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.

1998 ACM Subject Classification F.3 Logics and Meanings of Programs, F.1.1 Models of Computation

Keywords and phrases Broadcast Networks, Parameterized Verification, Local strategies

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

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 [14], 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 [11], token-passing [6, 2], message passing [5] or shared memory [12]. In his nice survey on such parameterized models [10], 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 [4].

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

Conference title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Broadcast protocols. Among the various parametric models of networks, broadcast protocols, originally studied by Esparza *et al.* [11], 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 [8]: 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 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 [7]. Another option to recover decidability of the reachability problem is to restrict the topologies to clique graphs [9], 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.

Due to lack of space, omitted details and proofs can be found in the companion research report [3].

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 [8, 9, 7], 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 [7]. 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 $\mathcal{T}_\mathcal{P} = (\Gamma, \Gamma_0, \to)$ 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 $\to \subseteq \Gamma \times \mathbb{N} \times \Delta \times 2^\mathbb{N} \times \Gamma$ is the transition relation defined as follows: $(\gamma, p, \delta, R, \gamma') \in \to$ (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 [9] and that such networks correspond to broadcast protocols with no rendez-vous [11]. 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 [7] and Reach$[\mathcal{C}]$ is decidable and Non-Primitive Recursive (NPR) [9, 11] (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 [7]. 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. 3]	Undecidable [Thm. 5] Decidable and NPR for complete protocols [Thm. 7]	NP-complete [Thm. 4]	Undecidable [Thm. 5]

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. 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 3.1) and Target$[\mathcal{L}]$ (Theorem 4.2). 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

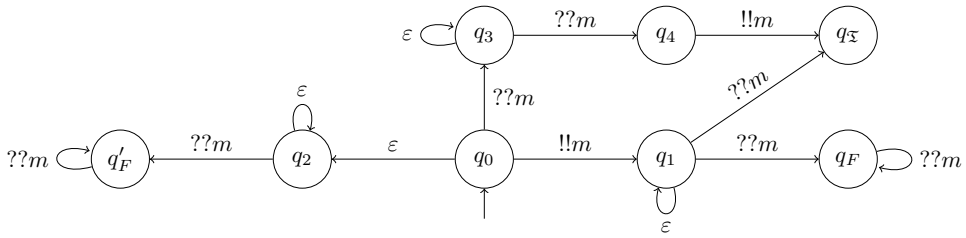

Figure 1 Example of a broadcast protocol.

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.

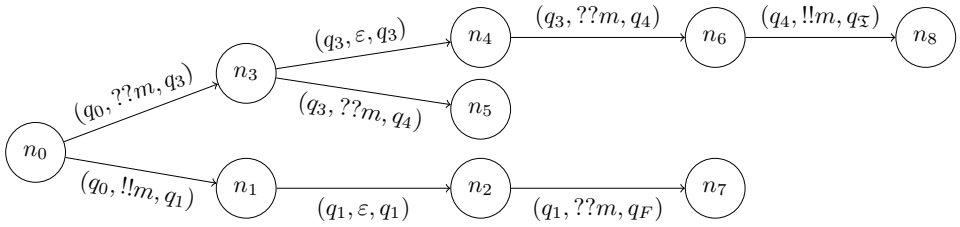

Figure 2 A strategy pattern for the broadcast protocol depicted Fig. 1.

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. 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.

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 3.1, 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 3.1 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 $\textsc{Reach}[\mathcal{L}]$ consists then in guessing a strategy pattern of polynomial size and an order and then verifying whether it is q_F-admissible.

▶ **Theorem 2.** 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 3.1), 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[$\mathcal{L}$].

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 $p \in [1..nbproc(\theta')]$.

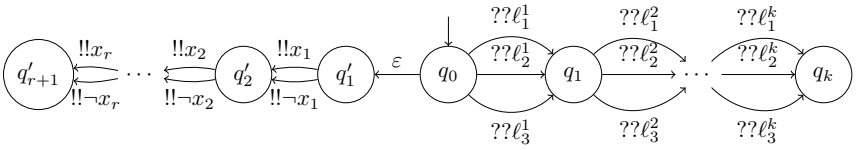

Figure 3 Encoding a 3-SAT formula into a broadcast protocol.

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 2, we obtain the precise complexity of REACH[$\mathcal{L}$].

▶ **Theorem 3.** 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

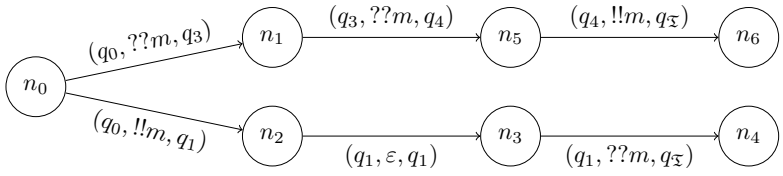

Figure 4 A $\mathfrak{T}$-coadmissible strategy pattern on the example protocol of Fig. 1.

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 4.** *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. 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 5.** 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 $\mathit{fl}_1 = \mathbf{s}$, $\mathit{card}(G_{(q_1,\mathit{fl}_1,!!_{ok})}(1)) = 0$ and if $\mathit{fl}_1 = \mathbf{m}$, then there exists $q' \in Q$ such that $G_{(q_1,\mathit{fl}_1,!!_{ok})}(1) = \{\{(q', \mathit{fl}_1, !!_{ok})\}\}$ and such that $(q, ??m, q') \in \Delta$;
b. if $(q, \mathit{fl}, b) \neq (q_1, \mathit{fl}_1, !!_{ok})$ or $i \neq 1$, then there exists $q' \in Q$ such that $G_{(q,\mathit{fl},b)}(i) = \{\{(q', \mathit{fl}, !!_{ok})\}\}$ 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 $(q_1, \mathit{fl}_1, !!_{ok})$ in $\mathbb{M}$. The transition $(q_1, !!m, q_2)$ tells us which broadcast is simulated. Then the functions $G_{(q,\mathit{fl},b)}$ 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 $\mathit{fl}_1 = \mathbf{m}$, 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 $(q, \mathit{fl}, b) \in Q \times \{\mathbf{m}, \mathbf{s}\} \times \{!!_{ok}, !!_{no}\}$, such that $F_{(q,\mathit{fl},b)} : [1..\mathbb{M}(q, \mathit{fl}, b)] \to \mathbf{HM}$, and

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

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

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

Intuitively the functions $F_{(q,\mathit{fl},b)}$ 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 $\mathit{fl} = \mathbf{s}$) of processes sharing the same history. Condition 1. states that if there are multiple processes ($\mathit{fl} = \mathbf{m}$) then they can be matched to more states in the protocol, but if it is single ($\mathit{fl} = \mathbf{s}$) 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 6.** *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, 13]. 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 7.** 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 [14] or shared memory [12]). 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 A. 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 *Proc. of VMCAI'14*, volume 8318 of *LNCS*, pages 262–281, 2014.

3 Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Distributed local strategies in broadcast networks. Research report, July 2015. https://hal.inria.fr/hal-01170796.

4 Benedikt Bollig. Logic for communicating automata with parameterized topology. In *Proc. of CSL-LICS'14*, page 18. ACM, 2014.

5 Benedikt Bollig, Paul Gastin, and Jana Schubert. Parameterized verification of communicating automata under context bounds. In *Proc. of RP'14*, volume 8762 of *LNCS*, pages 45–57, 2014.

6 Edmund M. Clarke, Muralidhar Talupur, Tayssir Touili, and Helmut Veith. Verification by network decomposition. In *Proc. of CONCUR'04*, volume 3170 of *LNCS*, pages 276–291, 2004.

7 Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *Proc. of FSTTCS'12*, volume 18 of *LIPIcs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

8 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *Proc. of CONCUR'10*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.

9 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *Proc. of FoSSaCS'11*, volume 6604 of *LNCS*, pages 441–455. Springer, 2011.

10 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *Proc. of STACS'14*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

11 Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *Proc. of LICS'99*, pages 352–359. IEEE Computer Society, 1999.

12 Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In *Proc. of CAV'13*, volume 8044 of *LNCS*, pages 124–140, 2013.

13 Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

14 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.

15 Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Proc. of FOCS'90*, pages 746–757. IEEE Computer Society, 1990.

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