Complexity of Coverability in Bounded Path Broadcast Networks

A. R. Balasubramanian
Technische Universität München, Germany

Abstract

Broadcast networks are a formalism of distributed computation that allow one to model networks of identical nodes communicating through message broadcasts over a communication topology that does not change over the course of executions. The parameterized verification problem for these networks amounts to proving correctness of a property for any number of nodes, and on all executions. Dually speaking, this problem asks for the existence of an execution of the broadcast network that violates a given property. One specific instance of parameterized verification is the coverability problem which asks whether there is an execution of the network in which some node reaches a given state of the broadcast protocol. This problem was proven to be undecidable by Delzanno, Sangnier and Zavattaro (CONCUR 2010). In the same paper, the authors also prove that, if we additionally assume that the underlying communication topology has a bound on the longest path, then the coverability problem becomes decidable.

In this paper, we provide complexity results for the above problem and prove that the coverability problem for bounded-path topologies is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the fast-growing hierarchy of complexity classes. This solves an open problem of Hasse, Schmitz and Schnoebelen (LMCS, Vol 10, Issue 4).

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Distributed computing models

Keywords and phrases Parameterized verification, Bounded path networks, Fast-growing complexity classes

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.35

Funding *A. R. Balasubramanian*: Supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).

1 Introduction

In recent years, significant effort has been put into understanding the precise computational complexity of problems which are *non-elementary*, i.e., problems whose running times cannot be upper bounded by any fixed tower of exponentials of the input size [13, 6, 20, 19, 1, 18, 8]. A well-known such problem is the satisfiability problem of the weak monadic theory of one successor (WS1S) [17]. A more recent addition to this collection is the reachability problem for Petri nets [7]. We refer the reader to the excellent survey by Schmitz [19] for a collection of various non-elementary problems from logic, automata theory and verification which have been proven to be complete for appropriate complexity classes in the *fast-growing hierarchy*. This hierarchy allows for a finer classification of problems lying beyond the elementary regime.

From a tractability perspective, these results are of course negative. However, there are non-elementary problems for which tools have been developed, for e.g. MONA for WS1S [11]; and considerable effort has been put into the development of fast heuristics to solve some non-elementary problems on realistic inputs, for e.g., there is a huge wealth of heuristics and special cases which have been studied for the Petri net reachability problem [3, 12, 14, 4, 5, 15]. Hence, understanding the precise complexity of a non-elementary problem can help us to solve it in practice by reducing it to various other well-studied non-elementary problems.

The fast-growing hierarchy mentioned above can help us in this goal of understanding the computational complexity of non-elementary problems. Proving a problem to be hard for one of these classes implies that that problem cannot have an efficient encoding into any of the non-elementary problems which lie strictly below this class. In their invited paper for CONCUR 2013 [21], Schmitz and Schnoebelen explicity state the program of populating the catalog of hard problems for classes in the fast-growing hierarchy, so that hardness proofs do not have to begin from Turing machines, but can instead rely on simpler reductions.

In this paper, we contribute to this program by considering a problem from the parameterized verification of broadcast networks and proving that it is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a complexity class in the fast-growing hierarchy. We now offer a brief overview of broadcast networks [9, 2]. Broadcast networks are a formalism of distributed computation that allow one to model networks of *identical nodes* communicating through message broadcasts. Each node runs the same protocol and an underlying communication topology specifies for each node, the set of neighbors that it can broadcast messages to. This topology remains invariant over the course of executions of the network. At any point, a node can broadcast a message which is received by all of its neighbors.

The parameterized verification problem for these networks amounts to proving correctness of a property for any number of nodes and over any communication topology. Dually, we ask for the existence of an execution of the network that violates a given property. One specific instance of parameterized verification is the coverability problem which asks whether there is an execution of the network in which some node reaches a given state of the broadcast protocol. This problem was proven to be undecidable by Delzanno, Sangnier and Zavattaro (Theorem 1 of [9]). In the same paper, the authors also prove that, if we additionally assume that the underlying communication topology has a bound on the longest path (bounded-path topologies), then the coverability problem becomes decidable (Theorem 5 of [9]). Our main result in this paper is that the coverability problem for bounded-path topologies is $\mathbf{F}_{\epsilon_0}$-complete, where $\mathbf{F}_{\epsilon_0}$ is a class in the aforementioned fast-growing hierarchy of complexity classes.

Our result settles a conjecture raised by Hasse, Schmitz and Schnoebelen (Section 8.3 of [16]) and also settles the complexity of the last remaining question from the original paper that initiated the study of parameterized verification problems for broadcast networks [9]. Moreover, we provide a new and rather natural problem to the list of $\mathbf{F}_{\epsilon_0}$-complete problems, which when compared to the list of $\mathbf{F}_{\omega}$-complete and $\mathbf{F}_{\omega^\omega}$-complete problems, is rather small currently (Section 6.4 of [19]). (Both $\mathbf{F}_{\omega}$ and $\mathbf{F}_{\omega^\omega}$ are classes in the fast-growing hierarchy which are much smaller than $\mathbf{F}_{\epsilon_0}$). Hence, in this sense, we contribute to the above-mentioned program of finding hard problems for classes in the fast-growing hierarchy. Further, we hope that the present work might prove to be useful in settling the complexity of other problems conjectured to be $\mathbf{F}_{\epsilon_0}$-complete (Section 8.3 of [16]), since all the problems mentioned there are concerned with infinite-state systems regarding bounded-path trees and graphs, and so those problems are in some sense "close" to the problem that we consider here.

2 Preliminaries

In this section, we recall the model of broadcast networks as defined in [2]. Intuitively, a broadcast network consists of several nodes, each executing the same finite-state *broadcast protocol.* A communication topology assigns to each node, a finite set of neighbors, to which it can communicate. At any point, some node can broadcast a message which is received by all of its neighbors. We now proceed to formalize this intuition.

Broadcast networks

▶ **Definition 1.** *A broadcast protocol is a tuple $\mathcal{P} = (Q, I, \Sigma, \Delta)$ where Q is a finite set of states, $I \subseteq Q$ is the set of initial states, Σ is a finite set of messages and $\Delta \subseteq Q \times \{!a, ?a, : a \in \Sigma\} \times Q$ is the transition relation.*

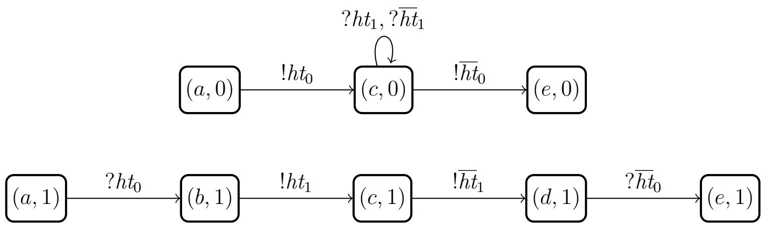
For ease of notation, we will write $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) for $(q, !a, q') \in \Delta$ (resp. $(q, ?a, q') \in \Delta$). A transition $q \xrightarrow{!a} q'$ (resp. $q \xrightarrow{?a} q'$) intuitively corresponds to broadcasting (resp. receiving) the message a. We will assume that broadcast protocols are complete, i.e. for every state q and every message a there exists q' such that $q \xrightarrow{?a} q'$.

As mentioned before, a broadcast network consists of several identical *nodes* running a broadcast protocol and each node has a finite set of neighbors. To formalize this, given a broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a *configuration* of $\mathcal{P}$ is a labelled graph $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ where N is a finite set of nodes, $\mathsf{E} \subseteq \mathsf{N} \times \mathsf{N}$ is a finite set of (undirected) edges specifying for every pair of nodes whether or not there is a communication link between them and $\mathsf{L} : \mathsf{N} \to Q$ is a labelling function that specifies the current state of each node. We let $\mathsf{L}(\gamma) = \{\mathsf{L}(\mathsf{n}) : \mathsf{n} \in \mathsf{N}\}$ be the set of labels appearing in the nodes of γ. We say that γ is *initial* if $\mathsf{L}(\gamma) \subseteq I$.

The semantics of the broadcast network of a protocol $\mathcal{P}$ is given by means of an infinite-state transition system $\mathcal{T}(\mathcal{P})$ which consists of all the configurations of the protocol $\mathcal{P}$. There is a step from the configuration $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ to the configuration $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ if $\mathsf{N}' = \mathsf{N}$, $\mathsf{E}' = \mathsf{E}$ and there exists a node n and a message $a \in \Sigma$ such that $(\mathsf{L}(\mathsf{n}), !a, \mathsf{L}'(\mathsf{n})) \in \Delta$, and for every other node n', if $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}$, then $(\mathsf{L}(\mathsf{n}), ?a, \mathsf{L}'(\mathsf{n}')) \in \Delta$; otherwise $\mathsf{L}(\mathsf{n}') = \mathsf{L}'(\mathsf{n}')$. In this case, we write $\gamma \xrightarrow{\mathsf{n}, a} \gamma'$ or simply $\gamma \to \gamma'$. Intuitively, a step consists of a node n broadcasting some message a which is then received by *all* of its neighbors; all the other nodes do nothing. Notice that between steps, the set of nodes and edges do not change.

A *run* from the configuration γ to the configuation γ' is a sequence of steps $\gamma \to \gamma_1 \to \gamma_2 \to \dots \gamma_{k-1} \to \gamma'$. If a run exists between configurations γ and γ' we denote it by $\gamma \xrightarrow{*} \gamma'$. An *execution* is a run starting from an initial configuration.

Given a state f and a configuration γ we say that γ covers f if $f \in \mathsf{L}(\gamma)$, i.e., if the state of some node in γ is f. We say that an execution $\gamma_0 \xrightarrow{*} \gamma$ covers f, if γ covers f. The *coverability* problem for broadcast protocols is to decide, given a broadcast protocol $\mathcal{P}$ and a state f, whether there is an execution from some initial configuration that covers f. It is known that the coverability problem is undecidable (Theorem 1 of [9]).


$$\begin{tikzcd}
(a,0) \arrow[r, "!ht_0"] & (c,0) \arrow[loop above, "?ht_1, ?\overline{ht}_1"] \arrow[r, "!\overline{ht}_0"] & (e,0)
\end{tikzcd}$$

$$\begin{tikzcd}
(a,1) \arrow[r, "?ht_0"] & (b,1) \arrow[r, "!ht_1"] & (c,1) \arrow[r, "!\overline{ht}_1"] & (d,1) \arrow[r, "?\overline{ht}_0"] & (e,1)
\end{tikzcd}$$

Figure 1 Example of a broadcast protocol where we set $I = \{(a, 0), (a, 1)\}$ and $\Sigma = \{ht_i, \overline{ht}_i : 0 \leq i \leq 1\}$. If for a state (f, i), we have not depicted what happens when message m is received at (f, i), we assume that $(f, i) \xrightarrow{?m} (\bot, i)$. Here $(\bot, 0)$ and $(\bot, 1)$ are new *sink* states, i.e., states with no outgoing transition.

▶ **Example 2.** We consider the broadcast protocol given in Figure 1. Figure 2 shows an execution in this protocol covering the state $(e, 0)$. Moreover, let $\gamma = (\mathsf{N}, \mathsf{E}, \mathsf{L})$ be *any* initial configuration and $\gamma' = (\mathsf{N}', \mathsf{E}', \mathsf{L}')$ be *any* configuration covering $(e, 0)$ such that $\gamma \xrightarrow{*} \gamma'$.

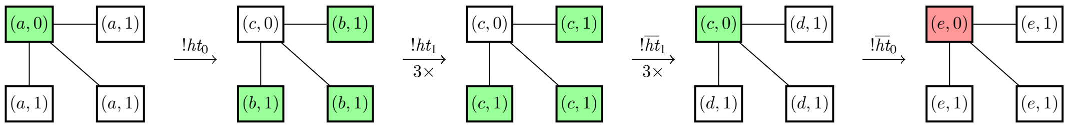


Figure 2 Example of an execution covering $(e,0)$ in the broadcast protocol given in Figure 1. The nodes marked in green make the broadcasts, i.e., first the node on the topmost left broadcasts ht_0, then all the other nodes broadcast ht_1 in some order, and then $\overline{ht}_1$ in some order, and then the node on the topmost left broadcasts $\overline{ht}_0$.

Hence, there is a node n such that $\mathsf{L}'(\mathsf{n}) = (e,0)$. Note that $\mathsf{L}(\mathsf{n})$ must be $(a,0)$. Hence n must have broadcasted both ht_0 and $\overline{ht}_0$ to move into the states $(c,0)$ and $(e,0)$ at different points during the run. This means that all of the neighbors of n received $\overline{ht}_0$ at some point, and so the labels of all of its neighbors in γ' must be either $(e,1)$ or $(\perp,0)$ or $(\perp,1)$.

Suppose n' is a neighbor of n such that $\mathsf{L}'(\mathsf{n}') = (e,1)$. Notice that if there is a neighbor $\mathsf{n}'' \neq \mathsf{n}$ of n' which was at $(c,0)$ during some point in the run, then n'' must have broadcasted ht_0 during the run. However, then n' would have received two ht_0 messages, which would have caused it to move into either $(\perp,0)$ or $(\perp,1)$. Hence, there is exactly one neighbor of n' which was labelled by $(c,0)$ at some point during the run.

This protocol along with the above discussion will prove useful later on for the lower bound reductions in section 5.

Bounded-path broadcast networks

Motivated by the undecidability of the coverability problem, the authors of [9] also study a different variant of the problem, which we now describe.

Let $\mathcal{P}$ be a broadcast protocol and let $k \geq 1$ be some number. Let γ be a configuration of $\mathcal{P}$. We say that γ is k-path bounded if the length of the *longest* simple path in γ is at most k. Now, let $\mathcal{T}_k(\mathcal{P})$ be the restriction of the transition system $\mathcal{T}(\mathcal{P})$ to only k-path bounded configurations. Notice that since the set of nodes and edges do not change during a run, $\mathcal{T}_k(\mathcal{P})$ is closed under the step relation. The *path bounded* coverability problem (Bounded-Path-Cover) is then defined as follows:

Given: A broadcast protocol $\mathcal{P} = (Q, I, \Sigma, \Delta)$, a state $f \in Q$ and a number k.
Decide: If there is an execution in $\mathcal{T}_k(\mathcal{P})$ which covers f.

The authors of [9] prove that this problem is decidable (Theorem 5 of [9]). The main result that we prove in this paper is that

▶ **Theorem 3.** *Bounded-Path-Cover is* $\mathbf{F}_{\epsilon_0}$*-complete.*

Here $\mathbf{F}_{\epsilon_0}$ is a member of the *fast-growing* complexity class hierarchy. We refer the reader to Section 2.3 of [19] for a description of the fast-growing hierarchy and the class $\mathbf{F}_{\epsilon_0}$. To prove the upper bound for our problem, we will consider the algorithm given in [9] and analyze its running time by means of *controlled-bad* sequences of a suitable well-quasi order, whose upper bounds will allow us to place Bounded-Path-Cover in the complexity class $\mathbf{F}_{\epsilon_0}$. The lower bound is proved by giving a logspace reduction from a known $\mathbf{F}_{\epsilon_0}$-hard problem, which we now proceed to describe.

3 Nested counter systems (NCS)

A nested counter system is a generalisation of a usual counter system with *higher-order counters*, i.e., counters which can themselves contain other (lower-order) counters. Intuitively, an one-dimensional counter is a usual counter, which can either add or subtract 1. A two-dimensional counter can either add or remove an one-dimensional counter, a three-dimensional counter can either add or remove a two-dimensional counter and so on. Here, we slightly alter the definition of nested counter systems as given in [8] so that it better suits our purposes. It can be easily verified that our altered definition does not affect the semantics of the system as given in [8].

A k-nested counter system (k-NCS) is a tuple $\mathcal{N} = (Q, \delta)$ where Q is a finite set of *states* and $\delta \subseteq \bigcup_{1 \le i,j \le k+1} (Q^i \times Q^j)$ is a set of *rules*. The set $\mathcal{C}_\mathcal{N}$ of *configurations* of $\mathcal{N}$ is defined to be the set of all labelled rooted trees of height atmost k, with labels from the set Q.

The operational semantics of $\mathcal{N}$ is defined in terms of the following transition relation $\rightarrow \subseteq \mathcal{C}_\mathcal{N} \times \mathcal{C}_\mathcal{N}$ on configurations: Let $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ be a rule with $i \le j \le k$. We say that a configuration C can move to the configuration C' using the rule r (denoted by $C \xrightarrow{r} C'$), if there is a path $v_0, v_1 \dots, v_i$ in C starting at the root such that for every $0 \le l \le i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \le l \le i$, changing the label of each v_l to q'_l and 2) for every $i+1 \le l \le j$, creating a new vertex v_l with label q'_l and adding it as a child to v_{l-1}.

Similarly, suppose $r := ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ is a rule with $j < i \le k$. Then $C \xrightarrow{r} C'$ if there is a path $v_0, v_1, \dots, v_i$ in C starting at the root such that for every $0 \le l \le i$, the label of v_l is q_l and, C' is obtained from C by 1) for every $0 \le l \le j$, changing the label of each v_l to q'_l and 2) removing the subtree rooted at the node v_{j+1}.

▶ **Example 4.** Let us consider the NCS $\mathcal{N}$ given by the states $Q = \{p_i, p'_i, q_i, q'_i : 0 \le i \le 4\}$ and consisting of the following rules: $r_1 = ((q_0, q_1), (q'_0, q'_1, q'_2)), r_2 = ((q'_0, q_3, q_2), (p_0)), r_3 = ((p_0), (p'_0))$. In Figure 3, we illustrate the application of these rules to a configuration of $\mathcal{N}$.

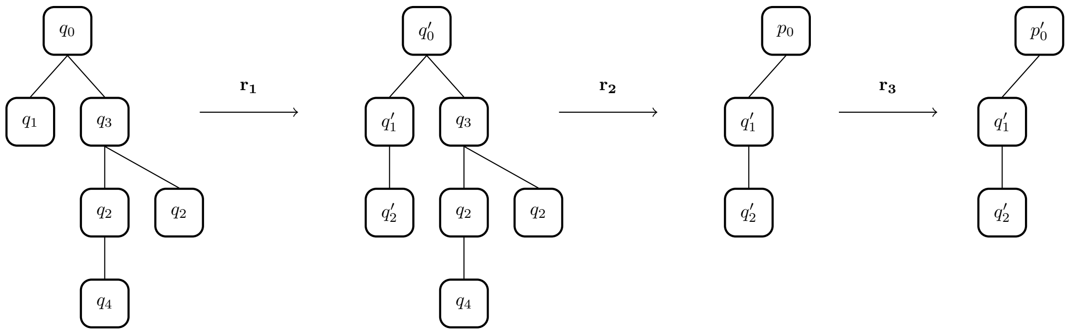


Figure 3 Application of the rules r_1, r_2 and r_3 to a configuration of $\mathcal{N}$, which is described in Example 4.

We say that $C \rightarrow C'$ if $C \xrightarrow{r} C'$ for some rule r. We let $\xrightarrow{*}$ denote the reflexive and transitive closure of $\rightarrow$ and we say that a configuration C reaches C' if $C \xrightarrow{*} C'$. Given two states $q_{in}, q_f \in Q$, we say that q_{in} can cover q_f if the (unique) configuration consisting of the single root vertex labelled with q_{in} can reach *some* configuration where the root is labelled by q_f. The coverability problem for an NCS is then the following: Given an NCS $\mathcal{N}$ and two states q_{in}, q_f, can q_{in} cover q_f? It is known that the coverability problem is $\mathbf{F}_{\epsilon_0}$-hard (Theorem 7 of [8]).

Lossy semantics. In addition to the "usual" semantics of an NCS that we have described in the previous section, we also need a *lossy semantics* which we now define here. Let $\mathcal{N} = (Q, \delta)$ be a k-NCS and let $q_{in}, q_f \in Q$. We say that there is a *lossy step* between configurations C and C', if C' can be obtained from C by deleting the subtree rooted at some vertex v in C. We let $C \dashrightarrow C'$ if either there is a lossy step between C and C' or $C \xrightarrow{r} C'$ for some rule r. As usual, we let $\overset{*}{\dashrightarrow}$ denote the reflexive and transitive closure of $\dashrightarrow$ and we say that C can reach C' in a lossy manner if $C \overset{*}{\dashrightarrow} C'$. We can then define the notion of the state q_{in} covering the state q_f in a straightforward manner.

For configurations C, C', we say that $C \geq C'$ iff C' can be obtained from C by a sequence of lossy steps. Since NCS do not have any zero tests, from the definition of the transition relation, we can easily infer the following proposition.

▶ **Proposition 5.** *If $C_1 \geq C_1'$ and $C_1' \overset{*}{\dashrightarrow} C_2'$ then there exists $C_2 \geq C_2'$ such that $C_1 \xrightarrow{*} C_2$.*

Hence, we get the following corollary.

▶ **Corollary 6.** *q_f can be covered from q_{in} in a lossy manner iff q_f can be covered from q_{in} under the usual semantics.*

This corollary will be useful later on in order to prove our hardness result.

4 A simulator protocol $\mathcal{P}_{\text{sim}}$

Throughout this section, let $\mathcal{N} = (Q, \delta)$ be a fixed k-NCS with two fixed states q_{in} and q_f. In this section, we will construct a broadcast protocol $\mathcal{P}_{\text{sim}} = (Q_{\text{sim}}, I_{\text{sim}}, \Sigma_{\text{sim}}, \delta_{\text{sim}})$, a state p of $\mathcal{P}_{\text{sim}}$, and define a notion of *good initial configurations* of $\mathcal{P}_{\text{sim}}$ such that the following property is satisfied: q_f can be covered from q_{in} in the NCS $\mathcal{N}$ iff p can be covered in $\mathcal{T}_{2k}(\mathcal{P}_{\text{sim}})$ by some execution starting at a *good initial configuration.* Intuitively, the protocol $\mathcal{P}_{\text{sim}}$ will *simulate* the NCS $\mathcal{N}$, *provided* that the initial configuration that it begins with is a good initial configuration.

States, alphabet and good configurations. For each $0 \leq i \leq k$, $\mathcal{P}_{\text{sim}}$ will have two states $(\textit{start}, i), (\textit{finish}, i)$. For each $0 \leq i \leq k$ and each $r \in \delta$, we will have five states $(\texttt{req-rec}[r], i), (\texttt{req-fwd}[r], i), (\texttt{wait}[r], i), (\texttt{ack-rec}[r], i), (\texttt{ack-fwd}[r], i)$. Finally, for each $0 \leq i \leq k$ and each $q \in Q$, $\mathcal{P}_{\text{sim}}$ will have a state (q, i). Notice that each state of $\mathcal{P}_{\text{sim}}$ is of the form (a, b) where $a \in Q \cup \{\textit{start}, \textit{finish}\} \cup \{\texttt{req-rec}[r], \texttt{req-fwd}[r], \texttt{wait}[r], \texttt{ack-rec}[r], \texttt{ack-fwd}[r]\}$ and $0 \leq b \leq k$. The first part "a" will be called the *base* of the state and the second part "b" will be called the *grade.* Sometimes we will abuse notation and refer to the base (resp. grade) of a node in a configuration to mean the base (resp. grade) of the label of that node.

The initial set of states I_{sim} will be the set $\{(q_{in}, 0)\} \cup \{(\textit{start}, i) : 1 \leq i \leq k\}$. (The asymmetry in the initial set of states between the case of 0 and others will be discussed in the following paragraphs). The alphabet Σ_{sim} will be the set $\{\textit{begin}_i^r, \textit{end}_i^r : r \in \delta, 0 \leq i \leq k\}$.

A configuration γ of $\mathcal{P}_{\text{sim}}$ is called *good* if γ is a tree of height at most k such that 1) the base of the label of every node is in the set $Q \cup \{\textit{start}, \textit{finish}\}$, 2) there is exactly one node n labelled by a state of grade 0, which will be called the root of γ and, 3) every node at distance i from n is labelled by a state of grade i. Notice that if γ is a good initial configuration then $\gamma \in \mathcal{T}_{2k}(\mathcal{P})$. Further, notice that in a good initial configuration, the root must be labelled by $(q_{in}, 0)$ and every node at distance i from the root is labelled by $(\textit{start}, i)$.

Intuition behind good configurations of $\mathcal{P}_{\text{sim}}$. Before we describe the transition relation of $\mathcal{P}_{\text{sim}}$, we describe some intuition behind the notion of a good configuration.

Let γ be a good configuration of $\mathcal{P}_{\text{sim}}$. Notice that there is a way to map γ to a configuration of $\mathcal{N}$: First, forget all the grades from the labels of each node in γ and just keep the bases. Next, remove all nodes whose label is either *start* or *finish* and from the resulting forest, pick the tree T containing the root. In this way, to every good configuration γ of $\mathcal{P}_{\text{sim}}$ we can define a configuration $\mathbb{E}(\gamma)$ of $\mathcal{N}$. Hence, we can use good configurations of $\mathcal{P}_{\text{sim}}$ to encode configurations of $\mathcal{N}$ and this is the reason behind defining good configurations of $\mathcal{P}_{\text{sim}}$. An example of this mapping is given in Figure 4.

Further, notice that if γ is any good initial configuration, then $\mathbb{E}(\gamma)$ is the initial configuration of $\mathcal{N}$. This is the reason behind the asymmetry in the definition of the initial set of states between the case of 0 and others.

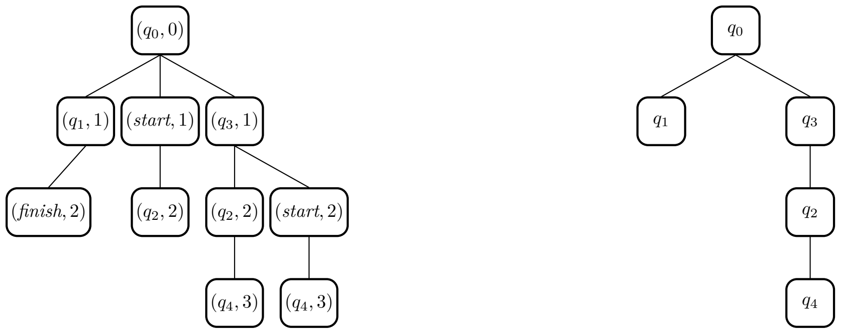


Figure 4 An example of the map $\mathbb{E}$ between good configurations of $\mathcal{P}$ and configurations of $\mathcal{N}$. On the left is a good configuration γ of $\mathcal{P}$ and on the right is its corresponding mapped configuration $\mathbb{E}(\gamma)$ of $\mathcal{N}$.

4.1 Transitions involving the letters $begin_i^r$ and end_i^r

For the rest of this section, let us fix a rule $r = ((q_0, \dots, q_i), (q'_0, \dots, q'_j)) \in \delta$ where $i, j \le k$ and let $w = \max(i, j)$. For the sake of uniformity, if $i < j$, then let $q_l = start$ for every $i < l \le j$. If $i > j$, then let $q'_l = finish$ for every $j < l \le i$.

Intuitively, the gadget that we will demonstrate will use the messages $begin_i^r$ and end_i^r to find a path $\mathsf{n}_0, \dots, \mathsf{n}_w$ labelled by $(q_0, 0), (q_1, 1), \dots, (q_w, w)$ and then change the labels along this path to $(q'_0, 0), (q'_1, 1), \dots, (q'_w, w)$. Notice that if $i \le j$, this means that a path of the form $(q_0, 0), \dots, (q_i, i), (start, i+1), \dots, (start, j)$ becomes $(q'_0, 0), \dots, (q'_i, i), (q'_{i+1}, i+1), \dots, (q'_j, j)$. Similarly, if $i > j$ then a path of the form $(q_0, 0), \dots, (q_j, j), \dots (q_i, i)$ becomes $(q'_0, 0), \dots, (q'_j, j), (finish, j+1), \dots, (finish, i)$. This would then allow us to simulate the rule r on good configurations of $\mathcal{P}_{\text{sim}}$.

Formally, we now describe the transitions involving the letters $\{begin_i^r, end_i^r : 0 \le i \le k\}$. First, we make a small remark:

▶ Remark 7. In the following, if we do not specify what happens upon receiving a message m from a state with base a and grade b, then it is to be assumed that $(a, b) \xrightarrow{?m} (finish, b)$.

The "gadget" for "simulating" the rule r. We now present the main transitions involving the messages $begin_i^r$ and end_i^r.

- First, we have four transitions

$$(q_0, 0) \xrightarrow{!begin_0^r} (\texttt{req-fwd}[r], 0) \xrightarrow{?begin_1^r} (\texttt{wait}[r], 0) \xrightarrow{?end_1^r} (\texttt{ack-rec}[r], 0) \xrightarrow{!end_0^r} (q_0', 0)$$

- Then, for every $1 \leq l \leq w-1$, we have

$$(q_l, l) \xrightarrow{?begin_{l-1}^r} (\texttt{req-rec}[r], l) \xrightarrow{!begin_l^r} (\texttt{req-fwd}[r], l) \xrightarrow{?begin_{l+1}^r} (\texttt{wait}[r], l) \xrightarrow{?end_{l+1}^r}$$
$$(\texttt{ack-rec}[r], l) \xrightarrow{!end_l^r} (\texttt{ack-fwd}[r], l) \xrightarrow{?end_{l-1}^r} (q_l', l)$$

- Finally, we have four transitions

$$(q_w, w) \xrightarrow{?begin_{w-1}^r} (\texttt{req-rec}[r], w) \xrightarrow{!begin_w^r} (\texttt{wait}[r], w) \xrightarrow{!end_w^r} (\texttt{ack-fwd}[r], w) \xrightarrow{?end_{w-1}^r} (q_w', w)$$

Self-loops. While the previous gadget comprised the main transitions involving $begin_i^r$ and end_i^r, for technical reasons we need the following self-loop transitions as well: For every state with base $a \in Q \cup \{start, finish\}$ and grade $1 \leq i \leq k$, there are two transitions $(a, i) \xrightarrow{?begin_{i-1}^r} (a, i)$ and $(a, i) \xrightarrow{?end_{i-1}^r} (a, i)$.

This finishes our description of the transition relation of $\mathcal{P}_{\text{sim}}$.

Intuition behind the transitions. We now give a brief intuition behind the gadget in the case of $w = 2$. Notice that only the root n_0 in a good configuration can be labelled by $(q_0, 0)$. Hence if n_0 broadcasts $begin_0^r$, it is *forwarding* its request of wanting to simulate the rule r to its children. The children have two choices: either stay where they are by means of the self-loops or *receive* the request and move to $(\texttt{req-rec}[r], 1)$. Atleast one child n_1 has to receive the request and move, otherwise the configuration enters into a deadlock. From $(\texttt{req-rec}[r], 1)$ n_1 can *forward* this request to its children by broadcasting $begin_1^r$ (and also let n_0 know that is has received its request, whereby it enters a waiting mode). Notice that if two children of n_0 forward the request, then n_0 will enter $(finish, 0)$ and the simulation of the rule r cannot happen. Similarly, some child n_2 of n_1 must receive the request of n_1, move to $(\texttt{req-rec}[r], 2)$, then broadcast $begin_2^r$. At this point, the base of each n_i is $\texttt{wait}[r]$.

Now n_2 can broadcast end_2^r, forwarding an *acknowledgment* to the request made by n_1. n_1 can receive this acknowledgment and broadcast end_1^r, forwarding an acknowledgment to n_0 which can broadcast end_0^r and move to $(q_0', 0)$. At this point, the labels of $\mathsf{n}_0, \mathsf{n}_1$ and n_2 are $(q_0', 0), (q_1', 1)$ and $(q_2', 2)$ respectively, which means that we have changed the labels along a path from $(q_0, 0), (q_1, 1)$ and $(q_2, 2)$ to $(q_0', 0), (q_1', 1)$ and $(q_2', 2)$.

4.2 Proof of correctness

The following lemma tells us that we can use good configurations of $\mathcal{P}_{\text{sim}}$ along with the gadget for the rule r described in the previous section to simulate steps of $\mathcal{N}$.

▶ **Lemma 8** ($\mathcal{P}_{\mathsf{sim}}$ simulates $\mathcal{N}$). *Suppose $C \xrightarrow{r} C'$ is a step in the NCS $\mathcal{N}$. Suppose γ is a good configuration such that 1) $\mathbb{E}(\gamma) = C$ and, 2) there is a path $\mathsf{n}_0, \ldots, \mathsf{n}_w$ in γ where the label of each n_l is (q_l, l). Then there is a good configuration γ' with $\gamma \xrightarrow{*} \gamma'$ such that 1) $\mathbb{E}(\gamma') = C'$ and, 2) γ' is the same as γ except the label of each n_l is (q_l', l).*

Proof sketch. For ease of presentation, we provide the proof in the case of $w = 2$. This proof can be generalized to any w in a straightforward manner.

The proof for $w = 2$ is essentially the same argument that is given in the intuition paragraph. Throughout the run that we are going to describe, if a node $\mathsf{n} \notin \{\mathsf{n}_0, \mathsf{n}_1, \mathsf{n}_2\}$ receives a message, then it will always take the self-loop transitions that we have constructed in the gadget for the rule r.

From γ, n_0 broadcasts $begin_0^r$ and moves to $(\texttt{req-fwd}[r], 0)$ and n_1 receives it and moves to $(\texttt{req-rec}[r], 1)$. Then, n_1 broadcasts $begin_1^r$ and moves to $(\texttt{req-fwd}[r], 1)$ and n_0 and n_2 receive it and move to $(\texttt{wait}[r], 0)$ and $(\texttt{req-rec}[r], 2)$ respectively. Then, n_2 broadcasts $begin_2^r$ and moves to $(\texttt{wait}[r], 2)$ and n_1 receives it and moves to $(\texttt{wait}[r], 1)$. Notice that at this point, the base of each n_i is $\texttt{wait}[r]$ and the labels of all the other nodes are unchanged, i.e., the same as the labels in γ.

Now, we proceed in the reverse direction. n_2 broadcasts end_2^r and moves to $(\texttt{ack-fwd}[r], 2)$ and n_1 receives it and moves to $(\texttt{ack-rec}[r], 1)$. Then, n_1 broadcasts end_1^r and moves to $(\texttt{ack-fwd}[r], 1)$ and n_0 and n_2 receive it and move to $(\texttt{ack-rec}[r], 0)$ and $(q_2', 2)$ respectively. Then, n_0 broadcasts end_0^r and moves to $(q_0', 0)$ and n_1 receives it and moves to $(q_1', 1)$. It is clear that the configuration reached at the end of this run satisfies the required properties. ◀

We now show a partial converse to the above lemma. It says that if there is a run of good configurations which uses only the transitions given in the gadget for the rule r and begins and ends with the root being in $(q_0, 0)$ and $(q_0', 0)$, then it is possible to "lift" that run back to the corresponding configurations in the NCS $\mathcal{N}$.

▶ **Lemma 9** ($\mathcal{N}$ simulates $\mathcal{P}_{\mathsf{sim}}$)**.** *Suppose $\gamma \xrightarrow{*} \gamma'$ where 1) γ is a good configuration, 2) the labels of the root in γ and γ' are $(q_0, 0)$ and $(q_0', 0)$ and 3) in all the configurations between γ and γ', the base of the root is in the set $\{\texttt{req-fwd}[r], \texttt{wait}[r], \texttt{ack-rec}[r]\}$. Then, 1) γ' is a good configuration and, 2) $\mathbb{E}(\gamma) \overset{*}{\dashrightarrow} \mathbb{E}(\gamma')$.*

Proof sketch. Let the run $\gamma \xrightarrow{*} \gamma'$ be of the form $\gamma = \gamma_0 \to \gamma_1 \to \dots \gamma_{m-1} \to \gamma_m = \gamma'$. By means of induction and some extensive case analysis on the gadget that we have constructed, we can first prove that there exists a path $\mathsf{n}_0, \mathsf{n}_1, \dots, \mathsf{n}_w$ in γ with the following properties:

- For each $0 \le l \le w$, the label of n_l is (q_l, l) in γ and (q_l', l) in γ'.
- For each $0 \le l \le w$, n_l broadcasts exactly two messages: $begin_l^r$ and end_l^r.
- For each $0 \le l < w$, the only child of n_l that broadcasts a message in the run is n_{l+1}.

We then let $Ch(\mathsf{n}_l)$ denote the set of children of n_l. Notice that the only node which could broadcast a message in γ_0 is n_0 and so it must be the case that $\gamma_0 \xrightarrow{\mathsf{n}_0, begin_0^r} \gamma_1$. Now, suppose, for some $0 \le l < w$, we have shown that it must be the case that $\gamma_0 \xrightarrow{\mathsf{n}_0, begin_0^r} \gamma_1 \dots \gamma_l \xrightarrow{\mathsf{n}_l, begin_l^r} \gamma_{l+1}$. Then, notice that the only nodes whose labels in γ_{l+1} could have an outgoing broadcast transition are the nodes in $\bigcup_{0 \le l' < l} (Ch(\mathsf{n}_{l'}) \setminus \{\mathsf{n}_{l'+1}\}) \cup Ch(\mathsf{n}_l)$. By our claim, among these only n_{l+1} broadcasts a message and so we must have that $\gamma_{l+1} \xrightarrow{\mathsf{n}_{l+1}, begin_{l+1}^r} \gamma_{l+2}$. Hence, in this way we get that $\gamma_0 \xrightarrow{\mathsf{n}_0, begin_0^r} \dots \gamma_w \xrightarrow{\mathsf{n}_w, begin_w^r} \gamma_{w+1}$. In exactly the same way, we can show that it must be the case that $\gamma_{w+1} \xrightarrow{\mathsf{n}_w, end_w^r} \gamma_{w+2} \xrightarrow{\mathsf{n}_{w-1}, end_{w-1}^r} \gamma_{w+3} \cdots \gamma_{2w+1} \xrightarrow{\mathsf{n}_0, end_0^r} \gamma_{2w+2} = \gamma_m$.

Let S be the set of all nodes whose base in γ belonged to $Q \cup \{start\}$ and whose base in γ' is *finish*. (Notice that $S \subseteq \bigcup_{0 \le l < w} Ch(\mathsf{n}_l)$ and $S \cup \{\mathsf{n}_0, \dots, \mathsf{n}_w\}$ are exactly the set of nodes whose labels have changed during the run). It is then easy to see that, by firing the rule r from $\mathbb{E}(\gamma)$ and then deleting all the subtrees whose roots are in S, we get $\mathbb{E}(\gamma) \overset{*}{\dashrightarrow} \mathbb{E}(\gamma')$. ◀

With these two "simulation" lemmas, we have the following main result.

▶ **Theorem 10.** *The state q_{in} can cover the state q_f in the NCS $\mathcal{N}$ iff $(q_f, 0)$ can be covered by some execution in $\mathcal{P}$ starting at a good initial configuration.*

5 A seeker protocol $\mathcal{P}_{\text{seek}}$

In the previous section, we have shown that given a k-NCS $\mathcal{N} = (Q, \delta)$ along with two states $q_{in}, q_f \in Q$, we can construct a simulator protocol $\mathcal{P}_{\text{sim}}$, such that q_{in} can cover q_f in $\mathcal{N}$ iff $(q_f, 0)$ can be covered in $\mathcal{P}_{\text{sim}}$ by an execution starting at a *good initial configuration*. In this section, we will construct a *seeker* protocol $\mathcal{P}_{\text{seek}}$ and "attach" it to $\mathcal{P}_{\text{sim}}$ which will let us get rid of the *goodness* assumption. The seeker protocol $\mathcal{P}_{\text{seek}}$ will begin at an arbitrary initial communication topology and *seek* for a subgraph to act as a good initial configuration for $\mathcal{P}_{\text{sim}}$. Hence, once we have deployed $\mathcal{P}_{\text{seek}}$ to find such a subgraph, we can then use $\mathcal{P}_{\text{sim}}$ to simulate the k-NCS $\mathcal{N}$ on this subgraph.

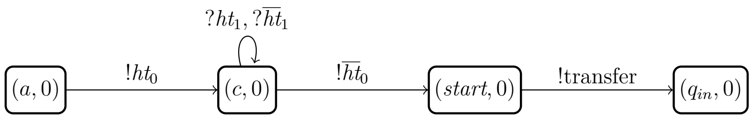
Formally, the seeker protocol $\mathcal{P}_{\text{seek}} = (Q_{\text{seek}}, I_{\text{seek}}, \Sigma_{\text{seek}}, \delta_{\text{seek}})$ will be a generalization of the protocol given in Figure 1 (with the exception that the (e, i) and $(\perp, i)$ states will be replaced by (start, i) and (finish, i) respectively).

States and alphabet. For each $0 \leq i \leq k$, $\mathcal{P}_{\text{seek}}$ will have six states of the form $(a, i), (b, i), (c, i), (d, i), (\mathit{start}, i)$ and (finish, i). Notice that (start, i) and (finish, i) are also present in $\mathcal{P}_{\text{sim}}$. $\mathcal{P}_{\text{seek}}$ will also have the state $(q_{in}, 0)$, which is a part of $\mathcal{P}_{\text{sim}}$ as well. Similar to $\mathcal{P}_{\text{sim}}$, we can define base and grade of a state.

The initial set of states will be $\{(a, i) : 0 \leq i \leq k\}$. For each $0 \leq i \leq k$, Σ_{seek} will have two letters: ht_i and $\overline{\mathit{ht}}_i$. Σ_{seek} will also have another additional letter: *transfer*.

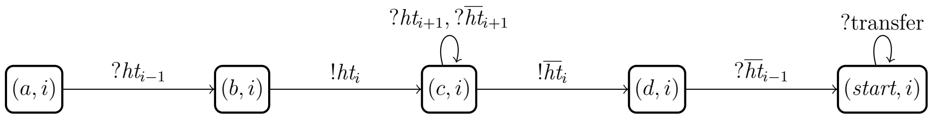
Transitions. Before we define the set of transitions, we make the same convention for $\mathcal{P}_{\text{seek}}$ that we had made in Remark 7 for $\mathcal{P}_{\text{sim}}$. Having stated this, we now describe the transitions:

- For the case of $i = 0$, we have the following transitions:



$$\begin{tikzcd}
(a,0) \arrow[r, "!\mathit{ht}_0"] & (c,0) \arrow[loop above, "?\mathit{ht}_1{,}\,?\overline{\mathit{ht}}_1"] \arrow[r, "!\overline{\mathit{ht}}_0"] & (\mathit{start},0) \arrow[r, "!\text{transfer}"] & (q_{in},0)
\end{tikzcd}$$

- For the case of $1 \leq i \leq k$, we have the following transitions: (The self-loops over the state (c, i) are not included when $i = k$).



$$\begin{tikzcd}
(a,i) \arrow[r, "?\mathit{ht}_{i-1}"] & (b,i) \arrow[r, "!\mathit{ht}_i"] & (c,i) \arrow[loop above, "?\mathit{ht}_{i+1}{,}\,?\overline{\mathit{ht}}_{i+1}"] \arrow[r, "!\overline{\mathit{ht}}_i"] & (d,i) \arrow[r, "?\overline{\mathit{ht}}_{i-1}"] & (\mathit{start},i) \arrow[loop above, "?\text{transfer}"]
\end{tikzcd}$$

Intuition behind the transitions. Let us give a brief intuition behind the transitions in the case of $k = 2$. A node n_0 which is at $(a, 0)$ aims to become the root of the good initial configuration that the seeker protocol should find, and so broadcasts ht_0, letting its neighbors know that it wants to be the root of the good subgraph. If any neighbor of n_0 is not in $(a, 1)$ then it immediately moves to a state with base *finish*. Otherwise, the set of all neighbors in $(a, 1)$ move to $(b, 1)$. From here, all of these nodes can broadcast ht_1, letting their neighbors know that they now want to become a child of the root. All these messages will also be received n_0 which will use the self-loop at $(c, 0)$ to ignore these messages. All the nodes which receive a ht_1 message can either move to a state with base *finish* or move to $(b, 2)$, from where they can broadcast ht_2 and thereby move to $(c, 2)$. At this point, we must have a tree subgraph in which n_0 is labelled by $(c, 0)$, its children are labelled by $(c, 1)$ and its children are labelled by $(c, 2)$.

Now the nodes labelled by $(c, 2)$ can all broadcast $\overline{ht}_2$, then the nodes labelled by $(c, 1)$ can all broadcast $\overline{ht}_1$ and then the node n_0 can broadcast $\overline{ht}_0$. This leads to a tree subgraph where n_0 is labelled by $(\mathit{start}, 0)$, its children are labelled by $(\mathit{start}, 1)$ and its children are labelled by $(\mathit{start}, 2)$. Now, n_0 can broadcast the letter "transfer" and move into $(q_{in}, 0)$, thereby *transferring* the control over to the simulator protocol $\mathcal{P}_{\text{sim}}$. In this manner, $\mathcal{P}_{\text{seek}}$ has found a good initial subgraph in which to run $\mathcal{P}_{\text{sim}}$.

Proof of correctness. Let $\mathcal{P} = (Q_{\text{seek}} \cup Q_{\text{sim}}, I_{\text{seek}}, \Sigma_{\text{seek}} \cup \Sigma_{\text{sim}}, \delta_{\text{seek}} \cup \delta_{\text{sim}})$ be the protocol obtained by taking the union of the seeker and the simulator protocols, such that the initial set of states is the initial set of states of the seeker protocol. Similar to the intuition given above, the protocol $\mathcal{P}$ first runs the seeker protocol till a node with label $(q_{in}, 0)$ is reached, at which point it runs the simulator protocol. The following lemma tells us that if a node gets labelled by $(q_{in}, 0)$ while running $\mathcal{P}$, then with that node as the root, there is a good initial configuration for the simulator protocol $\mathcal{P}_{\text{sim}}$. This then allows us the protocol $\mathcal{P}$ to run the simulator protocol on this good initial configuration.

▶ **Lemma 11.** *Suppose $\gamma \xrightarrow{*} \gamma' \xrightarrow{\mathsf{n}, transfer} \eta$ is an execution of $\mathcal{P}$. After removing all nodes whose label's base is finish in η, the connected component containing the node n is a good initial configuration for the simulator protocol $\mathcal{P}_{sim}$.*

Proof sketch. First, let us focus on the execution $\gamma \xrightarrow{*} \gamma'$. By definition of an execution, γ is an initial configuration for the protocol $\mathcal{P}_{\text{seek}}$ and so all the nodes in γ have their labels in the set $\{a_i : 0 \leq i \leq k\}$.

Let T be the connected component containing the node n in γ' after removing all nodes whose base is *finish*. Let $F := \{(\mathit{start}, i) : 0 \leq i \leq k\}$. First, we show that all nodes in T must have labels from the set F. Suppose there is a node n' in T whose label is not in F. Pick such an n' which is at the shortest distance from n and let $\mathsf{n} = \mathsf{n}_0, \mathsf{n}_1, \mathsf{n}_2, \ldots, \mathsf{n}_l, \mathsf{n}'$ be a shortest path from n to n'.

By a generalization of the argument given in Example 2, we can prove by induction that for each $1 \leq i \leq l$, the label of each n_i in T is (start, i) and the only neighbor of n_i which was labelled by $(c, i-1)$ at some point during the run is n_{i-1}. Using this, we can then show that n' must have moved to $(\mathit{start}, l+1)$ at some point during the run.

By assumption, the label of n' is not $(\mathit{start}, l+1)$ in T, and so it must moved out of $(\mathit{start}, l+1)$ to some state of the simulator protocol. By analysing the constructed protocol $\mathcal{P}$, we can then prove that n' must have received two ht_l messages. But any node that receives two ht_l messages must necessarily move to a state with base *finish*, contradicting the fact that $\mathsf{n}' \in T$.

Having proved that every node in T has its label in F, we can then show by examining the structure of the transitions, that T must be a tree of height atmost k such that n_0 is labelled by $(\mathit{start}, 0)$ and all nodes at distance i from n_0 are labelled by (start, i). This then implies that after removing all nodes with base *finish* in η, the connected component containing the node n is a good initial configuration for the simulator protocol. ◀

▶ **Theorem 12.** *The state q_{in} can cover q_f in the NCS $\mathcal{N}$ iff the state $(q_f, 0)$ can be covered from any initial configuration in $\mathcal{T}_{2k}(\mathcal{P})$.*

Proof sketch. Due to lack of space, we focus only on the right to left implication. Suppose $\gamma \xrightarrow{*} \gamma'$ is an execution of $\mathcal{P}$ such that some node n in γ' is labelled by $(q_f, 0)$. Let γ_0 be the configuration along this run when the node n first got the label $(q_{in}, 0)$. (Notice that such a configuration must exist because of the construction of $\mathcal{P}$). By Lemma 11, in γ_0, if we remove all nodes whose base is *finish*, then we get a good initial configuration T for $\mathcal{P}_{\text{sim}}$ with n as the root. Notice that no node with base *finish* can ever broadcast a message. Hence, in the run $\gamma_0 \xrightarrow{*} \gamma'$, none of the nodes in T ever receive a message from any node outside of T. It follows that we can restrict the run $\gamma_0 \xrightarrow{*} \gamma'$ to only the subtree T, to get a run of $\mathcal{P}_{\text{sim}}$ starting at a good initial configuration and covering $(q_f, 0)$. By Theorem 10, we get that q_f can be covered from q_{in} in $\mathcal{N}$. ◀

Hence, we get,

▶ **Corollary 13.** *Bounded-Path-Cover is $\mathbf{F}_{\epsilon_0}$-hard.*

6 Upper bound for Bounded-Path-Cover

In this section, we give a sketch of the proof that Bounded-Path-Cover is in $\mathbf{F}_{\epsilon_0}$. Let $\mathcal{P} = (Q, I, \Sigma, \Delta)$ be a fixed protocol.

▶ **Definition 14.** *Let $\gamma_1 = (\mathsf{N}_1, \mathsf{E}_1, \mathsf{L}_1)$ and $\gamma_2 = (\mathsf{N}_2, \mathsf{E}_2, \mathsf{L}_2)$ be two configurations of $\mathcal{P}$. We say that γ_1 is an induced subgraph of γ_2 (denoted by $\gamma_1 \preceq_{is} \gamma_2$) if there is a label preserving injection h from N_1 to N_2 such that $(\mathsf{n}, \mathsf{n}') \in \mathsf{E}_1$ if and only if $(h(\mathsf{n}), h(\mathsf{n}')) \in \mathsf{E}_2$.*

It is known that, for any $k \geq 1$, the set of all k-path bounded configurations of $\mathcal{P}$ is a well-quasi ordering under the induced subgraph relation $\preceq_{\text{is}}$ (Theorem 2.2 of [10]). Using this fact, the authors of [9] show that for every k, the transition system $\mathcal{T}_k(\mathcal{P})$ is a *well-structured transition system* (WSTS) and then apply the generic backward exploration algorithm for WSTS (See [20, 13]) to prove that Bounded-Path-Cover is decidable. By using the standard and generic complexity arguments for WSTS (See [20, 13, 21]), an upper bound on the running time of their procedure simply boils down to estimating the length of *controlled bad sequences* of k-path bounded configurations under the induced subgraph relation.

Let $H : \mathbb{N} \to \mathbb{N}$ be the successor function and let $n \in \mathbb{N}$. For each $i \in \mathbb{N}$, let H^i denote the i-fold application of H to itself i times, with H^0 being the identity function.

▶ **Definition 15.** *A sequence $\gamma_0, \gamma_1, \ldots,$ of k-path bounded configurations is (H, n)-controlled bad if the number of nodes in each γ_i is at most $H^i(n)$ and $\gamma_i \not\preceq_{is} \gamma_j$ for any $i < j$.*

Our main result is an upper bound on the length of (H, n)-controlled bad sequences of k-path bounded configurations, by embedding these configurations into the well-quasi ordering of *generalized priority alphabets* (See [16]). This encoding is inspired by a similar encoding given for bounded depth trees in Section 8.1 of [16]. This result then allows us to prove that

▶ **Theorem 16.** *Bounded-Path-Cover is in $\mathbf{F}_{\epsilon_0}$ and hence $\mathbf{F}_{\epsilon_0}$-complete.*

References

1 Sergio Abriola, Santiago Figueira, and Gabriel Senno. Linearizing well quasi-orders and bounding the length of bad sequences. *Theor. Comput. Sci.*, 603:3–22, 2015. doi:10.1016/j.tcs.2015.07.012.

2 Nathalie Bertrand, Patricia Bouyer, and Anirban Majumdar. Reconfiguration and message losses in parameterized broadcast networks. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: https://lmcs.episciences.org/7280.

3 Michael Blondin. The abcs of petri net reachability relaxations. *ACM SIGLOG News*, 7(3):29–43, 2020. doi:10.1145/3436980.3436984.

4 Michael Blondin, Alain Finkel, Christoph Haase, and Serge Haddad. The logical view on continuous petri nets. *ACM Trans. Comput. Log.*, 18(3):24:1–24:28, 2017. `doi:10.1145/3105908`.

5 Michael Blondin and Christoph Haase. Logics for continuous reachability in petri nets and vector addition systems with states. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. `doi:10.1109/LICS.2017.8005068`.

6 Pierre Chambart and Philippe Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 205–216, 2008. `doi:10.1109/LICS.2008.47`.

7 Wojciech Czerwinski, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for petri nets is not elementary. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 24–33. ACM, 2019. `doi:10.1145/3313276.3316369`.

8 Normann Decker and Daniel Thoma. On freeze LTL with ordered attributes. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2016. `doi:10.1007/978-3-662-49630-5_16`.

9 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR 2010 - Concurrency Theory, 21th International Conference,*, pages 313–327, 2010. `doi:10.1007/978-3-642-15375-4_22`.

10 Guoli Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16(5):489–502, 1992. `doi:10.1002/jgt.3190160509`.

11 Jacob Elgaard, Nils Klarlund, and Anders Møller. Mona 1.x: New techniques for ws1s and ws2s. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, pages 516–520, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

12 Javier Esparza, Ruslán Ledesma-Garza, Rupak Majumdar, Philipp J. Meyer, and Filip Niksic. An smt-based approach to coverability analysis. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2014. `doi:10.1007/978-3-319-08867-9_40`.

13 Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science*, pages 269–278, 2011. `doi:10.1109/LICS.2011.39`.

14 Estíbaliz Fraca and Serge Haddad. Complexity analysis of continuous petri nets. *Fundam. Informaticae*, 137(1):1–28, 2015. `doi:10.3233/FI-2015-1168`.

15 Christoph Haase and Simon Halfon. Integer vector addition systems with states. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2014. `doi:10.1007/978-3-319-11439-2_9`.

16 Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen. The power of priority channel systems. *Log. Methods Comput. Sci.*, 10(4), 2014. `doi:10.2168/LMCS-10(4:4)2014`.

17 Albert R. Meyer. Weak monadic second order theory of succesor is not elementary-recursive. In Rohit Parikh, editor, *Logic Colloquium*, pages 132–154, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

18 Sylvain Schmitz. Complexity bounds for ordinal-based termination - (invited talk). In *Reachability Problems - 8th International Workshop, RP 2014*, pages 1–19, 2014. doi:10.1007/978-3-319-11439-2_1.
19 Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Trans. Comput. Theory*, 8(1):3:1–3:36, 2016. doi:10.1145/2858784.
20 Sylvain Schmitz and Philippe Schnoebelen. Multiply-recursive upper bounds with higman's lemma. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011*, pages 441–452, 2011. doi:10.1007/978-3-642-22012-8_35.
21 Sylvain Schmitz and Philippe Schnoebelen. The power of well-structured systems. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013. doi:10.1007/978-3-642-40184-8_2.

A Appendix

A.1 Proofs for Section 6

First, let us describe the backward exploration algorithm for solving Bounded-Path-Cover that is given in Section 5 of [9]. Given a protocol $\mathcal{P} = (Q, I, \Sigma, \delta)$, a state f and a number k, we consider the set of all configurations in $\mathcal{T}_k(\mathcal{P})$ with the induced subgraph ordering $\preceq_{\text{is}}$. Given a set S of $\mathcal{T}_k(\mathcal{P})$ we let $\uparrow S := \{\gamma' : \exists \gamma \in S, \gamma \preceq_{\text{is}} \gamma'\}$. A set S is called *upward-closed* if $S = \uparrow S$.

In Section 5 of [9], the following results are proved about $\mathcal{T}_k(\mathcal{P})$:

- If S is upward-closed, then there exists a finite set B such that $\uparrow B = S$. Such a B will be called the basis of S.
- If S is upward-closed and if $Pre(S)$ is the set of all configurations $\gamma' \in \mathcal{T}_k(\mathcal{P})$ such that there is a configuration $\gamma \in S$ with $\gamma' \to \gamma$, then $S \cup Pre(S)$ is upward-closed. Moreover, given a basis B of S, we can compute a basis B' of $S \cup Pre(S)$ such that the number of nodes of each configuration in B' is at most one more than the maximum number of nodes in any configuration of B.

In Theorem 5 of [9] it is shown that the following algorithm terminates and is correct for Bounded-Path-Cover : Construct a sequence of finite sets $B_0, B_1, \ldots,$ such that each $B_i \subseteq \mathcal{T}_k(\mathcal{P})$, B_0 is the single node configuration labelled by f and B_{i+1} is a basis for $\uparrow B_i \cup Pre(\uparrow B_i)$. The algorithm then finds the first m such that $\uparrow B_m = \uparrow B_{m+1}$ and checks if there is an initial configuration in $\uparrow B_m$.

The running time complexity of the algorithm is mainly dominated by the length of the sequence $B_0, B_1, \ldots, B_m$. Since m is the first index such that $\uparrow B_m = \uparrow B_{m+1}$, we can find a minimal element $\gamma_i \in \uparrow B_{i+1} \setminus \uparrow B_i$ for each $i < m$.

Consider the sequence $\gamma_0, \ldots, \gamma_{m-1}$. Notice that $\gamma_i \not\preceq_{\text{is}} \gamma_j$ for any $j > i$ and further the number of nodes in each γ_i is at most $H^i(1)$, where H is the successor function. It follows that $\gamma_0, \ldots, \gamma_{m-1}$ is a controlled bad sequence. Our main result is that

▶ **Lemma 17.** *The length of (H, n)-controlled bad sequences over k-path bounded configurations of $\mathcal{P}$ is upper bounded by the function $F_{\epsilon_0}(p(|Q|, k, n))$.*

Here F_{ϵ_0} is the *fast-growing* function at level ϵ_0 and p is some fixed primitive recursive function. For our purposes, we do not need the actual definition of F_{ϵ_0}, but we only need to know that $\mathbf{F}_{\epsilon_0}$ contains the set of problems whose running time is upper bounded by the function F_{ϵ_0} composed with any primitive recursive function (See [19]). By the lemma above and the fact that the running time complexity of the algorithm for Bounded-Path-Cover is primarily dominated by the length of $(H, 1)$-controlled bad sequences we get,

▶ **Theorem 18.** *Bounded-Path-Cover is in* $\mathbf{F}_{\epsilon_0}$.

All that suffices is to prove Lemma 17. To do so, we will reduce the problem of estimating the length of controlled bad sequences over k-path bounded configurations to the problem of estimating the length of controlled bad sequences over another well-quasi order for which we already know upper bounds. We now proceed to recall this well-quasi order as it is defined in [16].

Generalized priority alphabets

Given a number $k \in \mathbb{N}$ called the priority level and a finite set Γ, a *generalised priority alphabet* is the set $\Sigma_{\Gamma,k} := \{(a,i) : a \in \Gamma, 0 \leq i \leq k\}$. Given $m = (a,i) \in \Sigma_{\Gamma,k}$, we say that i is the priority of m. Then for $x, y \in \Sigma^*_{\Gamma,k}$, we say that $x \sqsubseteq_{\Gamma,k} y$ if $x = (a_1,i_1),(a_2,i_2),\dots,(a_l,i_l)$ where each $(a_j,i_j) \in \Sigma_{\Gamma,k}$ and $y = y_1(a_1,i_1)y_2(a_2,i_2)y_3\dots y_l(a_l,i_l)$ such that $\forall 1 \leq j \leq l$, we have $y_j \in \Sigma^*_{\Gamma,i_j}$, i.e., x can be obtained from y by removing subwords in such a manner so that the priority of each removed subword is not bigger than the first preserved letter to its right. It is known that for every k and Γ, the ordering $\sqsubseteq_{\Gamma,k}$ is a well-quasi ordering. (Theorem 3.6 of [16]). Now, similar to controlled bad sequences for k-path bounded configurations, we can define (a slightly different notion of) controlled bad sequences for words over $\Sigma_{\Gamma,k}$. Let $Sq : \mathbb{N} \to \mathbb{N}$ be the squaring function and let Sq^i denote the squaring function composed with itself i times.

▶ **Definition 19.** *A sequence $w_0, w_1, \dots,$ of words over $\Sigma_{\Gamma,k}$ is (Sq,n)-controlled bad if the length of each w_i is at most $Sq^i(n)$ and $w_i \not\sqsubseteq_{\Gamma,k} w_j$ for any $i < j$.*

Encoding k-path bounded graphs using generalized priority alphabets

A labelled k-path bounded graph is any graph $G = (\mathsf{N},\mathsf{E},\mathsf{L})$ such that there is a labelling function $\mathsf{L} : \mathsf{N} \to A$ for some some finite set A. (Notice that the set of k-path bounded configurations of a protocol is a labelled k-path bounded graph where A is the set of states of the protocol). We have the following theorem regarding labelled k-path bounded graphs.

▶ **Theorem 20** (Lemma 2.1 of [10]). *Suppose G is a labelled k-path bounded graph for $k \geq 1$. Then there is a node n such that every connected component of $G \setminus \{\mathsf{n}\}$ is a labelled $(k-1)$-path bounded graph.*

This theorem suggests the following inductive encoding of labelled k-path bounded graphs as strings over a priority alphabet: Let $G = (\mathsf{N},\mathsf{E},\mathsf{L})$ be any labelled graph with labelling function $\mathsf{L} : \mathsf{N} \to A$ where A is some finite set. Let $\mathsf{e}, \bar{\mathsf{e}}$ be two symbols not in the finite set A and let $A^k := \cup_{0 \leq i \leq k} A \times \{\mathsf{e},\bar{\mathsf{e}}\}^i$. Notice that $A^0 := A$. By induction on k, we will now define a string $\langle G \rangle \in \Sigma_{A^k,k}$.

Base case: If G is a 0-path bounded configuration, then G is a single node n and can be encoded as $(\mathsf{L}(\mathsf{n}),0) \in \Sigma^*_{A^0,0}$.

Induction step: Suppose G is a k-path bounded configuration for some $k \geq 1$ such that G is not $(k-1)$-path bounded. Let n be a vertex such that all the connected components $C_1,\dots,C_l$ of $G \setminus \{\mathsf{n}\}$ are $(k-1)$-path bounded configurations. (Such a node exists by Theorem 20). For every node n' in every C_i, first change its label from $\mathsf{L}(\mathsf{n}')$ to $(\mathsf{L}(\mathsf{n}'),\mathsf{e})$ if n' is a neighbor of n in G and otherwise change its label to $(\mathsf{L}(\mathsf{n}'),\bar{\mathsf{e}})$. Call these new labelled graphs as $C^{\mathsf{n}}_1,\dots,C^{\mathsf{n}}_l$.

By induction hypothesis, for each C^{n}_i, we have a string $\langle C^{\mathsf{n}}_i \rangle \in \Sigma^*_{((A\times\{\mathsf{e},\bar{\mathsf{e}}\})^{k-1},k-1)} \subseteq \Sigma^*_{A^k,k-1}$. We now let $\langle G \rangle := \langle C^{\mathsf{n}}_1 \rangle(\mathsf{L}(\mathsf{n}),k)\langle C^{\mathsf{n}}_2 \rangle(\mathsf{L}(\mathsf{n}),k)\dots\langle C^{\mathsf{n}}_l \rangle(\mathsf{L}(\mathsf{n}),k)$.

Notice that if G is a labelled k-path bounded graph which is not $(k-1)$-path bounded, then $\langle G\rangle$ is of the form $\langle C_1^{\mathsf{n}}\rangle(a,k)\langle C_2^{\mathsf{n}}\rangle(a,k)\dots\langle C_l^{\mathsf{n}}\rangle(a,k)$ where 1) a is the label of some node n in G, 2) $C_1,\dots,C_l$ are connected components of $G\setminus\{\mathsf{n}\}$ which are labelled $(k-1)$-path bounded subgraphs of G. This will be called the *decomposition* of $\langle G\rangle$ and the node n will be called its *crown*.

We then have the following lemma:

▶ **Lemma 21.** *If G and H are such that $\langle G\rangle \sqsubseteq_{A^k,k} \langle H\rangle$ then $G \preceq_{is} H$.*

Proof. Notice that if $\langle G\rangle \sqsubseteq_{A^k,k} \langle H\rangle$, then the highest priority appearing in $\langle G\rangle$ and $\langle H\rangle$ must be the same, which, without loss of generality, we can assume to be k.

We prove the lemma by induction on k. The base case of 0 is clear.

For the induction step, let $\langle C_1^{\mathsf{n}}\rangle(a,k)\langle C_2^{\mathsf{n}}\rangle(a,k)\dots\langle C_m^{\mathsf{n}}\rangle(a,k)$ be the decomposition of $\langle G\rangle$ with crown n and let $\langle D_1^{\mathsf{n}'}\rangle(a',k)\langle D_2^{\mathsf{n}'}\rangle(a',k)\dots\langle D_n^{\mathsf{n}'}\rangle(a',k)$ be the decomposition of $\langle H\rangle$ with crown n'. Since $\langle G\rangle \sqsubseteq_{A^k,k} \langle H\rangle$, it must be the case that $a = a'$.

By definition of the $\sqsubseteq_{A^k,k}$ relation, it must be the case that for every C_j^{n}, there exists i_j such that $\langle C_j^{\mathsf{n}}\rangle \sqsubseteq_{A^k,k-1} \langle D_{i_j}^{\mathsf{n}'}\rangle$. Notice that the priority has reduced and we can apply the induction hypothesis to conclude that for each j, $C_j^{\mathsf{n}} \preceq_{\text{is}} D_{i_j}^{\mathsf{n}'}$ and so there exists a label preserving injection h_j from the nodes of C_j^{n} to the nodes of $D_{i_j}^{\mathsf{n}'}$ such that (u,v) is an edge in C_j^{n} iff $(h_j(u),h_j(v))$ is an edge in $D_{i_j}^{\mathsf{n}'}$.

Now, consider the following label preserving injection h from G to H: Map the crown n to the other crown n' and if n'' is any other node in any one of the connected components C_j, then map n'' to $h_j(\mathsf{n}'')$. Notice that if u and v are nodes in G which belong to the same connected component of $G\setminus\{\mathsf{n}\}$ then (u,v) is an edge in G iff $(h(u),h(v))$ is an edge in H. Similarly, if u and v are nodes in G which belong to different connected components of $G\setminus\{\mathsf{n}\}$ then $h(u)$ and $h(v)$ also belong to different connected components of $H\setminus\{\mathsf{n}'\}$ and so the statement "(u,v) is an edge in G iff $(h(u),h(v))$ is an edge in H" is vacously true.

Finally suppose $u = \mathsf{n}$ and v is some other node of G. Notice that the last field in the label of v is e if (u,v) is an edge in G and $\bar{\mathsf{e}}$ otherwise. By definition of h we have that $h(u) = \mathsf{n}'$ and also that the label of $h(v)$ is the same as v. But by definition of decomposition of $\langle H\rangle$, the last field in the label of $h(v)$ is e if $(\mathsf{n}',h(v))$ is an edge in H and $\bar{\mathsf{e}}$ otherwise. Hence, in this case as well, we have shown that (u,v) is an edge in G iff $(h(u),h(v))$ is an edge in H. This concludes the proof. ◀

Upper bound on the length of controlled bad sequences for k-path bounded configurations

Fix a protocol $\mathcal{P}$ with states Q and a number k and consider the set of configurations in $\mathcal{T}_k(\mathcal{P})$. By the previous lemma, we can infer that the length of the longest (H,n)-controlled bad sequence over the set of configurations of $\mathcal{T}_k(\mathcal{P})$ is at most the length of the longest (Sq,n)-controlled bad sequence over the generalized priority alphabet $\Sigma_{Q^k,k}$, which we know is at most $F_{\epsilon_0}(p(|Q|,k,n))$ where p is some primitive recursive function (Proposition 4.1 and Sections 4.1.1 and 4.1.2 of [16]). This then implies Lemma 17, which is what we wanted to prove.