appreciate working on both fundamental aspects and the validation by a prototypical implementation. The working language is English.

Benefits package
- Subsidised catering service
- Partially-reimbursed public transport
- Social security
- Paid leave
- French courses

Remuneration

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

Warning: you must enter your e-mail address in order to save your application to Inria. Applications must be submitted online on the Inria website. Processing of applications sent from other channels is not guaranteed.