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 decidablity.

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;

$r : \mathcal{C} \to \mathcal{P}$ is the reader function that assigns to each channel a process that can read from it;

$w : \mathcal{C} \to \mathcal{P}$ is the writer function that assigns to each channel a process that can write to it.

Channel

Each channel $c \in \mathcal{C}$ has a message alphabet Γ_c. We assume channel alphabets are disjoint. Operations on the channel:

- $c!m$ is a send operation. It appends the message m to the end of channel c
- $c?m$ is a read operation. It receives the message m from the head of channel c (enabled only if m is at the head of channel c)

Process

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

$$TS_p = (Q_p, \Delta_p, s_p),$$

where

Q_p is a finite set of states;

$\Delta_p \subseteq Q_p \times O_p \times Q_p$ is a finite set of transitions, where,

$O_p = \{c?m \mid r(c) = p, m \in \Gamma_c\} \cup \{c!m \mid w(c) = p, m \in \Gamma_c\}$;

s_p is the initial state.

2.2 Configuration Graph

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

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

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 the content of channel c. $\alpha[p \leftarrow q]$ to be the control state α' such that $\alpha'(p) = q$, and $\alpha'(p') = \alpha(p')$ if $p' \neq p$. Similarly, we define $\beta[c \leftarrow w] = \beta'$ such that $\beta'(c) = w$, and $\beta'(c') = \beta(c')$ if $c' \neq c$

Vertices in the configuration graph ($G = (V, E)$) are configurations, and the edges are labelled with channel operations. The edges need to satisfy the following conditions. Let $(\alpha, \beta), (\alpha', \beta') \in V, c \in \mathcal{C}$, then

- $(\alpha, \beta) \xrightarrow{c!m} (\alpha', \beta')$ if
  - $\alpha' = \alpha[w(c) \leftarrow \alpha'(w(c))]$
  - $(\alpha(w(c)), c!m, \alpha'(w(c))) \in \delta_{w(c)}$
  - $\beta' = \beta[c \leftarrow \beta(c).m]$
- $(\alpha, \beta) \xrightarrow{c?m} (\alpha', \beta')$ if
  - $\alpha' = \alpha[r(c) \leftarrow \alpha'(r(c))]$
  - $(\alpha(r(c)), c?m, \alpha'(r(c))) \in \delta_{r(c)}$
  - $\beta = \beta'[c \leftarrow m.\beta'(c)]$

A run is a path in this graph. $c_1 \xrightarrow[G]{*} c_2$ represents a run starting in c_1 and ending in c_2.

3 The Reachability Problem

A target control state α_t is said to be reachable if there is a run $c_0 \xrightarrow{*} c_t$, where, $c_0 = (\alpha_0, \beta_0)$, $\alpha_0(p) = s_p \; \forall p \in \mathcal{P}$, and $\beta_0(c) = \epsilon \; \forall c \in \mathcal{C}$; $c_t = (\alpha_t, \beta)$ for some β

The reachability problem asks whether a given target control state is reachable. 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

Reductions

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. 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]

Languages

Given the target control state α_t to be reached, we define for all $p \in \mathcal{P}$ the language $L_p = L(A_p)$. We get the finite automation A_p by taking the transition system TS_p and replacing the opertion in the transition with just the message and having final state as $\alpha_t(p)$. $A_p = (Q_p, \Sigma_p, \delta_p, s_p, \alpha_t(p))$, where, $\Sigma_p = \bigcup\limits_{c \in \mathcal{C}} \{\Gamma_c : r(c) = p \text{ or } w(c) = p\}$, $\delta_p = \{(q, m, q') : (q, c?m, q') \in \Delta_p \text{ or } (q, c!m, q') \in \Delta_p\}$

$\pi_c(w)$ is the word we get by deleting all letter not in Γ_c. This can be extended to languages as well i.e. $\pi_c(L) = \{\pi_c(w) : w \in L\}$.

TODO later: write a note and motivation Notice that for A_p to accept a word w which has letters from $r(p)$, the word $\pi_{r(c)}(w)$ needs to be available in the $r(c)$ channel.

We define for all $p \in \mathcal{P}$, L_P^e as follows :

- if p is a leaf node, $L_p^e = L_p \cap \Gamma^*_{\iota(p)}$, where $\iota(p)$ gives the channel that p reads from.
- if p is a non-leaf node, and children of p is given by the set $k(p) = \{k_1, k_2, .., k_l\}$, $L_p^e = L_p \cap \text{shuffle}(\{\pi_{\iota(i)}(L_i^e) : i \in k(p)\} \cup \{\Gamma^*_{\iota(p)}\})$

TODO Describe what Gp is, describe we want to reach alphat and betat where betat is epsilon, write somewhere that we think root has this dummy reading channel, define shuffle of sets and shuffle of words

▶ **Lemma 1.** *Let p be the root of the tree, then*

$$w \in L_p^e \implies (\alpha_0, \beta_0[\iota(p) \leftarrow \pi_{\iota(p)}(w)]) \xrightarrow[G_p]{*} (\alpha'_t, \beta_t),$$

where $\alpha'_t(p) = \alpha_t(p)$ for $p \in V_p$ and $\alpha'_t(p) = \alpha_0(p)$ for $p \notin V_p$

Proof. By induction on the no. of nodes in the tree graph G_p.

Base case: No. of nodes is 1. Let this node be p. $L_p^e = L_p \cap \Gamma^*_{\iota(p)}$. $w \in L_p^e \implies w \in L_p$, so there is a run $s_p \xrightarrow[A_p]{w} \alpha_t(p)$. Since p is a leaf node we have $\pi_{\iota(p)}(w) = w$. Therefore, in G_p we get the run $(\langle s_p \rangle, \langle \pi_{\iota(p)}(w) \rangle) \xrightarrow[G_p]{*} (\langle \alpha_t(p) \rangle, \langle \epsilon \rangle)$

Induction: Let p be a non-leaf node and children of p is given by the set $k(p) = \{k_1, k_2, .., k_l\}$ We have $w \in L_p \cap \text{shuffle}(\{\pi_{\iota(i)}(L_i^e) : i \in k(p)\} \cup \{\Gamma^*_{\iota(p)}\}) \implies w \in shuffle(w_1, .., w_l, \pi_{\iota(p)}(w))$ where $w_i \in \pi_{\iota(i)}(L_i^e)$. By induction hypothesis we have run $\rho_i : (\alpha_0, \beta_0[\iota(i) \leftarrow w_i]) \xrightarrow[G_i]{*} (\alpha'_t, \beta_t)$ for each $i \in k(p)$. Since $w \in L_p$ we have the run ρ: $s_p \xrightarrow[A_p]{*} \alpha_t(p)$. Using ρ and the runs for each i, ρ_i, we can construct a run in G_p as follows: $(\alpha_0, \beta_0[p \leftarrow \pi_{\iota(p)}(w)]) \xrightarrow[\text{p moves}]{*} (\alpha_0[p \leftarrow \alpha_t(p)], \beta_0[\{i \leftarrow w_i : i \in k(p)\}]) \xrightarrow[k_1 \text{ moves}]{*} \cdots \xrightarrow[k_l \text{ moves}]{*} (\alpha'_t, \beta_t)$ ◀

▶ **Lemma 2.** *Let p be the root of the tree, then*

$$(\alpha_0, \beta_0[\iota(p) \leftarrow \omega]) \xrightarrow[G_p]{*} (\alpha'_t, \beta_t) \implies \exists w \in L_p^e \text{ such that } \pi_{\iota(p)}(w) = \omega,$$

where $\alpha'_t(p) = \alpha_t(p)$ *for* $p \in V_p$ *and* $\alpha'_t(p) = \alpha_0(p)$ *for* $p \notin V_p$

Proof. Induction on the no. of nodes in the tree.
Base case: No. of nodes is 1. Let this node be p. We have $(\langle s_p \rangle, \langle \omega \rangle) \xrightarrow[G_p]{*} (\langle \alpha_t(p) \rangle, \langle \epsilon \rangle)$ implies we have the run $s_p \xrightarrow[A_p]{\omega} \alpha_t(p)$ implies $\omega \in L_p^e$ and we can take w as ω itself since $\pi_{\iota(p)}(\omega) = \omega$.

Induction: Let p be a non-leaf node and children of p is given by the set $k(p) = \{k_1, k_2, .., k_l\}$. We have the run $\rho : (\alpha_0, \beta_0[\iota(p) \leftarrow \omega]) \xrightarrow[G_p]{*} (\alpha'_t, \beta_t)$. We can construct another run ρ' in which all the p transitions are taken first and no other process moves until p finishes. Once p finishes let channel state be β' where $\beta'(\iota(k_i)) = \omega_i$ and the rest of the channels have ϵ.

Take the corresponding run in A_p and let that be $s_p \xrightarrow[A_p]{w'} \alpha_t(p)$

We can extract the following run from ρ' by taking only transitions of V_{k_i}

$(\alpha, \beta_0[\iota(k_i) \leftarrow \omega_i]) \xrightarrow[\text{only nodes in } V_{k_i} \text{ move}]{G_{k_i}} (\alpha'_t, \beta_t)$. By induction hypothesis we have some word $w_i \in L_{k_i}^e$ and $\omega_i = \pi_{\iota(k_i)}(w_i) \implies \omega_i \in \pi_{\iota(k_i)} L_{\iota(k_i)}^e$

We have $w' \in shuffle(\omega_1, ..., \omega_l, \omega) \implies w' \in L_p^e$ and $\pi_{\iota(p)}(w') = \omega$

◀

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

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

◀

4 Conclusions

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

References

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