INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

Decidability of communicating topologies

Adwait Godbole, Sriram Balasubramanian
November 11, 2019

1 INTRODUCTION

1.1 SOME DEFINITIONS

Definition 1.1. A *labelled transition system* is a tuple $\mathscr{A} = \langle S, S_I, S_F, A, \rightarrow \rangle$ where, S is a set of states, S_I and S_F are initial and final state sets, A is the finite set of actions and $\rightarrow \subseteq S \times A \times S$ is a labelled transition relation.

Definition 1.2. A *communication topology* is a tuple $\mathscr{T} = \langle P, C, M, \mathsf{src}, \mathsf{dst}, \mathsf{msg} \rangle$ where P is a finite set of communicating processes, C is a finite set of channels, M is the (finite) message alphabet, src and dst are functions from $C \rightarrow P$ marking the source and the destination for each channel and msg is the function that assigns to each C its message alphabet (a subset of M).

Definition 1.3. A *system of communicating processes* is a pair $\mathscr{S} = \langle \mathscr{T}, \{\mathscr{A}^p\}_{p \in P} \rangle$ of a topology and its constituent processes.

1.2 CHANNEL TYPES - BAG, FIFO, LOSSY

1. **Bag** channel is one where the messages do not have any order. The messages on the channel can simply be seen as a multiset.
2. **FIFO** channel is one where the channel is a FIFO queue with total order on the messages per channel (this will also be called a reliable channel).
3. **Lossy** channel is one where messages can be non-deterministically dropped from the channel.

Depending upon the type of channels, we can get decidability or undecidability. We will see how these can be effectively characterized.

2 A FEW PRELIMINARY RESULTS

Lemma 2.1 (Decidability of Bag-*only* systems)**.** *A communicating system where all channels are Bag type channels is decidable*

Proof. We can convert this problem into a Petri Net reachability problem. Note that with the order on messages removed, each channel can be viewed as a multiset of messages.

1. For each channel-message pair (c, m), construct a place s_m^c in the Petri-Net storing tokens corresponding to the number of such m type messages in c.
2. The control state can be modelled with additional places in the Petri net.
3. For action a corresponding to $c?m$ ($c!m$) is translated into a transition t with s_m^c being one of the input (output) places; the others being those for the control state.

□

A channel c is said to be a *unary* channel if $|\mathsf{msg}(c)| = 1$. Since, unary channels are just special cases of Bag channels, we have the following corollary.

Corollary 2.1.1 (Decidability of *unary*-channel systems)**.** *A communicating system where all channels are unary is decidable*

Lemma 2.2 (Decidability of FIFO-only channel systems)**.** *A communicating system where all channels are perfect FIFOs is decidable if and only if the topology is a polyforest, i.e, a directed acyclic graph whose underlying undirected graph is a tree.*

Proof. If is proved later. Only if is as follows:

1. We prove that reachability in all (two) topologies consisting of two processes which have a cycle in the underlying undirected graph is undecidable
2. Proceed by induction to prove undecidability for topologies with an n length cycle if the cycle has two consecutive channels in the same direction
3. If no two consecutive channels in the n length cycle are in the same direction, prove by simulating PCP.

□

3 FIFO + LOSSY CHANNEL SYSTEMS

3.1 SOME BASIC SYSTEMS

We will first look at some basic systems to get an intuitive understanding.

Consider the systems in Figure 3.1. We have seen that state reachability for T_1^d is decidable. In fact, so is T_2^d!

In order to show this, we consider the following variant(s) of Post's Correspondence Problem. We then will encode T_2^d as an instance of this problem.

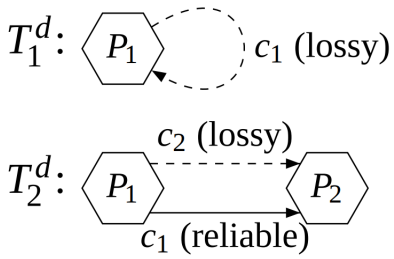


Figure 3.1: Two simple systems

Problem (Post's Embedding Problem (PEP))**.** For two finite alphabets Σ and Γ, and two morphisms $u, v : \Sigma^* \to \Gamma^*$, does there exist a $\sigma \in \Sigma^+$ such that $u(\sigma) \sqsubseteq v(\sigma)$?

However, PEP is too simple a problem to encode T_2^d in. The decidability of PEP is trivial. This statement rests on the following (easy) lemma.

Fact (Simple decomposition lemma)**.** If $u \cdot w \sqsubseteq v \cdot t$, then either $u \sqsubseteq v$ or $w \sqsubseteq t$.

Then, if there is a σ such that $u(\sigma) \sqsubseteq v(\sigma)$, there must be a $i \in \Sigma$ such that, $u(i) \sqsubseteq v(i)$. So, we look at a more powerful version of PEP, called as PEP$^r eg$.

Problem (Regular Post's Embedding Problem (PEP^{reg}))**.** For two finite alphabets Σ and Γ, and two morphisms $u, v : \Sigma^* \to \Gamma^*$, and a regular language $R \subseteq \Sigma^*$, does there exist a $\sigma \in R$ such that $u(\sigma) \sqsubseteq v(\sigma)$?

This problem is decidable as well. For a proof, we refer the reader to [1].

Lemma 3.1 (Decidability of T_2^d)**.** *The control-state reachability of T_2^d is decidable*

Proof. Let the FIFO and lossy channels be denoted by f and l respectively. Also let $\alpha(\delta)$ and $\beta(\delta)$ denote the messages read or written in transition δ. Let Δ_1 and Δ_2 denote the transitions of the sender and the receiver respectively. We will encode the problem as a PEP^{reg} instance. Unidirectionality allows for reordering of actions such that first transitions in Δ_1 are performed followed by those in Δ_2. Define the homomorphisms u and v as follows

1. $u(\delta) = \beta(\delta)$ for $\delta \in \Delta_2$ and ϵ otherwise
2. $v(\delta) = \alpha(\delta)$ for $\delta \in \Delta_1$ and ϵ otherwise

Let R_1 and R_2 capture the accepting state reachability of sender and receiver, i.e. s_0. This is basically the language accepted by P_1 and P_2 when viewed as finite state automata. R_3 captures language where writes on channel f are immediately matched by a read, i.e. $R_3 = \{\delta\gamma | \delta\gamma \in \Delta_1\Delta_2$ with $\alpha(\delta) = \alpha(\gamma)\}$. Let $R = (R_1 \bowtie R_2) \cap R_3^*$. Now, we claim that the PEP^{reg} instance corresponding to (u, v, R) encodes T_2^d. This is easy to see as R captures all the runs where P_1 and P_2 reach the final state while writing and reading the same word to the reliable channel. Moreover if we have a run $\sigma \in R$, $u(\sigma) \sqsubseteq v(\sigma)$ ensures that the word read from the lossy channel is a subword of that which was written, precisely capturing the lossy channel constraint. □

3.2 Some Undecidable Cases

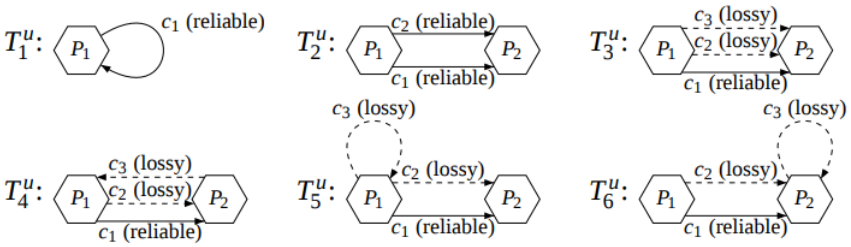


Figure 3.2: Undecidable Topologies

Proof for all of these is by PCP or its directed version PCP^{dir} with some nice tricks.

The decidable and undecidable cases, seen earlier are important as, we will reduce more complicated systems to these while giving the complete characterization for decidability.

3.3 Complete Characterisation

We start by some additional transformations and definitions.

1. $T - c$ is the topology obtained by deleting channel c
2. $T[P_1{=}P_2]$ is the topology obtained by combining processes P_1 and P_2
3. T/c is the topology obtained by contracting channel c

Definition 3.1. A channel c is *essential* if all directed paths from $src(c)$ to $dst(c)$ contain c. In particular, $src(c) \neq dst(c)$.

Theorem 3.2 (Synchronization of essential channels)**.** *If c is an essential channel, then every run that starts and ends with c empty is causal-equivalent to a run that is synchronous for c.*

A formal proof of this theorem is in [2], page 17. We provide a proof sketch here. For some intuition please see 4.1

Proof sketch. The proof is by induction on length of runs.

Base case: Length of run is 0 is a trivial case.

Inductive case: If length of run is n, and first action does not write on c, we are done. Otherwise, the run is of the form $\rho = x_1 \xrightarrow{\text{c!m}} x_1' \cdot \rho_1 \cdot x_2 \xrightarrow{\text{c?m}} x_2' \cdot \rho_2$. Let $p = src(c)$, $q = dst(c)$. Then, we can reorder $x_1 \xrightarrow{\text{c!m}} x_1' \cdot \rho_1$ to $z_1 \cdot x_1 \xrightarrow{\text{c!m}} x_1' \cdot z_2$ where z_1 does not contain any action in p and z_2 does not contain any action in q, because the channel c is essential. Therefore, $z_2 \cdot x_2 \xrightarrow{\text{c?m}} x_2'$ can be reordered into $x_2 \xrightarrow{\text{c?m}} x_2' \cdot z_2$. Now, send and receive of m is synchronized. Applying induction hypothesis on ρ_2, we get the result. □

Using Theorem 1, we can prove that effective channels can be replaced using two bag channels without affecting reachability and other important properties.

We can now revisit the characterization of FIFO-only channel systems, in particular the converse of Lemma 2.2.

DECIDABILITY OF POLYFOREST TOPOLOGIES In a polyforest, all channels are essential, or else there would be a cycle in their undirected graphs. Therefore, all channels can be synchronized and replaced by two bags. Reachability of all bag systems is decidable, and therefore, reachability in polyforest systems is decidable.

Now we look at a crucial topology transformation.

3.3.1 DECIDABILITY BY FUSION

Lemma 3.3 (Decidability by Fusion)**.** *Let c be an essential channel in T. Then*

1. *T has decidable reachability if T/c has decidable reachability*
2. *If in addition c is a reliable channel then T/c has decidable reachability if T has decidable reachability*

Proof sketch. This is in fact a consequence of Theorem 1. Consider S as the system with topology T/c. Due to Theorem 1, S only needs to simulate runs where c is synchronous. Hence the channel states can be assimilated into the state corresponding to P_1 an P_2 and the result follows due to decidability of S. □

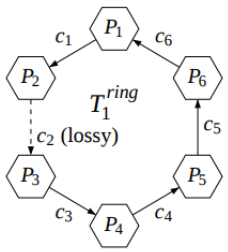


Figure 3.3: What can we say about control state reachability of T_1^{ring}?

SOME EXAMPLES OF FUSION T_1^{ring} (see Figure 3.3) is decidable. We can fuse all the reliable channels! The topology we get is identical to T_1^d, which is decidable.

T_2^{ring} (see Figure 3.4) is undecidable since we can only fuse any four out of the five reliable channels, since the last one does not remain essential. Then we have a topology similar to T_4^u which is undecidable. T_3^{ring} is decidable since we can fuse c_1, c_3, c_4, c_6 to get an all lossy-channel system, which we know is decidable.

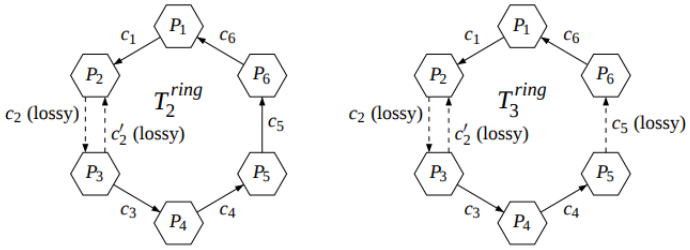


Figure 3.4: What can we say about control state reachability of T_2^{ring} and T_3^{ring}?

3.4 Decidability by Lossy Splitting

Definition 3.2 (Lossy Splitting)**.** A topology T that can be split into T_1 and T_2 by deleting unidirectional lossy channels between T_1 and T_2 is denoted as $T_1 \triangleright T_2$.

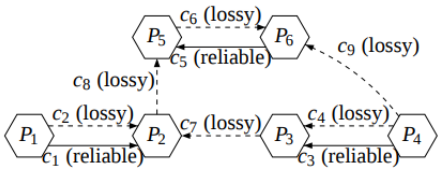


Figure 3.5: Example of lossy splitting - a topology that can be split into 3 components ($T_2 \triangleright (T_1 \triangleright T_3)$)

Lemma 3.4 (Decidability by Splitting)**.** *Reachability is decidable for $T_1 \triangleright T_2$ iff it is decidable for both T_1 and T_2*

3.5 Complete Characterisation of FIFO + Lossy Systems

Theorem 3.5 (**Main Theorem 1** Complete Characterisation Theorem)**.** *A network topology with FIFO + Lossy channels is decidable iff it can be reduced to T_2^d or LCS using fusion and splitting only*

Proof idea. The 'if' direction follows from the decidability by fusion and splitting that we saw earlier. The 'only if' direct requires an involved case analysis on the irreducible topology. The main idea is as follows.

1. Either all channels are lossy (in which case we have decidability).
2. Or there exists a reliable channel c_r linking some nodes A and B.
3. There must also exist an alternative path θ with atleast one lossy channel, else we could reduce the toplogy further. Then we can show that the system is equivalent either to T_2^d or one of these systems in Figure 3.2.

□

4 FIFO + BAG CHANNELS

We now deal with the case when the system of communicating processes have both FIFO channels and bag channels. To give a full characterization of decidable topologies, we first describe three main techniques used in the reduction of any topology to a known decidable (or undecidable) topology. The last of these, fusion, is a variant of that seen in the earlier section.

1. **Synchronization**: Replace a FIFO from p to q by two channels, one from p to q and the other from q to p. The communication on these two channels is *synchronized*, as the sender waits for an acknowledgement (ACK) from the receive before sending another message.
2. **Splitting**: Redirect a bag channel from p to q to a new process r, with a new bag channel from q to r. The reads of q are now writes on the new channel. The new process just matches messages of both channels.
3. **Fusion**: We can fuse a strongly connected component of processes into a single process. The SCC can simulate the new process, so $Reach(\mathscr{U}) \preceq Reach(\mathscr{T})$, where $\mathscr{U}$ is obtained after fusion from $\mathscr{T}$

4.1 SYNCHRONIZATION EXPLAINED

A channel c is *essential* if all directed paths from $src(c)$ to $dst(c)$ contain c. In particular, $src(c) \neq dst(c)$. We can prove that if c is an essential channel, then every run that starts and ends with c empty is causal-equivalent to a run that is synchronous for c.

The formal proof is in [2], page 17. The intuition behind the proof is as follows. Given a run on the original system, consider the first write on c by $p = src(c)$ and the first read on c by $q = dst(c)$. Any other writes on c after the first write but before the first read, and all other actions which depend on these writes, can be pushed *after* the first read by q. This is because c is essential, so the first read by q cannot depend on these extra writes, because the only path of information flow from p to q is through c.

4.2 SPLITTING EXPLAINED

We need the following concepts before we can describe when splitting works.

1. **Reversibility**: A channel c is *reversible* if there is a directed path from its destination $dst(c)$ to its source $src(c)$. A channel is *irreversible* if it is not reversible.
2. **Causal run**: A run $(x_0, a_1, x_1, \ldots a_n, x_n)$ is *causal* for a given process p if there is a directed path from q to p for every process q such that $a_i \in A_q$ and $a_j \in A_p$ for some $1 \leq i < j \leq n$. Here, the x_i's are the macrostates and a_i's are the actions done in the transitions between macrostates.

Given a process p, we can prove that every run is causal-equivalent to a run that is causal for p. The intuition here is that given any action a on a process q which does not have a directed path to p, there is no causal reason why a must occur before any action done on p, as a cannot influence p or those processes dependent on actions done by p.

Theorem 4.1 (**Main theorem 2**). *If b is an irreversible channel in $\mathcal{T}$, then it holds that $Reach(\mathcal{T}) \preceq Reach(\mathcal{U})$ where $\mathcal{U}$ results from the split of b in $\mathcal{T}$.*

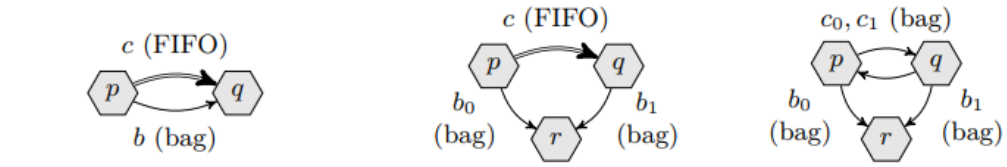
Proof sketch. The formal proof is in [3], page 19. The intuition is as follows. Let the source of original bag channel b be p and destination q. In the new construction, we have a new process r and channels b_1 from p to r and b_2 from q to r in place of b. Consider any run in this new topology, and transform it into a causally-equivalent run which is causal for p. In this run, all actions of q occur after all actions of p, since there is no directed path from q to p. This can be easily translated to a run in the original topology. □

4.3 Fusion

Consider a set of strongly connected set of processes, S. This implies that there is a directed cycle connecting all processes in S together. These processes can be fused into a single process/automaton where all incoming channels to any one of the fused processes are directed to the new process, similarly for outgoing channels. The original topology can simulate any system of processes with the new topology as follows.

1. Duplicate the automaton of the new process obtained after fusion in all automata in the strongly connected component.
2. Replace actions which cannot be performed by any process due to unavailability of channels with a dummy do-nothing action.
3. When any process takes a transition, it sends a message along the directed cycle instructing all other automata to undertake the same transition.

We illustrate the above ideas through these examples.



(a) Parallel channels b and c (b) Splitting b (c) Synchronising c

Figure 4.1: Some Two Channel Topologies

In the above example, we can split the bag channel into two with an additional synchronization process r that matches the incoming tokens from b_0 and b_1 (Figure 4.1b). Now, the FIFO channel c is essential and hence communication can be synchronized on this channel. Therefore, this topology is decidable.

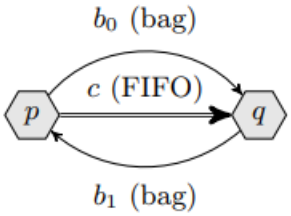


Figure 4.2: Another FIFO + Bag Topology

The topology in Figure 4.2 is undecidable, even when b_0 and b_1 are unary! This is because we can synchronize p and q and fuse them into a single automaton, which can then simulate a self-loop FIFO system.

4.4 Characterization of Decidability

We define the following terms.

1. Two processes p and q are *synchronizable* over D, written $p \approx_D q$, if there exists a directed path from p to q and a directed path from q to p, both using only channels in D.
2. A *jumping circuit* is a sequence $(p_0, c_1, q_1, p_1, \ldots, c_n, q_n, p_n)$ of processes $p_i, q_i \in P$ and channels $c_i \in C$, with $n \geq 1$, such that $c_1, \ldots, c_n$ are pairwise distinct non-unary channels, $p_0 = p_n$, and $p_{i-1} \overset{c_i}{=\!=} q_i \approx_D p_i$ for all $1 \leq i \leq n$, where $D = C \setminus \{c_1, \ldots, c_n\}$.
3. A *jumping cycle* is a jumping circuit such that $q_i \not\approx_D q_j$ for all $1 \leq i < j \leq n$.

We can immediately make the following observations:

1. Every jumping circuit can be transformed into a jumping cycle.
2. For every jumping cycle $p_0 \overset{c_0}{=\!=} q_1 \approx_D p_1 \ldots \overset{c_n}{=\!=} q_n \approx_D p_n$, there exist n pairwise disjoint subsets $D_1, \ldots, D_n$ of the set $D = C \setminus \{c_1, \ldots, c_n\}$ such that $q_i \approx_{D_i} p_i$ for all $1 \leq i \leq n$. Here, $a{=\!=}b$ means that $a \to b$ or $b \to a$

Proof. Proof left as exercise for the reader. □

Theorem 4.2 (Characterization of Decidability)**.** *Given a topology $\mathscr{T}$, $Reach(\mathscr{T})$ is decidable if, and only if, $\mathscr{T}$ has no jumping cycle.*

Proof. The idea is as follows. $\Leftarrow$ (proof of undecidability)

1. We fuse the set of supports (the D_is) for the directed paths between q_i and p_i one by one.
2. Just verify that after each step, the other D_is still act as supports for their respective channels, and at the end, we have a closed loop of FIFO channels.

$\Rightarrow$ (proof of decidability)

First, we introduce the notion of divided topology. A topology $\mathcal{T}$ is *divided* if the destination of every irreversible unary channel is a sink (i.e., is not the source of some channel) and is not the destination of some non-unary channel.

We then consider the following three lemmata.

Lemma 4.3. *Consider a topology $\mathcal{T}$ and an essential non-unary channel c therein. Let $\mathcal{U}$ be the topology obtained from $\mathcal{T}$ by adding an acknowledgement channel for $c(\overleftarrow{c})$. Then $\mathcal{T}$ contains a jumping cycle if U contains a jumping cycle.*

Lemma 4.4. *Consider a topology $\mathcal{T}$ and a non-unary channel c therein. If $\mathcal{T}$ is divided, then so is the topology resulting from the synchronisation of c in $\mathcal{T}$.*

Lemma 4.5. *If $\mathcal{T}$ is a divided topology with no jumping cycle, then every non-unary channel in $\mathcal{T}$ is essential.*

The intuition behind the above lemmata is as follows.

1. Assume $\mathcal{T}$ does not have jumping cycle but $\mathcal{U}$ does. If $\overleftarrow{c}$ is a part of the directed path which synchronizes p_i and q_i, then since c is essential, c is also a part of the directed path. This means that we can find a synchronization between q_i and $src(c)$ and $dst(c)$ and p_i. This means there exists a jumping cycle in $\mathcal{T}$.
2. Need to prove that the new channel does not destroy dividedness. Can be done after some casework.
3. Assume that some c is not essential. This means that there exists a directed path π from p to q that does not contain c. We can show that c is irreversible, else there would be jumping cycle. We can break down π into a sequence of reversible and irreversible channels, so $\pi = \chi_0 \cdot p_0 \Rightarrow q_1 \cdot \chi_1 \ldots p_{n-1} \Rightarrow q_n \cdot \chi_n$. All irreversible channels are non-unary, or else the destinations of these channels should be sinks, and χ_n cannot be 0 length, since q is the destination of non-unary channel c. A jumping circuit is obtained by using the channel c to complete the circuit.

We can now prove the decidability part of the main theorem, by showing that each of the following steps holds.

1. First, split all irreversible unary channels to get a divided topology
2. In a divided topology, by Lemma 4.5, all non-unary channels are essential.
3. We can synchronise each non-unary channel one by one to get a topology consisting only of unary channels, which is decidable.

Hence we are done. □

5 Conclusion

We thus have studied various topologies and the differences that arise due to different types of channels. We can provide complete characterizations of the decidability of control state reachabiity of these systems based on the structural propeties of their topologies.

REFERENCES

[1] CHAMBART, P., AND SCHNOEBELEN, P. Post embedding problem is not primitive recursive, with applications to channel systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science* (2007), Springer, pp. 265–276.

[2] CHAMBART, P., AND SCHNOEBELEN, P. Mixing lossy and perfect fifo channels. In *International Conference on Concurrency Theory* (2008), Springer, pp. 340–355.

[3] CLEMENTE, L., HERBRETEAU, F., AND SUTRE, G. Decidable topologies for communicating automata with fifo and bag channels. In *International Conference on Concurrency Theory* (2014), Springer, pp. 281–296.