Verifying arbitrary temporal formulas in the temporal logic of actions

Lucian Wischik

In SRC Technical Note 1999-003. This report describes a summer project undertaken by Lucian Wischik of the University of Cambridge, at Compaq Systems Research Center. He was supervised by Leslie Lamport and Yuan Yu.

The Temporal Logic of Actions

"Engineers should be able to specify and verify their systems directly and conveniently in logic." This aphorism provides a pragmatic motivation for Lamport's Temporal Logic of Actions, and for the associated model-checker TLC. It also distinguishes TLC from other comparable model-checkers, where specifications must be translated into a special-purpose language. (A model of a specified system, in this context, is a sample execution trace. To 'check' a model is to ensure that it satisfies the given verification conditions).

The goal of the project was to extend TLC to verify arbitrary temporal formulas. Before, it could verify only "always" formulas about states:

"always, the state will be such with no dangling pointers"

Now it can verify more general formulas:

"if ever I put a datum into my reliable-transport-protocol, then it must eventually come out the other end."

To verify arbitrary temporal formulas about states is a standard problem, addressed by other model-checkers as well, and is solved using the "tableau" technique of Clark and Emerson ("Design and synthesis of synchronization skeletons using branching time temporal logic", 1981). We briefly outline this technique below. (Temporal formulas are ones involving the predicates $\Box$ always, or $\Diamond$ eventually. For instance, $\Box(a \Rightarrow \Diamond b)$ means that every occurence of a must eventually be followed by one of b).

However, new problems arise in the application of this technique to the Temporal Logic of Actions. In fact, they arise as a direct consequence of the very *actions* that give the logic its name. (Actions relate the next state of the system to the current – they describe the transitions of the system. An example action is $x' = x + 1$.)

Checking fairness: disjunctive normal forms

The first problem concerns *fairness* criteria. A fair transition is one that must eventually be taken (assuming that it is possible). In convential model-checkers, fairness is specified explicitly in the special-purpose specification language. For example, to say that the scheduler must eventually allow the process to proceed, we might write:

`fair`; `x` := `x`+`1`;

In TLC, fairness is expressed as a logical predicate on actions. "Always, eventually, $x' = x + 1$."

$$\Box\Diamond(x' = x + 1)$$

Note that this is just a logic formula, and can appear anywhere in the specification or verification conditions (whereas the keyword `fair` can obviously appear only as part of the specification program).

The new technique we introduced to deal with such formulas, in specification or verification conditions, involves converting the problem into *disjunctive normal forms*. These forms always have the same structure:

$$\begin{array}{ll} & (\Diamond\Box \mathrm{ea}_1 \wedge \Box\Diamond \mathrm{ae}_1 \wedge \mathrm{misc}_1) \\ \vee & (\Diamond\Box \mathrm{ea}_2 \wedge \Box\Diamond \mathrm{ae}_2 \wedge \mathrm{misc}_2) \\ \vee & \ldots \end{array}$$

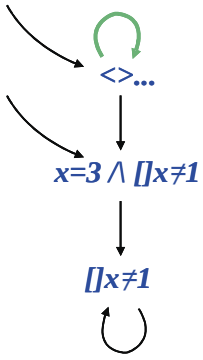
Observe that, within each disjunct, the fairness formulas $\Diamond\Box$ and $\Box\Diamond$ are all gathered together. This gives for them a straightforward decision procedure: an infinite cycle in the system's behaviour satisfies $\Diamond\Box p$ if p is true *everywhere* in the cycle; and it satisfies $\Box\Diamond q$ if q is true *somewhere* in the cycle. We provided an algorithm to convert arbitrary expressions into normal form, proved that the conversion preserves meaning and that the normal forms are unique, and implemented the decision procedure.

Checking arbitrary temporal formulas: tableau

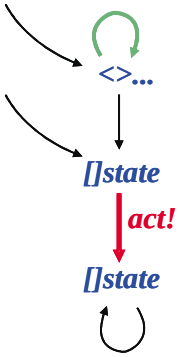
The fairness formulas above are a generalisation of the fariness specifications present in other model-checkers. But the Temporal Logic of Actions actually allows even greater generality – it allows arbitrary temporal formulas involving actions and states. To handle such formulas in full generality requires that the standard tableau technique be modified to handle actions. Unfortunately, this modification was not discovered until late in the summer, and there was not time fully to develop the theory. (The disjunctive normal forms, although just a special case, are still important: they are essentially an optimisation that reduces the exponential cost of fairness tableau).

The tableau technique is as follows. To check whether a sample execution trace satisfies a given temporal formula on states – for instance, $\neg(\Box(a \Rightarrow \Diamond b))$ – we construct a particular (non-deterministic) finite state machine. This

machine accepts only those traces which satisfy the formula. We run the sample execution-trace in parallel with the machine. If the trace is accepted by the machine, then it satisfies the temporal formula! The machine for the example formula is illustrated below.



The suggested modification of the technique, so that it can check arbitrary formulas on actions as well as on states, is illustrated in the example machine below.



Implementation

The techniques described above, for conversion into disjunctive normal form and for the tableau technique, were implemented within TLC. As presented, the techniques may be prohibitively expensive in time and space. They have not yet been tested on real-world examples. It also remains an open question as to whether there is any significant practical, engineering benefit to the verification of temporal formulas. Hopefully, the two new techniques introduced in this summer project will eventually lead to an answer.