Exposure to Deadlock for Communicating Processes is Hard to Detect

Thomas Raeuchle
Sam Toueg*

TR 83-555
May 1983

Department of Computer Science
Cornell University
Ithaca, New York 14853

*Partial support for this work was provided by the National Science Foundation under grant No. 83-03135.

EXPOSURE TO DEADLOCK FOR COMMUNICATING PROCESSES IS HARD TO DETECT

Thomas Raeuchle

Sam Toueg

Department of Computer Science, Cornell University

ABSTRACT

It is shown that the applicability of global state analysis as a tool for proving correctness of communication protocols is rather limited. Brand et al. showed that reachability of global deadlock states for protocols with unbounded FIFO channels is undecidable. It is shown, that the same is true for unbounded non-FIFO channels. For bounded FIFO channels the problem is shown to be PSPACE-hard. Protocols with bounded non-FIFO channels are shown to be analyzable in polynomial time.

1. Introduction

As protocols are becoming more and more sophisticated and complex, a need emerges for formal specification and verification methods. Protocols for distributed computations can be described by two characteristics: The party characteristics, a description of the possibly infinite set of input-output message sequences of the processes participating in the protocol, and the topology characteristics, the set of topologies the protocol is designed to work on. The protocol characteristics determine the applicability of theoretical models and verification methods to a particular protocol.

Various models have been proposed in the literature, among them finite state machines, Petri nets and high level programming languages. Properties to be verified include freedom from deadlock, progress, recovery from failure, self-synchronization, partial correctness and termination.

The problem of proving certain properties is well understood if the underlying message passing discipline is synchronous [Levin]. There are proof systems in the literature for asynchronous message passing [Schlichting, Zaf], but most of them are not complete in the sense that it is possible to prove partial correctness but not termination. Recently, [Yu] gave a polynomial-time algorithm for deadlock detection for communicating finite state machines that exchange only one type of message. In the sequel we show that, if a small number of message types are used, it is inherently difficult to show termination for this type of communicating processes. This result holds even if the communication channels are bounded.

The remainder of the paper is organized as follows. In section 2 we describe our model for protocols, in section 3 we present undecidability results for two-party protocols with unbounded channels. Section 4 shows the problem to be PSPACE-hard for

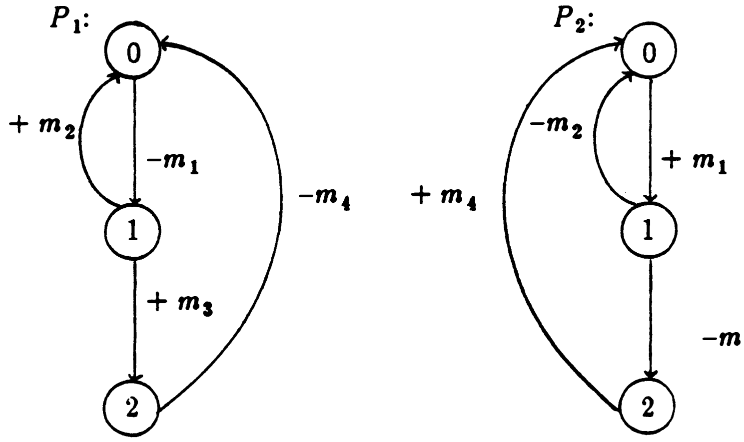

Figure 1. Resource Allocation Protocol

message m_1. Process P_2 receives the request and changes to state 1. It then either grants the request by sending message m_3 and changing to state 2, or it denies the request by sending message m_2 and changing back to state 0. P_1 releases the resource by sending m_4 and changing to state 0.

Protocol validation techniques using finite state models basically consist of the following steps. Construct a global system state S and perform a reachability analysis of states, based on perturbing the global state by all legal state transitions, and examining the resulting global states for certain properties. Following [Zaf] we model global states as 2 by 2 matrices[1]

$$S = \begin{pmatrix} s_1 & C_{12} \\ C_{21} & s_2 \end{pmatrix}$$

where s_i denotes the state of process P_i, and C_{ij} the content of the channel from process P_i to P_j.

[1]Note that we only consider two-party protocols.

non-FIFO channels.

Formal definitions and proofs of the following classical results about TMs can be found in [HU]. According to Rice's theorem any non-trivial property of the recursively enumerable languages, the languages accepted by Turing machines (TM's), is undecidable. It is a well known result that membership of the empty word ϵ in a recursively enumerable language is a non-trivial property. Therefore, it is undecidable whether a TM accepts ϵ.

An *off-line* TM is a TM with a special input tape. The input is given between a left endmarker $\cent$ and a right endmarker \$. The TM is not allowed to write on the input tape, nor can the head be moved beyond the endmarkers. Off-line TMs are equivalent to regular TMs.

A *counter machine* is an off-line TM with semi-infinite storage tapes and only two tape symbols, B (blank) and Z (bottom of stack). An integer i can be stored on a tape by moving the head i cells to the right of Z. To increment a stored number by k the head is moved k cells to the right, to decrement it is moved k cells to the left. The TM can only check whether a counter is 0, i.e., if the corresponding head is positioned at Z, but it cannot directly determine the value of a counter. Two-counter machines can simulate arbitrary TMs.

It follows from the above results that it is undecidable whether a two-counter machine accepts ϵ. We show how to construct a protocol that simulates a two-counter machine. The protocol will reach a deadlock state if and only if the two-counter machine accepts ϵ.

We first modify the two-counter machine so that it erases both its storage tapes channel. On receiving a special flush message f from P_1, P_2 enters a receiving loop until channel C_{12} is empty. It then sends back the message f. Figure 2 shows the finite state representation of process P_2, figure 3 shows the flush component.

Figure 3 is to be interpreted as follows: on receiving message f, P_2 enters the flush state s_{f_1}. It then continues to receive messages m_1 and m_2 and send them back to P_1, until channel C_{12} is empty, i.e., ϕ_i is returned. It then sends back the f message and enters the "normal" receiving state s_0.

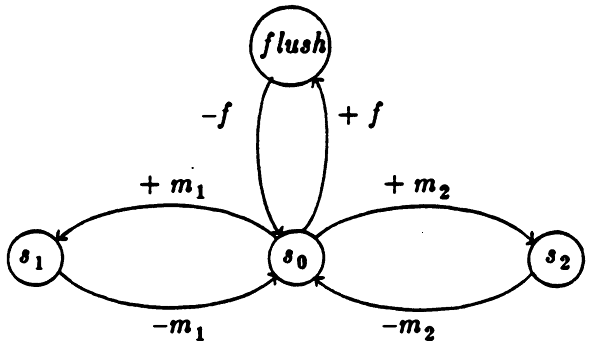

Figure 2. Message repeater P_2

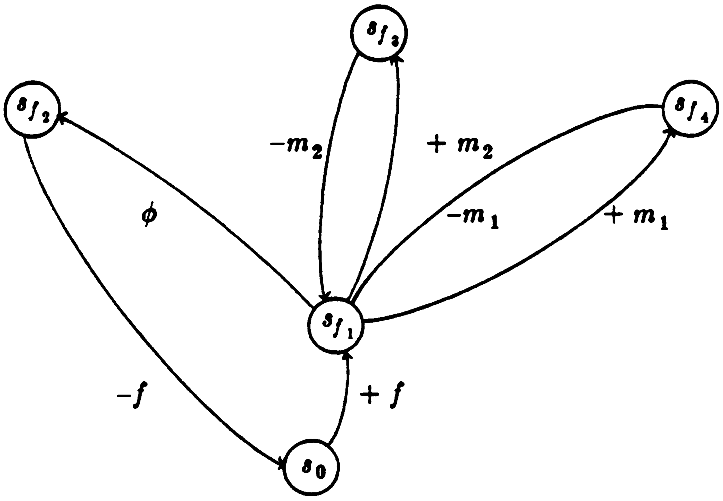

Figure 3. Flush component for P_2

Thus, a deadlock state is reachable if and only if the counter machine accepts ϵ. Therefore it is undecidable whether a global deadlock state is reachable.

4. Bounded FIFO channels

In practice, communication channels are bounded by bandwidth and the number of buffers available for storing incoming and outgoing messages. The "boundedness" is usually enforced by flow control algorithms. In this section we will show that determining exposure to deadlock for protocols with bounded FIFO channels is as hard as determining whether a *linear bounded automaton* (LBA) accepts a word w. This problem is known to be complete for PSPACE [Karp].

A LBA is a nondeterministic on-line TM with two special tape symbols, the left endmarker $\not c$ and the right endmarker \$. It cannot move its head to the left of $\not c$ nor to the right of \$, nor can it print over any of these markers. In other words, its available workspace is bounded by the length of the input. Since the LBA can have an arbitrary large, albeit fixed alphabet, the workspace is actually bounded by a constant times the length of the input.

LBA's recognize context-sensitive languages (CSL's) plus possibly the word ϵ which, by definition, is not in any CSL. For our considerations we exclude the case where w is ϵ. Let $G = (V, T, P, S)$ be a context-sensitive grammar for CSL L, where V is the set of nonterminal symbols, T is the set of terminal symbols, P is the set of productions, and $S \in V$ is the starting symbol. From G we can derive a LBA that can answer the question whether $w \in L(G)$ as follows. The LBA guesses a production $\alpha \rightarrow \beta$ and a position in w such that $w = w_1 \beta w_2$, and replaces β in w by α. This is always possible within the $\not c$ and \$ limits since, by definition, the length of α is always smaller channels is PSPACE-hard.

5. Bounded non-FIFO channels

For bounded non-FIFO channels the problem is solvable in time polynomial in the length of the description of the protocol, provided the channel bound is fixed rather than a parameter of the input.

Consider a protocol with state space S_1 for P_1 and S_2 for P_2. Let M be the set of message names with $|M| = n$,[3] and let k be the bound on the channel. As indicated above, k is considered a constant. A configuration of the protocol is a tuple $I = (s, s', C_{12}, C_{21})$ where $s \in S_1$ and $s' \in S_2$ and $C_{ij} \in M^k$.

The number of different configurations is bounded by $c * |S_1| * |S_2| * n^k$. Thus, exposure to deadlock can be checked by an exhaustive search of the configuration space.

6. Discussion

We showed that the applicability of global state analysis as a tool for proving protocols correct is rather limited. First we showed that determining exposure to deadlock for two-party protocols with unbounded non-FIFO channels is impossible. Unlike in [Brand], where FIFO channels are considered, our proof does not require both channels to be infinite. The semantics of our send operation allow us to consider only one unbounded channel. We also do not regard the receive operation as a "passive" operation, our receive operation is explicitly triggered by the process. The fact that only three message types are required in the proof of the undecidability result suggests that global state analysis is impractical protocols with unbounded channels.

[3] The size of a set M is denoted by |M|.

[Levin]
G. Levin "Proof Rules for Communicating Sequential Processes", *Ph.D. Thesis, Department of Computer Science, Cornell University, August 1980*

[Schlichting]
R. D. Schlichting, "Axiomatic Verification to Enhance Software Reliability", *Ph.D. Thesis, Department of Computer Science, Cornell University, January 1982*

[Yu] Y. T. Yu, M.G. Gouda, "Deadlock Detection for a Class of Communicating Finite State Machines", *IEEE Trans. on Comm., Vol. Comm-30, No. 12, December 1982*

[Zaf] P. Zafiropulo, C. West, H. Rudin, D. D. Cowan "Towards Analyzing and Synthesizing Protocols.", *IEEE Trans. on Comm., Vol. Comm 28, No. 4, April 1980*