20

Verification Model Reduction through Abstractions

Jean-Charles Grégoire[1]
INRS-Télécommunications, 16, Pl. du Commerce, Verdun, P.Q., CANADA H3E 1H6

1 Introduction

The major problem still facing researchers in the formal verification of finite state systems is the *state space explosion.* This problem occurs partly because the domain of each variable of the system model contributes to the global state and verification typically involves an exhaustive search through the state space, to recognize states corresponding to wanted or unwanted system configurations. In telecommunication or hardware systems, most practical problems have a number of states that exceeds the memory capacity of our computers. New techniques of state-space compression, "on-the-fly" computing, combination of various techniques such as so-called partial order methods or symbolic computing with Binary Decision Diagrams (BDDs) have allowed us to tackle problems several orders of magnitude greater. Yet, it is clear that the "brute force" approach to verification is unlikely to ever be sufficient, and we need to develop not only new techniques but also methods to achieve further practical reduction of the state space. Recent research on this topic has shown promising results (e.g. [Loi94]).

We describe here the salient features of a new verification tool dedicated mainly to the *expression* and the *verification* of communication protocols, but also meant to be suitable to verify distributed applications.

2 Expression

Our modeling language has an algorithmic flavour, and is vaguely related to PASCAL, combined with single-writer/multiple readers synchronous communication primitives and with the nondeterministic repetitive and alternative constructs proposed by Dijkstra [Dij75]. We have found that such a language is quite convenient to express complex problems such as protocols and distributed algorithms in a way that is fairly "natural" for designers. PROMELA[Hol89] is an example of a quite successful modeling language with similar aims. PROMELA and its tool, SPIN, as well some of their limitations, have actually inspired some of our developments.

[1]Prof. Grégoire is supported by the Chaire Cyrille Duquet en Génie logiciel des télécommunications

D. Hogrefe et al. (eds.), *Formal Description Techniques VII*
© IFIP International Federation for Information Processing 1995

3 Verification

We have integrated different forms of abstractions which result in a significant decrease in the size of the verification problem. We consider two categories of abstractions: those who reduce the number of transitions, and those who reduce the size of the state information.

3.1 Transition reduction

To achieve a reduction on the number of transitions, we use well-known techniques, namely functional ("compact") transitions, which allows a transition to reflect a complex transformation of the state, *symmetries*[ID93, CFJ93], which restrict the analysis of "similar" behaviours to a unique case, and *partial orders* [GW93], which restrict the amount of interleaving.

3.2 State information compression

State information compression means a reduction of the amount of state information. Such information can be superfluous when it does not contribute to the properties we want to verify, or it can be redundant, when the value of some state variables can be inferred from others.

3.2.1 Value domain abstractions

We make a distinction between two categories of variables in the state space. On the one hand, we have *control variables*, which capture all forms of interaction between processes (i.e. global variables and communication channels) and nondeterministic choices. On the other hand, we have *computation variables*, used mainly for the purpose of computing the values of control variables, or exchanged data.

Computation variables, and their manipulation, are very often removed from verification models, typically through nondeterministic selections where their values might influence behaviour. This may however contribute to generate otherwise unrealistic behaviours, or not capture enough of the model for the purpose of verification.

We manipulate those two types of variables in different ways, to guarantee that computation variables do not contribute to the growth of the state space while still taking their influence into account.

3.2.2 Redundancy elimination

There are cases in practice where state information is redundant, or superfluous. Let us consider for example a process implementing a buffer with a FIFO discipline. We can abstract out the buffer and use only the number of items held as a control variable. This information, combined with the state of a producer and that of a consumer is enough to generate all the data-independent behaviour. Indeed, the behaviour of FIFO buffers does not depend on the value of its content. However, in modeling the buffer this way, we lose the possibility to transmit the information itself to the receiving side.

If the data influences the behaviour of the receiver, we have several courses of action which preserve the expression of the model. Either we keep the buffer in the model, in spite of the redundancy, or we abstract it out and generate all the possible values at the receiver side, a form of nondeterminism which may create more behaviours than the original model would allow. We introduce a mechanism for a third course of action, which is to keep the information behind the scene, i.e. on the evaluation stack rather than in the state information, and give the exact value to the receiver when it is needed, i.e. currently with a read. These so-called "pnambic" techniques can be used with any kind of intermediate buffer structure.

Another technique that we use is the introduction of extra variables in the state space which are evaluated from computation variables but present a better abstraction of the property. One example is a"volume" variable which to count the number of items held in a buffer, and computed from pointer variables in a circular buffer implementation.

3.2.3 Server

In client-server systems, Remote Procedure Calls (RPCs) are very often used as an access mechanism to the operations of the server. This mechanism is a natural extension of function calls. By combining a blocking test mechanism, function calls and persistant (global) computational variables, we can abstract out of the state space (deterministic) servers while preserving their behaviour, and preserving once again the expression of the model.

4 Conclusions

We have presented several techniques of abstraction. These techniques can reduce the number of transitions and the size of the state information. A prototype of new tool integrating these techniques has been realized.

References

[CFJ93] E.M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. In *Proc. Conference on Computer Aided Verification*, Lecture Notes in Computer Science, pages 450–462. Springer Verlag, 1993.

[Dij75] E.W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

[GW93] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, April 1993.

[Hol89] G. Holzmann. Algorithms for automated protocol verification. *AT&T Technical Journal*, 68(1):32–44, 1989.

[ID93] C-W. Norris Ip and David L. Dill. Better verification through symmetry. In *International Conference on Computer Hardware Description Languages*, pages 87–100, 1993.

[Loi94] C. Loiseaux. *Vérification symbolique de programmes réactifs à l'aide d'abstractions.* PhD thesis, Université Joseph Fourier de Grenoble, Laboratoire VERIMAG, 1994.