

Decidability results of Communicating Finite State Machines over acyclic topology

G. Namratha Reddy

Chennai Mathematical Institute, India

<http://www.cmi.ac.in/~namratha>

Abstract

Models of concurrent finite state processes that communicate over queues allow different reachability results based on the underlying topology. The reachability is known to be undecidable for a general topology, but when we restrict to acyclic topologies it becomes decidable. The paper provides a proof of how it can be decided by reducing the problem to checking emptiness of a certain regular language.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases Distributed Systems, Reachability, Finite state machines with FIFO queues, Network of concurrent processes

1 Introduction

Distributed systems are becoming more and more popular. The ability to achieve with multiple cores/ computers for efficiency, reliability is highly enticing. Applications like block chain, database replication, ...

These concurrent nature of these applications/programs makes their verification challenging. Techniques to solve this problem are in high demand.

Formal methods uses mathematical formalism makes it easier to show that certain algorithms are not possible, maybe by showing that they are undecidable/decidable (atleast theoretically) which provides a motivation to develop efficient algorithms that can be used for practical purposes.

Communicating finite state processes (cite paper) over queues has been a standard model for modelling concurrent non-recursive programs.

One correctness idea for a concurrent program is to define via a safety property. Which states that a bad state should never be reached in my program. Which translates to asking in the model when we start with the initial configuration is it possible to reach this bad state. The answer to this question depends on the model we have considered. It turns out that for the communication finite state processes with queues it is undecidable to know if a state is reachable and this is shown because it is turing powerful (cite paper). But when restricted to certain class of topologies it allows for decidability.

2 System topology

2.1 Model

Topology

We work with a topology which we describe using the tuple

$$\mathcal{T} = (\mathcal{P}, \mathcal{C}, r, w),$$

where

\mathcal{P} is the set of processes;

\mathcal{C} is the set of channels;

42 $r : \mathcal{C} \rightarrow \mathcal{P}$ is the reader function that assigns to each channel a process that can read
 43 from it;
 44 $w : \mathcal{C} \rightarrow \mathcal{P}$ is the writer function that assigns to each channel a process that can write
 45 to it.

46 Channel

47 Each channel $c \in \mathcal{C}$ has a message alphabet Γ_c .
 48 Operations on the channel:
 49 ■ $c!m$ is a send operation. It appends the message m to the end of channel c
 50 ■ $c?m$ is a read operation. It receives the message m from the head of channel c (enabled
 51 only if m is at the head of channel c)

52 Process

53 Each process $p \in \mathcal{P}$ is modelled as a finite state transtion system

$$54 \quad TS_p = (Q_p, \delta_p, s_p),$$

55 where

56 Q_p is a finite set of states;
 57 $\delta_p \subseteq Q_p \times O_p \times Q_p$ is a finite set of transitions, where,
 58 $O_p = \{c?m \mid r(c) = p, m \in \Gamma_c\} \cup \{c!m \mid w(c) = p, m \in \Gamma_c\};$
 59 s_p is the initial state.

60 2.2 Configuration Graph

61 A configuration (α, β) of the system is a pair describing the states of all processes (control
 62 state) and the contents of all channels(channel state).

$$63 \quad \alpha \in \prod_{i \in \mathcal{P}} Q_i, \beta \in \prod_{c \in \mathcal{C}} \Gamma_c^*$$

64 For any $p \in \mathcal{P}$, We write $\alpha(p)$ gives the state of process p , and for any $c \in \mathcal{C}$, $\beta(c)$ gives
 65 the content of channel c . $\alpha[p \leftarrow q]$ to be the control state α' such that $\alpha'(p) = q$, and
 66 $\alpha'(p') = \alpha(p')$ if $p' \neq p$. Similarly, we define $\beta[c \leftarrow w] = \beta'$ such that $\beta'(c) = w$, and
 67 $\beta'(c') = \beta(c')$ if $c' \neq c$

68
 69 Vertices in the configuration graph $(G = (V, E))$ are configurations, and the edges are
 70 labelled with channel operations. The edges need to satisfy the following conditions.

71 Let $(\alpha, \beta), (\alpha', \beta') \in V, c \in \mathcal{C}$, then

72 A global configuration of the system would consist of the states of each process and the
 73 channel contents of each channel. So if we have n processes and m channels, a configuration
 74 would be the tuple $(q_0, q_1, \dots, q_n, \gamma_1, \gamma_2, \dots, \gamma_m)$

75 that is when a transition from one tuple to the next happens what all should align in the
 76 universe.

77 A run is a path in this graph

78 3 The Reachability Problem

79 The reachability problem is to ask whether we can reach a target configuration where one or
 80 more processes are in a target state.

We know that the problem is undecidable in general (cite), because if there is a loop in the topology then we can simulate a queue machine and state reachability is undecidable for queue machines.

So we ask the question of whether it is decidable for acyclic topologies.

What we mean by acyclic topologies is consider the network topology and ignore the direction of the edges, so we get an undirected graph and this graph should have no cycles.

We solve this with the help of two reductions

Reduction 1

given such an acyclic topology the reachability problem can be reduced to one in which the target state is reached only if the queue is empty [1]

Reduction 2

We reduce it to another isomorphic topology that looks like a tree, where every process has one incoming edge (except the root) i.e every process can read from one channel but write to multiple channels [1]

► **Lemma 1.** *Given a directed tree topology with $r=0$ as the root of the tree $w \in L_r^e \implies (s_0, s_1, \dots, s_n, w \downarrow_{r(0)}, \epsilon, \dots, \epsilon) \rightarrow_G^* (f_0, f_1, \dots, f_n, \epsilon, \epsilon, \dots, \epsilon)$*

Proof. Induction on the no. of nodes in the directed tree topology

Base case: We have only one node r , $L_r^e = \epsilon \implies \epsilon \in L_r \implies s_r = f_r$, clearly since we start in s_r it is reachable

Induction: Let r be a non-leaf node and let the children be k_1, k_2, \dots, k_m , $w \in L_r^e \implies w \in L_i \cap shufffle(L_{k_1}^e \downarrow_r, L_{k_2}^e \downarrow_r, \dots, L_{k_m}^e \downarrow_r, M_{r(i)}^*)$ w is of the form $shufffle(w_1, w_2, \dots, w_m)$ where $w_i \in L_{k_i}^e \downarrow_{r(i)}$

$w_i \downarrow_{r(k_i)} = w'_i \downarrow_{r(k_i)}$ where $w'_i \in L_{k_i}^e$

Let π_i be the tree that is rooted by k_i By induction hypothesis, since $w'_i \in L_{k_i}^e$ we have run from nodes in π_i are in start state, and reader channel of k_i has $w'_i \downarrow_{r(k_i)}$ to a final configuration where every node is in the final state and all channels are empty.

Since w is in L_r^e it means it is also in L_r we can take this run and convert to the corresponding run in the configuration graph where only r is moving, and this w can generate $w'_i \downarrow_{r(k_i)}$ in each k_i 's reader channel, so we can start from $(s_0, s_1, \dots, s_n, \epsilon, \dots) \rightarrow$ (only r moves and fills up all the channels) $\rightarrow (f_0, s_1, \dots, s_n, \epsilon, w_i \downarrow_{r(k_i)}, \dots, w_i \downarrow_{r(k_i)}) \rightarrow$ *stitch the k_i runs one after the other*

► **Lemma 2.** *Given a directed tree topology with $i=0$ as the root of the tree.*

$(s_0, s_1, \dots, s_n, \alpha, \epsilon, \dots, \epsilon) \rightarrow_G^* (f_0, f_1, \dots, f_n, \epsilon, \epsilon, \dots, \epsilon) \implies \exists w \in L_i^e, \alpha \downarrow_{r(0)} = w \downarrow_{r(0)}$

Proof. Base case : There is only one node α has to be epsilon

Induction: Let r be a non-leaf node, and k_1, k_2, \dots, k_n be the children of the r and let the tree rooted by them be $\pi_1, \pi_2, \dots, \pi_n$

Let r run and fill all the reader channels with α_i For each k_i we form a run from where only nodes in π_i are in start state and α_i is in the reader channel of k_i , there and final state is reached. Now we use induction hypothesis to say that there is a word in $w_i \in L_{k_i}^e$ and

$\alpha_i \downarrow_{r(0)} = w_i \downarrow_{r(0)}$

So we find a w_i for each k_i

so we have $w_i \downarrow_{r(i)} \in L_i^e$ so we have $\alpha_i \in L_i^e$

124 Consider the word $w \in \text{shuffle}(\alpha_1, \alpha_2, \dots, \alpha_n, \alpha)$ walsobelongsto L_r
 125 $w \in L_r^e$ and $w \downarrow_{r(r)} = \alpha$
 126

127 ► **Theorem 3.** *The Reachability problem is decidable for CFMs with FIFO channels over*
 128 *undirected topology.*

129 **Proof.** We use reductions 1 and 2 to get a tree topology
 130

131 4 Conclusions

132 Morbi eros magna, vestibulum non posuere non, porta eu quam. Maecenas vitae orci risus,
 133 eget imperdiet mauris. Donec massa mauris, pellentesque vel lobortis eu, molestie ac turpis.
 134 Sed condimentum convallis dolor, a dignissim est ultrices eu. Donec consectetur volutpat
 135 eros, et ornare dui ultricies id. Vivamus eu augue eget dolor euismod ultrices et sit amet nisi.
 136 Vivamus malesuada leo ac leo ullamcorper tempor. Donec justo mi, tempor vitae aliquet non,
 137 faucibus eu lacus. Donec dictum gravida neque, non porta turpis imperdiet eget. Curabitur
 138 quis euismod ligula.

139 — References —

- 140 1 La Torre, S., Madhusudan, P., Parlato, G. (2008). Context-Bounded Analysis of Concurrent
 141 Queue Systems. In: Ramakrishnan, C.R., Rehof, J. (eds) Tools and Algorithms for the
 142 Construction and Analysis of Systems. TACAS 2008. Lecture Notes in Computer Science, vol
 143 4963. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-78800-3_21