Incremental Inductive Verification, Does It Work? Does It Scale ?

Ahmed Rezine[1] and Arne Borälv[2]

[1] Linköping Universitet
[2] Prover Technology

Description

Powerful Multicore platforms are now widespread and promise to naturally scale performance by running programs in parallel. Except for optimizing one's code, this became the only way to speed up programs. As a result, a great amount of work focuses on coming up with clever ways to run programs in parallel. This thesis explores the applicability and the performance of an incremental verification technique (proposed by Aaron Bradely in his PhD thesis) that can be naturally run in parallel.

Verification explores all possible scenarios before validating a program or a design. It typically targets safety critical systems where missed bugs can have catastrophic consequences (transportation, medical instruments, etc). Schematically, to inductively show that a safety critical system satisfies a property P, one shows that P holds at the initial state and that it is preserved at each step of the system to be verified (consecution). Often, showing consecution is the challenging part. The traditional, and well established, approach is to come up with a property that is both inductive and that implies P. Apart from the fact that this involves a great deal of ingenuity, this approach is monolithic and is difficult to run in parallel. The incremental approach proposes instead to find series of properties (instead of one) that together imply the property P. In each such series, the latter properties are inductive if the earlier ones are assumed (with the first one being inductive without particular assumptions). This incremental aspect allows to build the series in parallel by loosely sharing the discovered inductive properties. It is this modularity with the resulting possibility of performing verification in parallel that is particularly appealing in the approach.

The thesis starts by gaining familiarity and experimenting with the new and promising incremental verification algorithm. There are already some open source implementations of the algorithm that may be used in the thesis, otherwise an implementation is to be carried out. An important part will be to collect existing benchmarks from international competitions and from Prover technology and to experiment with different heuristics (based on combinations with different abstract domains or with user supplied hints) when generating the intermediary properties. The applicability and the scalability of the resulting verification programs will then be compared to the existing verification tools.

Qualification

This 30 hp thesis can be carried by one or two Masters students that:

- Enjoy discrete mathematics and logics (related courses)
- Very good programming skills (for instance in c++)
- Good knowledge in scripting and in functional programming is a plus

Practical considerations

The student(s) will carry the thesis at the Department of Computer and Information Science in Linköping in collaboration with Prover Technology-Stockholm.