Exploit parallel computing on SMT solving

Natasha Sharygina, Matteo Marescotti

22 December 2015

The *Satisfiability Modulo Theories* (SMT) problem is the decision problem of determining whether a logical formula is satisfiable, given that some of the variables have an interpretation with respect to combinations of first-order background theories. The expressiveness of SMT makes it suitable for a vast range of application domains, including *software verification*, and for that reason it has recently attained significant interest from both industry and academia.

The computational cost of SMT can be very high given that just the problem of determining the propositional satisfiability, the *Boolean Satisfiability Problem* (SAT), is proven to be NP-complete, and the introduction of the background theories can only make the problem harder. Nevertheless relatively little research exists on how parallel computing can be used to speed up SMT solving.

The goal of this UROP project is to exploit parallel computing on SMT solving by iteratively partitioning the search-space and solving each partition independently. The work, including implementation and experimentation, will be carried out as an extension of the novel parallel version of OpenSMT, an SMT solver developed by the Verification Group at USI.

We are looking for a motivated student who wants to improve his/her knowledge on software verification and cloud-based parallel computing. This project will give the student an excellent overview of a quickly developing field while being sufficiently approachable. Prior knowledge of C++ language is required. The knowledge of POSIX threads API of C language is highly recommended before beginning. Prior knowledge in SMT modelling is not required, though is a plus.

The project is from Prof. Natasha Sharygina and it will be assisted by Matteo Marescotti. The aim of this project is to extend the novel parallel version of OpenSMT. The student will be coached while:

1. Getting familiar with the current research and implementation state.
2. Implementing and debugging the core functionalities in a cloud computing environment.
3. Designing and running a set of experiments.