On Quantitative Software Verification

Marta Kwiatkowska

On Quantitative Software Verification

Marta Kwiatkowska

Oxford University Computing Laboratory, Parks Road, Oxford, OX1 3QD

Abstract: Software verification has made great progress in recent years, resulting in several tools capable of working directly from source code, for example, SLAM and Astree. Typical properties that can be verified are expressed as Boolean assertions or temporal logic properties, and include whether the program eventually terminates, or the executions never violate a safety property. The underlying techniques crucially rely on the ability to extract from programs, using compiler tools and predicate abstraction, finite-state abstract models, which are then iteratively refined to either demonstrate the violation of a safety property (e.g. a buffer overflow) or guarantee the absence of such faults. An established method to achieve this automatically executes an abstraction-refinement loop guided by counterexample traces [1].

The vast majority of software verification research to date has concentrated on methods for analysing qualitative properties of system models. Many programs, however, contain randomisation, real-time delays and resource information. Examples include anonymity protocols and random back-off schemes in e.g. Zigbee and Bluetooth. *Quantitative verification* [2] is a technique for establishing quantitative properties of a system model, such as the probability of battery power dropping below minimum, the expected time for message delivery and the expected number of messages lost before protocol termination. Models are typically variants of Markov chains, annotated with reward structures that describe resources and their usage during execution. Properties are expressed in temporal logic extended with probabilistic and reward operators. Tools such as the probabilistic model checker PRISM are widely used to analyse system models in several application domains, including security and network protocols. However, at present the models are formulated in the modelling notations specific to the model checker. The key difficulty in transferring quantitative verification techniques to real software lies in the need to generalise the abstraction-refinement loop to the quantitative setting. Progress has been recently achieved using the idea of strongest evidence for counterexamples [3] and stochastic game abstractions [4].

In this lecture, we present a quantitative software verification method for ANSI-C programs extended with random assignment. The goal is to focus on system software that exhibits probabilistic behaviour, for example through communication failures or randomisation, and quantitative properties of software such as "the maximum probability of file-transfer failure" or "the maximum expected number of function calls during program execution". We use a framework based on SAT-based predicate abstraction, in which probabilistic programs are represented as Markov decision processes, and their abstractions as stochastic two-player games [5]. The abstraction-refinement loop proceeds in a quantitative

fashion, yielding lower and upper bounds on the probability/expectation values for the computed abstractions. The bounds provide a quantitative measure of the precision of the abstraction, and are used to guide the refinement process, which proceeds automatically, iteratively refining the abstraction until the interval between the bounds is sufficiently small. In contrast to conventional approaches, our quantitative abstraction-refinement method does not produce counterexample traces. The above techniques have been implemented using components from GOTO-CC, SATABS and PRISM and successfully used to verify actual networking software.

The lecture will give an overview of current research directions in quantitative software verification, concentrating on the potential of the method and outlining future challenges.

Acknowledgements: Supported in part by EPSRC grants EP/D07956X, EP/D076625 and EP/F001096, and FP7 project CONNECT-IP.

References

1. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Proc. CAV'00. Volume 1855., Springer (2000)
2. Kwiatkowska, M.: Quantitative verification: Models, techniques and tools. In: Proc. 6th ESEC/FSE, ACM Press (September 2007) 449–458
3. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Proc. CAV'08. LNCS 5123, Springer (2008)
4. Kwiatkowska, M., Norman, G., Parker, D.: Game-based abstraction for Markov decision processes. In: Proc. QEST'06, IEEE (2006)
5. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Abstraction refinement for probabilistic software. In Jones, N., Muller-Olm, M., eds.: Proc. 10th VMCAI'09. Volume 5403 of LNCS., Springer (2009) 182–197