数理解析研究所講究録
996 巻 1997 年 1-4

Fighting Livelock in the i-Protocol with the Concurrency Factory*

Y.S. Ramakrishna, Scott A. Smolka, Eugene W. Stark
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794–4400
USA
ysr,sas,stark@cs.sunysb.edu

Oleg Sokolsky
Computer Command and Control Company
2300 Chestnut Street
Philadelphia, PA 19103
USA
sokolsky@cccc.com

Abstract

We report on our efforts to detect, locate, and correct a non-trivial livelock error in the i-protocol, a bidirectional sliding-window protocol employed in the publicly available GNU UUCP file transfer utility. The Concurrency Factory's local model checker for the modal mu-calculus was used to verify the presence of the livelock in an instance of the protocol having a window size of 1, exploring about 1.7×10^5 states out of a total estimated global state space of 2.749×10^{14}. A simple inductive argument shows that the same error is present in the protocol for all larger window sizes.

Key to the Factory's success was the use of an abstraction to reduce the message sequence number space from 32 — the constant defined in the protocol's C-code — to $2W$, where W is the window size. This abstraction is shown to preserve the truthhood of all modal mu-calculus formulæ.

A performance comparison with the SPIN and SMV model checkers has also been carried out. Indeed, the i-protocol appears well-suited as a benchmark for verification tools aimed at real-life concurrent systems software.

*Research supported in part by NSF grant CCR-9505562, and AFOSR grants F49620-95-1-0508 and F49620-96-1-0087.

1 Introduction

Concurrent systems, in general, and communication protocols, in particular, are notoriously difficult to program, debug, and prove correct. Much of this difficulty stems from the large number of interleavings that a set of concurrently executing processes generate, resulting in the well-known *state space explosion* problem.

Model checking [CE81, CES86] has proved to be a particularly effective method for detecting bugs in the design of concurrent systems, and, to better cope with state explosion, a number of enhancements to the basic technique have been proposed; e.g., modular model checking [KV95], symbolic model checking [BC+90], local model checking [La92], partial model checking [HP94, Va93, GW91], compositional model checking [CLM89], the use of abstraction mappings [Wo86, DGG93], and the exploitation of symmetries [ES93].

In this abstract, we report on our experience in using the Concurrency Factory's model checking facility to detect and correct a non-trivial livelock in a communication protocol that forms part of a widely used set of communication tools. The Concurrency Factory [CG+94] is a joint project between the State University of New York at Stony Brook and North Carolina State University to develop an integrated toolset for the specification, verification, and implementation of concurrent and distributed systems. Like the Concurrency Workbench [CPS93], the Factory employs bisimulation, preorder, and model checking as its main avenues of analysis.

The protocol that we investigate, the i-protocol, is part of the GNU UUCP package, available from the Free Software Foundation, and is used for file transfers over serial lines, such as telephone lines. The i-protocol is part of a protocol stack; its purpose is to ensure ordered reliable duplex communication between two sites. At its lower interface, the i-protocol assumes unreliable (lossy) packet-based FIFO connectivity. To its upper interface, it provides *reliable* packet-based FIFO service. The GNU UUCP package also contains the g- and j-protocols, which are variants of the i-protocol.

A problem with the i-protocol was first noticed by Stark, while trying to transfer large files from a remote computer to his home PC over a modem line. In particular, under certain message loss conditions, the protocol would enter a "confused" state and drop the connection. In order to diagnose this problem, we extracted an abstract version of the i-protocol from its source code,[1] consisting of approximately 1500 lines of C-code, formalizing it in a Value Passing Language (VPL). VPL is the textual specification language of the Concurrency Factory and is derived from Milner's $\mathcal{M}_0$ [Mi89].

The VPL source of the i-protocol was then subjected to a series of model checking experiments using the Concurrency Factory's local model checker for the modal mu-calculus. This led us to the source of the problem: a livelock that occurs when a particular series of message losses drives the protocol into a state where the communicating parties enter into a cycle of fruitless message exchanges without any packets being delivered to the upper layer entities. Seeing no progress, the two sides close the connection, which must then be reestablished. If

[1]The source code is freely available from GNU software repositories; see, for instance, `prep.ai.mit.edu:/pub/gnu`.

the communication line (or virtual circuit) is sufficiently noisy, or if one of the sides is slow in emptying communication buffers, say due to disk waits, leading to buffer overflows, the chances of this scenario recurring are high.

Using the diagnostic facility that accompanies the Factory's model checker, we were able to pinpoint and subsequently "patch" the bug in the VPL code. The fix to the protocol consists of a fairly simple change in the way negative acknowledgements are handled by the protocol.

The verification was particularly challenging because of the large theoretical state space of the protocol. Indeed, most exhaustive verifiers, or global model checkers, would be inadequate for this verification, since the state space of the protocol, even for a window size of 1, is approximately 2.8×10^{14}. The Concurrency Factory's model checker was, however, able to detect and diagnose the livelock, since it is a *local* model checker. In effect, it exploits the structure of the formula expressing the property of interest to guide and limit its search to only the "relevant" portions of the global state space.

In addition, some judicious abstractions were used to reduce the size of the state space, retaining in our model only as much detail as appeared necessary for finding the bug. While many of these abstractions have been verified only in an informal sense, we verified formally the soundness of an abstraction that reduced the message sequence number space from 32 — the constant defined in the protocol's C-code — to $2W$, where W is the window size. This abstraction preserved the truth of all modal mu-calculus formulæ, and reduced the state space of the protocol by several orders of magnitude.

We also compare the performance of the Concurrency Factory on the i-protocol with the SPIN [Ho93] and SMV [Mc92] model checkers. Indeed, the i-protocol appears well-suited as a benchmark for verification tools aimed at real-life concurrent systems software.

References

[BC+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, L.J. Hwang, Symbolic model checking: 10^{20} states and beyond, in: *Proc. 5th IEEE LICS* (1990) 428-439.

[CE81] E. M. Clarke and E. A. Emerson, Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic, LNCS 131, 1981.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla, Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications, *ACM Trans. Prog. Lang. Syst.*, 8(2), 1986.

[CLM89] E.M. Clarke, D.E. Long, K.L. McMillan, Compositional Model Checking, in: *Proc. 4th IEEE LICS*, 1989.

[CG+94] R. Cleaveland, J. N. Gada, P. M. Lewis, S. A. Smolka, O. V. Sokolsky, S. Zhang, The Concurrency Factory – practical tools for specification, simulation, verification and implementation of concurrent systems, in *Proceedings of the DIMACS*

Workshop on Specification Techniques for Concurrent Systems, Princeton, NJ, 1994.

[CPS93] R. Cleaveland, J. Parrow, and B. Steffen, The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems, *ACM TOPLAS*, 15(1), 1993.

[DGG93] D. Dams and O. Grumberg, R. Gerth, Generation of reduced models for checking fragments of CTL, in *Proc. 5th CAV* (1993) LNCS 697.

[ES93] E.A. Emerson, A.P. Sistla, Symmetry and model checking, in: *Proc. 5th CAV* (1993) LNCS 697.

[GW91] P. Godefroid, P. Wolper, A partial approach to model checking, in: *Proc. 6th IEEE LICS* (1991) 406-415.

[Ho93] G.J. Holzmann, Design and validation of protocols: a tutorial, *Computer Networks and ISDN Systems* **25** (1993) 981-1017.

[HP94] G.J. Holzmann, D. Peled, An improvement in formal verification, in: *Proc. FORTE* (1994).

[KV95] O. Kupferman, M.Y. Vardi, On the complexity of branching modular model checking, in: Proc. 6th Conf. CONCUR (1995) LNCS 962, 408-422.

[La92] K.G. Larsen, Efficient local correctness checking, in: *Proc. 4th CAV* (1992) LNCS 663.

[Mc92] K. McMillan, Symbolic model checking — an approach to the state explosion problem, Ph.D. Thesis, School of Computer Science, Carnegie Mellon University (1992)

[Mi89] R. Milner, *Communication and concurrency.* International Series in Computer Science. Prentice Hall, 1989.

[Va93] A. Valmari, On-the-fly verification of stubborn sets, in: *Proc. 5th CAV* (1993), LNCS 697, 397-408.

[Wo86] P. Wolper, Expressing interesting properties of programs in propositional temporal logic, in: *Proc. 13th ACM POPL* (1986).