Using Constraints for Model Checking Buffered Systems
(Extended Abstract)

Sept. 13, 2005

Claus Traulsen and Martin Steffen
{ctr,ms}@informatik.uni-kiel.de

Christian-Albrechts-University Kiel, Germany

1 Motivation

In model checking asynchronous systems with buffered communications (via queues), the state space of the queue itself, beside the interleaving, can lead to state explosion, expecially in enumerative model checking. For a queue of length n, through which k different values are sent, k^n possible states exist. While partial order reduction decreases the relevant number of interleavings, it does not decrease the number of reachable combinations in the queue.

For illustration consider a simple protocol (Figure 1), where a process sends acknowledgment and reject messages randomly over a queue to another process, performs some computation and, after some time, takes a decision based on the value that was sent.

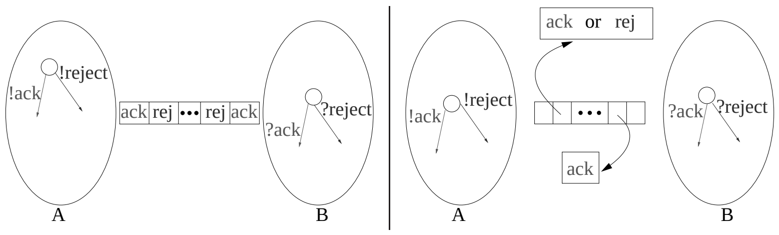

Fig. 1. A simple protocol: Process A sends reject and acknowledgment messages to Process B. The queue can either be represented explicitly or with pointers.

Computing the reachable states in a naive way would mean to compute all states where an acknowledgment was sent and all states where a reject was sent. For each state after a value was sent and before a decision is based on it, a similar state is considered which differs only in the value in the queue. Instead, one could just remember that a value was send, but not specify exactly which one. Then the number of states between sending the value and actually using it is halved. This can drastically reduce the number of states of the queue.

To exploit abstraction of values on the one hand and to keep the coupling between the processes on the other hand, one could use pointers into the lattice obtained from the power set of possible values instead of concrete values for variables. As soon as the value is is needed by either the sender or the receiver it is further specified, which is directly observable by the other process, too. The different cases do not need to be considered independently, until a real decision is based on the values. For the concrete implementation of this idea, we use constraints as a more formal way to obtain the same information as with pointers; thus we use a symbolic representation of states in enumerative model checking.

2 Results

Our contribution consists of both a theoretical formalization of this method and an experimental evaluation. The full results can be found in [3].

2.1 Formalization of the Method

To model the processes we give a semantics for labeled transition systems that uses constraints instead of explicit valuations for the variables. Since one constraint can represent multiple valuations, this can reduce the state-space. We show both soundness and completeness of this method, i. e., every valuation that is represented by a reachable state is indeed reachable and every reachable valuation is represented by a reachable state. Consequently, the semantics can be easily used to check whether a state formula without temporal operators holds for all states.

Since one state can represent multiple valuations, the exact information from which valuation another state is reachable might get lost and more possible behavior is introduced. Therefore, false negatives might occur when checking arbitrary LTL formulas, i. e., a formula might be considered not valid for a given program, even though it is indeed valid.

It is also more complicated to check whether a state was already reached before, since it must be tested whether all valuations are already represented by another state. In order to simplify this check, it might be useful to split states which represent various valuations. This can also be useful if it is too hard to represent one combined state, while it is easy to represent two states with the same information. Different heuristics when to split constraints have a strong influence on how hard it is to store a single state and how many states have to be considered. In the experiments we use the two most extreme strategies; we either split always, which is similar to a standard semantics with explicit valuations, or never.

2.2 Experimental Results

We compare the semantics with constraints with a semantics that uses explicit valuations for a simple example, consisting of just two processes which communicate via one queue. The number of reachable states and the time needed to compute them is measured with

1. a C^{++} implementation that uses explicit valuation to represent states
2. the same implementation, but the states are represented by constraints. The Integrated Canonizer and Solver (ICS, [1]) is used to solve the constraints.
3. the enumerative model checker Spin [2].

The comparison is shown in Figure 2.

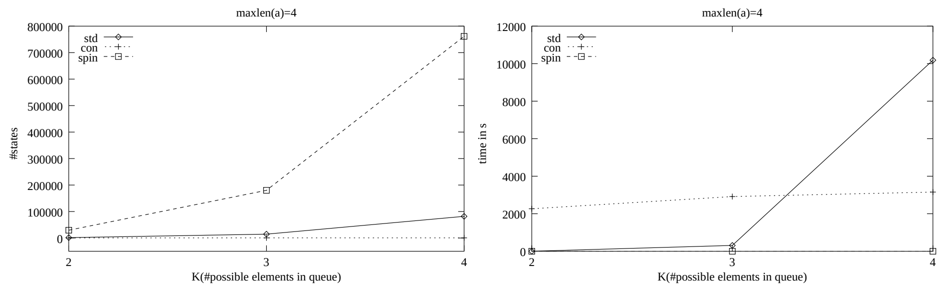

Fig. 2. Number of reachable states and time to compute them using the standard semantics, a semantics that uses constraints and the model checker Spin.

The example is parameterized by the number of different elements sent through the queue. When this number is increased the number of states used by explicit representation of the values is increased significantly, whereas it has no influence on the number of states when constraints are used. Even though more time is needed to compute every single state using constraints, the lower amount of states leads to a faster computation in total. While Spin is much faster and more efficient in representing the reachable states, the increase of the number of stored states shows that it suffers from the state explosion of this example, too.

The figures show that using constraints to represent states can be beneficial for model checking systems with queues, but further experiments, e. g. with different heuristics on how to split states, have still to be done.

References

1. J.-C. Filiâtre, S. Owre, H. Rueß, and N. Shankar. ICS: Integrated canonizer and solver. In G. Berry, H. Comon, and A. Finkel, editors, *Proceedings of CAV '01*, volume 2102 of *Lecture Notes in Computer Science*, pages 246–249. Springer-Verlag, 2001.
2. G. J. Holzmann. *The Spin Model Checker*. Addison-Wesley, 2003.
3. C. Traulsen. Using constraints for model checking asynchronous systems with queues. Master's thesis, Christian-Albrechts-Universität zu Kiel, Germany, 2005. Also available as `http://www.informatik.uni-kiel.de/~ctr/PostChoice.pdf`.