Systematic design of abstract model checking

Elisa Quintarelli

Politecnico di Milano, Dipartimento di Elettronica e Informazione.
Piazza Leonardo da Vinci, 20133 Milano (Italy)
E-mail: quintare@elet.polimi.it

Introduction

Model checking has emerged as a successful approach for automated verification of complex reactive systems, e.g., to estabilish the validity of security properties of protocols, tipically expressed using temporal logic [7, 13]. However, it is well known that verifying a temporal logic formula against a model, in particular finding all the system states that verify the formula, is in general a computationally hard problem. Moreover, model checking is usually applied to programs that consist of several concurrent processes, and thus, the number of states representing the whole programs behaviour may grow exponentially in the number of concurrent processes. This problem (i.e. the "state explosion problem") and the high complexity for verifying temporal formulae against a model, expecially for the temporal logic CTL*, are limiting factors that have to be tackled for any practical use of this technique.

According to a well established definition, "*Abstract interpretation is a general theory for approximating the semantics of discrete dynamic systems*" [3]. This theory offers an appropriate framework to approximate the model of a reactive system in order to obtain a simpler abstract model, over which the properties of interest can be checked for satisfaction. The idea here is that of verifying temporal properties in an abstract model which is systematically derived from the concrete semantic of the system we want to analyze, e.g., by abstracting the information contained in its states. Since the pioneering work on model checking and abstraction by Clarke et al. [1], a number of works have applied this idea to reduce the phenomenon of state explosion (e.g. [6, 12]).

The problem

One of the main challanges of current research in abstract model checking is the design of abstraction. Current applications of abstract model checking in fact are obtained by specifying equivalence relations on the structure of the state space, either by morphisms or by weaker notions like Galois connection-based abstract interpretations. While these simplifications may improve considerably the efficiency of model checking, they lack in most cases of a systematic design. Abstractions are usually derived by hand from an inspection of the properties of the system we want to analyze. It is in fact often the case that the abstraction is considered a secondary derived concept which is not formally related with the property, expressed in some temporal logic, of the system we want to analyze. Formal methods for tuning abstract model checking in accuracy and costs may instead provide the right tools for adopting it.

The main result

Abstract interpretation theory offers a number of methodologies that have not been applied yet in the field of abstract model checking. A number of authors recognized in the possibility of modifying abstract models by modifying abstractions a great potential for improving abstract model checking in precision and complexity (e.g. see Section 9 in [6]), but few applications of these techniques are known in abstract model checking. On the contrary, this practice is quite common in static program analysis by abstract interpretation. A number of operations have been studied both in theory and in practice to compose, decompose, refine and compress abstract domains and analyses (see [8, 9] for a survey), providing advanced algebraic methodologies and techniques for tuning analyses in accuracy and costs. This work is in progress and consists basically in two parts: (1) we study domain operations such as refinements and compressors for respectively systematically improving precision and reducing complexity in abstract model checking, and (2) applying these operations for systematically deriving *optimal* domains for abstract model checking.

Refining and compressing abstract model checking

In this work we study the impact of standard domain refinement operations in abstract model checking. The problem is that when a chosen abstract domain turns out to provide a too rough abstract model for verifying a given temporal property of interest, this model can be refined by refining the corresponding abstract domain. Conversely, any operation acting on domains which is devoted to their simplification (decomposition or compression) can play the dual rôle of reducing the complexity of the verification of temporal formulae, provided that the formulae of interest are verified in both abstract and concrete models. In both these situations, the key problem is to study the structure of temporal formulae which are preserved or lost by changing the abstract domain by means of domain refinement or simplification, and in particular the structure of those formulae that are verified in the new model and which were not verified in the former. We consider the universal fragment of the branching time temporal logic CTL* [7] and we characterize the structure of temporal formulae that are verified in a new abstract model which is obtained either by refining an abstract domain by means of standard operations for domain transformation introduced in [4], namely: reduced product [4], and

disjunctive completion [4], or by simplifying the domain by means of their inverse operations, namely by complementation for domain decomposition [2] or by least disjunctive bases for domain compression [10]. In particular we prove that relevant properties of systems can be checked compositionally by decomposing the abstract models by domain complementation and that disjunctive information is in some cases redundant in abstract model checking of CTL*. This may provide sensible simplification algorithms for improving abstract model checking in complexity yet maintaining accuracy.

Systematic design of abstract models

Correctness is a basic requirement of any approximation technique, and this holds also for abstract interpretation. In abstract model checking the notion of soundness is also required: suppose C is a transition system representing the behaviour of a reactive system and φ is a temporal logic formula which estabilishes a security property of the system. If we verify φ in an abstract model A which is derived from C, it must hold that $A \models \varphi$ implies $C \models \varphi$.
Although the notion of soundness is the basic requirement for any abstract interpretation, completeness is instead an ideal and uncommon situation [11]. Completeness means that, relatively to the semantic properties encoded by the abstract domains, no loss of information occurs. In this case, roughly speaking, the abstract semantic is able to take full advantage of the power of the underlying abstract domain. In abstract interpretation completeness is meant as the natural strenghtening of the notion of soundness, requiring its reverse relation hold.
In the case of abstract model checking by applying the results in [11], we provide a systematic methodology for deriving complete abstract models for verifying a temporal logic formula φ, namely the most abstract model A such that $A \models \varphi$ iff $C \models \varphi$. Moreover, the same methodology allows us to find, for any given abstraction, the structure of the most precise temporal calculus T (T could be a subset of some well known temporal logics as in [5]) such that $\forall \varphi \in T$ $A \models \varphi$ iff $C \models \varphi$.
All these methods are constructively driven by the specification (e.g. by a transition system) of the system we want to analyze and contribute to the systematic derivation of "*optimal*" abstract domains for abstract model checking.

References

[1] Clarke, E. M., Grumberg, O., and Long, D. E. Model checking and abstraction. *ACM Trans. Program. Lang. Syst. 16*, 5 (1994), 1512–1542.

[2] Cortesi, A., Filé, G., Giacobazzi, R., Palamidessi, C., and Ranzato, F. Complementation in abstract interpretation. *ACM Trans. Program. Lang. Syst. 19*, 1 (1997), 7–47.

[3] Cousot, P. Abstract interpretation. *ACM Comput. Surv. 28*, 2 (1996), 324–328.

[4] Cousot, P., and Cousot, R. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symp. on Principles of Programming Languages (POPL '79)* (1979), ACM Press, New York, pp. 269–282.

[5] Cousot, P., and Cousot, R. Temporal abstract interpretation. In *Conference Record of the 27th ACM Symp. on Principles of Programming Languages (POPL 2000)* (2000), ACM Press, New York.

[6] Dams, D., Gerth, R., and Grumberg, O. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst. 19*, 2 (1997), 253–291.

[7] Emerson, E. A. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed., vol. B: Formal Models and Semantics. Elsevier, Amsterdam and The MIT Press, Cambridge, Mass., 1990.

[8] Filé, G., Giacobazzi, R., and Ranzato, F. A unifying view of abstract domain design. *ACM Comput. Surv. 28*, 2 (1996), 333–336.

[9] Giacobazzi, R., and Ranzato, F. Refining and compressing abstract domains. In *Proc. of the 24th Internat. Colloq. on Automata, Languages and Programming (ICALP '97)* (1997), P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, Eds., vol. 1256 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 771–781.

[10] Giacobazzi, R., and Ranzato, F. Optimal domains for disjunctive abstract interpretation. *Sci. Comput. Program 32*, 1-3 (1998), 177–210.

[11] Giacobazzi, R., Scozzari, F., and Ranzato, F. Making abstract interpretation complete. *Journal of ACM* (March 2000).

[12] Kesten, Y., and Pnueli, A. Modularization and abstraction: The keys to formal verification. In *The 23rd International Symposium on Mathematical Foundations of Computer Science* (1998), L. Brim, J. Gruska, and J. Zlatuska, Eds., vol. 1450 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 54–71.

[13] Manna, Z., and Pnueli, A. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, Berlin, 1992.