Distributed local strategies in broadcast networks

Nathalie Bertrand, Paulin Fournier, Arnaud Sangnier

Distributed local strategies in broadcast networks*

Nathalie Bertrand[1], Paulin Fournier[2], and Arnaud Sangnier[3]

[1] Inria Rennes Bretagne Atlantique
[2] ENS Rennes, Univ Rennes 1
[3] LIAFA, Univ Paris Diderot, Sorbonne Paris Cité, CNRS

Abstract. We study the problems of reaching a specific control state, or converging to a set of target states, in networks with a parameterized number of identical processes communicating via broadcast. To reflect the distributed aspect of such networks, we restrict our attention to executions in which all the processes must follow the same *local strategy* that, given their past performed actions and received messages, provides the next action to be performed. We show that the reachability and target problems under such local strategies are NP-complete, assuming that the set of receivers is chosen non-deterministically at each step. On the other hand, these problems become undecidable when the communication topology is a clique. However, decidability can be regained for reachability under the additional assumption that all processes are bound to receive the broadcast messages.

1 Introduction

Parameterized models for distributed systems. Distributed systems are nowadays ubiquitous and distribution is one of the main paradigms in the conception of computing systems. Conceiving, analyzing, debugging and verifying such systems are tedious tasks which lately received an increased interest from the formal methods community. Considering parametric models with an unknown number of identical processes is a possible approach to tame distributed systems in which all processes share the same code. It has the advantages to allow one to establish the correctness of a system independently of the number of participants, and to ease bugs detection by the possibility to adapt the number of processes on demand.

In their seminal paper on distributed models with many identical entities [13], German and Sistla represent the behavior of a network by finite state machines interacting via 'rendezvous' communications. Variants have then been proposed, to handle different communication means, like broadcast communication [10], token-passing [5,2], message passing [4] or shared memory [11]. In his nice survey on such parameterized models [9], Esparza shows that minor changes, such as the presence or absence of a controller in the system, can drastically modify the complexity of the verification problems. Another perspective for parametric systems has been proposed by Bollig who studied their expressive power with respect to logics over Message Sequence Charts [3].

Broadcast protocols. Among the various parametric models of networks, broadcast protocols, originally studied by Esparza *et al.* [10], have later been analyzed under a new viewpoint, leading to new insights on the verification problems. Specifically, a low level model to represent the main characteristics of ad-hoc networks has been proposed [7]: the network is equipped with a communication topology and processes communicate via broadcast to their neighbors. It was shown that, given a protocol represented by a finite state machine performing internal actions, broadcasts and receptions of messages, the problem of deciding whether there exists an initial communication topology from which one of the processes can reach a specific control

* This work is partially supported by the ANR national research program ANR-14-CE28-0002 PACS.

state is undecidable. The same holds for the target problem, which asks whether all processes can converge to a set of target states. For both the reachability and the target problems, decidability can however be regained, by considering communication topologies that can change non-deterministically at any moment [6]. Another option to recover decidability of the reachability problem is to restrict the topologies to clique graphs [8], yielding a model equivalent to broadcast protocols.

Local distributed strategies. In this paper, we consider the reachability and target problems under a new perspective, which we believe could also be interesting for other 'many identical processes' models. In such models, the protocol executed by each process is often described by a finite state machine that can be non-deterministic. Therefore it may happen that two processes behave differently, even if they have the same information on what has happened so far in an execution. To forbid such non-truly distributed behaviors, we constrain processes to take the same decisions in case they fired the same sequence of transitions so far. We thus study the reachability and target problems in broadcast protocols restricted to *local strategies*. Interestingly, the notably difficult distributed controller synthesis problem [15] is relatively close to the problem of existence of a local strategy. Indeed a local strategy corresponds to a local controller for the processes executing the protocol and whose role is to resolve the non-deterministic choices.

Our contributions. First we show that the reachability and target problems under local strategies in reconfigurable broadcast networks are NP-complete. To obtain the upper bound, we prove that local strategies can be succinctly represented by a finite tree of polynomial size in the size of the input protocol. This result is particularly interesting, because deciding the existence of a local strategy is intrinsically difficult. Indeed, even with a fixed number of processes, the locality constraint cannot be simply tested on the induced transition system, and *a priori* local strategies may need unbounded memory. From our decidability proofs, we derive an upper bound on the memory needed to implement the local strategies. We also give cutoffs, *i.e.* upper bounds on the minimal number of processes needed to reach or converge to target states. Second we show the two problems to be undecidable when the communication topology is a clique. Moreover, the undecidability proof of the target problem holds even if the locality assumption is dropped. However, the reachability problem under local strategies in clique is decidable (yet non-primitive recursive) for complete protocols, *i.e.* when receptions are always possible from every state.

All proofs and details can be found in Appendix.

2 Networks of reconfigurable broadcast protocols

In this paper, given $i, j \in \mathbb{N}$ such that $i \leq j$, we let $[i..j] = \{k \mid i \leq k \leq j\}$. For a set E and a natural $\ell > 0$, let E^ℓ be the set of vectors $\mathbf{v}$ of size ℓ over E. For a vector $\mathbf{v} \in E^\ell$ and $i \in [1..\ell]$, $\mathbf{v}[i]$ is the i-th component of $\mathbf{v}$ and $|\mathbf{v}| = \ell$ its size. The notation $\mathcal{V}_E$ stands for the infinite set $\bigcup_{\ell \in \mathbb{N} \setminus \{0\}} E^\ell$ of all vectors over E. We will use the notation $\mathcal{M}(E)$ to denote the set of multi-sets over E.

2.1 Syntax and semantics

We begin by presenting our model for networks of broadcast protocols. Following [7,8,6], we assume that each process in the network executes the same (non-deterministic) broadcast protocol given by a finite state machine where the actions are of three kinds: broadcast of a

message m (denoted by $!!m$), reception of a message m (denoted by $??m$) and internal action (denoted by ε).

Definition 1. *A* broadcast protocol *is a tuple $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ with Q a finite set of control states; $q_0 \in Q$ the initial control state; Σ a finite message alphabet and $\Delta \subseteq Q \times (\{!!m, ??m \mid m \in \Sigma\} \cup \{\varepsilon\}) \times Q$ a finite set of edges.*

We denote by $A(q)$ the set $\{(q, \varepsilon, q') \in \Delta\} \cup \{(q, !!m, q') \in \Delta\}$ containing broadcasts and internal actions (called *active actions*) of $\mathcal{P}$ that start from state q. Furthermore, for each message $m \in \Sigma$, we denote by $R_m(q)$ the set $\{(q, ??m, q') \in \Delta\}$ containing the edges that start in state q and can be taken on reception of message m. We say that a broadcast protocol is *complete* if for every $q \in Q$ and every $m \in \Sigma$, $R_m(q) \neq \emptyset$. Whether protocols are complete or not may change the decidability status of the problems we consider (see Section 4).

We now define the semantics associated with such a protocol. It is common to represent the network topology by an undirected graph describing the communication links [6]. Since the topology may change at any time (such an operation is called reconfiguration), we decide here to simplify the notations by specifying, for each broadcast, a set of possible receivers that is chosen non-deterministically. The semantics of a network built over a broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ is given by a transition system $\mathcal{T}_\mathcal{P} = (\Gamma, \Gamma_0, \rightarrow)$ where $\Gamma = \mathcal{V}_Q$ is the set of configurations (represented by vectors over Q); $\Gamma_0 = \mathcal{V}_{\{q_0\}}$ is the set of initial configurations and $\rightarrow \subseteq \Gamma \times \mathbb{N} \times \Delta \times 2^{\mathbb{N}} \times \Gamma$ is the transition relation defined as follows: $(\gamma, p, \delta, R, \gamma') \in \rightarrow$ (also denoted by $\gamma \xrightarrow{p,\delta,R} \gamma'$) iff $|\gamma| = |\gamma'|$ and $p \in [1..|\gamma|]$ and $R \subseteq [1..|\gamma|] \setminus \{p\}$ and one of the following conditions holds:

Internal action: $\delta = (\gamma[p], \varepsilon, \gamma'[p])$ and $\gamma'[p'] = \gamma[p']$ for all $p' \in [1..|\gamma|] \setminus \{p\}$ (*the p-th process performs an internal action*).
Communication: $\delta = (\gamma[p], !!m, \gamma'[p])$ and $(\gamma[p'], ??m, \gamma'[p']) \in \Delta$ for all $p' \in R$ such that $R_m(\gamma[p']) \neq \emptyset$, and $\gamma'[p''] = \gamma[p'']$ for all $p'' \in [1..|\gamma|] \setminus (R \cup \{p\})$ and for all $p'' \in R$ such that $R_m(\gamma[p'']) = \emptyset$ (*the p-th process broadcasts m to all the processes in the reception set R*).

Obviously, when an internal action is performed, the reception set R is not taken into account. We point out the fact that the hypothesis $|\gamma| = |\gamma'|$ implies that the number of processes remains constant during an execution (there is no creation or deletion of processes). Yet, $\mathcal{T}_\mathcal{P}$ is an infinite state transition system since the number of possible initial configurations is infinite. An *execution* of $\mathcal{P}$ is then a finite sequence of consecutive transitions in $\mathcal{T}_\mathcal{P}$ of the form $\theta = \gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_\ell,\delta_\ell,R_\ell} \gamma_{\ell+1}$ and we denote by $\Theta[\mathcal{P}]$ (or simply Θ when $\mathcal{P}$ is clear from context) the set of all executions of $\mathcal{P}$. Furthermore, we use $nbproc(\theta) = |\gamma_0|$ to represent the number of processes involved in the execution θ.

2.2 Local strategies and clique executions

Our goal is to analyze executions of broadcast protocols under *local strategies*, where each process performs the same choices of edges according to its past history (*i.e.* according to the edges of the protocol it has fired so far).

A *finite path* in $\mathcal{P}$ is either the empty path, denoted by ϵ, or a non-empty finite sequence of edges $\delta_0 \cdots \delta_\ell$ such that δ_0 starts in q_0 and for all $i \in [1..\ell]$, δ_i starts in the state in which

δ_{i-1} ends. For convenience, we say that ϵ ends in state q_0. We write $\mathsf{Path}(\mathcal{P})$ for the set of all finite paths in $\mathcal{P}$.

For an execution $\theta \in \Theta[\mathcal{P}]$, we define, for every $p \in [1..nbproc(\theta)]$, the *past* of process p in θ (also referred to as its *history*), written $\pi_p(\theta)$, as the finite path in $\mathcal{P}$ that stores the sequences of edges of $\mathcal{P}$ taken by p along θ. We can now define local strategies which allow us to focus on the executions in which each process performs the same choice according to its past. A *local strategy* σ for $\mathcal{P}$ is a pair (σ_a, σ_r) of functions specifying, given a history, the next active action to be taken, and the reception edge to choose when receiving a message, respectively. Formally $\sigma_a : \mathsf{Path}(\mathcal{P}) \to (Q \times (\{!!m \mid m \in \Sigma\} \cup \{\varepsilon\}) \times Q)$ satisfies, for every $\rho \in \mathsf{Path}(\mathcal{P})$ ending in $q \in Q$, either $A(q) = \emptyset$ or $\sigma_a(\rho) \in A(q)$. Whereas $\sigma_r : \mathsf{Path}(\mathcal{P}) \times \Sigma \to (Q \times \{??m \mid m \in \Sigma\} \times Q)$ satisfies, for every $\rho \in \mathsf{Path}(\mathcal{P})$ ending in $q \in Q$ and every $m \in \Sigma$, either $R_m(q) = \emptyset$ or $\sigma_r(\rho, m) \in R_m(q)$.

Since our aim is to analyze executions where each process behaves according to the same local strategy, we now provide the formal definition of such executions. Given a local strategy σ, we say that a path $\delta_0 \cdots \delta_\ell$ *respects* σ if for all $i \in [0..\ell - 1]$, we have $\delta_{i+1} = \sigma_a(\delta_0 \dots \delta_i)$ or $\delta_{i+1} = \sigma_r(\delta_0 \cdots \delta_i, m)$ for some $m \in \Sigma$. Following this, an execution θ respects σ if for all $p \in [1..nbproc(\theta)]$, we have that $\pi_p(\theta)$ respects σ (i.e. we have that each process behaves as dictated by σ). Finally we define $\Theta_{\mathcal{L}} \subseteq \Theta$ as the set of *local executions* (also called local semantics), that is executions θ respecting a local strategy.

We also consider another set of executions where we assume that every message is broadcast to all the processes of the network (apart from the emitter). Formally, an execution $\theta = \gamma_0 \xrightarrow{p_0,\delta_0,R_0} \dots \xrightarrow{p_\ell,\delta_\ell,R_\ell} \gamma_{\ell+1}$ is said to be a *clique execution* if $R_k = [1, \dots, nbproc(\theta)] \setminus \{p_k\}$ for every $k \in [0..\ell]$. We denote by $\Theta_{\mathcal{C}}$ the set of clique executions (also called clique semantics). Note that clique executions of broadcast networks have been studied in [8] and that such networks correspond to broadcast protocols with no rendez-vous [10]. We will also consider the intersection of these subsets of executions and write $\Theta_{\mathcal{LC}}$ for the set $\Theta_{\mathcal{L}} \cap \Theta_{\mathcal{C}}$ of clique executions which respect a local strategy.

2.3 Verification problems

In this work we study the parameterized verification of the reachability and target properties for broadcast protocols restricted to local strategies. The first one asks whether there exists an execution respecting some local strategy and that eventually reaches a configuration where a given control state appears, whereas the latter problem seeks for an execution respecting some local strategy and that ends in a configuration where all the control states belong to a given target set. We consider several variants of these problems depending on whether we restrict to clique executions or not and to complete protocols or not.

For an execution $\theta = \gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_\ell,\delta_\ell,R_\ell} \gamma_{\ell+1}$, we denote by $\mathsf{End}(\theta) = \{\gamma_{\ell+1}[p] \mid p \in [1..nbproc(\theta)]\}$ the set of states that appear in the last configuration of θ. Reach$[\mathcal{S}]$, the parameterized reachability problem for executions restricted to $\mathcal{S} \in \{\mathcal{L}, \mathcal{C}, \mathcal{LC}\}$ is defined as follows:

Input: A broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ and a control state $q_F \in Q$.

Output: Does there exist an execution $\theta \in \Theta_{\mathcal{S}}$ such that $q_F \in \mathsf{End}(\theta)$?

In previous works, the parameterized reachability problem has been studied without the restriction to local strategies; in particular the reachability problem on unconstrained executions is in PTIME [6] and Reach$[\mathcal{C}]$ is decidable and Non-Primitive Recursive (NPR) [8,10] (it is in fact Ackermann-complete [16]).

TARGET[$\mathcal{S}$], the parameterized target problem for executions restricted to $\mathcal{S} \in \{\mathcal{L}, \mathcal{C}, \mathcal{LC}\}$ is defined as follows:
Input: A broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ and a set of control states $\mathfrak{T} \subseteq Q$.
Output: Does there exist an execution $\theta \in \Theta_{\mathcal{S}}$ such that $\mathsf{End}(\theta) \subseteq \mathfrak{T}$?
It has been shown that a generalization of the target problem, without restriction to local strategies, can be solved in NP [6]. In this work, we focus on executions under local strategies and we obtain the results presented in the following table:

REACH[$\mathcal{L}$]	REACH[$\mathcal{LC}$]	TARGET[$\mathcal{L}$]	TARGET[$\mathcal{LC}$]
NP-complete [Thm. 2]	Undecidable [Thm. 4] Decidable and NPR for complete protocols [Thm. 5]	NP-complete [Thm. 3]	Undecidable [Thm. 4]

Most of the problems listed in the above table are monotone: if, in a network of a given size, an execution satisfying the reachability or target property exists, then, in any bigger network, there also exists an execution satisfying the same property.

Proposition 1. *Let θ be an execution in $\Theta_{\mathcal{L}}$ [resp. $\Theta_{\mathcal{LC}}$]. For every $N \geq nbproc(\theta)$, there exists θ' in $\Theta_{\mathcal{L}}$ [resp. $\Theta_{\mathcal{LC}}$] such that $nbproc(\theta') = N$ and $\mathsf{End}(\theta) = \mathsf{End}(\theta')$ [resp. $\mathsf{End}(\theta) \subseteq \mathsf{End}(\theta')$].*

This monotonicity property allows us to look for cutoffs, *i.e.* minimal number of processes such that a local execution with a given property exists. In this work, we provide upper-bounds on these cutoffs for REACH[$\mathcal{L}$] (Proposition 4) and TARGET[$\mathcal{L}$] (Theorem 3.2). For REACH[$\mathcal{LC}$] restricted to complete protocols, given the complexity of the problem, such an upper-bound would be non-primitive recursive and thus would not be of any practical use.

2.4 Illustrative example

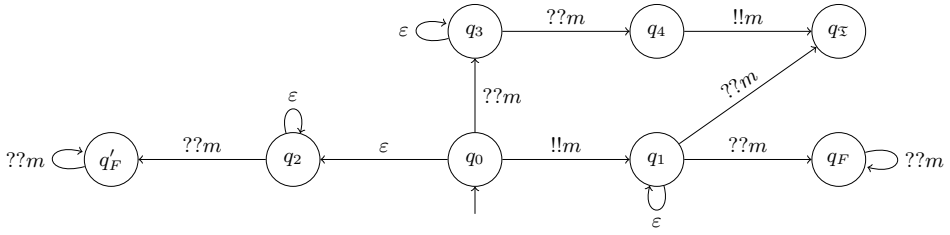

Fig. 1. Example of a broadcast protocol.

To illustrate the notions of local strategies and clique executions, we provide an example of a broadcast protocol in Fig. 1. On this protocol no clique execution can reach state q_F: as soon as a process in q_0 sends message m, all the other processes in q_0 receive this message, and move to q_3, because of the clique topology. An example of a clique execution is: $(q_0, q_0, q_0, q_0) \to (q_1, q_3, q_3, q_3)$ (where we omit the labels over $\to$). However, there exists a local execution reaching q_F: $(q_0, q_0) \to (q_1, q_0) \to (q_F, q_1)$. This execution respects a local strategy since, from q_0 with empty past, the first process chooses the edge broadcasting m with empty reception set and in the next step the second process, also with empty past, performs the same action, broadcasting the message m to the first process. On the other hand, no local

strategy permits to reach q'_F. Indeed, intuitively, to reach q'_F, in state q_0 one process with empty past needs to go to q_1 and another one to q_2, which is forbidden by locality. Finally $(q_0, q_0, q_0) \to (q_1, q_0, q_3) \to (q_1, q_1, q_4) \to (q_{\mathfrak{T}}, q_{\mathfrak{T}}, q_{\mathfrak{T}})$ is a local execution that targets the set $\mathfrak{T} = \{q_{\mathfrak{T}}\}$.

3 Verification problems for local executions

We begin with studying the parameterized reachability and target problems under local executions, *i.e.* we seek for a local strategy ensuring either to reach a specific control state, or to reach a configuration in which all the control states belong to a given set.

3.1 Solving Reach[$\mathcal{L}$]

To obtain an NP-algorithm for Reach[$\mathcal{L}$], we prove that there exists a local strategy to reach a specific control state if and only if there is a local strategy which can be represented thanks to a finite tree of polynomial size; the idea behind such a tree being that the paths in the tree represent past histories and the edges outgoing a specific node represent the decisions of the local strategy. The NP-algorithm will then consist in guessing such finite tree of polynomial size and verifying if it satisfies some conditions needed to reach the specified control state.

Representing strategies with trees. We now define our tree representation of strategies called strategy patterns, which are standard labelled trees with labels on the edges. Intuitively a strategy pattern defines, for some of the paths in the associated protocol, the active action and receptions to perform.

A *strategy pattern* for a broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ is a labelled tree $T = (N, n_0, E, \Delta, \mathsf{lab})$ with N a finite set of nodes, $n_0 \in N$ the root, $E \subseteq N \times N$ the edge relation and $\mathsf{lab} : E \to \Delta$ the edge-labelling function. Moreover T is such that if $e_1 \cdots e_\ell$ is a path in T, then $\mathsf{lab}(e_1) \cdots \mathsf{lab}(e_\ell) \in \mathsf{Path}(\mathcal{P})$, and for every node $n \in N$: there is at most one edge $e = (n, n') \in E$ such that $\mathsf{lab}(e)$ is an active action; and, for each message m, there is at most one edge $e = (n, n') \in E$ such that $\mathsf{lab}(e)$ is a reception of m.

Since all labels of edges outgoing a node share a common source state (due to the hypothesis on labelling of paths), the labelling function lab can be consistently extended to nodes by letting $\mathsf{lab}(n_0) = q_0$ and $\mathsf{lab}(n) = q$ for any $(n', n) \in E$ with $\mathsf{lab}((n', n)) = (q', a, q)$.

The strategy pattern represented in Fig. 2, for the broadcast protocol from Fig. 1, illustrates that strategy patterns somehow correspond to under-specified local strategies. For example, from node n_1 (labelled by q_1) no reception of message m is specified, and from node n_5 (labelled by q_4) no reception and no active action are specified.

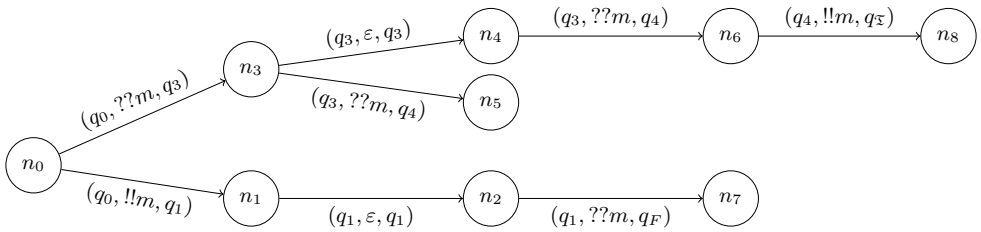

Fig. 2. A strategy pattern for the broadcast protocol depicted Fig. 1.

More generally, given $\mathcal{P}$ a broadcast protocol, and T a strategy pattern for $\mathcal{P}$ with edge-labelling function lab, a local strategy $\sigma = (\sigma_a, \sigma_r)$ for $\mathcal{P}$ is said to *follow* T if for every path $e_1 \cdots e_\ell$ in T, the path $\rho = \mathsf{lab}(e_1) \cdots \mathsf{lab}(e_\ell)$ in $\mathcal{P}$ respects σ. Notice that any strategy pattern admits at least one local strategy that follows it.

Reasoning on strategy patterns. We now show that one can test directly on a strategy pattern whether the local strategies following it can yield an execution reaching a specific control state. An *admissible strategy pattern* for $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ is a pair $(T, \prec)$ where $T = (N, n_0, E, \Delta, \mathsf{lab})$ is a strategy pattern for $\mathcal{P}$ and $\prec \subseteq N \times N$ is a strict total order on the nodes of T such that:

(1) for all $(n, n') \in E$ we have $n \prec n'$;
(2) for all $e = (n, n') \in E$, if $\mathsf{lab}(e) = (\mathsf{lab}(n), ??m, \mathsf{lab}(n'))$ for some $m \in \Sigma$, then there exists $e_1 = (n_1, n_1')$ in E such that $n_1' \prec n'$ and $\mathsf{lab}(e_1) = (\mathsf{lab}(n_1), !!m, \mathsf{lab}(n_1'))$.

In words, (1) states that $\prec$ respects the natural order on the tree and (2) that every node corresponding to a reception of m should be preceded by a node corresponding to a broadcast of m.

The example of strategy pattern on Fig. 2 is admissible with the order $n_i \prec n_j$ if $i < j$, whereas for any order including $n_3 \prec n_1$ it is not admissible (a broadcast of m should precede n_3). In general, given a strategy pattern T and a strict total order $\prec$, checking whether $(T, \prec)$ is admissible can be done in polynomial time (in the size of the pattern).

In order to state the relation between admissible strategy patterns and local strategies, we define $\mathsf{lab}(T) = \{\mathsf{lab}(n) \mid n \in N\}$ as the set of control states labelling nodes of T and $\mathsf{Occur}(\theta) = \{\gamma_i[p] \mid i \in [0..\ell+1] \text{ and } p \in [1..nbproc(\theta)]\}$ as the set of states that appear along an execution $\theta = \gamma_0 \to \cdots \to \gamma_{\ell+1}$. The next proposition tells us that admissible strategy patterns are necessary and sufficient to represent the sets of states that can be reached under local strategies.

Proposition 2. *For all $Q' \subseteq Q$, there exists an admissible strategy pattern $(T, \prec)$ such that $\mathsf{lab}(T) = Q'$ iff there exists a local strategy σ and an execution θ such that θ respects σ and $Q' = \mathsf{Occur}(\theta)$, furthermore σ follows T.*

Minimizing admissible strategy patterns. For $(T, \prec)$ an admissible strategy pattern, we denote by $\mathsf{last}(T, \prec)$ the maximal node w.r.t. $\prec$ and we say that $(T, \prec)$ is *q_F-admissible* if $\mathsf{lab}(\mathsf{last}(T, \prec)) = q_F$. We now show that there exist polynomial size witnesses of q_F-admissible strategy patterns. The idea is to keep only relevant edges that either lead to a node labelled by q_F or that permit a broadcast of a new message. Intuitively, a *minimal* strategy pattern guarantees that (1) there is a unique node labelled with q_F, (2) in every subtree there is either a node labelled by q_F or a broadcast of a new message (*i.e.* a broadcast of a message that has not been seen previously with respect to the order $\prec$), and (3) a path starting and ending in two different nodes labelled by the same state, cannot be compressed without losing a new broadcast or a path towards q_F (by compressing we mean replacing the first node on the path by the last one). These hypotheses allow us to seek only for q_F-admissible strategy patterns of polynomial size.

Proposition 3. *If there exists a q_F-admissible strategy pattern for $\mathcal{P}$, then there is one of size at most $(2|\Sigma|+1) \cdot (|Q|-1)$ and of height at most $(|\Sigma|+1) \cdot |Q|$.*

By Proposition 2, there exists an execution $\theta \in \Theta_{\mathcal{L}}$ such that $q_F \in \mathsf{Occur}(\theta)$ iff there exists a q_F-admissible strategy pattern and thanks to Proposition 3 it suffices to look only for

q_F-admissible strategy patterns of size polynomial in the size of the broadcast protocol. A non-deterministic polynomial time algorithm for Reach[$\mathcal{L}$] consists then in guessing a strategy pattern of polynomial size and an order and then verifying whether it is q_F-admissible.

Theorem 1. *Reach[$\mathcal{L}$] is in* NP.

We can furthermore provide bounds on the minimal number of processes and on the memory needed to implement local strategies. Given a q_F-admissible strategy pattern one can define an execution following the pattern such that each reception edge of the pattern is taken exactly once and active actions may be taken multiple times but in a row. Such an execution needs at most one process per reception edge. Together with the bound on the size of the minimal strategy patterns (see Proposition 3), this yields a cutoff property on the minimal size of network to reach the final state. Moreover the past history of every process in this execution is bounded by the depth of the tree, hence we obtain an upper bound on the size of the memory needed by each process for Reach[$\mathcal{L}$].

Proposition 4. *If there exists an execution $\theta \in \Theta_{\mathcal{L}}$ such that $q_F \in \mathsf{Occur}(\theta)$, then there exists an execution $\theta' \in \Theta_{\mathcal{L}}$ such that $q_F \in \mathsf{Occur}(\theta')$ and $nbproc(\theta') \leq (2|\Sigma| + 1) \cdot (|Q| - 1)$ and $|\pi_p(\theta')| \leq (|\Sigma| + 1) \cdot |Q|$ for every $p \in [1..nbproc(\theta')]$.*

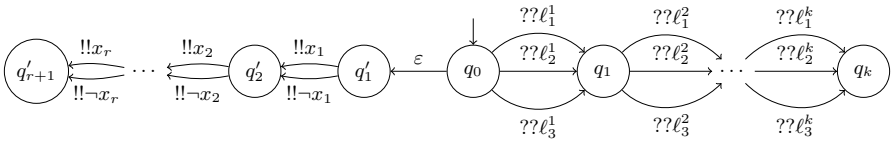

Fig. 3. Encoding a 3-SAT formula into a broadcast protocol.

By reducing 3-SAT, one can furthermore show Reach[$\mathcal{L}$] to be NP-hard. Let $\phi = \bigwedge_{1 \leq i \leq k} (\ell_1^i \vee \ell_2^i \vee \ell_3^i)$ be a 3-SAT formula such that $\ell_j^i \in \{x_1, \neg x_1, \ldots, x_r, \neg x_r\}$ for all $i \in [1..k]$ and $j \in \{1, 2, 3\}$. We build from ϕ the broadcast protocol $\mathcal{P}$ depicted at Fig. 3. Under this construction, ϕ is satisfiable iff there is an execution $\theta \in \Theta_{\mathcal{L}}$ such that $q_k \in \mathsf{Occur}(\theta)$. The local strategy hypothesis ensures that even if several processes broadcast a message corresponding to the same variable, all of them must take the same decision so that there cannot be any execution during which both x_i and $\neg x_i$ are broadcast. It is then clear that control state q_k can be reached if and only if each clause is satisfied by the set of broadcast messages. Together with Theorem 1, we obtain the precise complexity of Reach[$\mathcal{L}$].

Theorem 2. *Reach[$\mathcal{L}$] is* NP*-complete.*

3.2 Solving Target[$\mathcal{L}$]

Admissible strategy patterns can also be used to obtain an NP-algorithm for Target[$\mathcal{L}$]. As we have seen, given an admissible strategy pattern, one can build an execution where the processes visit all the control states present in the pattern. When considering the target problem, one also needs to ensure that the processes can afterwards be directed to the target set. To guarantee this, it is possible to extend admissible strategy patterns with another order on the nodes which ensures that (a) from any node there exists a path leading to the target

set and (b) whenever on this path a reception is performed, the corresponding message can be broadcast by a process that will only later on be able to reach the target.

We formalize now this idea. For $\mathfrak{T} \subseteq Q$ a set of states, a $\mathfrak{T}$-*coadmissible strategy pattern* for $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ is a pair $(T, \lhd)$ where $T = (N, n_0, E, \Delta, \mathsf{lab})$ is a strategy pattern for $\mathcal{P}$ and $\lhd \subseteq N \times N$ is a strict total order on the nodes T such that for every node $n \in N$ with $\mathsf{lab}(n) \notin \mathfrak{T}$ there exists an edge $e = (n, n') \in E$ with $n \lhd n'$ and either:

- $\mathsf{lab}(e) = (\mathsf{lab}(n), \varepsilon, \mathsf{lab}(n'))$ or,
- $\mathsf{lab}(e) = (\mathsf{lab}(n), !!m, \mathsf{lab}(n'))$ or,
- $\mathsf{lab}(e) = (\mathsf{lab}(n), ??m, \mathsf{lab}(n'))$ and there exists an edge $e_1 = (n_1, n_1') \in E$ such that $n \lhd n_1$, $n \lhd n_1'$ and $\mathsf{lab}(e_1) = (q_1, !!m, q_1')$.

Intuitively the order $\lhd$ in a $\mathfrak{T}$-coadmissible strategy pattern corresponds to the order in which processes must move along the tree towards the target; the conditions express that any node with label not in $\mathfrak{T}$ has an outgoing edge that is feasible. In particular, a reception of m is only feasible before all edges carrying the corresponding broadcast are disabled.

A strategy pattern T equipped with two orderings $\prec$ and $\lhd$ is said to be $\mathfrak{T}$-*biadmissible* whenever $(T, \prec)$ is admissible and $(T, \lhd)$ is $\mathfrak{T}$-coadmissible. To illustrate the construction

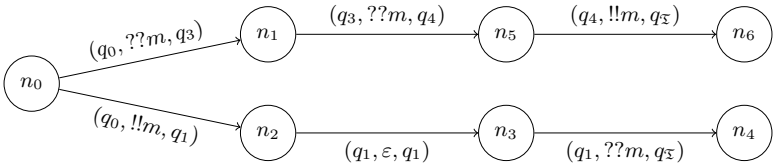

Fig. 4. A $\mathfrak{T}$-coadmissible strategy pattern on the example protocol of Fig. 1.

of $\mathfrak{T}$-coadmissble patterns, we give in Fig. 4 an example pattern, that, equipped with the natural order $n_i \lhd n_j$ iff $i < j$, is $\mathfrak{T}$-coadmissible for $\mathfrak{T} = \{q_\mathfrak{T}\}$. Indeed all leaves are labelled with a target state, and the broadcast edge $n_5 \xrightarrow{(q_4, !!m, q_\mathfrak{T})} n_6$ allows all processes to take the corresponding reception edges. This $\mathfrak{T}$-coadmissible pattern is in particular obtained from the execution $(q_0, q_0, q_0) \to (q_1, q_3, q_0) \to (q_1, q_3, q_0) \to (q_\mathfrak{T}, q_4, q_1) \to (q_\mathfrak{T}, q_4, q_1) \to (q_\mathfrak{T}, q_\mathfrak{T}, q_\mathfrak{T})$. Notice that $\lhd$ is not an admissible order, because $n_1 \lhd n_2$, however there are admissible orders for this pattern, for example the order $n_0 \prec n_2 \prec n_3 \prec n_4 \prec n_1 \prec n_5 \prec n_6$.

As for Reach$[\mathcal{L}]$, one can show polynomial size witnesses of $\mathfrak{T}$-*biadmissible* strategy patterns exist, yielding an NP-algorithm for Target$[\mathcal{L}]$. Also, the size of minimal $\mathfrak{T}$-*biadmissible* strategy patterns gives here also a cutoff on the number of processes needed to satisfy the target condition, as well as an upper bound on the memory size.

Theorem 3. *1.* Target$[\mathcal{L}]$ *is* NP*-complete.*

2. If there exists an execution $\theta \in \Theta_\mathcal{L}$ such that $\mathsf{End}(\theta) \subseteq \mathfrak{T}$, then there exists an execution $\theta' \in \Theta_\mathcal{L}$ such that $\mathsf{End}(\theta') \subseteq \mathfrak{T}$ and $nbproc(\theta') \leq 16|\Sigma| \cdot |Q| + 4|\Sigma| \cdot (|Q| - |\mathfrak{T}| + 1)$ and $|\pi_p(\theta')| \leq 4|\Sigma| \cdot |Q| + 2(|Q| - |\mathfrak{T}|) + 1$ for every $p \leq nbproc(\theta')$.

Remark 1. The NP-hardness derives from the fact that the target problem is harder than the reachability problem. To reduce Reach$[\mathcal{L}]$ to Target$[\mathcal{L}]$, one can add the broadcast of a new message from q_F, and its reception from any state to q_F.

Another consequence of this simple reduction is that TARGET[$\mathcal{L}$] in NP yields another proof that REACH[$\mathcal{L}$] is in NP, yet the two proofs of NP-membership allowed us to give an incremental presentation, starting with admissible strategy patterns, and proceeding with co-admissible strategy patterns.

4 Verification problems for local clique executions

4.1 Undecidability of Reach[$\mathcal{LC}$] and Target[$\mathcal{LC}$]

REACH[$\mathcal{LC}$] and TARGET[$\mathcal{LC}$] happen to be undecidable and for the latter, even in the case of complete protocols. The proofs of these two results are based on a reduction from the halting problem of a two counter Minsky machine (a finite program equipped with two integer variables which can be incremented, decremented and tested to zero). The main idea consists in both cases in isolating some processes to simulate the behavior of the machine while the other processes encode the values of the counters.

Thanks to the clique semantics we can in fact isolate one process. This is achieved by setting the first transition to be the broadcast of a message *start* whose reception makes all the other process change their state. Hence, thanks to the clique semantics, there is only one process that sends the message *start*, such process, called the controller, will be in charge of simulating the transitions of the Minsky machine. The clique semantics is also used to correctly simulate the increment and decrement of counters. For instance to increment a counter, the controller asks whether a process simulating the counter can be moved from state 0 to state 1 and if it is possible, relying on the clique topology only one such process changes its state (the value of the counter is then the number of processes in state 1). In fact, all the processes will receive the request, but the first one answering it, will force the other processes to come back to their original state, ensuring that only one process will move from state 0 to 1.

The main difficulty is that broadcast protocols (even under the clique semantics) cannot test the absence of processes in a certain state (which would be needed to simulate a test to 0 of one of the counters). Here is how we overcome this issue for TARGET[$\mathcal{LC}$]: the controller, when simulating a zero-test, sends all the processes with value 1 into a sink error state and the target problem allows to check for the reachability of a configuration with no process in this error state (and thus to test whether the controller has 'cheated', *i.e.* has taken a zero-test transition whereas the value of the associated counter was not 0). We point out that in this case, restricting to local executions is not necessary, we get in fact as well that TARGET[$\mathcal{C}$] is undecidable.

For REACH[$\mathcal{LC}$], the reduction is more tricky since we cannot rely on a target set of states to check that zero-test were faithfully simulated. Here in fact we will use two controllers. Basically, before sending a *start* message, some processes will be able to go to a waiting state (thanks to an internal transition) from which they can become controller and in which they will not receive any messages (this is where the protocol needs to be incomplete). Then we will use the locality hypothesis to ensure that two different controllers will simulate exactly the same run of the Minsky machine twice and with exactly the same number of processes encoding the counters. Restricting to local strategies guarantees the two runs to be identical, and the correctness derives from the fact that if in the first simulation the controller 'cheats' while performing a zero-test (and sending as before some processes encoding a counter value into a sink state), then in the second simulation, the number of processes encoding the counters

will be smaller (due to the processes blocked in the sink state), so that the simulation will fail (because there will not be enough processes to simulate faithfully the counter values).

Theorem 4. REACH[$\mathcal{LC}$] *is undecidable and* TARGET[$\mathcal{LC}$] *restricted to complete protocol is undecidable.*

The undecidability proof for REACH[$\mathcal{LC}$] strongly relies on the protocol being incomplete. Indeed, in the absence of specified receptions, the processes ignore broadcast messages and keep the same history, thus allowing to perform twice the same simulation of the run. In contrast, for complete protocols, all the processes are aware of all broadcast messages, therefore one cannot force the two runs to be identical. In fact, the reachability problem is decidable for complete protocols, as we shall see in the next section.

4.2 Decidability of Reach[$\mathcal{LC}$] for complete protocols

To prove the decidability of REACH[$\mathcal{LC}$] for complete protocols, we abstract the behavior of a protocol under local clique semantics by counting the possible number of different histories in each control state.

We identify two cases when the history of processes can differ (under local clique semantics): (1) When a process p performs a broadcast, its history is unique for ever (since all the other processes must receive the emitted message); (2) A set of processes sharing the same history can be split when some of them perform a sequence of internal actions and the others perform only a prefix of that sequence.

From a complete broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ we build an abstract transition system $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}} = (\Lambda, \lambda_0, \Rightarrow)$ where configurations count the number of different histories in each control state. More precisely the set of abstract configurations is $\Lambda = \mathcal{M}(Q \times \{\mathbf{m}, \mathbf{s}\} \times \{!!_{ok}, !!_{no}\}) \times \{\varepsilon, !!\}$. Abstract configurations are thus pairs where the first element is a multiset and the second element is a flag in $\{\varepsilon, !!\}$. The latter indicates the type of the next actions to be simulated (sequence of internal actions or broadcast): it prevents to simulate consecutively two incoherent sequences of internal actions (with respect to the local strategy hypothesis). For the former, an element $(q, \mathbf{s}, !!_{ok})$ in the multiset represents a single process (flag $\mathbf{s}$) in state q with a unique history which is allowed to perform a broadcast (flag $!!_{ok}$). An element $(q, \mathbf{m}, !!_{no})$ represents many processes (flag $\mathbf{m}$) in state q, all sharing the same unique history and none of them is allowed to perform a broadcast (flag $!!_{no}$). The initial abstract configuration λ_0 is then $(\{\{(q_0, \mathbf{m}, !!_{ok})\}\}, \varepsilon)$. In the sequel we will write $\mathbf{HM}$ for the set $\mathcal{M}(Q \times \{\mathbf{m}, \mathbf{s}\} \times \{!!_{ok}, !!_{no}\})$ of history multisets, so that $\Lambda = \mathbf{HM} \times \{\varepsilon, !!\}$, and typical elements of $\mathbf{HM}$ are denoted $\mathbb{M}$, $\mathbb{M}'$, etc.

In order to provide the definition of the abstract transition relation $\Rightarrow$, we need to introduce new notions, and notations. An ε-path ρ in $\mathcal{P}$ from q to q' is either the empty path (and in that case $q = q'$) or it is a non-empty finite path $\delta_0 \cdots \delta_n$ that starts in q, ends in q' and such that all the δ_i's are internal transitions.

An ε-path ρ in $\mathcal{P}$ is said to be a *prefix* of an ε-path ρ' if $\rho \neq \rho'$ and either ρ is the empty path or $\rho = \delta_0 \cdots \delta_n$ and $\rho' = \delta_0 \cdots \delta_n \delta_{n+1} \ldots \delta_{n+m}$ for some $m > 0$. Since we will handle multisets, let us give some convenient notations. Given E a set, and $\mathbb{M}$ a multiset over E, we write $\mathbb{M}(e)$ for the number of occurrences of element $e \in E$ in $\mathbb{M}$. Moreover, $card(\mathbb{M})$ stands for the cardinality of $\mathbb{M}$: $card(\mathbb{M}) = \sum_{e \in E} \mathbb{M}(e)$. Last, we will write $\oplus$ for the addition on multisets: $\mathbb{M} \oplus \mathbb{M}'$ is such that for all $e \in E$, $(\mathbb{M} \oplus \mathbb{M}')(e) = \mathbb{M}(e) + \mathbb{M}'(e)$.

The abstract transition relation $\Rightarrow \in \Lambda \times \Lambda$ is composed of two transitions relations: one simulates the broadcast of messages and the other one sequences of internal transitions. This will guarantee an alternation between abstract configurations flagged with ε and the ones flagged with !!. Let us first define $\Rightarrow_{!!} \subseteq (\mathbf{HM} \times \{!!\}) \times (\mathbf{HM} \times \{\varepsilon\})$ which simulates a broadcast. We have $(\mathbb{M}, !!) \Rightarrow_{!!} (\mathbb{M}', \varepsilon)$ iff there exists $(q_1, !!m, q_2) \in \Delta$ and $fl_1 \in \{\mathbf{s}, \mathbf{m}\}$ such that

1. $\mathbb{M}(q_1, fl_1, !!_{ok}) > 0$
2. there exists a family of functions G indexed by $(q, fl, b) \in Q \times \{\mathbf{m}, \mathbf{s}\} \times \{!!_{ok}, !!_{no}\}$, such that $G_{(q,fl,b)} : [1..\mathbb{M}(q, fl, b)] \to \mathbf{HM}$, and:

$$\mathbb{M}' = \{\{q_2, \mathbf{s}, !!_{ok}\}\} \oplus \bigoplus_{\{(q,fl,b) | \mathbb{M}(q,fl,b) \neq 0\}} \bigoplus_{i \in [1..\mathbb{M}(q,fl,b)]} G_{(q,fl,b)}(i)$$

   and such that for each (q, fl, b) verifying $\mathbb{M}(q, fl, b) \neq 0$, for all $i \in [1..\mathbb{M}(q, fl, b)]$, the following conditions are satisfied:
   (a) if $fl_1 = \mathbf{s}$, $card(G_{(q_1,fl_1,!!_{ok})}(1)) = 0$ and if $fl_1 = \mathbf{m}$, then there exists $q' \in Q$ such that $G_{(q_1,fl_1,!!_{ok})}(1) = \{\{(q', fl_1, !!_{ok})\}\}$ and such that $(q, ??m, q') \in \Delta$;
   (b) if $(q, fl, b) \neq (q_1, fl_1, !!_{ok})$ or $i \neq 1$, then there exists $q' \in Q$ such that $G_{(q,fl,b)}(i) = \{\{(q', fl, !!_{ok})\}\}$ and such that $(q, ??m, q') \in \Delta$.

Intuitively to provide the broadcast, we need to find a process which is 'allowed' to perform a broadcast and which is hence associated with an element $(q_1, fl_1, !!_{ok})$ in $\mathbb{M}$. The transition $(q_1, !!m, q_2)$ tells us which broadcast is simulated. Then the functions $G_{(q,fl,b)}$ associate with each element of the multiset $\mathbb{M}$ of the form (q, fl, b) a single element which can be reached thanks to a reception of the message m. Of course this might not hold for an element of the shape $(q_1, \mathbf{s}, !!_{ok})$ if it is the one chosen to do the broadcast since it represents a single process, and hence this element moves to q_2. Note however that if $fl_1 = \mathbf{m}$, then $(q_1, \mathbf{m}, !!_{ok})$ represents many processes, hence the one which performs the broadcast is isolated, but the many other ones have to be treated for reception of the message. Note also that we use here the fact that since an element $(q, \mathbf{m}, b)$ represents many processes with the same history, all these processes will behave the same way on reception of the message m.

We now define $\Rightarrow_{\varepsilon} \subseteq (\mathbf{HM} \times \{\varepsilon\}) \times (\mathbf{HM} \times \{!!\})$ which simulates the firing of sequences of ε-transitions. We have $(\mathbb{M}, \varepsilon) \Rightarrow_{\varepsilon} (\mathbb{M}', !!)$ iff there exists a family of functions F indexed by $(q, fl, b) \in Q \times \{\mathbf{m}, \mathbf{s}\} \times \{!!_{ok}, !!_{no}\}$, such that $F_{(q,fl,b)} : [1..\mathbb{M}(q, fl, b)] \to \mathbf{HM}$, and

$$\mathbb{M}' = \bigoplus_{\{(q,fl,b) | \mathbb{M}(q,fl,b) \neq 0\}} \bigoplus_{i \in [1..\mathbb{M}(q,fl,b)]} F_{(q,fl,b)}(i)$$

and such that for each (q, fl, b) verifying $\mathbb{M}(q, fl, b) \neq 0$, for all $i \in [1..\mathbb{M}(q, fl, b)]$, we have:

1. $card(F_{(q,fl,b)}(i)) \geq 1$ and if $fl = \mathbf{s}$, $card(F_{(q,fl,b)}(i)) = 1$;
2. If $F_{(q,fl,b)}(i)(q', fl', b') \neq 0$, then $fl' = fl$;
3. There exists a pair $(q_{!!}, fl_{!!}) \in Q \times \{\mathbf{m}, \mathbf{s}\}$ such that:
   - $F_{(q,fl,b)}(i)(q_{!!}, fl_{!!}, !!_{ok}) = 1$
   - for all $(q', fl') \neq (q_{!!}, fl_{!!})$ $F_{(q,fl,b)}(i)(q', fl', !!_{ok}) = 0$;
   - There exists a ε-path $\rho_{!!}$ from q to $q_{!!}$.
4. For all (q', fl') such that $F_{(q,fl,b)}(i)(q', fl', !!_{no}) = k > 0$, there exists k different ε-paths (strict) prefix of $\rho_{!!}$ from q to q'.

Intuitively the functions $F_{(q,fl,b)}$ associate with each element (q, fl, b) of the multiset $\mathbb{M}$ a set of elements that can be reached via internal transitions. We recall that each such element represents a set (or a singleton if $fl = \mathbf{s}$) of processes sharing the same history. Condition 1. states that if there are multiple processes ($fl = \mathbf{m}$) then they can be matched to more states in the protocol, but if it is single ($fl = \mathbf{s}$) it should be matched by an unique state. Condition 2. expresses that if an element in $\mathbb{M}$ represents many processes, then all its images represent as well many processes. Conditions 3. and 4. deal with the locality assumption. Precisely, condition 3. states that among all the elements of $\mathbb{M}'$ associated with an element of $\mathbb{M}$, one and only one should be at the end of a ε-path, and only one process associated with this element will be allowed to perform a broadcast. This justifies the use of the flag $!!_{ok}$. Last, condition 4. concerns all the other elements associated to this element of $\mathbb{M}$: their flag is set to $!!_{no}$ (they cannot perform a broadcast, because the local strategy will force them to take an internal transition), and their state should be on the previously mentioned ε-path.

As announced, we define the abstract transitive relation by $\Rightarrow = \Rightarrow_{\varepsilon} \cup \Rightarrow_{!!}$. Note that by definition we have a strict alternation of transitions of the type $\Rightarrow_{\varepsilon}$ and of the type $\Rightarrow_{!!}$. An *abstract local clique execution* of $\mathcal{P}$ is then a finite sequence of consecutive transitions in $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}}$ of the shape $\xi = \lambda_0 \Rightarrow \lambda_1 \cdots \Rightarrow \lambda_{\ell+1}$. As for concrete executions, if $\lambda_{\ell+1} = (\mathbb{M}_{\ell+1}, t_{\ell+1})$ we denote by $\mathsf{End}(\xi) = \{q \mid \exists fl \in \{\mathbf{m}, \mathbf{s}\}.\exists b \in \{!!_{ok}, !!_{no}\}.\mathbb{M}_{\ell+1}(q, fl, b) > 0\}$ the set of states that appear in the end configuration of ξ.

As an example, a possible abstract execution of the broadcast protocol from Fig. 1 is: $(\{\{(q_0, \mathbf{m}, !!_{ok})\}\}, \varepsilon) \Rightarrow (\{\{(q_0, \mathbf{m}, !!_{no}), (q_2, \mathbf{m}, !!_{no}), (q_2, \mathbf{m}, !!_{ok})\}\}, !!)$. This single-step execution represents that among the processes in q_0, some processes will take an internal action to q_2 and loop there with another internal action (they are represented by the element $(q_2, \mathbf{m}, !!_{ok})$), others will only move to q_2 taking a single internal action (they are represented by $(q_2, \mathbf{m}, !!_{no})$), and finally some processes will stay in q_0 (they are represented by $(q_0, \mathbf{m}, !!_{no})$); note that these processes cannot perform a broadcast, because due to the local strategy hypothesis, they committed to firing the internal action leading to q_2.

Another example of an abstract execution is: $(\{\{(q_0, \mathbf{m}, !!_{ok})\}\}, \varepsilon) \Rightarrow (\{\{(q_0, \mathbf{m}, !!_{ok})\}\}, !!)$ $\Rightarrow (\{\{(q_1, \mathbf{s}, !!_{ok}), (q_3, \mathbf{m}, !!_{ok})\}\}, \varepsilon) \Rightarrow (\{\{(q_1, \mathbf{s}, !!_{ok}), (q_3, \mathbf{m}, !!_{no}), (q_3, \mathbf{m}, !!_{ok})\}\}, \varepsilon)$. Here in the first step, no process performs internal actions, in the second step one of the processes in q_0 broadcasts m, moves to q_1 and we know that no other process will ever share the same history, it is hence represented by $(q_1, \mathbf{s}, !!_{ok})$; then all the other processes with the same history represented by $(q_0, \mathbf{m}, !!_{ok})$ must receive m and move to q_3, they are hence represented by $(q_3, \mathbf{m}, !!_{ok})$. The last step represents that some processes perform the internal action loop on q_3.

The definition of the abstract transition system $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}}$ ensures a correspondence between abstract local clique executions and local clique executions in $\mathcal{P}$. Formally:

Lemma 1. *Let $q_F \in Q$. There exists an abstract local clique execution ξ of $\mathcal{P}$ such that $q_F \in \mathsf{End}(\xi)$ iff there exists a local clique execution $\theta \in \Theta_{\mathcal{LC}}$ such that $q_F \in \mathsf{End}(\theta)$.*

Given the abstract transition system $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}}$, in order to show that Reach[$\mathcal{LC}$] is decidable, we then rely on the theory of well-structured transition systems [1,12]. Indeed, the natural order on abstract configurations is a well-quasi-order compatible with the transition relation $\Rightarrow$ of $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}}$ (bigger abstract configurations simulate smaller ones) and one can compute predecessors of upward-closed sets of configurations. This allows us to conclude that, in $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}}$, the set of all predecessors of a configuration where q_F appears is effectively computable, so that we can decide whether q_F is reachable in $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}}$, hence, thanks to the previous lemma, in $\mathcal{P}$.

We also show that Reach[$\mathcal{LC}$] is non-primitive recursive thanks to a PTIME reduction from Reach[$\mathcal{C}$] (which is Ackermann-complete [16]) to Reach[$\mathcal{LC}$]. We exploit the fact that the only difference between the semantics $\mathcal{C}$ and $\mathcal{LC}$ is that in the latter, processes with the same history take the same decision. We simulate this in $\mathcal{C}$ with a gadget which assigns a different history to each individual process at the beginning of the protocol making hence the reachability problem for $\mathcal{C}$ equivalent to the one with $\mathcal{LC}$ semantics.

Theorem 5. Reach*[$\mathcal{LC}$] restricted to complete protocols is decidable and* NPR.

5 Conclusion

We considered reconfigurable broadcast networks under local strategies that rule out executions in which processes with identical local history behave differently. Under this natural assumption for distributed protocols, the reachability and target problems are NP-complete. Moreover, we gave polynomial bounds on the cutoff and on the memory needed by strategies. When the communication topology is a clique, both problems become undecidable. Decidability is recovered for reachability if we further assume that protocols are complete.

To the best of our knowledge, this is the first attempt to take into account the local viewpoint of the processes in parameterized distributed systems. It could be interesting to study how the method we propose in this work can be adapted to parameterized networks equipped with other means of communication (such as rendez-vous [13] or shared memory [11]). In the future we also plan to deal with properties beyond simple reachability objectives, as for example linear or branching time properties.

References

1. Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Inf. Comput.*, 160(1-2):109–127, 2000.
2. Benjamin Aminof, Swen Jacobs, Ayrat Khalimov, and Sasha Rubin. Parameterized model checking of token-passing systems. In *VMCAI'14*, volume 8318 of *LNCS*, pages 262–281, 2014.
3. Benedikt Bollig. Logic for communicating automata with parameterized topology. In *CSL-LICS'14*, page 18. ACM, 2014.
4. Benedikt Bollig, Paul Gastin, and Jana Schubert. Parameterized verification of communicating automata under context bounds. In *RP'14*, volume 8762 of *LNCS*, pages 45–57, 2014.
5. Edmund M. Clarke, Muralidhar Talupur, Tayssir Touili, and Helmut Veith. Verification by network decomposition. In *CONCUR'04*, volume 3170 of *LNCS*, pages 276–291, 2004.
6. Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *FSTTCS'12*, volume 18 of *LIPIcs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
7. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR'10*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.

8. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi 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.
9. Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *STACS'14*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.
10. Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *LICS'99*, pages 352–359. IEEE Computer Society, 1999.
11. Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In *CAV'13*, volume 8044 of *LNCS*, pages 124–140, 2013.
12. Alain Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.
13. Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.
14. Marvin Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall International, 1967.
15. Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *FOCS'90*, pages 746–757. IEEE Computer Society, 1990.
16. Sylvain Schmitz and Philippe Schnoebelen. The power of well-structured systems. In *CONCUR'13*, volume 8052 of *LNCS*, pages 5–24. Springer, 2013.

A Complements for Section 2

A.1 Formal definition of past

Due to space constraints, we omitted the formal definition of past for processes, so we detail it here.

For an execution $\theta \in \Theta[\mathcal{P}]$, we inductively define for all $p \in [1..nbproc(\theta)]$ the *past* of process p in θ (also referred to as *history*), written $\pi_p(\theta)$, as follows: $\pi_p(\gamma_0) = \epsilon$ and $\pi_p(\gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_n,\delta_n,R_n} \gamma_{n+1})$ is equal to:

- $\pi_p(\gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_{n-1},\delta_{n-1},R_{n-1}} \gamma_n) \cdot \delta_n$ if $p_n = p$; We keep the active action performed by p.
- $\pi_p(\gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_{n-1},\delta_{n-1},R_{n-1}} \gamma_n) \cdot \delta$ if $p \in R$, $\delta = (\gamma_n[p], ??m, \gamma_{n+1}[p]) \in R_m(\gamma_n[p])$ and $\delta_n = (\gamma_n[p_n], !!m, \gamma_{n+1}[p_n])$; In the case of a broadcast received by p we keep the transition used for the reception.
- $\pi_p(\gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_{n-1},\delta_{n-1},R_{n-1}} \gamma_n)$ if $p \in R$ and $\delta_n = (\gamma_n[p_n], !!m, \gamma_{n+1}[p_n])$ but $R_m(\gamma_n[p]) = \emptyset$; In the case of a broadcast that p cannot receive its past does not change.
- $\pi_p(\gamma_0 \xrightarrow{p_0,\delta_0,R_0} \gamma_1 \dots \xrightarrow{p_{n-1},\delta_{n-1},R_{n-1}} \gamma_n)$ otherwise. In the case where p is not involved its past does not change.

Note that by definition of the transition relation $\rightarrow$, for every execution θ and every $p \in [1..nbproc(\theta)]$, the past of process p in θ is a finite path in $\mathcal{P}$.

A.2 Monotonicity property

Proposition 1. *Let θ be an execution in $\Theta_{\mathcal{L}}$ [resp. $\Theta_{\mathcal{LC}}$]. For every $N \geq nbproc(\theta)$, there exists θ' in $\Theta_{\mathcal{L}}$ [resp. $\Theta_{\mathcal{LC}}$] such that $nbproc(\theta') = N$ and $\mathsf{End}(\theta) = \mathsf{End}(\theta')$ [resp. $\mathsf{End}(\theta) \subseteq \mathsf{End}(\theta')$].*

Proof. We first prove that, given a local execution $\theta \in \Theta_{\mathcal{L}}$ there exists another local execution $\theta' \in \Theta_{\mathcal{L}}$ such that $nbproc(\theta') = nbproc(\theta) + 1$ and $\mathsf{End}(\theta) = \mathsf{End}(\theta')$. The proof is by induction on the length of θ. The idea is to add in θ' a process denoted p_{add} that behaves exactly as the first process (with process identifier 1) of θ and such that all other processes behave in θ' as in θ. Formally we define inductively a function $copycat(\theta)$, such that $copycat(\gamma_0) = \gamma_0'$ with $|\gamma_0'| = |\gamma_0| + 1$ and

$$copycat(\theta \xrightarrow{p,\delta,R} \gamma) = \begin{cases} copycat(\theta) \xrightarrow{p,\delta,R} \gamma' & \text{if } p \neq 1 \text{ and } 1 \notin R \\ copycat(\theta) \xrightarrow{p,\delta,R \cup \{p_{add}\}} \gamma' & \text{if } p \neq 1 \text{ and } 1 \in R \\ copycat(\theta) \xrightarrow{p,\delta,R} \gamma_{int} \xrightarrow{p_{add},\delta,\emptyset} \gamma' & \text{if } p = 1 \end{cases}$$

$$\text{with } \gamma'(p_{add}) = \gamma(1) \text{ and } \forall p' \in [1..nbproc(\theta)],\ \gamma'(p') = \gamma(p')$$

Intuitively if the transition did not affect the first process in θ, the exact same transition is fired in θ', and it affects neither 1 nor p_{add}. Otherwise in case 1 receives a message, p_{add} performs exactly the same reception (as specified by the reception set and the condition on γ'). Finally, if process 1 performs an active action in θ, p_{add} also performs that active action, yet the associated reception set is empty, so that execution θ can continue on the original processes.

Clearly enough, for any local execution $\theta \in \Theta_{\mathcal{L}}$, $copycat(\theta)$ is also a local execution and it satisfies $nbproc(copycat(\theta)) = nbproc(\theta) + 1$ and $\mathsf{End}(\theta) = \mathsf{End}(copycat(\theta))$. Applying iteratively the function $copycat$, one obtains a local execution θ' with arbitrarily many processes and such that $\mathsf{End}(\theta') = \mathsf{End}(\theta)$.

We now restrict to local clique executions, and similarly to the previous case, prove that given a local clique execution $\theta \in \Theta_{\mathcal{LC}}$ there exists $\theta' \in \Theta_{\mathcal{LC}}$ such that $nbproc(\theta') = nbproc(\theta) + 1$ and $\mathsf{End}(\theta) \subseteq \mathsf{End}(\theta')$. The proof is easier since we only require an inclusion of the set of states appearing in the last configuration. It suffices to add a new process p_{add} that receives all the messages (because of the clique topology) and that does not perform any active action, so that θ can be mimicked exactly, yet on a larger number of processes. Formally, given a strategy σ and an execution θ following σ we define inductively the function $passiv(\theta)$, such that $passiv(\gamma_0) = \gamma'_0$ with $|\gamma'_0| = |\gamma_0| + 1$ and $passiv(\theta \xrightarrow{p,\delta,R} \gamma) = passiv(\theta) \xrightarrow{p,\delta,R\cup\{p_{add}\}} \gamma'$ where $\forall p' \in [1..nbproc(\theta)],\ \gamma'(p') = \gamma(p')$ and in the case where δ is a broadcast of $m \in \Sigma$ we ask that p_{add} follows the local strategy: $\sigma_r(\pi_{p_{add}}(passiv(\theta)), m) = (\mathsf{dest}(\pi_{p_{add}}(passiv(\theta))), ??m, \gamma'(p_{add}))$.

Clearly enough for any local clique execution $\theta \in \Theta_{\mathcal{LC}}$, $passiv(\theta)$ is also a local clique execution and it satisfies $nbproc(passiv(\theta)) = nbproc(\theta) + 1$ and $\mathsf{End}(\theta) \subseteq \mathsf{End}(passiv(\theta))$. Applying iteratively the function $passiv$, one obtains a local execution θ' with arbitrarily many processes and such that $\mathsf{End}(\theta) \subseteq \mathsf{End}(\theta')$.

Note that in the case of a clique topology, one cannot preserve the set $\mathsf{End}(\theta)$ in general while increasing the number of processes, because processes are bound to receive all messages. Consider as an example the simple protocol composed only of two transitions $(q_0, !!m, q_1)$ and $(q_0, ??m, q_2)$. It admits a local clique execution $\theta \in \Theta_{\mathcal{LC}}$ with a single process such that $\mathsf{End}(\theta) = \{q_1\}$, yet any local clique execution θ' with at least two processes satisfies $\mathsf{End}(\theta') = \{q_1, q_2\}$ or $\mathsf{End}(\theta') = \{q_0\}$.

B Complements and proofs for Section 3.1

B.1 Formal definitions

We provide some formal definitions with relation to labelled trees used to represent strategy pattern. A *labelled tree* is a finite graph $T = (N, n_0, E, \Upsilon, \mathsf{lab})$ where N is a finite set of nodes, $n_0 \in N$ is called the *root* of T and $E \subseteq N \times N$ is the edge relation which satisfies the following conditions for all $n \in N$: $(n, n) \notin E$; $(n, n_0) \notin E$; if $n \neq n_0$ then there exists a unique $n' \in N$ such that $(n', n) \in E$, and $\mathsf{lab} : E \to \Upsilon$ is an edge-labelling function. For each edge $e = (n, n')$, we use the following notations: $\mathsf{src}(e) = n$ and $\mathsf{dest}(e) = n'$. A path in the tree is then either the empty path ϵ or a finite sequence of edges $e_1 \cdots e_\ell$ such that $\mathsf{src}(e_1) = n_0$ and $\mathsf{dest}(e_i) = \mathsf{src}(e_{i+1})$ for all $i \in [1..\ell - 1]$. A node n' is said to be the descendant of a node n if there exists a non-empty path $e_1 \cdots e_\ell$ and $i, j \in [1..\ell]$ such that $i \leq j$ and $\mathsf{src}(e_i) = n$ and $\mathsf{dest}(e_j) = n'$. We denote by $desc(T, n)$ the set of descendants of a node n in T. The *subtree* at node $n \in N$ of T, denoted by $\mathsf{Sub}(T, n)$ is the tree $(desc(T, n) \cup \{n\}, n, E', \Upsilon, \mathsf{lab}')$ such that $E' = E \cap ((desc(T, n) \cup \{n\}) \times desc(T, n))$ and lab' is the restriction of lab to E'. For a node $n \in N$ such that $n \neq n_0$, we use $\mathsf{pred}(n)$ to represent the unique edge $e \in E$ such that $\mathsf{dest}(e) = n$. Finally we define the size of T, denoted by $|T|$, as the number of its nodes.

We recall the definition of a strategy pattern.

Definition 2 (Strategy pattern). *A* strategy pattern *for a broadcast protocol* $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ *is a labelled tree* $T = (N, n_0, E, \Delta, \mathsf{lab})$ *such that if* $e_1 \cdots e_\ell$ *is a path in* T*, then* $\mathsf{lab}(e_1) \cdots \mathsf{lab}(e_\ell) \in \mathsf{Path}(\mathcal{P})$*, and for every node* $n \in N$ *and every message* $m \in \Sigma$*, we have:*

- *there is at most one edge* $e = (n, n') \in E$ *such that* $\mathsf{lab}(e)$ *is an active action;*
- *there is at most one edge* $e = (n, n') \in E$ *such that* $\mathsf{lab}(e)$ *is a reception of* m*.*

Given a strategy pattern $T = (N, n_0, E, \Delta, \mathsf{lab})$ for a broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$, let us define the *history* function $h : N \to \mathsf{Path}(\mathcal{P})$ that associates with each node of the strategy pattern the path in $\mathcal{P}$ it represents. Formally, for every $n \in N$ writing $e_1 \cdots e_\ell$ for the path in T with $\mathsf{dest}(e_\ell) = n$ then $h(n) = \mathsf{lab}(e_1) \cdots \mathsf{lab}(e_\ell)$. Given an execution θ, process p is said to be in node n if $h(n) = \pi_p(\theta)$.

We recall that given $\mathcal{P}$ a broadcast protocol, and T a strategy pattern for $\mathcal{P}$ with edge-labelling function lab, a local strategy $\sigma = (\sigma_a, \sigma_r)$ for $\mathcal{P}$ is said to *follow* T if for every path $e_1 \cdots e_\ell$ in T, the path $\rho = \mathsf{lab}(e_1) \cdots \mathsf{lab}(e_\ell)$ in $\mathcal{P}$ respects σ.

Finally we recall the definition of an admissible strategy pattern.

Definition 3 (Admissible strategy pattern). *An* admissible strategy pattern *for* $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ *is a pair* $(T, \prec)$ *where* $T = (N, n_0, E, \Delta, \mathsf{lab})$ *is a strategy pattern for* $\mathcal{P}$ *and* $\prec \subseteq N \times N$ *is a strict total order on the nodes of* T *such that:*

(1) for all $(n, n') \in E$ *we have* $n \prec n'$*;*
(2) for all $e = (n, n') \in E$*, if* $\mathsf{lab}(e) = (\mathsf{lab}(n), ??m, \mathsf{lab}(n'))$ *for some* $m \in \Sigma$*, then there exists* $e_1 = (n_1, n_1')$ *in* E *such that* $n_1' \prec n'$ *and* $\mathsf{lab}(e_1) = (\mathsf{lab}(n_1), !!m, \mathsf{lab}(n_1'))$*.*

Lemma 2. *Given a strategy pattern* $T = (N, n_0, E, \Delta, \mathsf{lab})$ *for a broadcast protocol* $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ *and a strict total order* $\prec \subseteq N \times N$*, checking whether* $(T, \prec)$ *is admissible can be done in polynomial time (in the size of the pattern).*

Proof. The proof is by induction on the size of the T.

For the base case, we consider the only pattern with a single node: $T = (\{n_0\}, n_0, \emptyset, \Delta, \mathsf{lab})$ the only strict total order is the trivial ordering $\prec = \emptyset$. The two conditions of the definition are trivially respected because $E = \emptyset$.

We now assume that for all strategy patterns of size K and for all strict total orders on the nodes we can check in polynomial time whether $(T, \prec)$ is admissible. We will now prove this property still holds for the strategy patterns of size $K + 1$.
Let $T = (N, n_0, E, \Delta, \mathsf{lab})$ be a strategy patterns of size $K + 1$ and $\prec \subseteq N \times N$ be a strict total order on the nodes. Let n be the maximal node with respect to $\prec$. First we can check in polynomial time if there exists a node $n' \in N$ such that $(n, n') \in E$. If such node exists then $(T, \prec)$ is not admissible (contradiction with condition *(1)*). Otherwise let T' be the pattern in which we remove the node n and its associated edges and $\prec'$ the total order $\prec$ without the node n. Formally, $T' = (N \setminus \{n\}, n_0, E \setminus \{\mathsf{pred}(n)\}, \Delta, \mathsf{lab}')$ and $\prec' = \prec \setminus \{(n', n) \mid n' \in N\}$. By induction hypothesis we can check in polynomial time whether $(T', \prec')$ is admissible. If it is admissible not, then one of the condition is violated and would also be violated for $(T, \prec)$. Otherwise, the only thing left to do in the case where $\mathsf{lab}(\mathsf{pred}(n))$ is a reception a message m, is to check wether there exists an edge $e_1 = (n_1, n_1') \in E$ such that $\mathsf{lab}(e_1) = (\mathsf{lab}(n_1), !!m, \mathsf{lab}(n_1'))$, otherwise *(2)* is not satisfied.

B.2 Link between admissibility and local executions

Lemma 3. *Given an admissible strategy pattern $(T, \prec)$, for all $M \in \mathbb{N} \setminus \{0\}$ and for all strategies σ that follows T, there exists an execution θ that respects σ and such that there are at least M processes in each node of the pattern in the last configuration of θ.*

Proof. The proof is by induction on the size of the strategy pattern T.

For the base case, we consider the only pattern with a single node: $T = (\{n_0\}, n_0, \emptyset, \Delta, \mathsf{lab})$ with trivial ordering $\prec = \emptyset$. Any local strategy σ follows T. For any $M \in \mathbb{N} \setminus \{0\}$ the execution consisting only of the initial configuration $\gamma_0 = \{q_0\}^M$ respects any local strategy σ and in the last configuration there are exactly M processes in node n_0.

We now assume that the property holds for all the admissible strategy patterns of size K and we will prove it holds for the admissible strategy patterns of size $K+1$. Let $(T, \prec)$ be an admissible strategy pattern of size $K+1$ with $T = (N, n_0, E, \Delta, \mathsf{lab})$. Let σ be a strategy following T and $M \in \mathbb{N} \setminus \{0\}$. We denote by $n \in N$ the maximal node according to the total order $\prec$. Note that n is necessarily a leaf thanks to the condition *(1)* of admissible strategy patterns. We denote by $(T', \prec')$ the admissible strategy pattern obtained from $(T, \prec)$ by removing the leaf n and its preceding edge $\mathsf{pred}(n) = (n', n)$.

First note that σ also follows T'. By induction hypothesis applied to T' and $M' = 2M+1$, there exists an execution θ such that θ respects σ and such that there are at least $2M+1$ processes in each node of T' in the last configuration of θ.

Let us now explain how θ can be extended depending on the type of the label of the deleted edge $\delta = \mathsf{lab}(\mathsf{pred}(n))$:

- If δ is an active action either internal (q, ε, q'), or a broadcast $(q, !!m, q')$ then we know that there are $2M+1$ processes in n'; hence we extend θ by choosing M processes among those processes to perform the active action δ with an empty reception set at each step;
- If $\delta = (q, ??m, q')$, then we know that since $(T, \prec)$ is an admissible strategy pattern, there exists an edge $e_1 = (n_1, n_1')$ in T such that $n_1' \prec n$ and $\mathsf{lab}(e_1) = (q_1, !!m, q_1')$. Furthermore, e_1 belongs also to T', hence there are $2M+1$ processes in node n' and $2M+1$ processes in node n_1. We extend θ by choosing one process to perform the broadcast of message m from n_1 and the associated reception set consists in M processes with history $h(n')$. This results in sending this M processes in node n.

In all the cases the obtained execution θ' respects σ and there are at least M processes in each node of the pattern in the last configuration of θ'.

We recall that we consider a broadcast protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$. Let us recall Proposition 2.

Proposition 2. *For all $Q' \subseteq Q$, there exists an admissible strategy pattern $(T, \prec)$ such that $\mathsf{lab}(T) = Q'$ iff there exists a local strategy σ and an execution θ such that θ respects σ and $Q' = \mathsf{Occur}(\theta)$, furthermore σ follows T.*

Proof. The first direction is a direct consequence of Lemma 3, taking *e.g.* $M = 1$.

To prove the second direction we suppose that there exists a local strategy σ and an execution θ such that θ respects σ and $Q' = \mathsf{Occur}(\theta)$. We assume that $\theta = \gamma_0 \xrightarrow{p_0, \delta_0, R_0} \dots \xrightarrow{p_\ell, \delta_\ell, R_\ell} \gamma_{\ell+1}$. We will explain how to build an admissible strategy pattern $(T, \prec)$ such that $\mathsf{lab}(T) = Q'$ from this execution. In the sequel, for every $i \in [1..\ell+1]$, we denote by

θ_i the execution $\gamma_0 \xrightarrow{p_0,\delta_0,R_0} \ldots \xrightarrow{p_{i-1},\delta_{i-1},R_{i-1}} \gamma_i$ consisting of the i first transitions in θ. We provide now the definition of a function *admtree* which, given a prefix of the execution θ, returns an admissible strategy pattern $(T, \prec)$. The idea is to build an admissible strategy pattern where the labelled paths characterize all possible pasts of the different processes involved in θ.

We proceed inductively as follows: $admtree(\gamma_0) = ((\{n_0\}, n_0, \emptyset, \Delta, \mathsf{lab}), \prec)$ with $\prec = \emptyset$ and for all $i \in [1..\ell+1]$, if $admtree(\theta_{i-1}) = (T, \prec)$ with $T = (N, n_0, E, \Delta, \mathsf{lab})$ then $admtree(\theta_i) = (T', \prec')$ where $T' = (N', n_0, E', \Delta, \mathsf{lab}')$ is obtained by completing T according to the following case analysis:

- if $\delta_{i-1} = (\gamma_{i-1}[p_{i-1}], \varepsilon, \gamma_i[p_{i-1}])$ and there does not exists a node n in T such that $\pi_{p_{i-1}}(\theta_i) = h(n)$, then let n' be the node in T such that $\pi_{p_{i-1}}(\theta_{i-1}) = h(n')$ (such node necessarily exists by definition of *admtree*). In that case, we add a new node n to T and we define $\mathsf{lab}'(n', n) = \delta_{i-1}$ and $\prec'$ is obtained from $\prec$ by defining n as the new maximal node.
- if $\delta_{i-1} = (\gamma_{i-1}[p_{i-1}], !!m, \gamma_i[p_{i-1}])$, then,
  - first, if there does not exists a node n in T such $\pi_{p_{i-1}}(\theta_i) = h(n)$, then let n' be the node in T such that $\pi_{p_{i-1}}(\theta_{i-1}) = h(n')$ (such node necessarily exists by definition of *admtree*). In that case, we add a new node n to T and we define $\mathsf{lab}'(n', n) = \delta_{i-1}$ and $\prec'$ is obtained from $\prec$ by defining n as the new maximal node.
  - afterwards for all $p \in R_{i-1}$ such that there does not exist a node n in T verifying $\pi_p(\theta_i) = h(n)$, let n' be the node in T such that $\pi_{p_{i-1}}(\theta_{i-1}) = h(n')$. Then we add a new node n to N', and lab' is extended such that $\mathsf{lab}'(n', n) = (\gamma_{i-1}[p], ??m, \gamma_i[p])$ and we extend $\prec'$ such that n is the new maximal of the order $\prec'$ (note that in that case it is important that the destination node of the broadcast is smaller w.r.t. $\prec'$ to the destination nodes of the performed receptions, but the order between these latter nodes is not relevant).

Note that if $admtree(\theta) = (T, \prec)$, then T is effectively a strategy pattern. The reason being that θ respects the local strategy σ, hence each path in $\mathcal{P}$ is associated via σ to a unique active action and a unique possible reception per message m. Furthermore, the fact that $admtree(\theta)$ is admissible follows directly from the inductive definition of the order. In fact condition *(1)* of admissible strategy patterns is verified since we add each time maximal nodes at the end of existing paths, and condition *(2)* is verified because each target node of a reception is bigger according to $\prec$ than a target node of a matching broadcast. Finally $\mathsf{lab}(T) = Q'$ since the labels of the nodes in T correspond exactly to all the control states seen in θ. It is furthermore clear by construction that σ follows T (since T is built following the choices given by σ in the execution θ).

B.3 Minimizing admissible strategy patterns

Given a q_F-admissible pattern $(T, \prec)$ where $T = (N, n_0, E, \Delta, \mathsf{lab})$, we denote by $\mathsf{NewBroad}(T, \prec) \subseteq N \setminus \{n_0\}$ the set of nodes such that $n \in \mathsf{NewBroad}(T, \prec)$ iff $\mathsf{lab}(\mathsf{pred}(n)) = (q, !!m, q')$ and for all $n' \in N \setminus \{n_0, n\}$ such that $\mathsf{lab}(\mathsf{pred}(n')) = (q'', !!m, q''')$, we have $n \prec n'$. We denote by $\mathsf{Imp}(T, \prec)$ the set of "important" nodes corresponding to $\mathsf{NewBroad}(T, \prec) \cup \{\mathsf{last}(T, \prec)\}$, *i.e.* the new broadcast and the last node labelled by q_F.

A q_F-admissible strategy pattern $(T, \prec)$, where $T = (N, n_0, E, \Delta, \mathsf{lab})$, is said to be *minimal* iff the following conditions are respected:

(a) for all $n \in N$, if $\mathsf{lab}(n) = q_F$ then $n = \mathsf{last}(T, \prec)$;
(b) for all $n \in N$, if $\mathsf{Sub}(T, n) = (N', n, E', \Delta, \mathsf{lab}')$ then $N' \cap \mathsf{Imp}(T, \prec) \neq \emptyset$;
(c) for all $n', n'' \in N$ such that $\mathsf{lab}(n') = \mathsf{lab}(n'')$ and $n' \neq n''$, if $\mathsf{Sub}(T, n') = (N', n', E', \Delta, \mathsf{lab}')$ and $\mathsf{Sub}(T, n'') = (N'', n'', E'', \Delta, \mathsf{lab}'')$ then $(N' \setminus \{n'\}) \cap \mathsf{Imp}(T, \prec) \neq N'' \cap \mathsf{Imp}(T, \prec)$.

Intuitively, condition **(a)** states that there is a unique node labelled with q_F, condition **(b)** that in every subtree there should be a node labelled by q_F or a new broadcast message, and, condition **(c)** that if two different subtrees have their root labelled by the same state, then there should be at least a new broadcast or the last state present in one of the subtree and not in the other one (the reason for this is basically that if one of the subtree is the subtree of the other one, then if all the new broadcasts or the last state are in the smaller subtree, we can replace the bigger subtree by the smaller one).

Lemma 4. *If there exists a q_F-admissible strategy pattern for $\mathcal{P}$, then there exists a minimal one.*

Proof. Let $(T, \prec)$ with $T = (N, n_0, E, \Delta, \mathsf{lab})$ be a q_F-admissible strategy pattern and assume that $(T, \prec)$ is not minimal.

First we suppose that there exists a node $n \in N$ such that $\mathsf{lab}(n) = q_F$ and $n \neq \mathsf{last}(T, \prec)$. Then let $n' \in N$ be the minimal node labelled by q_F, formally n' is such that for all nodes $n \in N \setminus \{n'\}$ verifying $\mathsf{lab}(n) = q_F$, we have $n' \prec n$. In that case we remove from T all the nodes n and the associated edges such that $n' \prec n$ and $n \neq n'$. The obtained structure is a q_F-admissible strategy pattern for $\mathcal{P}$, since thanks to the condition *(1)* respected by $\prec$, we know that such operation does not break the tree structure and furthermore since we only remove nodes bigger than n, we know that condition *(2)* of $\prec$ is still satisfied. Finally it is clear that the obtained admissible strategy pattern verifies the condition **(a)**.

Now we assume that condition **(a)** is satisfied by $(T, \prec)$ and we suppose that there exists a node $n \in N$ such that $\mathsf{Sub}(T, n) = (N', n, E', \Delta, \mathsf{lab}')$ and $N' \cap \mathsf{Imp}(T, \prec) = \emptyset$. The operation to get a pattern satisfying condition **(b)** is easy: for all nodes n such that $\mathsf{Sub}(T, n) = (N', n, E', \Delta, \mathsf{lab}')$ and $N' \cap \mathsf{Imp}(T, \prec) = \emptyset$ we remove from T the nodes N' and the associated edges leading to that nodes. Since we remove subtrees, the obtained structure is still a strategy pattern. Also since $\mathsf{Imp}(T, \prec) = \mathsf{NewBroad}(T, \prec) \cup \{\mathsf{last}(T, \prec)\}$, this allows us to deduce that this strategy pattern with the restriction of $\prec$ to the remaining nodes is still a q_F-admissible strategy pattern (condition *(1)* for admissibility is trivially satisfied and for what concerns condition *(2)*, since we keep for each message m the minimal node associated with the broadcast of this message, then condition *(2)* is also satisfied). Note finally that the obtained admissible strategy pattern satisfies the condition **(a)** and **(b)**.

We assume now that conditions **(a)** and **(b)** are satisfied by $(T, \prec)$ and that the condition **(c)** is not satisfied. Then until condition **(c)** is not satisfied, we perform the following operations: assume there are two nodes $n', n'' \in N$ such that $\mathsf{lab}(n') = \mathsf{lab}(n'')$ and $n' \neq n''$ with $\mathsf{Sub}(T, n') = (N', n', E', \Delta, \mathsf{lab}')$ and $\mathsf{Sub}(T, n'') = (N'', n'', E'', \Delta, \mathsf{lab}'')$ and $N' \cap \mathsf{Imp}(T, \prec) = N'' \cap \mathsf{Imp}(T, \prec)$. Then necessarily either $N' \subset N''$ or $N'' \subset N'$. Suppose we have the second case, *i.e.*, that $\mathsf{Sub}(T, n'')$ is a subtree of $\mathsf{Sub}(T, n')$, since $N' \cap \mathsf{Imp}(T, \prec) = N'' \cap \mathsf{Imp}(T, \prec)$ and since only important nodes matter (as we have seen with the previous case) and since $\mathsf{lab}(n') = \mathsf{lab}(n'')$, we can replace in T the subtree $\mathsf{Sub}(T, n')$ by its subtree $\mathsf{Sub}(T, n'')$ (and doing so, remove from T the nodes in $N' \setminus N''$). The obtained structure is still a q_F-admissible strategy pattern: in fact it is a strategy tree pattern because $\mathsf{lab}(n') = \mathsf{lab}(n'')$, it is still q_F-admissible because $\mathsf{Sub}(T, n'')$ is a subtree of $\mathsf{Sub}(T, n')$ and

because we did not remove any important nodes. Repeating this operation allows us to finally get a q_F-admissible strategy pattern which respects conditions **(a)**, **(b)** and **(c)** and hence which is minimal.

Proposition 3. *If there exists a q_F-admissible strategy pattern for $\mathcal{P}$, then there is one of size at most $(2|\Sigma|+1)\cdot(|Q|-1)$ and of height at most $(|\Sigma|+1)\cdot|Q|$.*

Proof. Given a strategy pattern T we call *intersection* node a node n in T from which at least two actions are defined (either two different receptions or a reception and an active action). We gather under the term *noticeable* nodes, the nodes that are important nodes or intersection nodes.

Thanks to Lemma 4, to establish Proposition 3, it sufffices to bound the size and height of minimal q_F-admissible strategy patterns. Let $(T,\prec)$ with $T=(N,n_0,E,\Delta,\mathsf{lab})$ be a minimal q_F-admissible strategy pattern for $\mathcal{P}=(Q,q_0,\Sigma,\Delta)$. First, from condition **(b)**, there are at most $|\mathsf{Imp}(T,\prec)|-1\leq|\Sigma|$ intersection nodes. Otherwise, there would be a subtree that would not contain any important node. Therefore, there are at most $|\Sigma|+|\Sigma|+1$ noticeable nodes. Second, from conditions **(a)** and **(c)**, we know that there are no more than $|Q|-2$ nodes between two noticeable nodes. Otherwise there would be two nodes with the same label that share the same set of important nodes, or there would be a node labelled with q_F. We thus derive the desired bound on the size of minimal q_F-admissible strategy patterns. We also obtain a bound on the height of minimal q_F-admissible strategy patterns: in the worst case, all important nodes belong to the same branch of T. We conclude by recalling that between two important nodes there are at most $|Q|-2$ nodes, and that the number of important nodes is bounded by $|\Sigma|+1$ (one per message type, plus the final state).

B.4 Bounds on cutoff and past length

We recall here Proposition 4.

Proposition 4. *If there exists an execution $\theta\in\Theta_{\mathcal{L}}$ such that $q_F\in\mathsf{Occur}(\theta)$, then there exists an execution $\theta'\in\Theta_{\mathcal{L}}$ such that $q_F\in\mathsf{Occur}(\theta')$ and $nbproc(\theta')\leq(2|\Sigma|+1)\cdot(|Q|-1)$ and $|\pi_p(\theta')|\leq(|\Sigma|+1)\cdot|Q|$ for every $p\in[1..nbproc(\theta')]$.*

The proof of Proposition 4 calls for the introduction of the following notations. Given an execution θ, a process p and a path $\rho\in\mathsf{Path}(\mathcal{P})$, we consider the execution built from θ in which there is an additional process p_{add} that behaves exactly as process p until its history is ρ. Formally, $follow(\theta,p,\rho)$[1] is defined inductively: $follow(\gamma_0,p,\rho)=\gamma_0'$ with $|\gamma_0'|=|\gamma_0|+1$ and if $\pi_{p_{add}}(follow(\theta,p,\rho))=\rho$ then $follow(\theta\xrightarrow{p',\delta,R}\gamma,p,\rho)=follow(\theta,p,\rho)\xrightarrow{p',\delta,R}\gamma'$ where $\gamma'(p'')=\gamma(p'')$ for every $p''\neq p_{add}$, otherwise

$$follow(\theta\xrightarrow{p',\delta,R}\gamma,p,\rho)=\begin{cases}\theta'\xrightarrow{p',\delta,R}\gamma' & \text{if } p'\neq p \text{ and } p\notin R\\ \theta'\xrightarrow{p',\delta,R\cup\{p_{add}\}}\gamma' & \text{if } p'\neq p \text{ and } p\in R\\ \theta'\xrightarrow{p,\delta,R}\gamma_{int}\xrightarrow{p_{add},\delta,\emptyset}\gamma' & \text{if } p=p'\end{cases}$$

$$\text{with } \theta'=follow(\theta,p,\rho),\ \gamma'(p_{add})=\gamma(p) \text{ and } \forall p''\neq p_{add},\ \gamma'(p'')=\gamma(p'')$$

[1] Notice that *follow* refines *copycat* defined page 16.

Intuitively in case p_{add} already reached its destination (*i.e.* has history ρ), the execution continues exactly as θ. Otherwise, in case of a broadcast by another process than p, the new process p_{add} behaves as p, *i.e.* receives the message in θ' if and only if p receives it in θ. Last, if p is responsible for the active action, p_{add} performs exactly the same active action, yet the associated reception set is empty, so that execution θ can continue on the original processes. Note that in all cases, the inductive definition ensures that θ' exists and is unique, so that $\mathit{follow}(\theta, p, \rho)$ is well defined. Moreover, assuming θ is a local execution, since p_{add} "follows" the process p, $\mathit{follow}(\theta, p, \rho)$ is a local execution too, and $\pi_{p_{add}}(\mathit{follow}(\theta, p, \rho)) = \pi_p(\theta)$ or $\pi_{p_{add}}(\mathit{follow}(\theta, p, \rho)) = \rho$ (and then $\pi_{p_{add}}(\mathit{follow}(\theta, p, \rho))$ is a prefix of $\pi_p(\theta)$).

Let $T = (N, n_0, E, \Delta, \mathsf{lab})$ be a strategy pattern. Given a subset of nodes $N_1 \subseteq N$ we define the restriction of T to the predecessors of nodes in N_1; formally $T_{\downarrow N_1} = (N', n_0, E', \Delta, \mathsf{lab})$ where $N' = \{n \in N \mid \exists n' \in \mathit{desc}(n, T) \cap N_1\}$ and $E' = E \cap (N' \times N')$. Assuming T is equipped with an order $\prec$, for any $k \in [1..|\mathsf{Imp}(T, \prec)|]$ the set of the k first important nodes with respect to $\prec$ is denoted $\mathsf{Imp}(T, \prec, k)$. Formally we have $\mathsf{Imp}(T, \prec, k) \subseteq \mathsf{Imp}(T, \prec)$ and $|\mathsf{Imp}(T, \prec, k)| = k$ and if $n \in \mathsf{Imp}(T, \prec, k)$ and $n' \in \mathsf{Imp}(T, \prec)$ with $n' \prec n$ then $n' \in \mathsf{Imp}(T, \prec, k)$. The following lemma bounds the number of processes needed to reach a configuration with processes in each of the nodes of $\mathsf{Imp}(T, \prec, k)$, by the number of predecessors of these nodes.

Recall that $h(n) \in \mathsf{Path}(\mathcal{P})$ is the labelling of the path in T from the root to node n.

Lemma 5. *Let $(T, \prec)$ be an admissible strategy pattern and $k \in [1..|\mathsf{Imp}(T, \prec)|]$. Then there exists an execution $\theta \in \Theta_{\mathcal{L}}$ with the following properties:*

1. $\{\pi_p(\theta) \mid p \in [1..\mathit{nbproc}(\theta)]\} = \{h(n) \mid n \in \mathsf{Imp}(T, \prec, k)\}$
2. $\mathit{nbproc}(\theta) \leq |T_{\downarrow \mathsf{Imp}(T, \prec, k)}|$

Proof. The proof is by induction on k. For the base case $k = 1$, the first important node is reachable only via active actions, by definition of admissibility. Therefore we can define θ as the execution with a single process p such that $\pi_p(\theta) = h(n_1)$ where n_1 is the first important node.

Assume now that the lemma holds for $k - 1$. Let $\theta_{k-1} \in \Theta_{\mathcal{L}}$ be an execution such that $\{\pi_p(\theta_{k-1}) \mid p \in [1..\mathit{nbproc}(\theta_{k-1})]\} = \{h(n) \mid n \in \mathsf{Imp}(T, \prec, k-1)\}$ and $\mathit{nbproc}(\theta_{k-1}) \leq |T_{\downarrow \mathsf{Imp}(T, \prec, k-1)}|$. Intuitively, in order from θ_{k-1} to build θ_k that proves the induction step, one additional process will follow some process of θ_{k-1} until it meets some node n on the way to n_k the k-th important node of T. Then the additional process p_{add} will aim at reaching n_k from n. To do so, since there might be receptions steps between n and n_k, some more processes are needed that will broadcast the corresponding messages. This will work smoothly by using other additional processes, since the needed broadcast precisely lead to important nodes in $\mathsf{Imp}(T, \prec, k-1)$. These additional processes will be defined thanks to an auxilliary function fill that we define now.

We consider a local execution θ such that for every node $n \in \mathsf{Imp}(T, \prec, k-1)$ there exists a process $p \in [1..\mathit{nbproc}(\theta)]$ with $\pi_p(\theta) = h(n)$. Moreover, for $p \in [1..\mathit{nbproc}(\theta)]$ and for a sequence of edges $e_1 \dots e_l$ in T such that $\pi_p(\theta) = h(\mathsf{src}(e_1))$, we define a function $\mathsf{fill}(\theta, e_1 \dots e_l, p)$ inductively as follows: $\mathsf{fill}(\theta, \epsilon, p) = \theta$ and otherwise

- if $\mathsf{lab}(e_1) = \delta$ is an active action, we let $\mathsf{fill}(\theta, e_1 \dots e_l, p) = \mathsf{fill}(\theta', e_2 \dots e_l, p)$ with $\theta' = \theta \xrightarrow{p, \delta, \emptyset} \gamma$; *i.e.* the process p performs the active action with an empty reception set. Notice that $\mathit{nbproc}(\theta') = \mathit{nbproc}(\theta)$, and $\forall p' \in [1..\mathit{nbproc}(\theta)] \setminus \{p\}, \pi_{p'}(\theta') = \pi_{p'}(\theta)$.

– if $\mathsf{lab}(e_1) = \delta$ is a reception of message m, we consider $e_m = (n_1, n_2)$ such that $\mathsf{lab}(e_m)$ is a broadcast of m and such that $n_2 \in \mathsf{Imp}(T, \prec, k-1)$. Such an edge exists because T is admissible. By assumption on θ, there exists $p' \in [1..nbproc(\theta)]$ such that $\pi_{p'}(\theta) = h(n_2)$. We consider the execution $follow(\theta, p', h(n_1))$, and write p_m for the additional process in that execution compared to θ. Then, we let $\theta' = follow(\theta, p', h(n_1)) \xrightarrow{p_m, \mathsf{lab}(e_m), \{p\}} \gamma$, and finally $\mathsf{fill}(\theta, e_1 \dots e_l, p) = \mathsf{fill}(\theta', e_2 \dots e_l, p)$ *i.e.* the additional process p_m broadcasts message m to p only. Notice that $nbproc(\theta') = nbproc(\theta) + 1$, $\pi_{p_m}(\theta) = h(n_2)$, for every $p' \in [1..nbproc(\theta_1)] \setminus \{p\}$, $\pi_{p'}(\theta') = \pi_{p'}(\theta)$, and $n_2 \in \mathsf{Imp}(T, \prec, k-1)$.

To conclude the induction step of the proof, we now explain how to obtain θ_k applying fill to θ_{k-1}. Let n_k be the k-th important node in T with respect to $\prec$. Let n be the last node appearing on the path to n_k and such that n is visited by some process p_1 along θ_{k-1}. We write $e_1 \dots e_l$ for the sequence of edges between n and n_k and consider $\theta' = follow(\theta_{k-1}, p_1, h(n))$ the execution in which an additional process goes to node n, and we denote by p' this process. The situation is illustrated on Figure 5 where important nodes are circled.

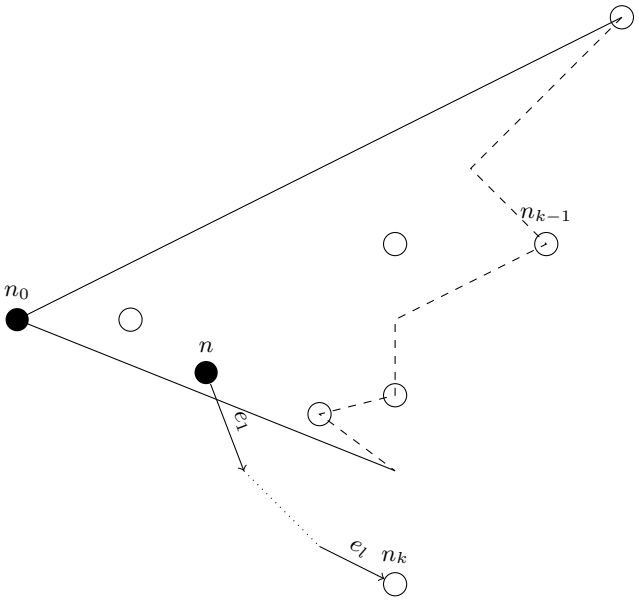

Fig. 5. Illustration for the construction of θ_k from θ_{k-1}.

By induction hypothesis on θ_{k-1} we know that for every node $n \in \mathsf{Imp}(T, \prec, k-1)$, there exists $p \in [1..nbproc(\theta_{k-1})]$ such that $\pi_p(\theta_{k-1}) = h(n)$. This also holds for the execution θ' extended with the additional process p'. As a consequence, we can apply fill to θ' for the sequence of edges $e_1 \dots e_l$ and the process p'. The resulting execution $\theta'' = \mathsf{fill}(\theta', e_1 \dots e_l, p')$ satisfies that for every $n \in \mathsf{Imp}(T, \prec, k-1)$ there exists $p \in [1..nbproc(\theta'')]$ with $\pi_p(\theta'') = h(n)$, and since the process p fired the sequence of edges $e_1 \dots e_l$, θ'' moreover satisfies $\pi_p(\theta'') = h(n_k)$. By definition of fill, all additional processes end in important nodes, so that $\{\pi_p(\theta'') \mid p \in [1..nbproc(\theta'')]\} = \{h(n) \mid n \in \mathsf{Imp}(T, \prec, k)\}$. Last, applying fill adds at most one process per edge of the sequence $e_1 \dots e_l$ from n to the k-th important node n_k. As a consequence, we obtain the following bound on the number of processes: $nbproc(\theta'') \leq |T_{\downarrow \mathsf{Imp}(T, \prec, k-1)}| + n = |T_{\downarrow \mathsf{Imp}(T, \prec, k)}|$.

Lemma 5 together with Proposition 3 implies Proposition 4.

C Complements and proofs for Section 3.2

C.1 Formal definitions

In order to obtain an NP algorithm for TARGET[$\mathcal{L}$], we develop the notion of $\mathfrak{T}$-coadmissible strategy patterns, that intuively refine admissible strategy patterns by ensuring that from every node of the pattern, there is a way to reach the target set. We first recall the definition of co-admissible strategy patterns.

Definition 4 (Co-admissible strategy pattern). *For $\mathfrak{T} \subseteq Q$ a set of states, a* $\mathfrak{T}$-coadmissible strategy pattern *for $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ is a pair $(T, \lhd)$ where $T = (N, n_0, E, \Delta, \mathsf{lab})$ is a strategy pattern for $\mathcal{P}$ and $\lhd \subseteq N \times N$ is a strict total order on the nodes T such that for every node $n \in N$ with $\mathsf{lab}(n) \notin \mathfrak{T}$ there exists an edge $e = (n, n') \in E$ with $n \lhd n'$ and either:*

- $\mathsf{lab}(e) = (\mathsf{lab}(n), \varepsilon, \mathsf{lab}(n'))$ *or,*
- $\mathsf{lab}(e) = (\mathsf{lab}(n), !!m, \mathsf{lab}(n'))$ *or,*
- $\mathsf{lab}(e) = (\mathsf{lab}(n), ??m, \mathsf{lab}(n'))$ *and there exists an edge $e_1 = (n_1, n_1') \in E$ such that $n \lhd n_1$, $n \lhd n_1'$ and $\mathsf{lab}(e_1) = (q_1, !!m, q_1')$.*

When convenient, in order to manipulate the order $\lhd$ more easily we will equivalently use an injective rank function $rk_\lhd : N \to \mathbb{Z}$ such that for every pair of nodes (n, n') we have $rk_\lhd(n) < rk_\lhd(n')$ iff $n \lhd n'$. Similarly to Lemma 2 for admissible patterns, we can establish the following:

Lemma 6. *Given a strategy pattern $T = (N, n_0, E, \Delta, \mathsf{lab})$ for a broadcast protocol $\mathcal{P}$ and a strict total order $\lhd \subseteq N \times N$, checking whether $(T, \lhd)$ is $\mathfrak{T}$-coadmissible (for a set $\mathfrak{T}$) can be done in polynomial time (in the size of the pattern).*

A strategy pattern T equipped with two orderings $\prec$ and $\lhd$ is said to be $\mathfrak{T}$-*biadmissible* whenever $(T, \prec)$ is admissible and $(T, \lhd)$ is $\mathfrak{T}$-coadmissible.

C.2 Link between biadmissibility and local executions

The relation between biadmissible strategy patterns, and local strategies satisfying a target objective is stated in the next proposition.

Proposition 5. *There exists a $\mathfrak{T}$-biadmissible pattern $(T, \prec, \lhd)$ iff there exists a local strategy σ and an execution θ such that θ respects σ and $\mathsf{End}(\theta) \subseteq \mathfrak{T}$; furthermore σ follows T.*

Proof. First we suppose that there exists a local strategy σ and an execution θ such that θ respects σ and $\mathsf{End}(\theta) \subseteq \mathfrak{T}$. We write $\theta = \gamma_0 \xrightarrow{p_0,\delta_0,R_0} \dots \xrightarrow{p_\ell,\delta_\ell,R_\ell} \gamma_{\ell+1}$. Let us define a function *coadmorder* which, given a prefix of the execution θ, returns an order $\lhd$ on the nodes of $admtree(\theta)$[2] . The idea of the order $coadmorder(\theta)$ is that if $n \lhd n'$ then, the last time, during θ, some processes are in node n stands before the last time some processes are in n'. Let $admtree(\theta) = (T, \prec)$ with $T = (N, n_0, E, \Delta, \mathsf{lab})$. The order is defined inductively as follows: $coadmorder(\gamma_0) = \emptyset$, and for all $i \in [1..\ell + 1]$, if $coadmorder(\theta_{i-1}) = \lhd$, $coadmorder(\theta_i)$ is obtained by completing $\lhd$ according to the following case analysis [3]:

[2] See Proof of Proposition 2 for a definition of *admtree*.

[3] θ_i is defined as in the proof of Proposition 2.

- if $\delta_{i-1} = (\gamma_{i-1}[p_{i-1}], \varepsilon, \gamma_i[p_{i-1}])$ by definition of *admtree* there exists two nodes n and n' in T such that $\pi_{p_{i-1}}(\theta_{i-1}) = h(n)$ and $\pi_{p_{i-1}}(\theta_i) = h(n')$ we obtain $\mathit{coadmorder}(\theta_i)$ by defining n as the second maximal node and n' as the maximal node. Formally, $\mathit{coadmorder}(\theta_i) = \lhd \setminus (\{n, n'\} \times N \cup N \times \{n, n'\}) \cup \{(n_1, n) \mid n_1 \in N \setminus \{n'\}\} \cup \{(n_1, n') \mid n_1 \in N\}$.
- if $\delta_{i-1} = (\gamma_{i-1}[p_{i-1}], !!m, \gamma_i[p_{i-1}])$, then,
  - first, for all $p \in R_{i-1}$ by definition of *admtree* there exists two nodes n and n' in T such that $\pi_p(\theta_{i-1}) = h(n)$ and $\pi_p(\theta_i) = h(n')$ we obtain $\mathit{coadmorder}(\theta_i)$ by defining n as the second maximal node and n' as the maximal node. Formally, $\mathit{coadmorder}(\theta_i) = \lhd \setminus (\{n, n'\} \times N \cup N \times \{n, n'\}) \cup \{(n_1, n) \mid n_1 \in N \setminus \{n'\}\} \cup \{(n_1, n') \mid n_1 \in N\}$.
  - afterward, we treat the process that did the broadcast as in the case of the internal transition. That is: by definition of *admtree* there exists two nodes n and n' in T such that $\pi_{pi-1}(\theta_{i-1}) = h(n)$ and $\pi_{pi-1}(\theta_i) = h(n')$ we obtain $\mathit{coadmorder}(\theta_i)$ by defining n as the second maximal node and n' as the maximal node. Formally, $\mathit{coadmorder}(\theta_i) = \lhd \setminus (\{n, n'\} \times N \cup N \times \{n, n'\}) \cup \{(n_1, n) \mid n_1 \in N \setminus \{n'\}\} \cup \{(n_1, n') \mid n_1 \in N\}$.

The fact that $(T, \mathit{coadmorder}(\theta))$ is $\mathfrak{T}$-coadmissible follows directly from the inductive definition of the order. In fact the condition is verified since each time a transition is taken in the execution we make sure that it is possible to take it according to the order. And, since $\mathsf{End}(\theta) \subseteq \mathfrak{T}$, from all states that do not belong to $\mathfrak{T}$ there must be a position in the execution from which any state out of $\mathfrak{T}$ does no longer appears in the execution, yielding a desired outgoing edge.

We now show the other implication: if there exists a $\mathfrak{T}$-biadmissible strategy pattern $(T, \prec, \lhd)$ then there exists a local strategy σ following T and there exists an execution θ that respects σ such that $\mathsf{End}(\theta) \subseteq \mathfrak{T}$. In order to relate more precisely coadmissible strategy patterns to local strategies, we define a partition of the nodes of T according to the position with respect to a given node n: $\mathit{part}(T, \lhd, n) = (S, G)$ with $S = \{n' \mid n' \lhd n\}$ the set of all nodes smaller than n and $G = \{n' \mid n \lhd n'\}$ the set of all nodes greater than n. Our proof is then based on the following technical lemma.

Lemma 7. *Let $(T, \lhd)$ be a coadmissible strategy pattern, n a node of T with $\mathsf{lab}(n) \notin \mathfrak{T}$ and θ an execution such that, writing $(S, G) = \mathit{part}(T, \lhd, n)$ and $M = |G|$, there are more than M processes in each node of G. Then there exists an execution θ' that extends θ and such that the number of processes in each node of S does not change, the number of processes in each node of G is at least $M - 1$ and no processes are in node n anymore.*

Proof. By definition of co-admissibility, there exists $e = (n, n') \in E$ such that $n \lhd n'$ and either:

- $\mathsf{lab}(e) = (\mathsf{lab}(n), \varepsilon, \mathsf{lab}(n'))$ or, $\mathsf{lab}(e) = (\mathsf{lab}(n), !!m, \mathsf{lab}(n'))$ or,
- $\mathsf{lab}(e) = (\mathsf{lab}(n), ??m, \mathsf{lab}(n'))$ and there exists an edge $e_1 = (n_1, n_1') \in E$ such that $n \lhd n_1$, $n \lhd n_1'$ and $\mathsf{lab}(e_1) = (q_1, !!m, q_1')$.

In the first case one can extend θ in an execution θ' by considering all the processes such that $\pi_p(\theta) = h(n)$ and let each of them perform the active action $\mathsf{lab}(e)$ with an empty reception set (in case of a broadcast). In the second case one builds θ' by considering one process such that $\pi_p(\theta) = h(n_1)$ and let it perform the broadcast of m with as reception set the set all processes with $\pi_p(\theta) = h(n)$. In both cases the processes in nodes of S are not concerned. At most one process that was in a node of G moved to an other node of G and all the processes in node n moved to a node of G yielding the desired properties on θ'.

To conclude the proof of Proposition 5, we observe that given a $\mathfrak{T}$-biadmissible strategy pattern, by Lemma 3, there exists an execution with an arbitrary number of processes in each node. From this initial execution and iterating Lemma 7 to every node by increasing order for $\lhd$, we obtain an execution for which the last configuration satisfies that all the processes are in a state belonging to $\mathfrak{T}$.

C.3 Minimizing biadmissible strategy patterns

As for reachability, the size of $\mathfrak{T}$-biadmissible strategy patterns can be minimized by keeping only relevant edges that permit a broadcast of either a new message or of the last message of this type. More formally, given a $\mathfrak{T}$-biadmissible pattern $(T, \prec, \lhd)$ where $T = (N, n_0, E, \Delta, \mathsf{lab})$, we denote as we did before for admissible patterns, $\mathsf{NewBroad}(T, \prec) \subseteq N \setminus \{n_0\}$ the set of nodes such that $n \in \mathsf{NewBroad}(T, \prec)$ iff $\mathsf{lab}(\mathsf{pred}(n)) = (q, !!m, q')$ and for all $n' \in N \setminus \{n_0\}$ such that $\mathsf{lab}(\mathsf{pred}(n')) = (q'', !!m, q''')$, we have $n \prec n'$. We further denote by $\mathsf{LastBroad}(T, \lhd) \subseteq N \setminus \{n_0\}$ the set of nodes such that $n \in \mathsf{LastBroad}(T, \lhd)$ iff $\mathsf{pred}(n) = (n', n) = e$, $n' \lhd n$, $\mathsf{lab}(e) = (q, !!m, q')$ and for all $e' = (n_1, n_1') \in E$ such that $n_1 \lhd n_1'$ and $\mathsf{lab}(e') = (q'', !!m, q''')$, we have $n_1 \lhd n'$. Finally, the set of important nodes is defined as $\mathsf{Imp}(T, \prec, \lhd) = \mathsf{NewBroad}(T, \prec) \cup \mathsf{LastBroad}(T, \lhd)$, *i.e.* consists of the new broadcasts and the last broadcasts.

A $\mathfrak{T}$-biadmissible strategy pattern $(T, \prec, \lhd)$, where $T = (N, n_0, E, \Delta, \mathsf{lab})$, is said to be *minimal* iff the following conditions are fulfilled:

- **(a)** for all $n \in N$, if $\mathsf{lab}(n) \in \mathfrak{T}$ and $\mathsf{Sub}(T, n) = (N', n, E', \Delta, \mathsf{lab}')$ with $N' \cap \mathsf{Imp}(T, \prec, \lhd) \subseteq \{n\}$ then $N' = \{n\}$;
- **(b)** for all $n \in N$ and all $n_1, n_2 \in N$ with $(n, n_1) \in E$ and $(n, n_2) \in E$, if $\mathsf{Sub}(T, n_1) = (N_1, n_1, E_1, \Delta, \mathsf{lab}_1)$ and $\mathsf{Sub}(T, n_2) = (N_2, n_2, E_2, \Delta, \mathsf{lab}_2)$ then either $N_1 \cap \mathsf{Imp}(T, \prec, \lhd) \neq \emptyset$, $N_2 \cap \mathsf{Imp}(T, \prec, \lhd) \neq \emptyset$, or $n_1 = n_2$;
- **(c)** for all $n_1, n_2, n_3 \in N$ pairwise different such that $\mathsf{lab}(n_1) = \mathsf{lab}(n_2) = \mathsf{lab}(n_3)$, if for $i \in \{1, 2, 3\}$, $\mathsf{Sub}(T, n_i) = (N_i, n_i, E_i, \Delta, \mathsf{lab}_i)$ and $n_3 \in N_2$ and $n_2 \in N_1$ then there exists $i, j \in \{1, 2, 3\}$ such that $N_i \cap \mathsf{Imp}(T, \prec, \lhd) \neq N_j \cap \mathsf{Imp}(T, \prec, \lhd)$.

Intuitively, these conditions state that **(a)** the branches of the tree end at the first target node not followed by an important node, **(b)** for any branching there cannot be two subtrees without important nodes and, **(c)** if three different subtrees have their root labelled by the same state, then there should be at least one important node in one of the subtrees and not in the other one (the reason is the same as for admissible pattern *i.e.* otherwise we can replace a bigger subtree by a smaller one).

Proposition 6. *If there exists a $\mathfrak{T}$-biadmissible strategy pattern for $\mathcal{P}$, then there exists a minimal one.*

Proof. Let $(T, \prec, \lhd)$ with $T = (N, n_0, E, \Delta, \mathsf{lab})$ be a $\mathfrak{T}$-biadmissible strategy pattern and assume that $(T, \prec, \lhd)$ is not minimal.

First we suppose that there exists a set of nodes $S \subseteq N$ such that for every node $n \in S$, $\mathsf{lab}(n) \in \mathfrak{T}$ and $\mathsf{Sub}(T, n) = (N', n, E', \Delta, \mathsf{lab}')$ with $N' \cap \mathsf{Imp}(T, \prec, \lhd) \subseteq \{n\}$. For each $n \in S$, we remove from T all the nodes in its subtree (*i.e.* every $n' \in N' \setminus \{n\}$), except n, and their associated edges. The resulting object is a $\mathfrak{T}$-biadmissible strategy pattern for $\mathcal{P}$. Indeed

thanks to condition (1) of admissibility[4] on $(T, \prec)$, such an operation preserves the tree structure. Moreover, since we only remove nodes bigger than nodes of S with respect to $\prec$, we know that condition (2) of $\prec$ is also preserved. Finally, since no important node was removed, and since the pattern was pruned at a node labelled by a state in $\mathfrak{T}$, we deduce that the conditions for coadmissibility[5] are still respected. In the end, the obtained strategy pattern is admissible, co-admissible and verifies condition **(a)** of minimality.

Now we assume that condition **(a)** is satisfied by $(T, \prec, \lhd)$ and we suppose that there exists a node $n \in N$ such that there exists $n_1 \neq n_2 \in N$ with $(n, n_1) \in E$ and $(n, n_2) \in E$, and $\mathsf{Sub}(T, n_1) = (N_1, n_1, E_1, \Delta, \mathsf{lab}_1)$ and $\mathsf{Sub}(T, n_2) = (N_2, n_2, E_2, \Delta, \mathsf{lab}_2)$ and with $N_1 \cap \mathsf{Imp}(T, \prec, \lhd) = \emptyset$, $N_2 \cap \mathsf{Imp}(T, \prec, \lhd) = \emptyset$. Getting a pattern that satisfies in addition condition **(b)** is easy: for every such node n we remove from T the nodes in N_1 if $n_1 \lhd n_2$ (symmetrically those of N_2 in case $n_2 \lhd n_1$) and their associated edges. Here again, since we remove entire subtrees, the resulting structure is still a strategy pattern. Furthermore since between n_1 and n_2, the maximal node w.r.t. $\lhd$ was kept, the condition for coadmissibility concerning node n still holds. Also, since no important node was removed, the pattern is still biadmissible. Finally the obtained admissible strategy pattern thus satisfies conditions **(a)** and **(b)**.

Last, we assume that conditions **(a)** and **(b)** are satisfied by $(T, \prec)$ whereas condition **(c)** is not satisfied. Until condition **(c)** is not satisfied, we perform the following operations. Suppose that there are three pairwise distinct nodes $n_1, n_2, n_3 \in N$ such that $\mathsf{lab}(n_1) = \mathsf{lab}(n_2) = \mathsf{lab}(n_3)$, for $i \in \{1, 2, 3\}$, $\mathsf{Sub}(T, n_i) = (N_i, n_i, E_i, \Delta, \mathsf{lab}_i)$ and $n_3 \in N_2$ and $n_2 \in N_1$ and such that $N_1 \cap \mathsf{Imp}(T, \prec) = N_2 \cap \mathsf{Imp}(T, \prec) = N_3 \cap \mathsf{Imp}(T, \prec)$. We proceed by case inspection:

- First assume that $n_1 \lhd n_2$. Since $N_1 \cap \mathsf{Imp}(T, \prec, \lhd) = N_2 \cap \mathsf{Imp}(T, \prec, \lhd)$, since only important nodes matter, and since $\mathsf{lab}(n_1) = \mathsf{lab}(n_2)$, we can replace in T the subtree $\mathsf{Sub}(T, n_1)$ by its subtree $\mathsf{Sub}(T, n_2)$, and doing so, remove from T the nodes in $N_1 \setminus N_2$. The resulting object is still a $\mathfrak{T}$-biadmissible strategy pattern because no important node was removed, and the predecessor of n_1 is now connected to a bigger node, hence still satisfies the coadmissibility condition.
  The cases $n_2 \lhd n_3$ and $n_1 \lhd n_3$ are treated similarly.
- Assume now the hardest situation: $n_3 \lhd n_2 \lhd n_1$. Note that $N_1 \cap \mathsf{Imp}(T, \prec, \lhd) = N_2 \cap \mathsf{Imp}(T, \prec, \lhd) = N_3 \cap \mathsf{Imp}(T, \prec, \lhd)$ hence that $N_1 \setminus N_3 \cap \mathsf{Imp}(T, \prec, \lhd) = \emptyset$. Let $n'_1, \ldots, n'_k \in N$ be the nodes on the path from n_1 to n_2 (included). Formally the nodes such that $n'_1 = n_1$, $n'_k = n_2$ and $\forall i \in [1..k-1], (n'_i, n'_{i+1}) \in E$. Since $n_2 \lhd n_1$ we know that there exist $n' = n'_i$ such that $n'_{i+1} \lhd n'_i$, we denote $\mathsf{Sub}(T, n') = (N', n', E', \Delta, \mathsf{lab}')$ for clarity. We now modify $\lhd$ in an order $\lhd'$ such that $n_2 \lhd' n_3$ and $(T, \lhd')$ is $\mathfrak{T}$-coadmissible. The idea is that we can decrease the rank of all the nodes in $(N' \setminus N_2) \cup \{n_2\}$ by the same value without falsifying the coadmissibility property (in fact none of this node is an important node). Formally, letting $B = \min_{n \in N}\{rk(n)\} - \max_{n \in N}\{rk(n)\} - 1$ we define $\lhd'$ as the order associated with the following rank function: for every node $n \in N \setminus ((N' \setminus N_2) \cup \{n_2\})$, $rk_{\lhd'}(n) = rk_{\lhd}(n)$ and for all the other nodes $n \in (N' \setminus N_2) \cup \{n_2\}$, $rk_{\lhd'}(n) = rk_{\lhd}(n) + B$. Let us argue that $(T, \lhd')$ is $\mathfrak{T}$-coadmissible. Indeed, the only edges that could cause a problem (to maintain the existence of an edge $e = (n, n') \in E$ with $n \lhd n'$ for each n such that $\mathsf{lab}(n) \notin \mathfrak{T}$) are (1) (n'_{i+1}, n'_i) since the rank of n'_i was decreased, but since $n'_{i+1} \lhd n'_i$

[4] See Definition 3, on page 18.
[5] See Definition 4, on page 25.

we are safe; and (2) the edges leaving n_2 but since the rank of n_2 was decreased and not the one of its successor we are also safe. Finally the rank of the important nodes is left unchanged. As a consequence, $(T, \lhd')$ is $\mathfrak{T}$-coadmissible and it satisfies $n_2 \lhd' n_3$, so that we can apply the transformation described for the first case.

Repeating this operation allows us to finally get a $\mathfrak{T}$-biadmissible strategy pattern which respects conditions **(a)**, **(b)** and **(c)** and hence which is minimal.

Lemma 8. *If there exists a $\mathfrak{T}$-biadmissible strategy pattern for $\mathcal{P}$, then there is one of size at most $16|\Sigma| \cdot |Q| \cdot (|Q| - |\mathfrak{T}| + 1)$ and of height at most $4|\Sigma| \cdot |Q| + 2(|Q| - |\mathfrak{T}|) + 1$.*

Proof. We proceed in two steps: in the first step we bound the number of nodes that precede important nodes, and in the second step we bound the number of nodes that do not lead to important nodes, but can be useful to reach the target.

The first step is very similar to the proof of the bound given for admissible trees. Let $(T, \prec, \lhd)$ with $T = (N, n_0, E, \Delta, \mathsf{lab})$ be a $\mathfrak{T}$-biadmissible strategy pattern for $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$. By Proposition 6, we can assume $(T, \prec, \lhd)$ to be minimal. We consider $T' = (N', n_0, E', \Delta, \mathsf{lab}') = T_{\downarrow \mathsf{Imp}(T, \prec, \lhd)}$ the pattern restricted to predecessors of important nodes. Pattern T' contains at most $|\mathsf{Imp}(T, \prec, \lhd)| - 1 \leq 2|\Sigma|$ intersection nodes. One can thus bound the number of noticeable nodes (recall that noticeable nodes gather intersection nodes and important nodes) by $2|\Sigma| + 2|\Sigma| = 4|\Sigma|$. Moreover, from condition **(c)** of minimality, we deduce that there are no more than $2|Q| - 1$ nodes between two noticeable nodes. Otherwise there would be three nodes with the same label that share the same set of important nodes. This implies that $|N'| \leq 4|\Sigma|(2|Q| - 1) + 4|\Sigma| = 8|\Sigma||Q|$, and concludes the first step.

For the second step, from condition **(b)** of minimality, from every node $n' \in N'$, there can be a node $n_2 \in N$ with $(n', n_2) \in E$ and such that $\mathsf{Sub}(T, n_2) = (N_2, n_2, E_2, \Delta, \mathsf{lab}_2)$ and $N_2 \cap \mathsf{Imp}(T, \prec, \lhd) = \emptyset$ and $\mathsf{Sub}(T, n_2)$ has a unique branch (by condition **(b)**). Yet, conditions **(a)** and **(c)** ensure in that case that $|N_2| \leq 2(|Q| - |\mathfrak{T}|) + 1$. Otherwise there are three nodes with the same label (impossible because of **(c)**) or a node with a label in target that is not a leaf (impossible because of **(a)**).

To conclude we deduce that for each node $n' \in N'$, there might be at most a unique node $n_2 \in N$ with $(n', n_2) \in E$ such that $\mathsf{Sub}(T, n_2) = (N_2, n_2, E_2, \Delta, \mathsf{lab}_2)$ and $N_2 \cap \mathsf{Imp}(T, \prec, \lhd) = \emptyset$ and $N_2 \cap N' = \emptyset$ and $|N_2| \leq 2(|Q| - |\mathfrak{T}|) + 1$. Since $|N'| \leq 8|\Sigma||Q|$, we deduce that the size of a minimal $\mathfrak{T}$-biadmissible strategy pattern for $\mathcal{P}$ is at most $16|\Sigma| \cdot |Q| \cdot (|Q| - |\mathfrak{T}| + 1)$.

For what concerns the height, the worst case is a unique branch contains all the important nodes (separated by $2|Q|$ nodes, thanks to **(c)**) and finish with $2(|Q| - |\mathfrak{T}|) + 1$ nodes to reach the target. Since the number of important nodes is bounded by twice the number of letters in Σ, we obtain that the height is bounded by $4|\Sigma| \cdot |Q| + 2(|Q| - |\mathfrak{T}|) + 1$.

C.4 Proof of Theorem 3

Theorem 3. *1.* TARGET*[$\mathcal{L}$] is* NP*-complete.*

2. If there exists an execution $\theta \in \Theta_{\mathcal{L}}$ such that $\mathsf{End}(\theta) \subseteq \mathfrak{T}$, then there exists an execution $\theta' \in \Theta_{\mathcal{L}}$ such that $\mathsf{End}(\theta') \subseteq \mathfrak{T}$ and $nbproc(\theta') \leq 16|\Sigma| \cdot |Q| + 4|\Sigma| \cdot (|Q| - |\mathfrak{T}| + 1)$ and $|\pi_p(\theta')| \leq 4|\Sigma| \cdot |Q| + 2(|Q| - |\mathfrak{T}|) + 1$ for every $p \leq nbproc(\theta')$.

Proof of Theorem 3.1

Proof. Using Proposition 5 we deduce that there exists an execution $\theta \in \Theta_{\mathcal{L}}$ such that $\mathsf{End}(\theta) \subseteq \mathfrak{T}$ iff there exists a $\mathfrak{T}$-biadmissible strategy pattern, and Lemma 8 allows us to look only for $\mathfrak{T}$-biadmissible strategy patterns whose size is polynomial in the size of the broadcast protocol $\mathcal{P}$. Thus we deduce a non-deterministic polynomial time algorithm which consists in guessing a strategy pattern (equipped with two orders) of polynomial size and then verifying whether it is $\mathfrak{T}$-biadmissible (this can be done in polynomial time thanks to Lemma 6). This proves that $\textsc{Target}[\mathcal{L}]$ is in NP. Moreover, since $\textsc{Target}[\mathcal{L}]$ is harder than $\textsc{Reach}[\mathcal{L}]$ (see Remark 1), we establish that $\textsc{Target}[\mathcal{L}]$ is NP-complete.

Proof of Theorem 3.2

Proof. We consider a minimal $\mathfrak{T}$-biadmissible strategy pattern $(T, \prec, \lhd)$. The cutoff on the number of processes is proved by constructing an execution that has two phases.

The first phase consists in filling all the important nodes (and only them) with at least one process. This can be done with less processes than $|T_{\downarrow \mathsf{Imp}(T,\prec,\lhd)}|$ using the same technique as the ones in the proof of Lemma 5 and using broadcasts that lead to nodes of $\mathsf{NewBroad}(T, \prec, \lhd)$). The minimality of T implies an upper bound of $8|\Sigma||Q|$ on the number of processes needed (see proof of Proposition 8 to get the bound on the size of $T_{\downarrow \mathsf{Imp}(T,\prec,\lhd)}$) .

The second phase consists in emptying the nodes towards target nodes. This is also done as in the proof of Lemma 5 but this time using broadcasts that lead to nodes of $\mathsf{LastBroad}(T, \prec, \lhd)$ and in the order defined by $\lhd$ rather than $\prec$. Since at the end of the first phase all the processes are in important nodes, we only need to consider the pattern restricted to $T_{\downarrow \mathsf{Imp}(T,\prec,\lhd)}$ plus at most one branch per important node that does not contain an important node and leads to a target node. For $k = |\mathsf{Imp}(T, \prec, \lhd)|$, we let $\theta_{imp} \in \Theta_{\mathcal{L}}$ be an execution obtained following the same techniques as for the proof of Lemma 5 and which respects the following properties:

1. $\{\pi_p(\theta_{imp}) \mid p \in [1..nbproc(\theta_{imp})]\} = \{h(n) \mid n \in \mathsf{Imp}(T, \prec, \lhd)\}$
2. $nbproc(\theta_{imp}) \leq |T_{\downarrow \mathsf{Imp}(T,\prec,\lhd)}|$

We define inductively the function empt that, given a node $n \in N$ and an execution such that there is a process in all the important nodes greater than n w.r.t. $\lhd$, extends the execution emptying all the non target nodes in the order given by $\lhd$. Formally, $\mathsf{empt}(n, \theta) = \theta$ if n is the maximal node. Otherwise, letting n' for the successor of n w.r.t. $\lhd$, we define $\mathsf{empt}(n, \theta)$ by:

- $\mathsf{empt}(n', \theta)$ if $\{p \mid \pi_p(\theta) = h(n)\} = \emptyset$; If there are no processes in n we continue with the following node.
- $\mathsf{empt}(n', \theta)$ if $\mathsf{lab}(n) \in \mathfrak{T}$; If node n is labelled by a state of $\mathfrak{T}$ we leave processes in node n unchanged and we continue with the following node.
- $\mathsf{empt}(n', \theta')$ if $\{p \mid \pi_p(\theta) = h(n)\} = \{p_1, \ldots, p_k\}$ and there exists an edge $e = (n, n_1)$ with $n \lhd n_1$ and such that e is labelled with an active action $\delta = \mathsf{lab}(e)$ and where $\theta' = \theta \xrightarrow{p_1,\delta,\emptyset} \ldots \xrightarrow{p_k,\delta,\emptyset} \gamma$. If there is an edge labelled by an active action outgoing of n, all the processes in n perform this action with an empty reception set.
- $\mathsf{empt}(n', \theta')$ if $\{p \mid \pi_p(\theta) = h(n)\} = P$ and there exists an edge $e = (n, n_1)$ with $n \lhd n_1$ and such that e is labelled with a reception of message m, $\delta = (\mathsf{lab}(n), ??m, q)$. We consider $e_m = (n_1, n_2)$ such that $\mathsf{lab}(e_m)$ is a broadcast of m and such that $n_2 \in \mathsf{Imp}(T, \prec, \lhd)$ with

$n \lhd n_2$. Such an edge exists because T is coadmissible. By assumption on θ, there exists $p' \in [1..nbproc(\theta)]$ such that $\pi_{p'}(\theta) = h(n_2)$. We consider the execution $follow(\theta, p', h(n_1))$, and write p_m for the additional process in that execution compared to θ. Then, we let $\theta' = follow(\theta, p', h(n_1)) \xrightarrow{p_m, \mathsf{lab}(e), P} \gamma$ where $\forall p \in P, \gamma(p) = q$. *i.e.* the additional process p_m broadcasts message m to the processes of P only, and they move along edge e. Here $\pi_{p_m}(\theta) = h(n_2)$.

By definition of the function empt, only in the last case the number of processes is incremented by 1.

Applying iteratively empt on θ_{imp} starting from the minimal node (with respect to $\lhd$), one obtains a local execution in which all the processes end in a target state. An additional process is added (through the function *follow*) only in the case of a reception, and at most one process is added by edge labelled by a reception. Moreover in θ_{imp} all the processes are in important nodes and while applying empt the processes remain together, hence the execution $\theta = \mathsf{empt}(n_0, \theta_{imp})$ visits only $|\mathsf{Imp}(T, \prec, \lhd)|$ branches that do not belong to $T_{\downarrow \mathsf{Imp}(T, \prec, \lhd)}$. As a consequence we can bound the number of processes by $nbproc(\theta) \leq nbproc(\theta_{imp}) + |T_{\downarrow \mathsf{Imp}(T, \prec, \lhd)}| + |\mathsf{Imp}(T, \prec, \lhd)|(2(|Q| - |\mathfrak{T}|) + 1) \leq 16|\Sigma| \cdot |Q| + 4|\Sigma|(|Q| - |\mathfrak{T}| + 1)$ (in fact the term $|T_{\downarrow \mathsf{Imp}(T, \prec, \lhd)}|$ is the number of additive broadcast that needs to be performed to bring all the nodes out of $T_{\downarrow \mathsf{Imp}(T, \prec, \lhd)}$ and then for each of the $|\mathsf{Imp}(T, \prec, \lhd)|$ branches, there might be at most $2(|Q| - |\mathfrak{T}|) + 1$ broadcast necessary to bring the processes at the end of the branch).

To conclude, as in the case of Reach[$\mathcal{L}$], the upper bound on the past of each process trivially coincides with the upper bound on the height of T.

D Proofs for Section 4

D.1 Undecidability of Reach[$\mathcal{LC}$] and Target[$\mathcal{LC}$]

Theorem 4. Reach*[$\mathcal{LC}$] is undecidable and* Target*[$\mathcal{LC}$] restricted to complete protocol is undecidable.*

Proof. We begin by recalling the definition of Minsky machines [14]. A *deterministic Minsky machine* manipulates two integer variables c_1 and c_2, which are called counters, and it is composed of a finite set of instructions. Each of the instructions is either of the form (1) $L : c_i := c_i + 1$; `goto` L' or (2) L : `if` $c_i = 0$ `then goto` L' `else` $c_i := c_i - 1$; `goto` L'', where $i \in \{1, 2\}$ and L, L', L'' are labels preceding each instruction. Furthermore there is a special label L_F from which nothing can be done. The halting problem then asks whether or not the execution starting from L_0 with both counters equal to 0 reaches L_F. Without loss of generality, we can assume that when the machines reaches L_F, the values of the two counters are equal to 0.

We begin with proving that Reach[$\mathcal{LC}$] is undecidable. For this, we encode a Minsky machine in the broadcast protocol $\mathcal{P}$ given in the Fig. 6, 7, 8, 9 and 10. The protocol $\mathcal{P}$ is built so as to simulate twice the run of the Minsky machine. In order to do so, in $\mathcal{P}$, (at least) two processes will decide the sequence of instructions of the Minsky machine (represented on Figs. 7 and 8). The remaining processes will encode the values of the counters (see Fig. 9): precisely, the number of processes in the state 1_i will represent the value of counter c_i.

Here are some key points on how this protocol is working along a local clique execution:

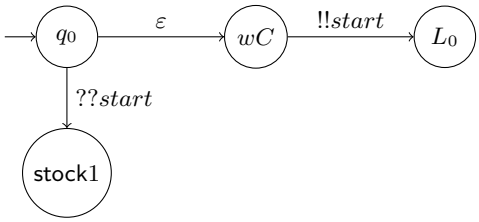

Fig. 6. Initialization phase for REACH[$\mathcal{LC}$].

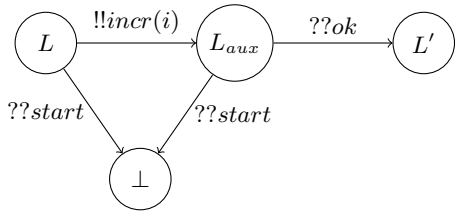

Fig. 7. Encoding $L : c_i := c_i + 1;$ `goto` L'.

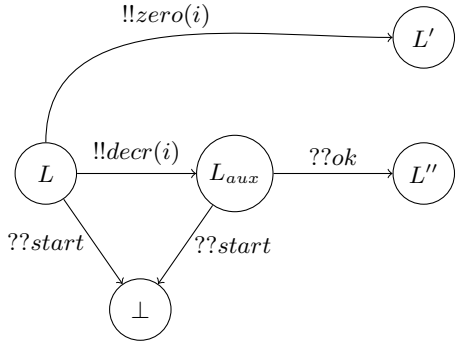

Fig. 8. Encoding $L :$ `if` $c_i = 0$ `then goto` L' `else` $c_i := c_i - 1;$ `goto` L''.

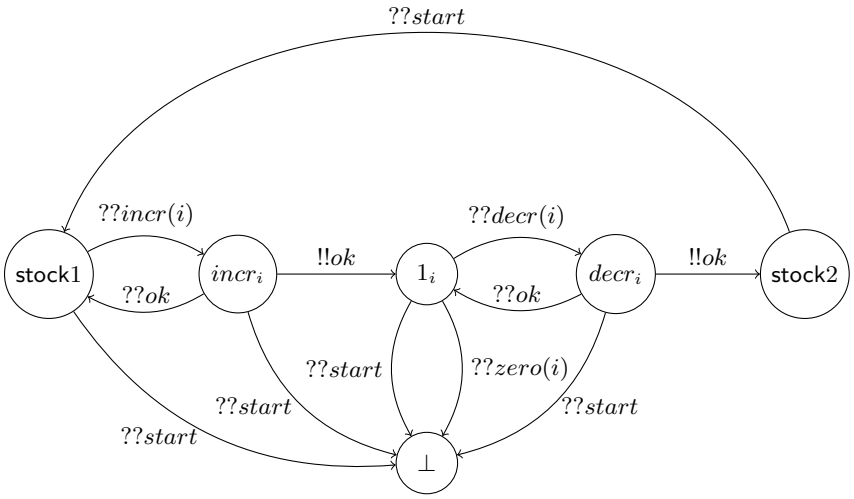

Fig. 9. Encoding counter c_i.

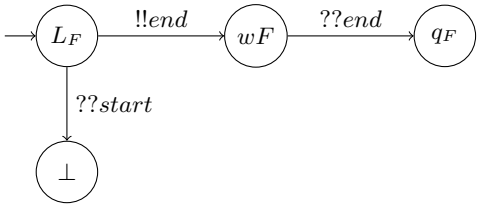

Fig. 10. Ending phase for REACH[$\mathcal{LC}$].

- During the initialization phase (see Fig. 6), using the internal actions, some processes may stay in q_0, some other may move to a waiting state wC (standing for waiting controller) where they will wait to be the next controller.
- As soon as a process in wC moves to L_0, it broadcasts message *start* and the simulation properly begins. Note that after this step no process can be in state q_0, the clique semantics guarantees that they moved to stock1. Yet there can be some processes in wC.
- Intuitively the process in L_0 will simulate the sequence of instructions of the Minsky machine while the processes in stock1 will be used to encode the counter in the first simulation of the run. Later, some of the processes from state wC will be used to perform the simulation a second time.
- Let us first comment on the following. What happens if a process in wC moves to L_0 while another process is already acting as the controller of the counter machine (*i.e.* is in state L or L_{aux} of the Figs. 7 or 8)? Then the first controller will receive the message *start* and move to state $\perp$ which is a deadlock (as shown in the above mentioned figures). The processes simulating the counters will all move either to $\perp$ or to stock1 (see Fig. 9). As a consequence, a new simulation starts and it cannot interfere with the previous one that has been stopped.
- Then let us explain how works precisely the simulation of each action:
  Incrementing a counter. To simulate an increment instruction of the form $L : c_i := c_i + 1$; `goto` L', the controller behaves as represented in Fig. 7. It broadcasts the message $incr(i)$, which is received by all the processes in state stock1 which all move to $incr_i$ (see Fig. 9). Then it waits to receive an acknowledgement message ok, this message is broadcast by one process in $incr_i$; as a consequence, the controller moves to state L' and exactly one process moves to 1_i (the one which performed the broadcast of ok), whereas all the other processes in $incr_i$ move back to stock1.
  Decrementing a counter. The decrement of a counter (see Fig. 8) is pretty similar to the increment, and only one process will move from 1_i to stock2, the pool of processes for the second simulation.
  Zero testing. When the controller mimics an instruction of the shape L : `if` $c_i = 0$ `then goto` L' `else` $c_i := c_i - 1$; `goto` L'', it has no way to know whether some processes are in state 1_i or not. However, it can choose to broadcast $zero(i)$, even if the counter value is not 0, *i.e.* if there are some processes in 1_i. A consequence of this is that all the processes in 1_i are sent to $\perp$. If the process performs the broadcast of $zero(i)$ while some processes are in 1_i, it *cheats*: it assumes that the counter value is 0 although it is not the case.
- If we observe the behavior of the processes simulating the counters, we see that for each pair increment-decrement, exactly one counter process is sent from stock1 to stock2 and possibly some processes are lost (*i.e.* moved to $\perp$) when the controller cheats.

– What happens at the end of a simulation of the two-counter machine run? When a process controller reaches L_F, it may move to state wF (standing for waiting final) (see Fig. 10).
– A new simulation can then begin with a process moving from wC to L_0 and broadcasting *start*. When the message *start* is sent (remember that the first simulation has to be finished otherwise, it will correspond to a new first simulation as explained previously), all the processes in stock1 are moved to $\perp$ and all the processes in stock2 are moved to stock1. Hence there are at most as many processes in stock1 as the number of decrements during the first simulation.
  Because we restrict to local executions, the new controller will perform exactly the same choices as the previous one, mimicking hence the exact same run. Consequently there are two options:
  1. Either the first controller has cheated at some point, then the second one will be eventually stuck, because there will not be enough processes in stock1 to answer an increment request.
  2. Or the first controller did not cheat, which means that the simulation was correct, and the second simulation will then also be correct. The second controller will reach L_F, then it will reach wF broadcasting *end* and so the first controller will move from wF to q_F (in case the first controller was not in wF, another simulation has to be performed).

Under this construction, the Minksy machine halts if and only if there is an execution $\theta \in \Theta_{\mathcal{LC}}$ such that $q_F \in \mathsf{End}(\theta)$. This proves that $\textsc{Reach}[\mathcal{L}]$ is undecidable.

Remark 2. Note that this proof heavily relies on three features: (1) the execution is local, (2) the execution is a clique execution and (3) the protocols can be incomplete. Indeed, restricting to clique executions allows one to distinguish controllers from processes encoding counters, and to be sure that exactly one process answers to increment/decrement requests. Second, restricting to local execution ensures that the second sequence of instructions exactly repeats the first one. Last, we use the fact that the protocol is not complete to ensure that some processes can stay in wC and that all the processes arriving for the first time in L_0 share the same history.

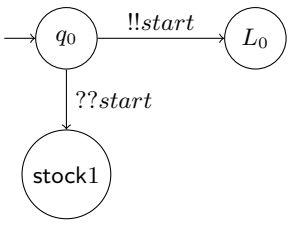

Fig. 11. Initialization phase for $\textsc{Target}[\mathcal{LC}]$.

To prove that $\textsc{Target}[\mathcal{LC}]$ restricted to complete protocol is undecidable, we use the same kind of reasoning and we reuse Fig. 7, 8 and 9 for the simulation of the actions and of the counters. The initialization phase differs and is represented in Fig. 11. Note that in that case, a single process controller will reach L_0 and we will guarantee that it does not cheat by

defining as target set of states $\mathfrak{T} = \{\mathsf{stock1}, \mathsf{stock2}, L_F\}$. Following the same argumentation as the one before, one can show that the target set of states allows us to ensure that the unique controller process (which reaches L_F) did not cheat otherwise there will be some processes in $\bot$. Hence for the broadcast protocol $\mathcal{P}$ we build here, we can show that there exists an execution $\theta \in \Theta_{\mathcal{LC}}$ such that $\mathsf{End}(\theta) \subseteq \mathfrak{T}$ iff the Minsky machine halts. Note that the protocol we describe is not actually complete but it will not harm the reduction to complement it by adding an edge to $\bot$ for each unspecified reception.

Remark 3. In contrast to the undecidability proof for REACH[$\mathcal{LC}$] in which the run of the Minsky machine is simulated twice in a row, restricting to local executions is not necessary here, so that we can show that TARGET[$\mathcal{C}$] is undecidable.

D.2 Decidability of Reach[$\mathcal{LC}$] for complete broadcast protocols

We provide here details and proofs for Subsection 4.2.

Reasoning on the abstract transition system. We define a natural order $\preceq$ on the set of abstract configurations Λ as follows: $(\mathbb{M}, t) \preceq (\mathbb{M}', t')$ if and only if $t = t'$ and $\mathbb{M}(q, fl, b) \leq \mathbb{M}'(q, fl, b')$ for all (q, fl, b).

Since the set $Q \times \{\mathbf{m}, \mathbf{s}\} \times \{!!_{ok}, !!_{no}\}$ is finite and since the set $\{\varepsilon, !!\}$ is also finite, then Dickson's lemma allows us to say that $(\Lambda, \preceq)$ is a well-quasi-order (wqo). From this we know that for any infinite sequence $(\lambda_i)_{i\in\mathbb{N}} \in \Lambda^{\mathbb{N}}$, there exists $i < j$ such that $\lambda_i \preceq \lambda_j$.

For a set $S \subseteq \Lambda$, we denote by $\uparrow S$ its *upward-closure* (with respect to $\preceq$) defined by $\uparrow S = \{\lambda' \mid \exists \lambda \in S \text{ s.t. } \lambda \preceq \lambda'\}$. A set $S \subseteq \Lambda$ is said *upward-closed* if $S = \uparrow S$. Since $(\Lambda, \preceq)$ is a wqo, for each upward-closed set $S \subseteq \Lambda$ there exists a finite basis $\{b_0, \ldots, b_k\} \subseteq S$ such that $S = \uparrow \{b_0, \ldots, b_k\}$. This provides a way to finitely represent infinite subsets of Λ. We will now show that the abstract transition system $\mathcal{T}_{\mathcal{P}}^{\mathcal{LC}} = (\Lambda, \Lambda_0, \Rightarrow)$ equipped with the wqo $\preceq$ is a well-structured transition system [1,12] and that one can effectively compute the predecessors of upward-closed sets.

The following monotonicity lemma is immediate given the definition of the transition relation $\Rightarrow$.

Lemma 9 (Monotonicity lemma). *Given $\lambda_1, \lambda_2, \lambda_1' \in \Lambda$ such that $\lambda_1 \Rightarrow \lambda_2$ and $\lambda_1 \preceq \lambda_1'$, there exists $\lambda_2' \in \Lambda$ such that $\lambda_1' \Rightarrow \lambda_2'$ and $\lambda_2 \preceq \lambda_2'$.*

Now we define an operator which allows to compute a finite basis for the set of one-step predecessors of an upward-closed set. For a set $S \subseteq \Lambda$, we define $Pre(S) = \{\lambda \in \Lambda \mid \exists \lambda' \in S \text{ s.t. } \lambda \Rightarrow \lambda'\}$. We will now see that given an abstract configuration λ, we can compute (using the definition of $\Rightarrow$) a finite basis of the set $Pre(\uparrow \{\lambda\})$. Let $\lambda = (\mathbb{M}, t)$ be an abstract configuration in λ. We define $p(\lambda)$ as follows:

- if $t = !!$, then $p(\lambda) = \{\lambda' \mid \lambda' \Rightarrow_{\varepsilon} \lambda\}$.
- if $t = \varepsilon$, then

$$p(\lambda) = \{\lambda' \mid \lambda' \Rightarrow_{!!} \lambda\} \cup$$
$$\bigcup_{(q_1, !!m, q_2) \in \Delta} \{\lambda' \mid \lambda' \Rightarrow_{!!} (\mathbb{M} \oplus \{\{(q_2, \mathbf{s}, !!_{ok})\}\}, \varepsilon)\} \cup$$
$$\bigcup_{(q_1, !!m, q_2), (q_1, ??m, q_3) \in \Delta} \{\lambda' \mid \lambda' \Rightarrow_{!!} (\mathbb{M} \oplus \{\{(q_2, \mathbf{s}, !!_{ok}), (q_3, \mathbf{m}, !!_{ok})\}\}, \varepsilon)\}$$

Let us explain the second part of this definition. Since our aim is to compute a basis of $Pre(\uparrow \{\lambda\})$, in order to compute the predecessors of $\uparrow \{\lambda\}$, we need to take into account for the transition relation $\Rightarrow_{!!}$, that the element witness of the broadcast might not be in λ but in a configuration belonging to its upward closure, this is the reason why we include the sets $\bigcup_{(q_1,!!m,q_2)\in\Delta}\{\lambda' \mid \lambda' \Rightarrow_{!!} (\mathbb{M} \oplus \{\{(q_2, \mathbf{s}, !!_{ok})\}\}, \varepsilon)\}$ and $\bigcup_{(q_1,!!m,q_2),(q_1,??m,q_3)\in\Delta}\{\lambda' \mid \lambda' \Rightarrow_{!!} (\mathbb{M} \oplus \{\{(q_2, \mathbf{s}, !!_{ok}), (q_3, \mathbf{m}, !!_{ok})\}\}, \varepsilon)\}$. This kind of assumptions to compute the predecessor basis of an upward closed set is similar to the one proposed in [10] to solve Reach[$\mathcal{C}$]. Note however that, for the transition relation $\Rightarrow_\varepsilon$, such trick is not necessary. Note also that given a configuration $\lambda \in \Lambda$, $p(\lambda)$ is finite since it contains abstract configurations, where the cardinality of the multiset is at most the cardinal of the multiset of λ plus 2. Using the definition of $\Rightarrow$, one obtains the following lemma.

Lemma 10. *For all $\lambda \in \Lambda$, $p(\lambda)$ is finite, effectively computable and $\uparrow p(\lambda) = Pre(\uparrow \{\lambda\})$.*

Proof of Theorem 5. We consider the following upward closed set: $F = \bigcup_{(q_F,fl,b)} \uparrow \{(\{\{(q_F, fl, b)\}\}, \varepsilon)\} \cup \uparrow \{(\{\{(q_F, fl, b)\}\}, !!)\}$. Using the methodology presented in [1,12], thanks to Lemmas 10 and 9, we know it is possible to compute a finite basis for the set $Pre^*(F) = \{\lambda \in \Lambda \mid \exists \lambda' \in F \text{ s.t. } \lambda \Rightarrow^* \lambda'\}$. Hence we can decide whether there exists an abstract local clique execution ξ of $\mathcal{P}$ such that $q_F \in \mathsf{End}(\xi)$: it suffices to test whether $\lambda_0 \in pre^*(F)$. Applying Lemma 1, we conclude that Reach[$\mathcal{LC}$] is decidable.

Proposition 7. Reach*[$\mathcal{LC}$] is* NPR.

Proof. In order to prove that Reach[$\mathcal{LC}$] is non-primitive recursive, we provide a PTIME reduction from Reach[$\mathcal{C}$] to Reach[$\mathcal{LC}$], and conclude since Reach[$\mathcal{C}$] is Ackermann-complete [16]. Intuitively the only difference between the semantics $\mathcal{C}$ and $\mathcal{LC}$ is that within $\mathcal{LC}$, processes with the same history take the same decisions. However, with the simple initialization gadget, represented in Fig. 12, one can assign a different history to each individual process, before they actually start the protocol. Indeed, in a clique topology, if a process performs a broadcast, it has a unique history forever. Hence to have k processes in q_0 with a different history, it suffices to perform k broadcasts of init and then one broadcast of start making all the processes move to q_0 thanks to the clique topology.

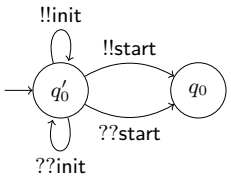

Fig. 12. Initialization gadget.

This provides a PTIME reduction from Reach[$\mathcal{C}$] to Reach[$\mathcal{LC}$] (which also works for the target problem). Let us formally define the construction illustrated in Fig. 12. Given a protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ we let $\mathcal{P}' = (Q', q'_0, \Sigma', \Delta')$ with

- $Q' = Q \cup \{q'_0\}$;
- $\Sigma' = \Sigma \cup \{\mathsf{init}, \mathsf{start}\}$;

– $\Delta' = \Delta \cup \{(q'_0, !!\mathsf{init}, q'_0), (q'_0, ??\mathsf{init}, q'_0), (q'_0, !!\mathsf{start}, q_0), (q'_0, ??\mathsf{start}, q_0)\}$

We now show that reaching q_F under clique semantics in $\mathcal{P}$ is equivalent to reaching q_F under local strategies and clique semantics in $\mathcal{P}'$. Formally, we prove that there exists an execution $\theta \in \Theta_{\mathcal{C}}[\mathcal{P}]$ such that $q_F \in \mathsf{Occur}(\theta)$ iff there exists an execution $\theta' \in \Theta_{\mathcal{LC}}[\mathcal{P}']$ such that $q_F \in \mathsf{Occur}(\theta')$.

($\Leftarrow$) Let $\theta' \in \Theta_{\mathcal{LC}}[\mathcal{P}']$ be a local clique execution of $\mathcal{P}'$ such that $q_f \in \mathsf{Occur}(\theta')$. From the definition of $\mathcal{P}'$, this execution must start with a series of broadcasts of init followed by the broadcast of start. We define θ as the suffix of execution θ' after the broadcast of start. Notice that $\theta \in \Theta_{\mathcal{C}}[\mathcal{P}]$ is a clique execution of $\mathcal{P}$, and that $q_f \in \mathsf{Occur}(\theta') = \mathsf{Occur}(\theta) \cup \{q'_0\}$.

($\Rightarrow$) Let $\theta \in \Theta_{\mathcal{C}}[\mathcal{P}]$ be a clique execution of $\mathcal{P}$ such that $q_f \in \mathsf{Occur}(\theta)$. In the local clique semantics for protocol $\mathcal{P}'$ we define the following execution θ'. First, each process broadcasts the message init in turn, then one process broadcasts start. Recall that, in local clique executions, when a process performs a broadcast, its history becomes unique for ever. Therefore, at this stage, each of the processes has its own history. Moreover, by definition of $\mathcal{P}'$, all the processes are in state q_0. Hence, from then on, one can reproduce execution θ. This execution θ' is thus a local clique execution in $\mathcal{P}'$ such that $q_f \in \mathsf{Occur}(\theta')$.