Information Processing Letters 22 (1986) 307–309
North-Holland

30 May 1986

# LIMITS FOR AUTOMATIC VERIFICATION OF FINITE-STATE CONCURRENT SYSTEMS

Krzysztof R. APT * and Dexter C. KOZEN **

*IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.*

Communicated by David Gries
Received 1 May 1985
Revised 29 July 1985

*Keywords*: Logics of programs, program verification, concurrency

In one of the most exciting developments in the theory of program correctness, Clarke and Emerson [1] and Queille and Sifakis [4] developed systems which automatically check whether a given finite-state concurrent program satisfies a given temporal formula. These systems allow us to verify the correctness of various nontrivial concurrent programs automatically. For example, automatic verification of correctness of the alternating bit protocol on the system of [2] took just over 19 seconds of CPU time.

An obvious limitation of these systems comes from the fact that only finite-state programs can be checked for correctness. Although many interesting concurrent programs are in fact finite-state, they are often given schematically in terms of a parameter n, representing the number of concurrent processes. Such a schematic program really represents an infinite sequence of uniformly defined finite-state programs. Consider, for example, solutions to the mutual exclusion problem. A solution for n processes competing for access to a critical section is to be correct for any value of $n \geqslant 2$. To check correctness of such a solution using the systems of [1,2,4] (provided, for each $n \geqslant 2$, the solution uses only a finite number of states), one would have to verify the solutions for 2 processes, 3 processes, etc. obtained from the schematic one by plugging in the values 2, 3, etc. for n. This cannot be done in finite time. An obvious question is whether the systems of [1,2,4] can be extended to handle schematic programs.

We show in this short article that in general such an extension cannot exist. The proof of this fact is obvious and the main difficulty lies rather in a satisfactory formalization of the problem. In subsequent considerations we abstract from the particular representation of the finite-state programs and from the particular form of the temporal logic considered. all functions considered are unary unless stated otherwise.

Let P, $\phi$ denote (codes of) total recursive functions which for each n produce a finite-state program P(n) and a temporal formula $\phi(n)$ in a given temporal logic, respectively. We call the pair (P, $\phi$) a *verification problem*.

A typical example of a verification problem is the following (incomplete) solution to the mutual exclusion problem for n processes: P(n) is the schematic program $[P_1 \| \ldots \| P_n]$ where for $i = 1, 2, \ldots, n$, $P_i$ is the program

```
$z_i := i$;
while true do
    while $i \neq \min\{j \mid z_j = j\}$ do skip od;
    $CS_i$;   / * critical section * /
    $z_i := n$
od
```

* On leave from LITP, Université Paris 7, 75251 Paris, France.
** Present affiliation: Computer Science Department, Cornell University, Ithaca, NY 14853, U.S.A.

and $\phi(n)$ is the temporal formula

$$\bigwedge_{1 \leqslant i \leqslant n} z_i = 0 \rightarrow \bigwedge_{1 \leqslant i < j \leqslant n} \Box \neg (at\ \mathrm{CS}_i \wedge at\ \mathrm{CS}_j).$$

Here, $\mathrm{CS}_i$ for $i = 1, 2, \ldots, n$ is a further unspecified finite-state program which does not modify $z_1, \ldots, z_n$. Observe that, for each n, P(n) is a finite-state program satisfying $\phi(n)$.

Let R be any recursive set of pairs of codes $(P, \phi)$ of total recursive functions containing all pairs of finite-state transducers. Let $Q_R$ be the relation

$$Q_R(P, \phi) \Leftrightarrow \begin{cases} \forall n\ (P(n) \text{ satisfies } \phi(n)) & \\ \qquad \text{if } (P, \phi) \in R, & \\ \mathit{false} \quad \text{otherwise.} & \end{cases}$$

We now show that $Q_R$ is not semi-decidable. In fact, the following stronger result holds.

**Theorem.** *Any relation* $Q_R$ *is* $\Pi_1^0$*-complete* (*in the terminology of Rogers* [5]).

**Proof.** By the results of [1,2,4] there exists an algorithm whose running time is polynomial in the number of states and the length of the temporal formula which allows us to decide whether a given finite-state program satisfies a given temporal formula. (Clarke et al. [1,2,4] use branching time logic, but the Theorem holds for any temporal logic for which this validity problem is decidable.) Thus, $Q_R$ is in $\Pi_1^0$. To show that $Q_R$ is $\Pi_1^0$-hard, let M be an arbitrary Turing machine with no input, and let P(n) be the following finite-state program:

```
begin
flag := false;
for i := 1 to n do
    simulate one step of M
od;
if M has not yet halted then flag := true
end
```

Then the final value of the Boolean variable *flag* is *true* iff M has not halted within n steps. Thus,

$$\neg\mathrm{HALT}(M) \Leftrightarrow \forall n\ P(n) \text{ satisfies the formula}$$
$$\Box(at\ \mathbf{end} \rightarrow \mathit{flag})$$
$$\Leftrightarrow (P, \Box(at\ \mathbf{end} \rightarrow \mathit{flag})) \in Q_R,$$

where $\Box$ stands for the usual 'always' operator of linear temporal logic. We can also write

$$\neg\mathrm{HALT}(M) \Leftrightarrow \forall n\ P(n) \text{ satisfies the formula } \Diamond\, \mathit{flag}$$
$$\Leftrightarrow (P, \Diamond \mathit{flag}) \in Q_R,$$

where $\Diamond$ stands for the usual 'eventually' operator of linear temporal logic (see, e.g., [3]).

Under a suitable encoding, P can be computed by a finite-state transducer, since it need only attach a bounded integer counter to a description of M; and the function $\phi$ is constant. This proves the Theorem, since $\neg$ HALT is $\Pi_1^0$-complete. $\Box$

Note that we have in fact proved that there is a fixed formula $\phi$ such that the relation $Q_R(\cdot, \phi)$ is $\Pi_1^0$-complete.

The proof does not make use of parallelism; for each n, P(n) is a sequential program. By using instead of P(n) a parallel program $[P(n) \parallel \ldots \parallel P(n)]$ consisting of n identical components, we get the same result addressing parallel programs explicitly.

The Theorem shows that even very simple temporal properties of schematic finite-state programs are not semi-decidable. Clearly, the result does not depend on the logic chosen. Indeed, for the case of deterministic programs, linear and branching time temporal logics coincide. The program properties considered in the proof can also be expressed in Hoare's logic, dynamic logic, or any other reasonable for formalism appropriate for studying program correctness.

**Acknowledgment**

We thank Martin Abadi for helpful comments on an earlier draft. We have recently learned that a similar result was proved independently by Lenore Zuck, who showed undecidability by transforming register machine programs into symmetric finite-state concurrent programs.

## References

[1] E.M. Clarke and E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: Logics of Programs, Lecture Notes in Computer Science 131 (Springer, Berlin, 1982).

[2] E.M. Clarke, E.A. Emerson and A.P. Sistla, Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach, in: Proc. 10th POPL, Austin, TX, 1983.

[3] Z. Manna and A. Pnueli, Verification of concurrent programs: Temporal proof principles, in: Logics of Programs, Lecture Notes in Computer Science 131 (Springer, Berlin, 1982).

[4] J.P. Queille and J. Sifakis, Specification and verification of concurrent systems in CESAR, in: Proc. 5th Internat. Symp. on Programming, Torino, Lecture Notes in Computer Science 137 (Springer, Berlin, 1982).

[5] H. Rogers, Jr., Theory of Recursive Functions and Effective Computability (McGraw-Hill, New York, 1967).