Phase-Bounded Broadcast Networks over Topologies of Communication

Lucie Guillou
IRIF, CNRS, Université Paris Cité, France

Arnaud Sangnier
DIBRIS, Università di Genova, Italy

Nathalie Sznajder
LIP6, CNRS, Sorbonne Université, Paris, France

Abstract

We study networks of processes that all execute the same finite state protocol and that communicate through broadcasts. The processes are organized in a graph (a *topology*) and only the neighbors of a process in this graph can receive its broadcasts. The coverability problem asks, given a protocol and a state of the protocol, whether there is a topology for the processes such that one of them (at least) reaches the given state. This problem is undecidable [6]. We study here an under-approximation of the problem where processes alternate a bounded number of times k between phases of broadcasting and phases of receiving messages. We show that, if the problem remains undecidable when k is greater than 6, it becomes decidable for $k = 2$, and EXPSPACE-complete for $k = 1$. Furthermore, we show that if we restrict ourselves to line topologies, the problem is in P for $k = 1$ and $k = 2$.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Parameterized verification, Coverability, Broadcast Networks

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2024.26

Related Version *Long Version*: `https://arxiv.org/abs/2406.15202` [15]

Funding *Lucie Guillou*: ANR project PaVeDyS (ANR-23-CE48-0005)

1 Introduction

Verifying networks with an unbounded number of entities. Ensuring safety properties for concurrent and distributed systems is a challenging task, since all possible interleavings must be taken into account; hence, even if each entity has a finite state behavior, the verification procedure has to deal with the state explosion problem. Another level of difficulty arises when dealing with distributed protocols designed for an unbounded number of entities. In that case, the safety verification problem consists in ensuring the safety of the system, for any number of participants. Here, the difficulty comes from the infinite number of possible instantiations of the network. In their seminal paper [13], German and Sistla propose a formal model to represent and analyze such networks: in this work, all the processes in the network execute the same protocol, given by a finite state automaton, and they communicate thanks to pairwise synchronized rendez-vous. The authors study the parameterized coverability problem, which asks whether there exists an initial number of processes that allow an execution leading to a configuration in which (at least) one process is in an error state (here the parameter is the number of processes). They show that it is decidable in polynomial time. Later on, different variations of this model have been considered, by modifying the communication means: token-passing mechanism [1,5], communication through shared register [8,11], non-blocking rendez-vous mechanism [14], or adding a broadcast mechanism to send a message to all the entities [9]. The model of population protocol proposed in [2] and for which verification methods have been developed recently in [10,12] belongs also to this family of systems. In this latter model, the properties studied are different, and more complex than safety conditions.

Broadcast networks working over graphs. In [6], Delzanno et. al propose a new model of parameterized network in which each process communicates *with its neighbors* by broadcasting messages. The neighbors of an entity are given thanks to a graph: the *communication topology*. This model was inspired by ad hoc networks, where nodes communicate with each other thanks to radio communication. The difficulty in proving safety properties for this new model lies in the fact that one has to show that the network is safe for all possible numbers of processes and all possible communication topologies. So the verification procedure not only looks for the number of entities, but also for a graph representing the relationship of the neighbours to show unsafe execution. As mentioned earlier, it is not the first work to propose a parameterized network with broadcast communication; indeed the parameterized coverability problem in networks with broadcast is decidable [9] and non-primitive recursive [24] when the communication topology is complete (each entity is a neighbor of all the others). However, when there is no restriction on the allowed communication topologies the problem becomes undecidable [6] but decidability can be regained by providing a bound on the length of all simple paths in allowed topologies [6]. This restriction has then been extended in [7] to allow also cliques in the model. However, with this restriction, the complexity of parameterized coverability is non-primitive recursive [7].

Bounding the number of phases. When dealing with infinite-state systems with an undecidable safety verification problem, one option consists in looking at under-approximations of the global behavior, restricting the attention to a subset of executions. If proving whether the considered subset of executions is safe is a decidable problem, this technique leads to a sound but incomplete method for safety verification. Good under-approximation candidates are the ones that can be extended automatically to increase the allowed behavior. For instance, it is known that safety verification of finite systems equipped with integer variables that can be incremented, decremented, or tested to zero is undecidable [19], but if one considers only executions in which, for each counter, the number of times the execution alternates between an increasing mode and a decreasing mode is bounded by a given value, then safety verification becomes decidable [16]. Similarly, verifying concurrent programs manipulating stacks is undecidable [22] but decidability can be regained by bounding the number of allowed context switches (a context being a consecutive sequence of transitions performed by the same thread) [20]. Context-bounded analysis has also been applied to concurrent programs with stacks and dynamic creation of threads [3]. Another type of underapproximation analysis has been conducted by [17] (and by [4] in another context), by considering bounded round-robin schedules of processes. Inspired by this work, we propose here to look at executions of broadcast networks over communication topologies where, for each process, the number of alternations between phases where it broadcasts messages and phases where it receives messages is bounded. We call such protocols k-phase-bounded protocols where k is the allowed number of alternations.

Our contributions. We study the parameterized coverability problem for broadcast networks working over communication topologies. We first show in Section 2 that it is enough to consider only tree topologies. This allows us to ease our presentation in the sequel and is also an interesting result by itself. In Section 3, we prove that the coverability problem

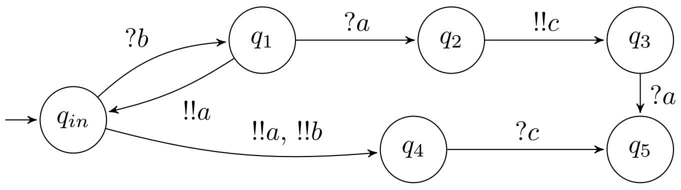


Figure 1 Example of a broadcast protocol denoted P.

is still undecidable when considering k-phase-bounded broadcast protocols with k greater than 6. The undecidability proof relies on a technical reduction from the halting problem for two counter Minsky machines. We then show in Sections 4 and 5 that if the number of alternations is smaller or equal to 2, then decidability can be regained. More precisely, we show that for 1-phase-bounded protocols, we can restrict our attention to tree topologies of height 1, which provides an ExpSpace-algorithm for the coverability problem. To solve this problem in the case of 2-phase-bounded protocols, we prove that we can bound the height of the considered tree and rely on the result of [6] which states that the coverability problem for broadcast networks is decidable when considering topologies where the length of all simple paths is bounded. We furthermore show that if we consider line topologies then the coverability problem restricted to 1- and 2-phase-bounded protocols can be solved in polynomial time.

Due to lack of space, omitted proofs and reasonings can be found in [15].

2 Preliminaries

Let A be a countable set, we denote A^* as the set of finite sequences of elements taken in A. Let $w \in A^*$, the length of w is defined as the number of elements in the sequence w and is denoted $|w|$. For a sequence $w = a_1 \cdot a_2 \cdots a_k \in A^+$, we denote by $w[-1]$ the sequence $a_1 \cdot a_2 \cdots a_{k-1}$. Let $\ell, n \in \mathbb{N}$ with $\ell \leq n$, we denote by $[\ell, n]$ the set of integers $\{\ell, \ell+1, \dots, n\}$.

2.1 Networks of processes

We study networks of processes where each process executes the same protocol given as a finite-state automaton. Given a finite set of messages Σ, a transition of the protocol can be labelled by three types of actions: (1) the broadcast of a message $m \in \Sigma$ with label $!!m$, (2) the reception of a message $m \in \Sigma$ with label $?m$ or (3) an internal action with a special label $\tau \notin \Sigma$. Processes are organised according to a topology which gives for each one of them its set of neighbors. When a process broadcasts a message $m \in \Sigma$, the only processes that can receive m are its neighbors, and the ones having an output action $?m$ have to receive it. Furthermore, the topology remains fixed during an execution.

Let Σ be a finite alphabet. In order to refer to the different types of actions, we write $!!\Sigma$ for the set $\{!!m \mid m \in \Sigma\}$ and $?\Sigma$ for $\{?m \mid m \in \Sigma\}$.

▶ **Definition 2.1.** *A* Broadcast Protocol *is a tuple $P = (Q, \Sigma, q_{in}, \Delta)$ such that Q is a finite set of states, Σ is a finite alphabet of messages, q_{in} is an initial state and $\Delta \subseteq Q \times (!!\Sigma \times ?\Sigma \cup \{\tau\}) \times Q$ is a finite set of transitions.*

We depict an example of a broadcast protocol in Figure 1. Processes are organised according to a topology, defined formally as follows.

▶ **Definition 2.2.** *A* topology *is an undirected graph, i.e. a tuple $\Gamma = (V, E)$ such that V is a finite set of vertices, and $E \subseteq V \times V$ is a finite set of edges such that $(u, v) \in E$ implies $(v, u) \in E$ for all $(u, v) \in V^2$, and for all $u \in V$, $(u, u) \notin E$ (there is no self-loop).*

We will use $\mathsf{V}(\Gamma)$ and $\mathsf{E}(\Gamma)$ to denote the set of vertices and edges of Γ respectively, namely V and E. For $v \in V$, we will denote $\mathsf{N}_\Gamma(v)$ the set $\{u \mid (v, u) \in E\}$. When the context is clear, we will write $\mathsf{N}(v)$. For $u, v \in \mathsf{V}(\Gamma)$, we denote $\langle v, u\rangle$ for the two pairs $(v, u), (u, v)$. We name Graphs the set of topologies. In this work, we will also be interested in some families of topologies: line and tree topologies. A topology $\Gamma = (V, E)$ is a *tree topology* if V is a set of words of $\mathbb{N}^*$ which is prefix closed with $\epsilon \in V$, and if $E = \{\langle w[-1], w\rangle \mid w \in V \cap \mathbb{N}^+\}$. This way, the *root* of the tree is the unique vertex $\epsilon \in V$ and a node $w \in V \cap \mathbb{N}^+$ has a unique parent $w[-1]$. The *height* of the tree is $\max\{n \in \mathbb{N} \mid |w| = n\}$. We denote by Trees the set of tree topologies. A topology $\Gamma = (V, E)$ is a *line topology* if V is such that $V = \{v_1, \ldots, v_n\}$ for some $n \in \mathbb{N}$ and $E = \{\langle v_i, v_{i+1}\rangle \mid 1 \leq i < n\}$. We denote by Lines the set of line topologies.

Semantics. A *configuration* C of a broadcast protocol $P = (Q, \Sigma, q_{in}, \Delta)$ is a tuple (Γ, L) where Γ is a topology, and $L : \mathsf{V}(\Gamma) \to Q$ is a labelling function associating to each vertex v of the topology its current state of the protocol. In the sequel, we will sometimes call processes or nodes the vertices of Γ. A configuration C is *initial* if $L(v) = q_{in}$ for all $v \in \mathsf{V}(\Gamma)$. We let $\mathcal{C}_P$ be the set of all configurations of P, and $\mathcal{I}_P$ the set of all initial configurations. When P is clear from the context, we may drop the subscript and simply use $\mathcal{C}$ and $\mathcal{I}$. Given a protocol $P = (Q, \Sigma, q_{in}, \Delta)$, and a state $q \in Q$, we let $R(q) = \{m \in \Sigma \mid \exists q' \in Q, (q, ?m, q') \in \Delta\}$ be the set of messages that can be received when in the state q.

Consider $\delta = (q, \alpha, q') \in \Delta$ a transition of P, and $C = (\Gamma, L)$ and $C' = (\Gamma', L')$ two configurations of P, and let $v \in \mathsf{V}(\Gamma)$ be a vertex. The transition relation $\xrightarrow{v,\delta} \in \mathcal{C} \times \mathcal{C}$ is defined as follows: we have $C \xrightarrow{v,\delta} C'$ if and only if $\Gamma = \Gamma'$, and one of the following conditions holds:

- $\alpha = \tau$ and $L(v) = q$, $L'(v) = q'$ and $L'(u) = L(u)$ for all $u \in \mathsf{V}(\Gamma) \setminus \{v\}$: vertex v performs an internal action;
- $\alpha = !!m$ and $L(v) = q$, $L'(v) = q'$ (vertex v performs a broadcast), and for each process $u \in \mathsf{N}(v)$ neighbor of v, either $(L(u), ?m, L'(u)) \in \Delta$ (vertex u receives message m from v), or $m \notin R(L(u))$ and $L(u) = L'(u)$ (vertex u is not in a state in which it can receive m and stays in the same state). Furthermore, $L'(w) = L(w)$ for all other vertices $w \in \mathsf{V}(\Gamma) \setminus (\{v\} \cup \mathsf{N}(v))$ (vertex w does not change state).

We write $C \to C'$ whenever there exists $v \in \mathsf{V}(\Gamma)$ and $\delta \in \Delta$ such that $C \xrightarrow{v,\delta} C'$. We denote by $\to^*$ [resp. $\to^+$] for the reflexive and transitive closure [resp. transitive] of $\to$. An *execution* of P is a sequence of configurations $C_0, \ldots, C_n \in \mathcal{C}_P$ such that for all $0 \leq i < n$, $C_i \to C_{i+1}$.

▶ **Example 2.3.** We depict in Figure 2 an execution of protocol P (from Figure 1): it starts with an initial configuration with three processes v_1, v_2, v_3, organised as a clique (each vertex is a neighbour of the two others), each on the initial state q_{in}. More formally, $\Gamma = (V, E)$ with $V = \{v_1, v_2, v_3\}$ and $E = \{\langle v_1, v_2\rangle, \langle v_2, v_3\rangle, \langle v_1, v_3\rangle\}$. From the initial configuration, the following chain of events happens: $C_0 \xrightarrow{v_1,(q_{in},!!b,q_4)} C_1 \xrightarrow{v_2,(q_1,!!a,q_{in})} C_2 \xrightarrow{v_3,(q_2,!!c,q_3)} C_3$.

2.2 Verification problem

In this work, we focus on the *coverability problem* which consists in ensuring a safety property: we want to check that, no matter the number of processes in the network, nor the topology in which the processes are organised, a specific error state can never be reached.

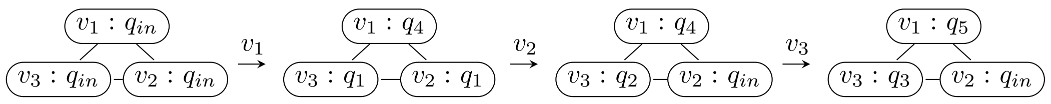


Figure 2 Example of an execution of protocol P (Figure 1).

The coverability problem over a family of topologies $\mathcal{S} \in \{\mathsf{Graphs}, \mathsf{Trees}, \mathsf{Lines}\}$ is stated as follows:

COVER[$\mathcal{S}$]	
Input:	A broadcast protocol P and a state $q_f \in Q$;
Question:	Is there $\Gamma \in \mathcal{S}$, $C = (\Gamma, L) \in \mathcal{I}_P$ and $C' = (\Gamma, L') \in \mathcal{C}_P$ and $v \in \mathsf{V}(\Gamma)$ such that $C \rightarrow^* C'$ and $L'(v) = q_f$?

For a family $\mathcal{S}$, if indeed there exist $C = (\Gamma, L)$ and $C' = (\Gamma, L')$ such that $C \rightarrow^* C'$ and $L'(v) = q_f$ for some $v \in \mathsf{V}(\Gamma)$, we say that q_f is *coverable* (in P) with Γ. We also say that the execution $C \rightarrow^* C'$ *covers* q_f. For short, we write COVER instead of COVER[Graphs]. Observe that COVER is a generalisation of COVER[Trees] which is itself a generalisation of COVER[Lines]. In [6], the authors proved that the three problems are undecidable, and they later showed in [7] that the undecidability of COVER still holds when restricting the problem to families of topologies with bounded diameter.

However, in [6], the authors show that COVER becomes decidable when searching for an execution covering q_f with a K-bounded path topology for some $K \in \mathbb{N}$, i.e. for a topology in which all simple paths between any pair of vertices $v_1, v_2 \in V$ have a length bounded by K. In [7], it is also shown that COVER is Ackermann-hard when searching for an execution covering q_f with a topology where all maximal cliques are connected by paths of bounded length. We establish the first result.

▶ **Theorem 2.4.** *COVER[Graphs] and COVER[Trees] are equivalent.*

Indeed, if it is obvious that when a state is coverable with a tree topology, it is coverable with a topology from Graphs, we can show that whenever a state is coverable, it is coverable with a tree topology. If a set q_f of a protocol P is coverable with a topology $\Gamma \in \mathsf{Graphs}$, let $\rho = C_0 \rightarrow \cdots \rightarrow C_n = (\Gamma, L_n)$ be an execution covering q_f, and a vertex $v_f \in \mathsf{V}(\Gamma)$ such that $L_n(v_f) = q_f$. We can build an execution covering q_f with a tree topology Γ' where the root reaches q_f. Actually, Γ' is the unfolding of Γ in a tree of height n.

3 Phase-Bounded Protocols

As COVER[Graphs], COVER[Trees] and COVER[Lines] are undecidable in the general case, we investigate a restriction on broadcast protocols: phase-bounded protocols.

For $k \in \mathbb{N}$, a k-phase-bounded protocol is a protocol that ensures that each process alternates at most k times between phases of broadcasts and phases of receptions. Before giving our formal definition of a phase-bounded protocol, we motivate this restriction.

Phase-bounded protocols can be seen as a semantic restriction of general protocols in which each process can only switch a bounded number of times between phases where it receives messages and phases where it broadcasts messages. When, usually, restricting the behavior of processes immediately yields an underapproximation of the reachable states, we highlight in [15] the fact that preventing messages from being received can in fact lead to new reachable states. Actually, the reception of a message is something that is not under

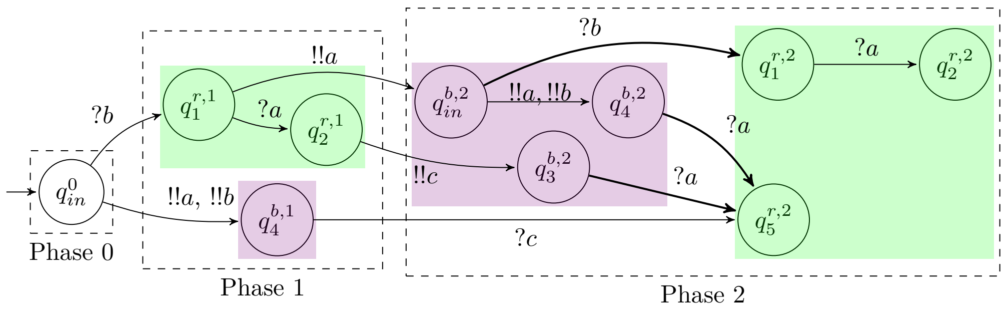


Figure 3 P_2: the 2-unfolding of protocol P (Figure 1).

the control of a process. If another process broadcasts a message, a faithful behavior of the system is that all the processes that can receive it indeed do so, no matter in which phase they are in their own execution. Hence, in a restriction that attempts to limit the number of switches between broadcasting and receiving phases, one should not prevent a reception to happen. This motivates our definition of phase-bounded protocols, in which a process in its last broadcasting phase, can still receive messages. A k-unfolding of a protocol P is then a protocol in which we duplicate the vertices by annotating them with the type and the number of phase (b or r for broadcast or reception and an integer between 0 and k for the number).

▶ **Example 3.1.** Figure 3 pictures the 2-unfolding of protocol P (Figure 1). Observe that from state $q_4^{b,2}$, which is a broadcast state, it is still possible to receive message a and go to state $q_5^{r,2}$. However, it is not possible to send a message from $q_5^{r,2}$ (nor from any reception state of phase 2).

We show in [15] that this definition of unfolding can be used as an underapproximation for Cover. In the remaining of the paper, we study the verification problems introduced in Section 2.2 when considering phase-bounded behaviors. We turn this restriction into a syntactic one over the protocol, defined as follows.

▶ **Definition 3.2.** *Let $k \in \mathbb{N}$. A broadcast protocol $P = (Q, \Sigma, q_{in}, \Delta)$ is* k-phase-bounded *if Q can be partitioned into $2k+1$ sets $\mathcal{Q} = \{Q_0, Q_1^b, Q_1^r, \dots Q_k^b, Q_k^r\}$, such that $q_{in} \in Q_0$ and for all $(q, \alpha, q') \in \Delta$ one of the following conditions holds:*

1. *there exist $0 \leq i \leq k$ and $\beta \in \{r, b\}$ such that $q, q' \in Q_i^\beta$ and $\alpha = \tau$ (for ease of notation, we take $Q_0 = Q_0^b = Q_0^r$);*
2. *there exists $1 \leq i \leq k$ such that $q, q' \in Q_i^b$ and $\alpha \in !!\Sigma$;*
3. *there exists $1 \leq i \leq k$ such that $q, q' \in Q_i^r$ and $\alpha \in ?\Sigma$;*
4. *there exists $0 \leq i < k$ such that $q \in Q_i^b$, $q' \in Q_{i+1}^r$ and $\alpha \in ?\Sigma$;*
5. *there exists $0 \leq i < k$ such that $q \in Q_i^r$, $q' \in Q_{i+1}^b$ and $\alpha \in !!\Sigma$;*
6. *$q \in Q_k^b$, $q' \in Q_k^r$ and $\alpha \in ?\Sigma$*

A protocol P is phase-bounded if there exists $k \in \mathbb{N}$ such that P is k-phase-bounded.

▶ **Example 3.3.** Observe that the protocol P displayed in Figure 1 is not phase-bounded: by definition, it holds that $Q_0 = \{q_{in}\}$, and $q_1 \in Q_1^r$ (because of the transition $(q_{in}, ?b, q_1)$). As a consequence $q_{in} \in Q_2^b$, because of the transition $(q_1, !!a, q_{in})$. This contradicts the fact that $Q_2^b \cap Q_0 = \emptyset$. Intuitively, P does not ensure that every vertex alternates at most a bounded number of times between receptions and broadcasts, in particular, for any integer $k \in \mathbb{N}$, it might be that there exists an execution where a process alternates $k+1$ times between reception of a message b from state q_{in}, and broadcast of a message a from state q_1. Removing the transition $(q_1, !!a, q_{in})$ from P would give a 2-phase-bounded protocol P': $Q_0 = \{q_{in}\}$, $Q_1^r = \{q_1, q_2\}$, $Q_1^b = \{q_4\}$, $Q_2^b = \{q_3\}$ and $Q_2^r = \{q_5\}$.

The following table summarizes our results (PB stands for phase-bounded).

	1-PB Protocols	**2-PB Protocols**	**PB Protocols**
COVER[Lines]	$\in P$ (Section 6.2)		Undecidable ($k \geq 4$) (Sec 4)
COVER[Graphs] COVER[Trees]	EXPSPACE-complete (Section 5)	Decidable (Section 6.1)	Undecidable ($k \geq 6$) (Section 4)

4 Undecidability Results

We prove that COVER restricted to k-phase-bounded protocols (with $k \geq 6$) is undecidable by a reduction from the halting problem of a Minksy machine [19]: a Minsky machine is a finite-state machine (whose states are called locations) with two counters, x_1 and x_2 (two variables that take their values in $\mathbb{N}$). Each transition of the machine is associated with an instruction: increment one of the counters, decrement one of the counters or test if one of the counters is equal to 0. The halting problem asks whether there is an execution that ends in the halting location. In a first step, the protocol will enforce the selection of a line of nodes from the topology. All other nodes will be inactive. In a second step, the first node of the line (that we call the head) visits the different states of the machine during an execution, while all other nodes (except the last one) simulate counters' values: they are either in a state representing value 0, or a state representing x_1 (respectively x_2). The number of processes on states representing x_1 gives the actual value of x_1 in the execution. The last node (called the tail) checks that everything happens as expected. When the head has reached the halting location of the machine, it broadcasts a message which is received and forwarded by each node of the line until the tail receives it and reaches the final state to cover.

When the head of the line simulates a transition of the machine, it broadcasts a message (the instruction for one of the counters), which is transmitted by each node of the line until the tail receives it. A classical way of forwarding the message through receptions and broadcasts would not give a phase-bounded protocol. Hence, during the transmission, the tail only receives messages and all other nodes only broadcast and do not receive any message. The main idea is that we do not use the reception of messages to move into the next state of the execution but to detect errors (and in that case, go to a bad sink state from which the process can not do anything). The processes will have to guess the correct message to send, and the correct instant to send it, otherwise some of them will go to the sink state upon the reception of this "wrong" message. Hence, when everyone makes the correct guesses, the only reception that occurs in the transmission is done by the tail process, whereas when someone makes an incorrect guess, a process goes to a bad state with a reception. In the reduction, if the halting state of the Minsky Machine is not reachable, there will be no way to make a correct guess that allows to cover the final state. In the next subsection, we explain how this is achieved. To do so, we explain the mechanism by abstracting away the actual instruction, and just show how to transmit a message.

4.1 Propagating a message using only broadcasts in a line

In a line, a node has at most two neighbors, but cannot necessarily distinguish between the two (its left and its right one). To do so, nodes broadcast messages with subscript 0, 1 or 2, and we ensure that: if a node broadcasts with subscript 1, its right [resp. left] neighbor broadcasts with subscript 0 [resp. subscript 2]. Similarly, if a node broadcasts with subscript

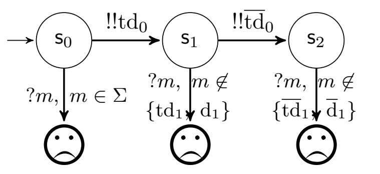

s0
s1
s2
!!td0
!!$\overline{\text{td}}_0$
?m, m ∈ Σ
?m, m ∉ {td1, d1}
?m, m ∉ {$\overline{\text{td}}_1$, $\overline{\text{d}}_1$}

Figure 4 Protocol P_h executed by v_0.

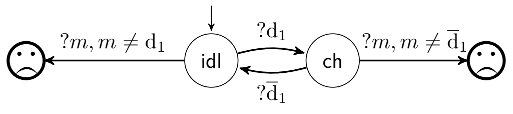


Figure 5 Protocol P_t executed by v_n.

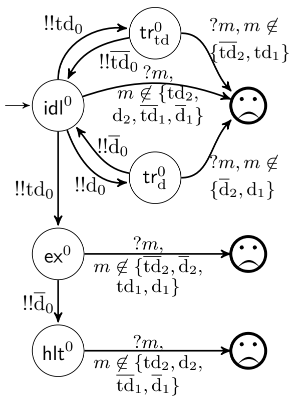

idl0
tr0td
tr0d
ex0
hlt0
!!td0
!!$\overline{\text{td}}_0$
!!d0
!!$\overline{\text{d}}_0$
?m, m ∉ {$\overline{\text{td}}_2$, td1}
?m, m ∉ {td2, d2, $\overline{\text{td}}_1$, $\overline{\text{d}}_1$}
?m, m ∉ {$\overline{\text{d}}_2$, d1}
?m, m ∉ {$\overline{\text{td}}_2$, $\overline{\text{d}}_2$, td1, d1}
?m, m ∉ {td2, d2, $\overline{\text{td}}_1$, $\overline{\text{d}}_1$}

Figure 6 P_0.

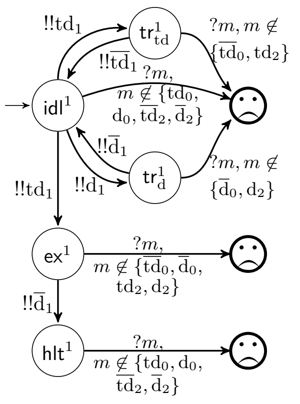

idl1
tr1td
tr1d
ex1
hlt1
!!td1
!!$\overline{\text{td}}_1$
!!d1
!!$\overline{\text{d}}_1$
?m, m ∉ {$\overline{\text{td}}_0$, td2}
?m, m ∉ {td0, d0, $\overline{\text{td}}_2$, $\overline{\text{d}}_2$}
?m, m ∉ {$\overline{\text{d}}_0$, d2}
?m, m ∉ {$\overline{\text{td}}_0$, $\overline{\text{d}}_0$, td2, d2}
?m, m ∉ {td0, d0, $\overline{\text{td}}_2$, $\overline{\text{d}}_2$}

Figure 7 P_1.

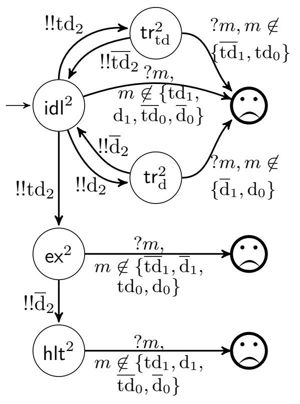

idl2
tr2td
tr2d
ex2
hlt2
!!td2
!!$\overline{\text{td}}_2$
!!d2
!!$\overline{\text{d}}_2$
?m, m ∉ {$\overline{\text{td}}_1$, td0}
?m, m ∉ {td1, d1, $\overline{\text{td}}_0$, $\overline{\text{d}}_0$}
?m, m ∉ {$\overline{\text{d}}_1$, d0}
?m, m ∉ {$\overline{\text{td}}_1$, $\overline{\text{d}}_1$, td0, d0}
?m, m ∉ {td1, d1, $\overline{\text{td}}_0$, $\overline{\text{d}}_0$}

Figure 8 P_2.

0 [resp. 2], its right neighbor broadcasts with subscript 2 [resp. 1] and its left one with subscript 1 [resp. 0].

Consider the five protocols displayed in Figures 4–8. The states marked as initial are the ones from which a process enters the protocol. Protocol P_h is executed by the head of the line, P_t by the tail of the line and other nodes execute either P_0, P_1 or P_2. Observe that messages go by pairs: $\mathrm{td}_i, \overline{\mathrm{td}}_i$ and d_i, $\overline{\mathrm{d}}_i$ for all $i \in \{0, 1, 2\}$.

The head broadcasts a request to be done with the pair of messages td_0, $\overline{\mathrm{td}}_0$. Each process in one of the P_i starts in idl^i and has a choice: either it transmits a message without executing it, or it "executes" it and tells it to the others. When it transmits a message not yet executed, it broadcasts the messages td_i and $\overline{\mathrm{td}}_i$ and visits states $\mathsf{tr}^i_{\mathrm{td}}$ and idl^i. When it executes the request, it broadcasts the messages td_i and $\overline{\mathrm{d}}_i$ and visits states ex^i and hlt^i. Finally, when it transmits a request already done, it broadcasts the messages d_i and $\overline{\mathrm{d}}_i$ and visits states $\mathsf{tr}^i_{\mathrm{d}}$ and idl^i. Once a process has executed the request (i.e. broadcast a pair td_j, $\overline{\mathrm{d}}_j$ for some $j \in \{0, 1, 2\}$), only pairs d_j, $\overline{\mathrm{d}}_j$, with $j \in \{0, 1, 2\}$, are transmitted in the rest of the line.

Correct transmission of a request. Take for instance the configuration C_0 depicted in Figure 9 for $n = 5$ (i.e. there are six vertices). We say that a configuration is *stable* if the head is in s_0 or s_2, the tail is in idl and other nodes are in idl^i or hlt^i for $i \in \{0, 1, 2\}$. Note that C_0 is stable. We depict a transmission in Figures 10a and 10b, starting from C_0. We denote the successive depicted configurations $C_0, C_1, \dots C_{11}$. Note that C_{11} is stable. Between C_0 and

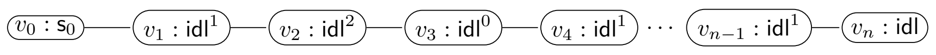
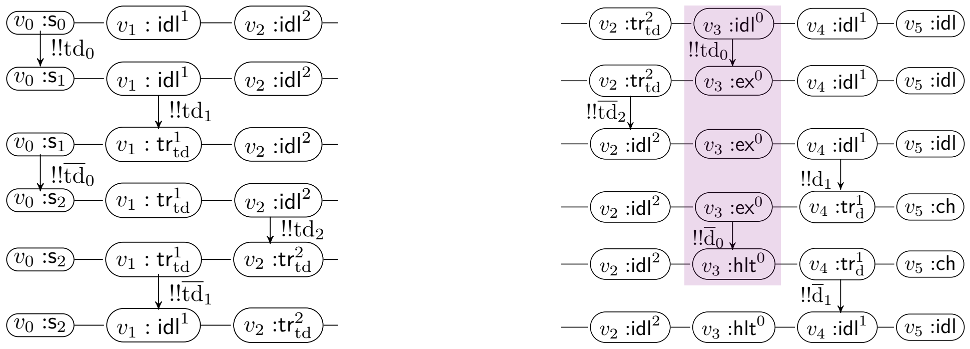


Figure 9 A configuration from which the transmission can happen: a node in state idl^i can only broadcast messages with subscript i.



(a) $C_0 \to C_1 \to \cdots \to C_5$. **(b)** $C_6 \to C_7 \to \cdots \to C_{11}$.

Figure 10 Example of correct transmission.

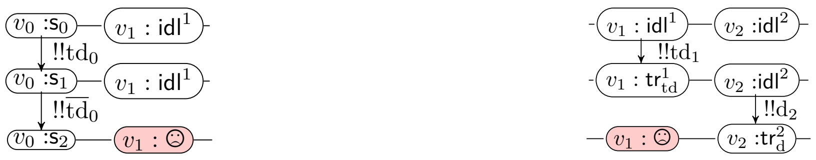
C_{11}, the following happens: Between C_0 and C_3, v_0 broadcasts the request with messages td_0 and $\overline{\text{td}}_0$. Between C_1 and C_8, v_1 and v_2 successively repeat the request to be done with messages td_1 and $\overline{\text{td}}_1$ for v_1 and td_2 $\overline{\text{td}}_2$ for v_2. Between C_6 and C_{10}, v_3 executes the request by broadcasting messages td_0 and $\overline{\text{d}}_0$. Between C_7 and C_{11}, v_4 transmits the done request with messages d_1 and $\overline{\text{d}}_1$. Hence, the request is executed by exactly one vertex (namely v_3), as highlighted in Figure 10b. Observe that the processes sort of spontaneously emit broadcast to avoid to receive a message. A correct guess of when to broadcast yields the interleaving of broadcasts that we have presented in this example.

How to prevent wrong behaviors? Observe that, when a node is in state idl^1, if one of its neighbor broadcasts a message which is not td_0, d_0 or $\overline{\text{td}}_2, \overline{\text{d}}_2$, then the node in idl^1 reaches ☹. We say that a process *fails* whenever it reaches ☹. We have the following lemma:

▶ **Lemma 4.1.** *Let $C \in \mathcal{C}$ be a* stable *configuration such that $C_0 \to^+ C$. Then in C, it holds that v_0 is in s_2, and there is exactly one vertex $v \in \{v_1, v_2, v_3, v_4\}$ on a state hlt^j for some $j \in \{0, 1, 2\}$.*

Indeed, let C be a *stable* configuration such that $C_0 \to^+ C$. It holds that:

1. *From C_0, the first broadcast is from v_0 and it broadcasts td_0.*
   Indeed, if another vertex than v_0 broadcasts a message m with subscript i from C_0, its left neighbor would fail with transition $(\mathsf{idl}^j, ?m, ☹)$ as $j = (i-1) \mod 3$ and $m \in \{\text{td}_i, \text{d}_i\}$. Let us consider an example depicted in Figure 11b: Assume v_1 is in state idl^1 and v_2 broadcasts td_2 or d_2 (it issues a request whereas v_1 is not broadcasting any request), then v_1 receives the message with transition that goes from idl^1 to ☹, as depicted in Figure 7. Hence, we can not reach a stable configuration from there.
2. *Each vertex (except the tail) broadcasts one pair of messages between C_0 and C.*
   Assume for instance that v_1 does not broadcast anything. From Item 1, v_0 broadcasts td_0, and so at some point it will also broadcasts $\overline{\text{td}}_0$ otherwise it would not be in s_0 or s_2 in C. Hence v_1 fails as depicted in Figure 11a. Actually, each vertex (except the tail) broadcasts exactly one pair: if it broadcasts more, its left neighbor would fail as well.
3. *When a node broadcasts a pair (td_j, $\overline{td}_j$), its right neighbor broadcasts either a pair (td_i, $\overline{td}_i$) or (td_i, $\overline{d}_i$), for $j, i \in \{0, 1, 2\}$.*
   Assume its right neighbor broadcasts d_i, it must be that $i = (j+1) \mod 3$. Such an example is depicted in Figure 11b: v_1 fails with $(\mathsf{tr}^1_{\text{td}}, ?\text{d}_2, ☹)$. Similarly, we have:
4. *When a node broadcasts a pair (td_j, $\overline{d}_j$) or a pair (d_j, $\overline{d}_j$), its right neighbor broadcasts a pair (d_i, $\overline{d}_i$), for $j, i \in \{0, 1, 2\}$.*



(a) v_1 does not transmit the request.

(b) v_2 broadcasts the wrong pair of messages.

Figure 11 Example of wrong behaviors during the transmission.

4.2 Putting everything together

We adapt the construction of Section 4.1 to propagate operations on counters of the machine issued by the head of the line. Counters processes will evolve in three different protocols as in Section 4.1. They can be either in a zero state, from which all the types of instructions can be transmitted, or in a state 1_{x} for x one of the two counters, from which all the types of operations can be transmitted, except 0-tests of x. Increments and decrements of a counter x are done in a similar fashion as in Section 4.1 (exactly one node changes its state). 0-tests are somewhat easier: no node changes state nor executes anything, and the tail accepts the same pair as the one broadcast by the head. However, if a node is in a 1_{x} when x is the counter compared to 0, it fails when its left neighbor broadcasts the request.

We ensure that we can select a line with a similar structure as the one depicted in Figure 9 thanks to a first part of the protocol where each node: (*i*) receives an announcement message from its predecessor with a subscript j (except the head which broadcasts first), (*ii*) broadcasts an announcement message with the subscript $(j+1) \mod 3$ (head broadcasts with subscript 0) and (*iii*) waits for the announcement of its successor with subscript $(j+2) \mod 3$ (except for the tail). If it receives any new announcement at any point of its execution, it fails. When considering only line topologies, as each node has at most two neighbors, this part can be achieved with fewer alternations. We get the two following theorems.

▶ **Theorem 4.2.** *Cover and Cover[Trees] are undecidable for k-phase-bounded protocols with $k \geq 6$.*

▶ **Theorem 4.3.** *Cover[Lines] is undecidable for k-phase-bounded protocols with $k \geq 4$.*

5 Cover in 1-Phase-Bounded Protocols

We show that Cover[Graphs] restricted to 1-phase-bounded protocols is ExpSpace-complete.

We begin by proving that for such protocols Cover[Graphs] and Cover[Stars] are equivalent (where Stars correspond to the tree topologies of height one). To get this property, we first rely on Theorem 2.4 (stating that Cover and Cover[Trees] are equivalent) and without loss of generality we can assume that if a control state can be covered with a tree topology, it can be covered by the root of the tree. We then observe that when dealing with 1-phase-bounded protocols, the behaviour of the processes of a tree which are located at a height strictly greater than 1 have no incidence on the root node. Indeed if a process at depth 2 performs a broadcast received by a node at depth 1, then this latter node will not be able to influence the state of the root because in 1-phase-bounded protocols, once a process has performed a reception, it cannot broadcast anymore. In the sequel we fix a 1-phase-bounded protocol $P = (Q, \Sigma, q_{in}, \Delta)$ and a state $q_f \in Q$. We then have:

▶ **Lemma 5.1.** *There exist $\Gamma \in$ Graphs, $C = (\Gamma, L) \in \mathcal{I}_P$ and $D = (\Gamma, L') \in \mathcal{C}_P$ and $v \in \mathsf{V}(\Gamma)$ such that $C \rightarrow^* D$ and $L'(v) = q_f$ iff there exists $\Gamma' \in$ Stars, $C' = (\Gamma', L'') \in \mathcal{I}$ and $D' = (\Gamma', L''') \in \mathcal{C}_P$ such that $C' \rightarrow_P^* D'$ and $L'''(\epsilon) = q_f$.*

To solve COVER[Stars] in EXPSPACE, we proceed as follows (1) we first propose an abstract representation for the configurations reachable by executions where the root node does not perform any reception, and that only keeps track of states in Q_0 and Q_1^b (2) we show that we can decide in polynomial space whether a configuration corresponding to a given abstract representation can be reached from an initial configuration (3) relying on reduction to the control state reachability problem in VASS (Vector Addition System with States), we show how to decide whether there exists a configuration corresponding to a given abstract representation from which q_f can be covered in an execution where the root node does not perform any broadcast. This reasoning relies on the fact that a process executing a 1-phase-bounded protocol first performs only broadcast (or internal actions) and then performs only receptions (or internal actions).

We use Q^b to represent the set $Q_0 \cup Q_1^b$ and we say that a configuration $C = (\Gamma, L)$ in $\mathcal{C}_P$ is a *star-configuration* whenever $\Gamma \in$ Stars. For a star-configuration $C = (\Gamma, L)$ in $\mathcal{C}_P$ such that $L(\epsilon) \in Q^b$, the *broadcast-print* of C, denoted by $\mathbf{bprint}(C)$, is the pair $(L(\epsilon), \{L(v) \in Q^b \mid v \in \mathsf{V}(\Gamma) \setminus \{\epsilon\}\})$ in $Q^b \times 2^{Q^b}$. We call such a configuration C a b-configuration. Note that any initial star-configuration $C_{in} = (\Gamma_{in}, L_{in}) \in \mathcal{I}$ is a b-configuration verifying $\mathbf{bprint}(C_{in}) \in \{(q_{in}, \emptyset), (q_{in}, \{q_{in}\})\}$ (the first case corresponding to $\mathsf{V}(\Gamma) = \{\epsilon\}$). We now define a transition relation $\Rightarrow$ between broadcast-prints. Given (q, Λ) and (q', Λ') in $Q^b \times 2^{Q^b}$, we write $(q, \Lambda) \Rightarrow (q', \Lambda')$ if there exists two *b*-configurations C and C' such that $\mathbf{bprint}(C) = (q, \Lambda)$ and $\mathbf{bprint}(C') = (q', \Lambda')$ and $C \rightarrow C'$. We denote by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$.

One interesting point of this abstract representation is that we can compute in polynomial time the $\Rightarrow$-successor of a given broadcast-print. The intuition is simple: either the root performs a broadcast of $m \in \Sigma$, and in that case we have to remove from the set Λ all the states from which a reception of m can be done (as the associated processes in C' will not be in a state in Q^b anymore) or one process in a state of Λ performs a broadcast and in that case it should not be received by the root node (otherwise the reached configuration will not be a b-configuration anymore).

▶ **Lemma 5.2.** *Given $(q, \Lambda) \in Q^b \times 2^{Q^b}$, we can compute in polynomial time the set $\{(q', \Lambda') \mid (q, \Lambda) \Rightarrow (q', \Lambda')\}$.*

In order to show that our abstract representation can be used to solve COVER[Stars], we need to rely on some further formal definitions. Given two star-configurations $C = (\Gamma, L)$ and $C' = (\Gamma', L')$, we write $C \preceq C'$ iff the two following conditions hold (i) $L(\epsilon) = L'(\epsilon)$, and, (ii) $|\{v \in \mathsf{V}(\Gamma) \setminus \{\epsilon\} \mid L(v) = q\}| \leq |\{v \in \mathsf{V}(\Gamma') \setminus \{\epsilon\} \mid L'(v) = q\}|$ for all $q \in Q^b$. We then have the following lemma where the two first points show that when dealing with star-configurations, the network generated by 1-phase-bounded protocol enjoys some monotonicity properties. Indeed, if the root node performs a broadcast received by other nodes, then if we put more nodes in the same state, they will also receive the message. On the other hand if it is another node that performs a broadcast, only the root node is able to receive it. The last point of the lemma shows that we can have as many processes as we want in reachable states in Q^b (as soon as the root node does not perform any reception) by duplicating nodes and mimicking behaviors.

▶ **Lemma 5.3.** *The following properties hold:*

(i) *If C_1, C_1' and C_2 are star-configurations such that $C_1 \rightarrow C_1'$ and $C_1 \preceq C_2$ then there exists a star-configuration C_2' such that $C_1' \preceq C_2'$ and $C_2 \rightarrow^* C_2'$.*

(ii) *If C_1, C_1' and C_2 are b-configurations such that $C_1 \rightarrow C_1'$ and $\mathbf{bprint}(C_1) = \mathbf{bprint}(C_2)$ and $C_1 \preceq C_2$ then there exists a b-configuration C_2' such that $C_1' \preceq C_2'$ and $\mathbf{bprint}(C_1') = \mathbf{bprint}(C_2')$ and $C_2 \rightarrow^* C_2'$.*

(iii) *If C is a b-configuration such that $C_{in} \rightarrow^* C$ for some initial configuration C_{in} then for all $N \in \mathbb{N}$, there exists an initial configuration C_{in}' and a b-configuration $C' = (\Gamma', L')$ such that $C_{in}' \rightarrow^* C'$ and $\mathbf{bprint}(C) = \mathbf{bprint}(C') = (q, \Lambda)$ and $|\{v \in \mathsf{V}(\Gamma') \setminus \{\epsilon\} \mid L'(v) = q'\}| \geq N$ for all $q' \in \Lambda$.*

We can now prove that we can reason in a sound and complete way with broadcast prints to characterise the b-configurations reachable from initial star-configurations. To prove this next lemma, we rely on the two last points of the previous lemma and reason by induction on the length of the $\Rightarrow$-path leading from (q_{in}, Λ_{in}) to (q, Λ).

▶ **Lemma 5.4.** *Given $(q, \Lambda) \in Q^b \times 2^{Q^b}$, we have $(q_{in}, \Lambda_{in}) \Rightarrow^* (q, \Lambda)$ with $\Lambda_{in} \in \{\emptyset, \{q_{in}\}\}$ iff there exist two b-configurations $C_{in} \in \mathcal{I}$ and $C \in \mathcal{C}$ such that $C_{in} \rightarrow^* C$ and $\mathbf{bprint}(C) = (q, \Lambda)$.*

Finally, we show that we can verify in exponential space whether there exists a configuration with a given broadcast-print (q, Λ) from which we can reach a configuration covering q_f thanks to an execution where the root node does not perform any broadcast. This result is obtained by a reduction to the control state reachability problem in (unary) VASS which is known to be ExpSpace-complete [18, 21]. VASS are finite state machines equipped with variables (called counters) taking their values in $\mathbb{N}$, and where each transition of the machine can either change the value of a counter, by incrementing or decrementing it, or do nothing. In our reduction, we encode the state of the root in the control state of the VASS and we associate a counter to each state of Q^b to represent the number of processes in this state. In a first phase, the VASS generates a configuration with (q, Λ) as broadcast-print and in a second phase it simulates the network. For instance, if a process performs a broadcast received by the root node, then we decrement the counter associated to the source state of the broadcast, we increment the one associated to the target state and we change the control state of the VASS representing the state of the root node accordingly. We need a last definition to characterise executions where the root node does not perform any broadcast: given two star-configurations $C = (\Gamma, L)$ and $C' = (\Gamma, L')$, we write $C \rightarrow_r C'$ whenever there exist $v \in \mathsf{V}(\Gamma)$ and $\delta \in \Delta$ such that $C \xrightarrow{v,\delta} C'$ and either $v \neq \epsilon$ or $\delta = (q, \tau, q')$ for some $q, q' \in Q$. We denote by $\rightarrow_r^*$ the reflexive and transitive closure of $\rightarrow_r$.

▶ **Lemma 5.5.** *Given $(q, \Lambda) \in Q^b \times 2^{Q^b}$, we can decide in ExpSpace whether there exist a b-configuration $C = (\Gamma_f, L)$ and a star-configuration $C_f = (\Gamma_f, L_f)$ such that $\mathbf{bprint}(C) = (q, \Lambda)$ and $L_f(\epsilon) = q_f$ and $C \rightarrow_r^* C_f$.*

Combining the results of the previous lemmas leads to an ExpSpace-algorithm to solve Cover[Stars]. We first guess a broadcast-print (q, Λ) and check in polynomial space whether it is $\Rightarrow$-reachable from an initial broadcast-print in $\{(q_{in}, \emptyset), (q_{in}, \{q_{in}\})\}$ thanks to Lemma 5.2 (relying on a non-deterministic polynomial space algorithm for reachability). Then we use Lemma 5.5 to check the existence of a b-configuration C with $\mathbf{bprint}(C) = (q, \Lambda)$ from which we can cover q_f. By Savitch's theorem [23], we conclude that the problem is in ExpSpace. The completeness of this method is direct. For the soundess, we reason as follows: using Lemma 5.4, there exists a configuration C reachable from an initial star-configuration such that $\mathbf{bprint}(C) = (q, \Lambda)$, and by Lemma 5.5, there is a configuration C' such that $\mathbf{bprint}(C') = (q, \Lambda)$ from which we cover q_f. Thanks to Lemma 5.3.(iii), there is a configuration C'' reachable from an initial configuration such that $C \preceq C''$ and $C' \preceq C''$ and $\mathbf{bprint}(C'') = (q, \Lambda)$. Thanks to Lemma 5.3.$(i)$ applied to each transition, we can build an execution from C'' that covers q_f. The lower bound is obtained by a reduction from the control state reachability in VASS.

▶ **Theorem 5.6.** *COVER[Graphs] and COVER[Trees] are EXPSPACE-complete for 1-phase-bounded protocols.*

6 Decidability Results for 2-Phase-Bounded Protocols

6.1 Cover and Cover[Trees] are Decidable on 2-PB Protocols

A *simple path between u and u'* in a topology $\Gamma = (V, E)$ is a sequence of distinct vertices $v_0, \ldots, v_k$ such that $u = v_0$, $u' = v_k$, and for all $0 \leq i < k$, $(v_i, v_{i+1}) \in E$. Its length is denoted $d(v_0, \ldots, v_k)$ and is equal to k. Given an integer K, we say that a topology Γ is K-bounded path (and we write $\Gamma \in K - \mathsf{BP}$) if there is no simple path $v_0, \ldots, v_k$ such that $d(v_0, \ldots, v_k) > K$ The result of this subsection relies on the following theorem.

▶ **Theorem 6.1** ([6],Theorem 5). *For $K \geq 1$, COVER[K-BP] is decidable.*

Hence, we show that if a state q_f of a protocol P is coverable with a tree topology, then q_f is actually coverable with a tree topology that is also $2(|Q| + 1) - \mathsf{BP}$. To establish this result, consider a coverable state q_f of a protocol P with a tree topology Γ, such that Γ is minimal in the number of nodes needed to cover q_f. We can suppose wlog that q_f is covered by the root of the tree. We argue that all nodes (except maybe the root) in the execution covering q_f broadcast something, as otherwise they are useless and could then be removed. We also argue that, since P is 2-phase-bounded, a node that would first broadcast after the first broadcast of its father would also be useless for the covering of q_f: this broadcast will only be received by its father in its *last phase of reception*, hence it will have no influence on the behavior of the root. These two properties are the key elements needed to establish the following lemma.

▶ **Lemma 6.2.** *Let $P = (Q, \Sigma, q_{in}, \Delta)$ be a 2-phase-bounded protocol and $q_f \in Q$. If q_f can be covered with a tree topology, then it can be covered with a topology $\Gamma \in$ Trees such that, for all $u \in V(\Gamma)$, $|u| \leq |Q| + 1$.*

Indeed, a counting argument implies that if this is not the case, there exist two nodes u_1 and u_2 on the same branch, different from the root, with u_1 a prefix of u_2, that both execute their first broadcast from the same state q. In this case, we could replace the subtree rooted in u_1 by the subtree rooted in u_2, and still obtain an execution covering q_f. Once u_1 has reached q (possibly by receiving broadcasts from the children of u_2), it will behave as in the initial execution. Behaviors of the children of u_1 might differ in this second part, but it can only influence u_1 in its reception phase, which will be the last phase, and hence will not influence the behavior of the root. Thanks to Theorems 2.4 and 6.1, we can then conclude.

▶ **Theorem 6.3.** *COVER and COVER[Trees] are decidable for 2-phase-bounded protocols.*

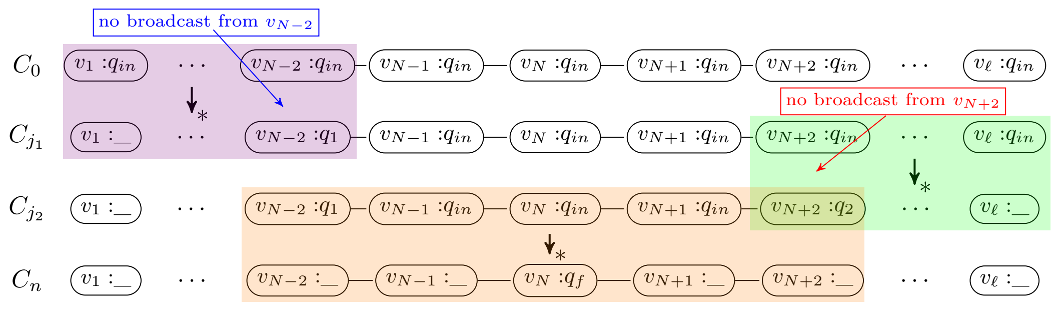


Figure 12 Illustration of execution ρ obtained from Lemma 6.4.

6.2 Polynomial Time Algorithm for Cover[Lines] on 2-PB Protocols

In the rest of this section, we fix a 2-phase-bounded protocol $P = (Q, \Sigma, q_{in}, \Delta)$ and a state $q_f \in Q$ to cover. For an execution $\rho = C_0 \to C_1 \to \cdots \to C_n$ with $C_n = (\Gamma, L_n)$, for all $v \in \mathsf{V}(\Gamma)$, we denote by $b_{\mathsf{first}}(v, \rho)$ the *smallest* index $0 \leq i < n$ such that $C_i \xrightarrow{v,t} C_{i+1}$ with $t = (q, !!m, q') \in \Delta$. If v never broadcasts anything, $b_{\mathsf{first}}(v, \rho) = -1$. We also denote by $t_{\mathsf{last}}(v, \rho)$ the *largest* index $0 \leq i < n$, such that $C_i \xrightarrow{v,t} C_{i+1}$ for some transition $t \in \Delta$. If v never issues any transition, we let $t_{\mathsf{last}}(v, \rho) = -1$.

The polynomial time algorithm relies on the fact that to cover a state, one can consider only executions that have a specific shape, described in the following lemma.

▶ **Lemma 6.4.** *If q_f is coverable with a line topology Γ such that $\mathsf{V}(\Gamma) = \{v_1, \ldots, v_\ell\}$ then there exists an execution $\rho = C_0 \to C_1 \to \cdots \to C_n$ such that $C_n = (\Gamma, L_n)$, and $3 \leq N \leq \ell - 2$ with $L_n(v_N) = q_f$, and*

1. *there exist $0 \leq j_1 < j_2 < n$ such that for all $0 \leq j < n$, if we let $C_j \xrightarrow{v^j, t^j} C_{j+1}$:*
   (a) *if $0 \leq j < j_1$, then $v^j \in \{v_1, \ldots, v_{N-2}\}$ and if $v^j = v_{N-2}$, then $t^j = (q, \tau, q')$ for some $q, q' \in Q$; and*
   (b) *if $j_1 \leq j < j_2$, then $v^j \in \{v_{N+2}, \ldots, v_\ell\}$ and if $v^j = v_{N+2}$, then $t^j = (q, \tau, q')$ for some $q, q' \in Q$; and*
   (c) *if $j_2 \leq j < n$, then $v^j \in \{v_{N-2}, \ldots, v_{N+2}\}$.*
2. **(a)** *for all $1 \leq i \leq N - 2$, $t_{\mathsf{last}}(v_i, \rho) \leq b_{\mathsf{first}}(v_{i+1}, \rho)$, and*
   (b) *for all $N + 2 \leq i \leq \ell$, $t_{\mathsf{last}}(v_i, \rho) \leq b_{\mathsf{first}}(v_{i-1}, \rho)$.*

Figure 12 illustrates the specific form of the execution described in Item 1 of Lemma 6.4: the first nodes to take actions are the ones in the purple part (on the left), then, only nodes in the green part (on the right) issue transitions), and finally the nodes in the orange central part take actions in order to reach q_f. The fact that P is 2-phase bounded allows us to establish Item 2 of Lemma 6.4: when v_{i+1} starts broadcasting, no further broadcasts from v_i will influence v_{i+1}'s broadcasts (it can only receive them in its last reception phase).

Figure 12 highlights why we get a polynomial time algorithm: when we reach the orange part of the execution, the nodes v_{N-1}, v_N and v_{N+1} are still in the initial state of the protocol. Moreover, in the orange part (which is the one that witnesses the covering of q_f), only five nodes take actions. Once one has computed in which set of states the nodes v_{N-2} and v_{N+2} can be at the beginning of the orange part, it only remains to compute the set of reachable configurations from a finite set of configurations. Let H be the set of possible states in which

v_{N-2} and v_{N+2} can be at the beginning of the last part of the execution, and for $q_1, q_2 \in H$, let $C_{q_1,q_2} = (\Gamma_5, L_{q_1,q_2})$ where Γ_5 is the line topology with five vertices $\{v_1, v_2, v_3, v_4, v_5\}$ and $L_{q_1,q_2}(v_1) = q_1$, $L_{q_1,q_2}(v_5) = q_2$ and for all other vertex v, $L_{q_1,q_2}(v) = q_{in}$.

Our algorithm is then: (1) Compute H; (2) For all $q_1, q_2 \in H$, explore reachable configurations from C_{q_1,q_2}; (3) Answer yes if we reach a configuration covering q_f, answer no otherwise. It remains to explain how to compute H. This computation relies on Item 2 of Lemma 6.4: locally, each node v_i at the left of v_{N-1} (resp. at the right of v_{N+1}) stops issuing transitions once its right neighbor v_{i+1} (resp. its left neighbor v_{i-1}) starts broadcasting.

Hence we compute iteratively set of coverable pairs of states $S \subseteq Q \times Q$ by relying on a family $(S_i)_{i\in\mathbb{N}}$ of subsets of $Q \times Q$ formally defined as follows:

$$\begin{aligned}
&S_0 = \{(q_{in}, q_{in})\}\\
&S_{i+1} = S_i \ \cup \{(q_1, q_2) \mid \text{there exist } (p_1, p_2) \in S_i, j \in \{1, 2\} \text{ s.t. } (p_j, \tau, q_j) \ \in \Delta \text{ and } p_{3-j} = q_{3-j}\}\\
&\cup \{(q_1, q_2) \mid \text{there exists } (p_1, p_2) \in S_i,\ \text{ s.t. } (p_2, !!m, q_2) \ \in \Delta, (p_1, ?m, q_1) \ \in \Delta, m \in \Sigma\}\\
&\cup \{(q_1, q_2) \mid \text{there exists } p_2 \in Q \text{ s.t. } (q_1, p_2) \in S_i,\ \text{ and } (p_2, !!m, q_2) \ \in \Delta \text{ and } m \notin R(q_1)\}\\
&\cup \{(q_{in}, q) \mid \text{there exists } (q, q') \in S_i \ \text{ for some } q' \in Q\}.
\end{aligned}$$

We then define $S = \bigcup_{n\ \in\mathbb{N}} S_n$, and $H = \{q \in Q \mid \text{ there exists } q' \text{ and } (q, q') \in S\}$. Observe that $(S_i)_{i\in\mathbb{N}}$ is an increasing sequence bounded by $|Q|^2$. The computation reaches then a fixpoint and S can be computed in polynomial time. We define $H = \{q \mid \exists q' \in Q, (q, q') \in S\}$. Note that $H \subseteq Q_0 \cup Q_1^r$, as expected by Item 2 of Lemma 6.4. We also state that our construction is complete and correct, leading to the following theorem.

▶ **Theorem 6.5.** *Cover[Lines] is in P for k-phase-bounded protocols with $k \in \{1, 2\}$.*

Proof. We explain why the algorithm takes a polynomial time: step 1 (computing H) is done in polynomial time as explained above. For step 2, there are at most $|H| \times |H| \leq |Q|^2$ pairs, and for each pair, we explore a graph of at most $|Q|^5$ nodes in which each vertex represents a configuration $C = (\Gamma_5, L)$. Accessibility in a graph can be done non-deterministically in logarithmic space, and so in polynomial time. Observe that all the lemmas of this section hold true when considering 1-phase-bounded protocols, hence the theorem. ◀

References

1 B. Aminof, S. Jacobs, A. Khalimov, and S. Rubin. Parametrized model checking of token-passing systems. In *VMCAI'14*, volume 8318 of *LNCS*, pages 262–281. Springer-Verlag, 2014.

2 D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC'04*, pages 290–299. ACM, 2004.

3 M. F. Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *Log. Methods Comput. Sci.*, 7(4), 2011.

4 B. Bollig, M. Lehaut, and N. Sznajder. Round-bounded control of parameterized systems. In *ATVA'18*, volume 11138 of *Lecture Notes in Computer Science*, pages 370–386. Springer, 2018.

5 E. M. Clarke, M. Talupur, T. Touili, and H. Veith. Verification by network decomposition. In *CONCUR'04*, volume 3170 of *LNCS*, pages 276–291. Springer-Verlag, 2004.

6 G. Delzanno, A.Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR'10*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.

7 G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *FOSSACS'11*, volume 6604 of *LNCS*, pages 441–455. Springer, 2011.

8 A. Durand-Gasselin, J. Esparza, P. Ganty, and R. Majumdar. Model checking parameterized asynchronous shared-memory systems. *Formal Methods Syst. Des.*, 50(2-3):140–167, 2017.

9 J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *LICS'99*, pages 352–359. IEEE Computer Society, 1999.

10 J. Esparza, P. Ganty, J. Leroux, and R. Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017.

11 J. Esparza, P. Ganty, and R. Majumdar. Parameterized verification of asynchronous shared-memory systems. *J. ACM*, 63(1):10:1–10:48, 2016.

12 J. Esparza, S. Jaax, M. A. Raskin, and C. Weil-Kennedy. The complexity of verifying population protocols. *Distributed Comput.*, 34(2):133–177, 2021.

13 S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

14 L. Guillou, A. Sangnier, and N. Sznajder. Safety analysis of parameterised networks with non-blocking rendez-vous. In *CONCUR'23*, volume 279 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

15 L. Guillou, A. Sangnier, and N. Sznajder. Phase-bounded broadcast networks over topologies of communication, 2024. arXiv:2406.15202.

16 O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978.

17 S. La Torre, P. Madhusudan, and G. Parlato. Model-checking parameterized concurrent programs using linear interfaces. In *CAV'10*, volume 6174 of *LNCS*, pages 629–644. Springer, 2010.

18 R.J. Lipton. *The reachability problem requires exponential space*. Research report (Yale University. Department of Computer Science). Department of Computer Science, Yale University, 1976.

19 M. L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., 1967.

20 S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. In *TACAS'05*, volume 3440 of *LNCS*, pages 93–107. Springer, 2005.

21 C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:223–231, 1978.

22 G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. *ACM Trans. Program. Lang. Syst.*, 22(2):416–430, 2000.

23 W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

24 S. Schmitz and P. Schnoebelen. The power of well-structured systems. In *CONCUR'13*, volume 8052 of *LNCS*, pages 5–24. Springer, 2013.