DECIDING LIVENESS FOR SPECIAL CLASSES OF COMMUNICATING FINITE STATE MACHINES

Chung-Kuo Chang, Mohamed G. Gouda and Louis E. Rosier

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

1. Introduction

Consider a network of two finite state machines that communicate exclusively by exchanging messages via two one-directional, error-free, unbounded, FIFO channels. Each machine has a finite number of states and state transitions. Each state transition in a machine is accompanied either by sending one message via the machine's output channel or by receiving one message from its input channel.

Networks of communicating finite state machines are useful in modeling[4], analysis[3,7], and synthesis[6,8,15] of communication protocols and distributed systems. The *analysis problem* for these networks can be stated as follows. "Characterize a class of these networks, and construct an efficient algorithm that can decide for any network in the given class whether its communication satisfies some desirable properties." Most of the work to solve this problem has concentrated so far on properties such as boundedness[2,12,14], freedom of deadlocks[2,12,13], and freedom from unspecified receptions[2,12]. These are all safety properties[8]; i.e. they merely guarantee that nothing bad will happen during the course of communication.

In this paper we solve the same problem for liveness properties of such networks. In particular, we characterize two classes of networks (Class 1-k networks whose communication is known to be bounded by some constant k in one direction, and Class 2 networks, where one machine sends one type of message), and construct efficient algorithms to decide three liveness properties for any network in these classes. Actually, each of the three liveness properties can be stated as requiring that some node in a given machine in the network under consideration be reached infinitely often during the course of communication under some fairness assumption. We characterize three graduated fairness assumptions that lead to three degrees of liveness: weak liveness, liveness, and strong liveness as defined in [6].

Following the introduction, the paper is organized as follows: Networks of communicating finite state machines are defined in Section 2. The liveness properties of these networks are presented in Section 3. Then, efficient algorithms in q where message g is the head message in x_i (y_i), and if u has an outgoing receiving edge, labelled +g, then edge e must occur infinitely often in q.

A sequence q of a network (M,N) is called *strongly fair* iff the following condition is satisfied: For any node u in M or N, if u occurs infinitely often in q, then each outgoing edge of u must occur infinitely often in q.

Let (M,N) be a network. A node u, in M or N, is said to be *weakly live* (*live*, *strongly live*, respectively) in (M,N) iff it occurs infinitely often in every strongly fair (fair, weakly fair, respectively) sequence of (M,N).

The following theorem states that node liveness for networks of communicating finite state machines are undecidable in general.

Theorem 1: It is undecidable whether a node u in an arbitrary network (M,N) is weakly live (live or strongly live).

In the next two sections, we discuss procedures to decide for any Class 1-k or Class 2 network whether a given node is weakly live (live or strongly live). But first, we present a result that will prove useful later.

Let q be a sequence of (M,N). Then, q_M (q_N) is the sequence of moves (i.e. edges) made by M (N) in q. Two sequences q and q' of (M,N) are said to *correspond* iff $q_M = q'_M$ and $q_N = q'_N$.

Let R' be the set of all sequences q of (M,N) such that if a state [u,v,x,y] is in q, then |y|=0 or 1. Also, let R" be the set of all sequences q of (M,N) such that if a state [u,v,x,y] is in q where |y|>1, then no subsequent receive move is made by N in q. The following lemma was essentially proved in [12].

Lemma 1: A sequence q is in R iff a corresponding sequence is in R' U R".

It follows that a node u, in M or N, is weakly live (live, strongly live, respectively) iff it occurs infinitely often in every strongly fair (fair, weakly fair, respectively) sequence in R' U R".

4. Deciding Liveness for Class 1-k Networks

In this section, let (M,N) be an arbitrary Class 1-k network, and let R, R' and R" be defined with respect to (M,N) as in the previous section.

Lemma 2: There is no weakly fair, fair, or strongly fair sequence of (M,N) in R". (The proof is by contradiction.)

The following lemma follows immediately from Lemmas 1 and 2.

Lemma 3: A node in M or N is weakly live (live, strongly live, respectively) in (M,N) iff it occurs infinitely often in every strongly fair (fair, weakly fair, respectively) sequence in R'.

The proof of the following lemma is based on the fact that the number of distinct states in each sequence in R' is bounded by a polynomial in 2^k, m and n, where m (n) is the number of nodes in machine M (N).

Lemma 4: If there exists a weakly fair (fair, strongly fair) sequence in R' in which a node u occurs at most a finite number of times, then there exists another such sequence q such that the sequence of moves (i.e. edges) made by M and N in q is of the form $w_1(w_2)^w$ where both $|w_1|$ and $|w_2|$ are bounded by a polynomial in 2^k, m and n.

From Lemma 4, it is useful to distinguish between the two cases of whether the constant k is a parameter of the problem or is a known fixed value. For example, since all the algorithms to decide weak liveness, liveness and strong liveness in this section run in time exponential in k, the next theorem indicates that this probably cannot be improved in the case where k is a parameter rather than a known fixed value.

Theorem 2: Deciding weak liveness, liveness, and strong liveness for Class 1-k networks is PSPACE-Complete provided that k is a parameter rather than a known fixed value.

Proof: The hardness proof is a reduction from the recognition problem for linear bounded automata and is similar in style to the PSPACE-hardness results given in [12]. The fact that the problem is doable in PSPACE follows directly from Lemma 4.

If k is, however, considered to be a fixed value, the problem's complexity needs to be analyzed with respect to parameters m and n; this is somewhat more involved. If a node u is not weakly live (live, strongly live, respectively) in (M,N), then there must exist a strongly fair (fair, weakly fair, respectively) sequence (as described in Lemma 4) in which u occurs only finitely often. Our approach is to determine if such a sequence exists. In order to do this, we need to categorize, in some sense, the possible sequences q mentioned in Lemma 4. This is done basically by categorizing the possible choices for w_2 with respect to each type of fairness. The categorization is as follows:

weakly fair: If a sequence q whose sequence of moves is of the form $w_1(w_2)^w$ is weakly fair, then w_2 must contain at least one edge from M and one edge from N; i.e. w_2 can be categorized by this pair of edges. (Note that w_2 may belong to one or more categories.)

fair or strongly fair: If a sequence q whose sequence of moves is of the form $w_1(w_2)^w$ is fair (strongly fair), then w_2 can be characterized by the "set" of nodes and edges visited in M and N.

In other words, each of the possible sequences q in Lemma 4 is characterized by a set of nodes and edges in M and N, that satisfies some conditions depending on whether we are interested in weakly fair, fair, or strongly fair sequences. We thus call these sets *solution categories*. Notice that for weakly fair sequences, there are O(m*n) different solution categories, but for fair or strongly fair sequences, there are an exponential (in m and n) number of different solution categories.

Lemma 5: There is an algorithm, that runs in nondeterministic logspace, to check for any given solution category whether there is a sequence that belongs to this category.

Thus, an algorithm to check whether a node is not weakly live (live or strongly live) would be to test each solution category to see whether there is a strongly fair (fair or weakly fair) sequence that satisfies Lemma 4. Consequently, we have almost immediately:

Theorem 3: There is an algorithm, that runs in nondeterministic logspace, to decide whether a node in a Class 1-k network is "not" strongly live, provided that k is a known fixed value. Moreover, the problem is complete for this complexity class.

The theorem immediately yields that strong liveness for Class 1-k networks is co-complete for nondeterministic logspace, and so is decidable in polynomial time.

For the case of fair or strongly fair sequences the problem becomes somewhat harder since there are an exponential number of solution categories to check. In order to solve this problem, in polynomial time, we need to develop fast ways to avoid looking at large numbers of candidate categories. This can be done for strongly fair sequences.

Theorem 4: There is a polynomial time algorithm to decide whether a node in a Class 1-k network is weakly live, provided that k is a known fixed value.

The proof involves showing that the nodes and edges in a solution category for strongly fair sequences must constitute strongly connected components of M and N. Since the strongly connected components of M and N can be found in O(m + n) time[1], the result follows.

We do not know, at the present time, whether a corresponding result holds for deciding whether a node is not live.

5. Deciding Liveness for Class 2 Networks

In this section, we describe algorithms that decide whether a node is *not* weakly live (live or strongly live) in a Class 2 network. For the ensuing discussion, let (M,N) be an arbitrary Class 2 network, and let u be an arbitrary node in M or N.

The results in this section are derived from similar (although more involved) arguments as those presented in the previous section. In particular, Lemmas 4 and 5 can be appropriately modified for Class 2 networks, but it is more difficult than it was in Section 4.

Theorem 5: There is an algorithm, that runs in nondeterministic logspace, to decide whether a node in a Class 2 network is "not" strongly live.

Theorem 6: There is a polynomial time algorithm to decide whether a node in a Class 2 network is weakly live.

Once again we do not know, at the present time, whether a result corresponding to Theorem 6 holds for deciding whether a node is live. We can, however, show a direct correspondence between this problem for Class 2 and Class 1-k networks.

References

[1] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, Mass., 1974.

[2] Brand, D. and P. Zafiropulo, "On communicating finite-state machines," *Journal ACM*, Vol. 30, No. 2, April 1983, pp. 323-342.

[3] Bochmann, G. V., "Finite state description of communication protocols," *Computer Networks*, Vol. 2, 1978, pp. 361-371.

[4] Bochmann, G. V. and C. Sunshine, "Formal methods in communication protocol design," *IEEE Trans. on Commun.*, April 1980, pp. 624-631.

[5] Chow, C. H., M. G. Gouda, and S. S. Lam, "An exercise in constructing multi-phase communication protocols," in *Proc. of SIGCOMM'84 Symposium*, June 1984. pp. 42-49.

[6] Gouda, M. G. and C. K. Chang, "A technique for proving liveness of communicating finite state machines with examples," in *Proc. of the 3rd Annual ACM Symp. on Principles of Distributed Computing*, Aug. 1984, pp. 38-49.

[7] Gouda, M. G. and Y. T. Yu, "Protocol validation by maximal progress state exploration," *IEEE Trans. on Commun.*, Vol. COM-32. No. 1, Jan. 1984, pp. 94-97.

[8] Gouda, M. G. and Y. T. Yu, "Synthesis of communicating finite state machines with guaranteed progress", *IEEE Trans. on Commun.*, Vol. COM-32. No. 7, July 1984, pp. 779-788.

[9] Hailpern, B. T. and S. S. Owicki, "Modular verification of computer communication protocols," *IEEE Trans. on Commun.*, Vol. COM-31, No. 1, Jan. 1983, pp. 56-68.

[10] Misra, J., K. M. Chandy and T. Smith, "Proving safety and liveness of communicating processes with examples," in *Proc. of ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, Aug. 1982, pp. 18-20.

[11] Pnueli, A., "The temporal semantics of concurrent programs," *Theoretical Computer Science*, Vol. 13, 1981, pp. 45-60.

[12] Rosier, L.E. and M. G. Gouda, "Deciding progress for a class of communicating finite state machines," in *Proc. of the Eighteenth Annual Conference on Information Sciences and Systems*, March 1984, Princeton, New Jersey.

[13] Yu, Y. T. and M. G. Gouda, "Deadlock detection for a class of communicating finite state machines," *IEEE Trans. on Commun.*, Dec. 1982, pp. 2514-2519.

[14] Yu, Y. T. and M. G. Gouda, "Unboundedness detection for a class of communicating finite-state machines," *Information Processing Letters*, Vol. 17, Dec. 1983, pp. 235-240.

[15] Zafiropulo, P., C. H. West, H. Rudin, D. Brand, and D. Cowan, "Towards analyzing and synthesizing protocols," *IEEE Trans. on Communn.*, Vol. COM-28, No. 4, April 1980, pp. 651-661.