M2 Internship Proposal

Formally Verified Synthetic Graph Generators

January 2020

Research Unit: ENSIE & Samovar Laboratory

Supervisors: Stefania Dumbrava (METHODES team) & Catherine Dubois (METHODES team)

Keywords: property-based testing, formal proofs, graphs.

Context: The problem of generating synthetic graphs under constraints is an important one, as it is crucial to benchmarking and experimental analysis, in all areas that deal with graph-shaped data. As these are ubiquitous in numerous computer science fields, such as networks [1], Semantic Web [2], machine learning [3], and databases [4, 5], ensuring conformance to their desired specification warrants the usage of formal methods. Indeed, none of the recently surveyed graph generation systems in [5, 6] benefit from the usage of constraint solvers or of logic-based verification tools to guarantee that the outputs meet the desired requirements [7].

We propose a novel approach for addressing the need for strong structural compliance guarantees for synthetically generated graphs. This relies on property-based testing, a method through which one can define properties and use a generative-testing engine to create randomized inputs, ensuring their correctness. This methodology has been integrated in the Coq proof assistant through the QuickChick framework [8], which provides a generic infrastructure for writing testing code, using built-in combinators. While the tool provides automation support, users can still make errors when writing their custom generators. This can introduce bugs in the verification process and overall reduce the benefits of testing. To establish the veracity of the testing code, we propose to go a step further and formally prove the correctness of the graph generators themselves.

Internship Objective This internship’s goal is to develop a proof-of-concept methodology for verifying synthetic graph generators using the Coq proof assistant and the QuickChick tool.

We propose the following milestones:

1. Generator Implementation. We plan to implement graph generators, tailored to specific application scenarios, within the QuickChick framework. Depending also on the student’s preferences, we will look into algorithms for generating regular, diverse [9], or random graphs, based on the Barabási–Albert (scale-free preferential attachment) or on the Erdos–Rényi graph models. As a particularity of this approach, we will leverage the reflective properties of the SSReflect proof language to design generators amenable to deductive verification.

2. Constraint Specification. We will then formally specify practically relevant graph constraints. Specifically, we will focus on both topological and schema (labeling/typing) properties. With respect to the former, we will formalize common constraints, i.e., restricting the number of vertices and edges, degree, or density, scaling up to more intricate ones, involving probability distributions or combinatorial
graph enumeration properties [10]. With respect to the latter, we will look into the formalization of scheme used to constrain data in conceptual modelling [11], which lie at the basis of modern graph schema languages.

3. **Soundness Verification.** Finally, we aim to use similar techniques to those described in [8, 12] and prove that the implemented generators indeed satisfy the constraints we specified. To ease the verification effort, we plan to develop a modular, light-weight graph library, based on the MathComp library of formalized mathematical theories.

To achieve these milestones, the suggested work program roughly entails:

- Reviewing fundamental concepts concerning property-based testing for Coq, following relevant chapters in the Software Foundations book [13];
- Implementing a modular library for graph generators, building on the existing QuickChick combinators. To this end, inspired by [8], automating generation will be privileged, by using an inductive approach and, if needed, providing suitable extensions to the plug-in;
- Specifying topological properties, leveraging formalized concepts from state-of-the-art Coq libraries for probabilistic programming (Polaris [14], Infotheo [15], Alea [16]);
- Specifying schema properties, i.e., typing (labeling) and key/value properties for both vertices and nodes, as well as other participation and cardinality constraints, drawing from conceptual models;
- Proving the soundness of the generators with respect to the specified constraints;
- Writing the internship report that describes the proposed methodology and evaluates its feasibility, in terms of the proof effort (l.o.c) and efficiency (generation runtime).

**Internship Period:** 6 months.

**Opportunities:** This internship is an opportunity for the student to acquire the following skills:

- familiarity with property-based testing, which is increasingly used in industry: in functional software development, as well as in the development of mission-critical telecommunication components (e.g., Ericsson 4G base stations or Motorola gateways), RESTful web services, databases, or automotive control (AUTOSAR project) [17].
- familiarity with proof engineering - crucial to large-scale industrial verification projects [18], as a result of using the Coq proof assistant and its extensions, such as the SSReflect proof language, as well as the QuickChick property-based random testing tool.
- familiarity with pursuing formal methods research to advance the state-of-the-art in formally verified property-based testing. This is especially valuable for students pursuing a research masters and can lead to a PhD thesis further exploring this topic.

**Required Skills:** Familiarity with formal proofs and knowledge of functional programming (OCaml).

**Application:**
Please send grade transcripts and a CV to stefania.dumbrava@ensiie.fr or catherine.duboist@ensiie.fr.

**References:**


