Parameterized Verification of Ad Hoc Networks

Giorgio Delzanno[1], Arnaud Sangnier[1], and Gianluigi Zavattaro[2]

[1] University of Genova [2] University of Bologna

Abstract. We study decision problems for parameterized verification of a formal model of Ad Hoc Networks with selective broadcast and spontaneous movement recently proposed by Singh, Ramakrishnan, and Smolka. The communication topology of a network is represented here as a graph. Nodes represent states of individual processes. Adjacent nodes represent single-hop neighbors. Processes are finite state automata that communicate via selective broadcast messages. Reception of a broadcast is restricted to single-hop neighbors. For this model we consider verification problems that can be expressed as reachability of configurations with one node (resp. all nodes) in a certain state. All decision problems are parametric both on the size and on the form of the communication topology of the initial configurations. We draw a complete picture of the decidability boundaries of these problems according to various assumptions on the communication topology of the network, namely static vs mobile and unbounded- vs bounded-path topologies.

1 Introduction

In recent years there has been an increasing interest in the formal verification of protocols used in Ad Hoc Networks. In this setting a node of the network can communicate only with the subset of nodes that are within the range of the its own transmission device. Movement or external factors can dynamically modify the configuration of the network. Building on previous works like [18,6,17,20,21], Singh, Ramakrishnan and Smolka define the ω-calculus [22] as a formal model of Ad Hoc Networks with selective broadcast and spontaneous movement. The structure underlying a configuration is a finite graph that defines the communication topology of the network. Specifically, in this model each node represents an individual process. Each process has an interface. An interface contains a set of group names. Nodes communicate through selective broadcast, i.e., a broadcast message is received only from nodes whose interfaces share names in common with that of the sending node (single-hop neighbors). When the number of nodes is fixed a priori, formal models of Ad Hoc Networks like those proposed in [22] can be verified by using symbolic model checking [9,22]. The study of verification problems for networks of arbitrary size and unknown topology is an interesting and challenging problem for this class of systems.

In this paper we investigate decidability and undecidability of parameterized verification problems for an automata-based model, we named AHN, inspired by the ω-calculus of [22]. In all decision problems we study, initial configurations may have an arbitrary (finite) number of nodes connected with an arbitrary topology. Our investigation takes into account different assumptions on the communication topology. Specifically, we consider configurations with unknown but static topology, with unknown but mobile topology, and with unknown static but bounded path topology. In the latter case we assume that there is an upper bound to the length of simple paths in the network topology.

For these three parameterized cases we present a systematic analysis of the decidability of the following verification problems: (COVER) reachability of a configuration with *one* node in a given state, (TARGET) reachability of a configuration with *all nodes* in a given state, (REPEAT-COVER) existence of a computation traversing *infinitely often* configurations with at least one node in a given state.

Our main negative result is that all three problems are undecidable for arbitrary static topology. The proofs are based on a simulation of a Turing complete formalism which is correct only for topologies of a given type. As the topology is arbitrary, the simulation is preceded by a protocol able to explore the current topology and to start the simulation only if it is of the expected form.

Perhaps surprisingly, all three problems become decidable in the mobile case. This result is similar to what happens in channel systems where introducing lossiness simplifies the verification task [3]. On the contrary, for static bounded path topologies, TARGET and REPEAT-COVER turn out to be undecidable while COVER is still decidable. This last result is similar to what happens in point-to-point communication networks with bounded communication paths [23], but due to broadcast communication we need to resort to a different proof technique. Namely, even if we use the theory of Well-Structured Transition Systems (WSTS) [1,2,10] as in [13,23], we need to consider a stronger ordering on states based on the induced subgraph ordering [4] instead of the subgraph-embedding. To the best of our knowledge, this is the first case of application of the induced subgraph ordering in the context of WSTS.

Related Work. Ethernet-like broadcast communication has been analyzed by Prasad [18] using the Calculus of Broadcasting Systems, in which all processes receive a broadcast message at once. A similar type of broadcast mechanism is used in the Broadcast Protocols of Emerson and Namjoshi [5]. In our setting, this is similar to the case in which all nodes share a common group (the underlying graph is a clique). Ene and Muntean presented the $b\pi$-calculus [6], an extension of the π-calculus [15] with a broadcast such that only nodes listening on the right channel can receive. Wireless broadcast communication has been investigated in the context of process calculi by Nanz and Hankin [17], Singh, Ramakrishnan and Smolka [20,21], Mezzetti and Sangiorgi [14], Godskesen [11], and Merro [12]. In particular Nanz and Hankin [17] consider a graph representation of node localities to determine the receivers of a message, while Godskesen [11] makes use of a neighbour relation. On the contrary, Mezzetti and Sangiorgi [14] and Merro [12] associate physical locations to processes so that the receivers depend on the location of the emitter and its transmission range. As already mentioned, we have been directly inspired by the ω-calculus of Singh, Ramakrishnan and Smolka [20,21]. The ω-calculus is based on the π-calculus. The π-calculus [15] intermixes the communication and mobility of processes by expressing mobility as change of interconnection structure among processes through communication. In the ω-calculus mobility of processes is abstracted from their communication actions, i.e., mobility is spontaneous and it does not involve any communication. In [22] the same authors define a constraint-based analysis for configurations with fixed topologies and a fixed number of nodes. The authors also mention that checking reachability of a configuration from an initial one is decidable for the fragment without restriction. This property is an immediate consequence of the fact that there is no dynamic generation or deletion of processes (i.e. it boils down to a finite-state reachability problem). The symbolic approach in [22] seems to improve verification results obtained with more standard model checking techniques. For instance, in [9] model checking is used for automatic verification of finite-state and timed models of Ad Hoc Networks. In these works the number of nodes in the initial configurations is known and fixed a priori. In order to detect protocol vulnerabilities tools like Uppaal are executed on all possible topologies (modulo symmetries) for a given number of nodes. In [19] Saksena et al. define a symbolic procedure based on graph-transformations to analyze routing protocol for Ad Hoc Networks. The symbolic representation is based on upward closed sets of graphs ordered w.r.t. subgraph inclusion. Their procedure is not guaranteed to terminate. In our paper we consider a non trivial class of graphs (bounded path configurations) for which backward analysis with a similar symbolic representation (upward closure of graphs w.r.t. induced subgraph ordering) is guaranteed to terminate for finite-state descriptions of individual nodes.

Due to lack of space, omitted proofs can be found in Appendix.

2 A Formal Model for Ad Hoc Network Protocols

Syntax. Following [22], a configuration of an Ad Hoc Network is modeled as a graph in which nodes represent processes and edges represent the underlying communication topology. We assume that nodes cannot dynamically be created or deleted. The behavior of a single node is described by a finite-state automaton, called *process*, with either local, broadcast, or reception actions.

Definition 1. *A process P is a tuple $\langle Q, \Sigma, E, Q_0 \rangle$ where: Q is a finite set of control states; Σ is a finite alphabet; $E \subseteq Q \times (\{\tau\} \cup \{\mathbf{b}(a), \mathbf{r}(a) \mid a \in \Sigma\}) \times Q$ is the transition relation; $Q_0 \subseteq Q$ is a set of initial control states.*

The label τ represents an internal action of a process, the label $\mathbf{b}(a)$ represents the broadcast message a sent to all single-hop (or adjacent) neighbors, and the label $\mathbf{r}(a)$ represents the reception of message a. A convenient way to describe the connectivity of the network is to associate to each node an interface that defines the set of group names to which the node belongs. Two nodes are connected if their interfaces share at least one common group name.

Definition 2. *An Ad Hoc Network Protocol (shortly AHN) is a pair $\langle P, \mathcal{G} \rangle$ where P is a process and $\mathcal{G}$ is a denumerable set of group names.*

Semantics. Given an AHN $\langle P, \mathcal{G} \rangle$ with $P = \langle Q, \Sigma, E, Q_0 \rangle$, a *node* n is represented by a pair $\langle q, I \rangle$, where $q \in Q$ is its current *state* and $I \subseteq \mathcal{G}$ is its *interface*. A configuration γ of $\langle P, \mathcal{G} \rangle$ is then a tuple $\langle n_1, \ldots, n_k \rangle$ of nodes with $k \geq 1$.

We use $\mathcal{C}$ to denote the set of configurations associated to $\langle P, \mathcal{G} \rangle$. We define functions σ and ι to extract the state and the interface of a node, i.e., $\sigma(\langle q, I \rangle) = q$ and

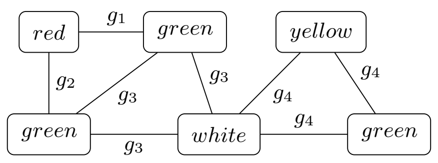

Fig. 1. Graph associated to a configuration.

$\iota(\langle q, I\rangle) = I$. We extend σ and ι to configurations in the natural way. Given a configuration γ, we sometimes consider $\sigma(\gamma)$ as a set rather than a vector and use $q \in \sigma(\gamma)$ to denote that there exists a node n_i in γ such that $\sigma(n_i) = q$. A configuration γ defines a given network topology specified by the graph $G(\gamma)$. The vertices in $G(\gamma)$ are in bijection with the nodes of γ. The label of a vertex is the state of the corresponding node in γ. Furthermore, there exists an edge between two vertices in $G(\gamma)$ if and only if the intersection of the interfaces of the corresponding nodes in γ is not empty. For instance, consider a configuration γ with six nodes such that $n_1 = \langle red, \{g_1, g_2\}\rangle$, $n_2 = \langle green, \{g_1, g_3\}\rangle$, $n_3 = \langle green, \{g_2, g_3\}\rangle$, $n_4 = \langle white, \{g_3, g_4\}\rangle$, $n_5 = \langle yellow, \{g_4\}\rangle$, and $n_6 = \langle green, \{g_4\}\rangle$, the communication topology induced by γ is depicted in Figure 1.

We then define the set of single-hop neighbors of node n_i in a configuration $\gamma = \langle n_1, \ldots, n_k\rangle$ as $Shn(\gamma, i) = \{j \in [1..k] \mid \iota(n_i) \cap \iota(n_j) \neq \emptyset \text{ and } j \neq i\}$, i.e., the set of nodes adjacent to n_i in $G(\gamma)$. Furthermore, given a broadcast message $a \in \Sigma$, we define the set of indexes $Rec(\gamma, a) = \{j \in [1..k] \mid (\sigma(n_j), \mathbf{r}(a), q) \in E \text{ for some } q \in Q\}$. The set of nodes in γ enabled by a broadcast a sent by node n_i is then defined as $Enabled(\gamma, i, a) = Shn(\gamma, i) \cap Rec(\gamma, a)$.

The semantics of an AHN $\langle P, \mathcal{G}\rangle$ with $P = \langle Q, \Sigma, E, Q_0\rangle$ is given by its associated transition system $TS(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow, \mathcal{C}_0\rangle$. $\mathcal{C}$ is the set of configurations associated to $\langle P, \mathcal{G}\rangle$, $\mathcal{C}_0$ is the set of initial configurations defined as $\mathcal{C}_0 = \{\gamma \in \mathcal{C} \mid \sigma(\gamma) \subseteq Q_0\}$ and $\Rightarrow \subseteq \mathcal{C} \times \mathcal{C}$ is the transition relation defined as follows: for $\gamma = \langle n_1, \ldots, n_k\rangle$ and $\gamma' = \langle n'_1, \ldots, n'_k\rangle$, $\gamma \Rightarrow \gamma'$ iff one of the following conditions holds:

local action there exists $i \in [1..k]$ such that $(\sigma(n_i), \tau, \sigma(n'_i)) \in E$, $\iota(n_i) = \iota(n'_i)$, and for all $j \in [1..k] \setminus \{i\}$, $n'_j = n_j$ (**local action**);

broadcast there exists $i \in [1..k]$ such that $(\sigma(n_i), \mathbf{b}(a), \sigma(n'_i)) \in E$, $\iota(n_i) = \iota(n'_i)$, and for all $j \in Enabled(\gamma, i, a)$, $(\sigma(n_j), \mathbf{r}(a), \sigma(n'_j)) \in E$, and for all $l \notin (Enabled(\gamma, i, a) \cup \{i\})$ $n'_l = n_l$.

We denote by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$. An execution is a sequence $\gamma_0\gamma_1 \ldots$ such that $\sigma(\gamma_0) \in Q_0$ and $\gamma_i \Rightarrow \gamma_{i+1}$ for $i \geq 0$. As an example, consider the following set of rules:

$$\begin{array}{cc} (white, \tau, yellow) & (yellow, \mathbf{b}(m), red) \\ (green, \mathbf{r}(m), yellow) & (white, \mathbf{r}(m), yellow) \end{array}$$

Starting with a configuration with white and green nodes, once a red alarm is detected by a white node (i.e. the node turns yellow), it is flooded to all single-hop neighbors. In turn, they forward the red alarm to their neighbors, and so on. In Figure 2 we show an

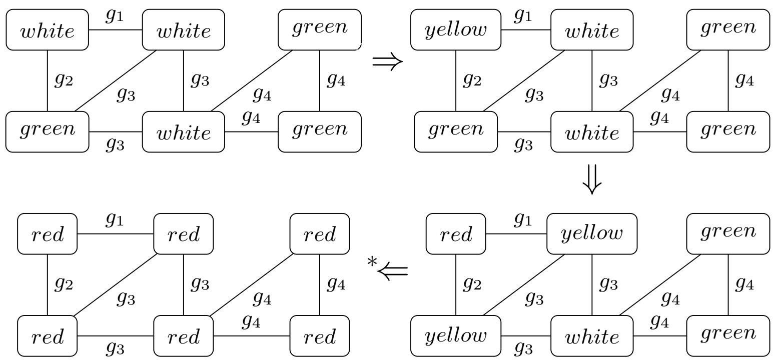

Fig. 2. Example of execution

execution of the previous rules. After some steps, the alarm reaches all the nodes of the communication graph.

Decision Problems. In this section we consider decision problems related to verification of safety and liveness properties studied in the literature for models like Petri nets [7,8]. All the problems are formulated in the parameterized case in which the size and the topology of the networks are not known. In the following definitions we assume an AHN $\langle P, \mathcal{G}\rangle$ with transition system $TS(P,\mathcal{G}) = \langle \mathcal{C}, \Rightarrow, \mathcal{C}_0\rangle$.

The first problem is *control state reachability* (COVER) defined as follows: given a control state q of P, do there exist $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ such that $\gamma \Rightarrow^* \gamma'$ and $q \in \sigma(\gamma')$? We recall that a configuration γ is initial if $\sigma(\gamma) \subseteq Q_0$. Notice that being initial does not enforce any particular constraint on the topology. Thus, assume that the state q represents an error state for a node of the network. If we can solve COVER, then we can decide if there exists a topology of the network and a sufficient number of processes from which we can generate a configuration in which the error is exposed.

The second problem is *target reachability problem* (TARGET) which we define as follows: given a subset of control states F of P, do there exist $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ such that $\gamma \Rightarrow^* \gamma'$ and $\sigma(\gamma') \subseteq F$?
Assume that the subset F represents blocking states for nodes of the network. If we can solve TARGET, then we can decide if there exists a topology of the network and a sufficient number of processes from which we can reach a configuration in which processes can no longer move.

Finally we will also study the *repeated control state reachability problem* (REPEAT-COVER): given a control state q of P, does there exist an infinite execution $\gamma_0 \Rightarrow \gamma_1 \Rightarrow \ldots$ such that the set $\{i \in \mathbb{N} \mid q \in \sigma(\gamma_i)\}$ is infinite?
This problem is a classical extension of the COVER problem that can be used, for instance, to verify whether a protocol is able to react to the occurrence of errors by reaching a state from which errors do not occur any longer. Assume that q represents the error state. If we can solve REPEAT-COVER, then we can decide if there exists a topology of the network and a sufficient number of processes that can generate a computation including infinitely many error states.

3 Static Topology

In this section, we will prove that COVER, TARGET and REPEAT-COVER are all undecidable problems. We first recall that in our decision problems there are no assumptions on the number of nodes and on the communication topology of the initial configurations. Furthermore, the model does not admit dynamic reconfigurations of the topology. Broadcast communication can be used however to ensure that a specific protocol succeeds only if the network topology has a certain form. To be more precise, consider the

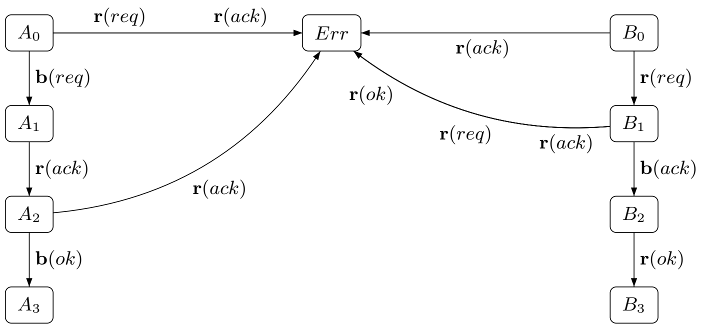

Fig. 3. The RAO (Req/Ack/Ok) protocol.

protocol specified by the process Req/Ack/Ok (RAO) of Figure 3 where A_0 and B_0 are the initial states.

Proposition 1. *Let $\mathcal{G}$ be a denumerable set of group names and γ an initial configuration of the AHN $\langle RAO, \mathcal{G}\rangle$. If γ' is a configuration such that $\gamma \Rightarrow^* \gamma'$ and such that $B_3 \in \sigma(\gamma')$, then the graph $G(\gamma')$ has the following properties:*

– *each node n labelled with B_3 is adjacent to a unique node labelled with A_3 (we will denote $f(n)$ this node)*[1]*;*
– *for each node n labelled with B_3, all the nodes adjacent to n or $f(n)$ are labelled with Err (except of course n and $f(n)$).*

Proof. Assume n is a node of γ' in state B_3. Since n has received a message ok to reach B_3, it is necessarily adjacent to a node in state A_3. No other node adjacent to n can be in state A_3. Indeed, if n receives two req messages before sending an ack, then n moves to state Err. Furthermore, if n sends an ack, then all adjacent nodes that are in states A_0 (ready to send a req) move to state Err. Rule $(A_0, \mathbf{r}(req), Err)$ ensures that, in $G(\gamma')$, no node labeled A_i is adjacent to a node labeled A_3. Rules $(B_0, \mathbf{r}(ack), Err)$ and $(B_1, \mathbf{r}(ack), Err)$ ensure that, when n has label B_3, its single-hop neighbors cannot have label B_i. Rule $(B_1, \mathbf{r}(ok), Err)$ ensures that a node different from n but adjacent to $f(n)$ must have state different from B_i. Indeed, if such a node is in state B_1, then the broadcast ok sent by $f(n)$ sends it to Err, and if such a node moves to B_2 sending ack then it sends node $f(n)$ to Err before it can reach A_3. $\square$

[1] Two nodes are adjacent iff there is an edge between these two nodes

Using an extension of the RAO protocol, we can define an AHN which simulates the execution of a deterministic two-counter Minsky machine and reduce the halting problem to COVER. 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 instuction 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 consists then in deciding whether or not the execution that starts from L_0 with counters equal to 0 reaches L_F.

The intuition behind the reduction is as follows. In a first phase we adapt the RAO protocol to ensure that a given control node is connected to two distinct lists of nodes used to simulate the content of the counters. Each node in the list associated to counter c_i is either in state Z_i or NZ_i The current value of the counter c_i equals the number of NZ_i nodes in the list. The length of each list is guessed non-deterministically during the execution of the first phase (i.e. before starting the simulation) and it corresponds to the maximum value store in a counter for the simulation to succeed. Initially, all nodes must encode zero (state Z_i). Note however that the RAO protocol can only be used to connect pairs of nodes with distinct ending state. For this reason, we assume that adjacent nodes in the list have distinct states (e.g. Z_i is connected to a node with state Z'_i, Z'_i is connected to a node in state Z''_i, Z''_i is connected to a node in state Z_i, and so on). This way each node has only one predecessor and one successor node among all neighbors (all other nodes, if present, are sent to error states).

In the second phase the control node starts the simulation of the instructions. It operates by querying and changing the state of the nodes in the two lists according to the type of instructions to be executed. In this phase all nodes in the same list behave in the same way (i.e., Z'_i, Z''_i and Z_i are all treated as zero units). Requests are propagated back and forth a list by using broadcast sent by a node to its (unique) single-hop successor/predecessor node. The protocols that define the two phases are fairly complicated; the corresponding automata are described in detail in Appendix A. Since it has been shown in [16] that the halting problem for deterministic two-counter Minsky machine is undecidable, we obtain the following result.

Theorem 1. COVER *is an undecidable problem.*

Furthermore, we have the following corollary.

Corollary 1. TARGET *and* REPEAT-COVER *are undecidable problems.*

Proof. Let $P = \langle Q, \Sigma, E, Q_0 \rangle$ be a process, $\mathcal{G}$ a denumerable set of group names and $q \in Q$. The reduction from COVER to REPEAT-COVER is classical and is performed by adding a loop of the form (q, τ, q) to E. To reduce COVER to TARGET, we build the process $P' = \langle Q', \Sigma', E', Q'_0 \rangle$ as follows:

- $Q' = Q \uplus \{r_0, r_1, r_F\}$ (with $Q \cap \{r_0, r_1, r_F\} = \emptyset$);
- $\Sigma' = \Sigma \uplus \{F_1, F_2\}$ (with $\Sigma \cap \{F_1, F_2\} = \emptyset$);
- $E' = E \uplus \{(q, \mathbf{b}(F_1), r_F), (r_0, \mathbf{r}(F_1), r_1), (r_1, \mathbf{b}(F_2), r_F)\} \cup \{(q', \mathbf{r}(F_2), r_F) \mid q' \in Q\}$;
- $Q'_0 = Q_0 \uplus \{r_0\}$.

Let $TS(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow, \mathcal{C}_0 \rangle$ and $TS(P', \mathcal{G}) = \langle \mathcal{C}', \Rightarrow', \mathcal{C}'_0 \rangle$. It is then easy to see that there exist $\gamma_2 \in \mathcal{C}'_0$ and $\gamma'_2 \in \mathcal{C}'$ such that $\gamma_2 \Rightarrow'^* \gamma'_2$ and $\sigma(\gamma'_2) \subseteq \{r_F\}$ if and only if there exists $\gamma_1 \in \mathcal{C}_0$ and $\gamma'_1 \in \mathcal{C}$ such that $\gamma_1 \Rightarrow^* \gamma'_1$ and $q \in \sigma(\gamma'_1)$. In fact, in $TS(P', \mathcal{G})$ after being in the state q a node can broadcast the message F_1 which launches a protocol whose goal is to send all the other nodes in the state r_F. $\square$

4 Mobile topology

In this section we consider a variant on the semantics of AHN obtained by adding spontaneous movement of nodes as in [22]. Node mobility is modeled by non-deterministic updates of their interfaces. Formally, let $\langle P, \mathcal{G} \rangle$ be a AHN with $TS(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow, \mathcal{C}_0 \rangle$. The semantics of $\langle P, \mathcal{G} \rangle$ with mobility is given by the transition system $TS_M(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow_M, \mathcal{C}_0 \rangle$ where the transition $\Rightarrow_M$ is defined as follows.

Definition 3 (Transition Relation with Mobility). *For $\gamma, \gamma' \in \mathcal{C}$ with $\gamma = \langle n_1, \ldots, n_k \rangle$ and $\gamma' = \langle n'_1, \ldots, n'_k \rangle$, we have $\gamma \Rightarrow_M \gamma'$ iff one the following condition holds:*

- *$\gamma \Rightarrow \gamma'$ **(no movement)**;*
- *there exists $i \in [1..k]$ such that, $\sigma(n'_i) = \sigma(n_i)$ (state does not change), $\iota(n'_i) \subseteq \mathcal{G}$ (interface changes in an arbitrary way), and for all $j \in [1..k] \setminus \{i\}$, $n'_j = n_j$ (all other nodes remain unchanged) **(movement)**.*

We prove next that COVER, REPEAT-COVER and TARGET are decidable for AHN with mobility. Intuitively, this follows from the observation that the topology of the network changes in an unpredictable and uncontrollable manner. Hence, every time a node broadcasts a message this is received by a non-deterministically chosen set of nodes, namely those in the communication range of the emitter at the time the message is broadcast. Formally, we reduce COVER, TARGET and REPEAT-COVER respectively to the *marking coverability*, *marking reachability* and *repeated marking coverability* problems for Petri nets, which are known to be decidable (see e.g. the survey [7] and [8] for the repeated coverability).

A *Petri net* (see e.g. [7]) is a tuple $N = (S, T, m_0)$, where S and T are finite sets of *places* and *transitions*, respectively. A finite multiset over the set S of places is called a *marking*, and m_0 is the initial marking. Given a marking m and a place p, we say that the place p contains $m(p)$ *tokens* in the marking m if there are $m(p)$ occurrences of p in the multiset m. A transition is a pair of markings written in the form $m' \mapsto m''$. The marking m of a Petri net can be modified by means of transitions firing: a transition $m' \mapsto m''$ can fire if $m(p) \geq m'(p)$ for every place $p \in S$; upon transition firing the new marking of the net becomes $n = (m \setminus m') \uplus m''$ where $\setminus$ and $\uplus$ are the difference and union operators for multisets, respectively. This is written as $m \to n$. We use $\to^*$ [resp. $\to^+$] to denote the reflexive and transitive closure [resp. the transitive closure] of $\to$. We say that m' is *reachable from* m if $m \to^* m'$. The *coverability* problem for marking m consists of checking whether $m_0 \to^* m'$ with $m'(p) \geq m(p)$ for every place $p \in S$. The *reachability* problem for marking m consists of checking whether $m_0 \to^* m$. Finally, the *repeated coverability problem* for marking m consists of checking wether there exists an infinite execution $m_0 \to^+ m_1 \to^+ m_2 \to^+ \ldots$ such

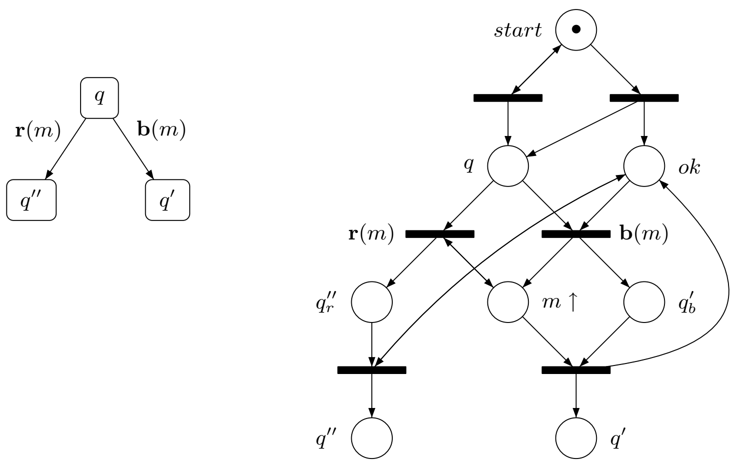

Fig. 4. A Petri net which simulates an AHN with mobility

that for all $i \in \mathbb{N}$, $m_i(p) \geq m(p)$ for every place $p \in S$. The coverability, reachability and repeated coverability problems are decidable for Petri nets [7,8].

We now show how to build a Petri net which simulates the behavior of an AHN with mobility. The Figure 4 gives an example of a Petri net associated to a process. In the Petri net, each control state q has a corresponding place q, and each node $\langle q, I \rangle$ of the network is represented by a token in the place q. The nodes interfaces (thus also the network topology) are abstracted away in the Petri net. In a first phase, before to put a token in the place ok, the Petri net put non-deterministically tokens in the places corresponding to the initial control states of the process. Then it produces a token in the place ok and the simulation begins. The broadcast communication is modeled by a *broadcast protocol* whose effect is to deliver the emitted message to a non-deterministically chosen set of potential receivers. More precisely, the broadcast protocol can be started by a token in a place q such that $(q, \mathbf{b}(m), q')$; after starting the protocol the token is moved to a transient place q'_b and a token is produced in the place $m \uparrow$. During the execution of the protocol, every token in a place r such that $(r, \mathbf{r}(m), r')$ can receive the message moving in a transient place r'_r. The protocol ends when the token in the transient place q'_b moves to the place q'. The tokens in the transient places r'_r can move to the corresponding places r' only when no broadcast protocol is running (when a broadcast protocol is running, there is no token in the place ok). This broadcast protocol does not faithfully reproduces the broadcast as formalized in the AHN model: in fact, in the Petri net there is no guarantee that the tokens in the transient places r'_r move to the corresponding places r' at the end of the execution of the protocol. A token that remains in those transient places (thus losing the possibility to interact with the other tokens in the Petri net) corresponds to a node in the AHN model that disconnects, due to mobility, from the other nodes in the system. Testing whether there is an execution in the AHN with mobility which ends in a configuration where one of the nodes is in the control state q can be done by testing whether the marking $\{q, ok\}$ can be covered in the associated Petri net. Hence:

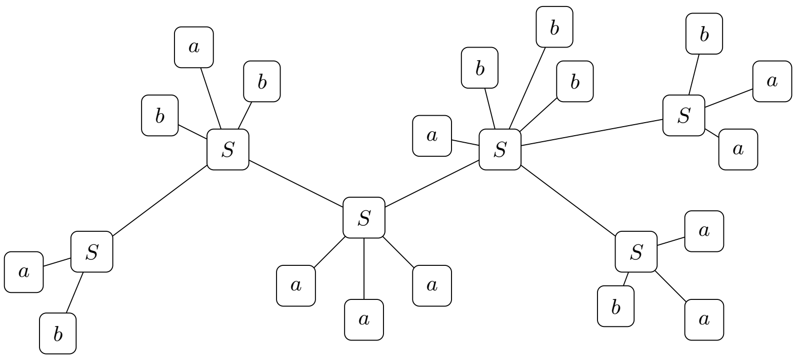

Fig. 5. Arborescence.

Theorem 2. *There exists a reduction from the* COVER *problem for AHN with mobility to the marking coverability problem for Petri nets.*

Using the same construction, we also obtain:

Theorem 3. *There exists a reduction from the* REPEAT-COVER *problem for AHN with mobility to the marking repeated coverability problem for Petri nets.*

In order to reduce TARGET to marking reachability we need to extend the Petri net associated to an AHN, adding a final stage in the computation, dedicated to the elimination of tokens from the places corresponding to the final states in F. Intuitively we add a transition of the form $\{ok\} \mapsto \{end\}$ and for each $q \in F$ we add a transition $\{end, q\} \mapsto \{end\}$ and we then test if the marking where all the places are empty execpt the place end is reachable.

Theorem 4. *There exists a reduction from the* TARGET *problem for AHN with mobility to the marking reachability problem for Petri nets.*

From these three last theorems and from the fact that the marking coverability, marking repeated coverability and marking reachability problems are decidable for Petri nets, we finally deduce:

Corollary 2. COVER, REPEAT-COVER *and* TARGET *are decidable for AHN with mobility.*

5 AHN restricted to Bounded Path Configurations

Let us go back to the AHN model with static topology. The possibility for a message to pass through an unbounded number of new nodes is a key feature in the proof of Theorem 1 (undecidability of COVER for static topology). For this reason, it seems natural to study COVER, TARGET and REPEAT-COVER for a restricted class of configurations in

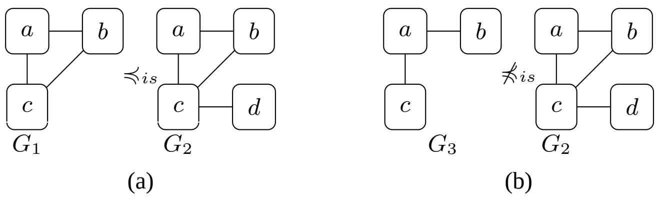

Fig. 6. Examples of the induce subgraph relation.

which, for a fixed K, a message can pass through at most K-different nodes. Formally, given an AHN $\langle P, \mathcal{G}\rangle$ with $P = \langle Q, \Sigma, E, Q_0\rangle$ and $TS(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow, \mathcal{C}_0\rangle$ our class of restricted configurations is defined as follows:

Definition 4 (K-bounded Path Configuration). *Given an integer $K \geq 1$, a configuration γ is a K-bounded path configuration if the longest simple path in the associated graph $G(\gamma)$ has length at most K.*

We denote by $\mathcal{C}^K$ the set of K-bounded path configurations. The semantics of the AHN $\langle P, \mathcal{G}\rangle$ resticted to K-bounded path configurations is given by the transition system $TS_K(P, \mathcal{G}) = \langle \mathcal{C}^K, \Rightarrow_K, \mathcal{C}_0^K\rangle$ where the transition relation $\Rightarrow_K$ is the restriction of $\Rightarrow$ to $\mathcal{C}^K \times \mathcal{C}^K$ and $\mathcal{C}_0^K = \mathcal{C}_0 \cap \mathcal{C}^K$. For fixed K, the class of K-bounded path configurations contains an infinite set of graphs. In Figure 5 we show an example in which nodes of type S (server) are connected via bounded paths and nodes of type a and b (clients) are locally connected to each server (using private group names). Here we could add any number of client nodes (connected with private group names to a single server) without breaking the bounded path property.

Decidability of COVER**.** In order to study COVER restricted to bounded path configurations, we first introduce some definitions and prove auxiliary properties. First of all we give the definition of the *induced subgraph* relation.

Definition 5 (Induced Subgraph Relation). *For configurations γ_1 and γ_2, we define $\gamma_1 \preccurlyeq_{is} \gamma_2$ if there exists a label preserving injection h from nodes of $G_1 = G(\gamma_1)$ to nodes of $G_2 = G(\gamma_2)$ such that (n, n') is an edge in G_1* **if and only if** *$(h(n), h(n'))$ is an edge in G_2, i.e., G_1 is* isomorphic *to an induced subgraph of G_2.*

Notice that the induced subgraph relation is stronger than the usual subgraph relation that requires only an homomorphic embedding of G_1 into G_2. In Fig. 6 (a) G_1 is isomorphic to an induced subgraph of G_2, thus $G_1 \preccurlyeq_{is} G_2$. In (b) G_3 is obtained from G_1 by removing the edge from node a to node c. The induced graph of G_2 with nodes a, b, c is no more isomorphic to G_3, hence $G_3 \not\preccurlyeq_{is} G_2$ Notice, however, that G_3 is still a subgraph of G_2.

The following lemma then holds.

Lemma 1. *Given $K \geq 1$, $(\mathcal{C}^K, \preccurlyeq_{is})$ is a well-quasi ordering (shortly wqo), i.e., for every infinite sequence of K-bounded path configurations $\gamma_1\gamma_2\ldots$ there exist $i < j$ s.t. $\gamma_i \preccurlyeq_{is} \gamma_j$.*

Proof. We can apply here Ding's Theorem (Theorem 2.2 in [4]). Let $\mathcal{P}_n$ be the class of graphs (with wqo labels) that do not contain P_n subgraphs, where P_n represents simple paths with n nodes (over the same labels). Ding's Theorem states that, for any natural number $n \geq 0$, the class $\mathcal{P}_n$ equipped with the induced subgraph relation is a wqo. Since $\mathcal{P}_{K+1}$ corresponds to the class of graphs with longest simple path of length at most K, by taking as set of labels control states equipped with $=$, we obtain the wqo of $(\mathcal{C}^K, \preccurlyeq_{is})$. □

Given a subset $S \subseteq \mathcal{C}^K$ we define $S\uparrow = \{\gamma' \in \mathcal{C}^K \mid \gamma \in S \text{ and } \gamma \preccurlyeq_{is} \gamma'\}$, i.e., $S\uparrow$ is the set of configurations generated by those in S via $\preccurlyeq_{is}$. A set $S \subseteq \mathcal{C}^K$ is an *upward closed set* w.r.t. to $(\mathcal{C}^K, \preccurlyeq_{is})$ if $S\uparrow = S$. Since $(\mathcal{C}^K, \preccurlyeq_{is})$ is a wqo, we obtain that every set of configurations that is upward closed w.r.t. $(\mathcal{C}^K, \preccurlyeq_{is})$ has a finite basis, i.e., it can be finitely represented by a finite number of K-bounded path configurations. We can exploit this property to define a decision procedure for COVER. For this purpose, we apply the methodology proposed in [1]. The first property we need to prove is that the transition relation induced by our models is compatible with $\preccurlyeq_{is}$.

Lemma 2 (Monotonicity). *For every $\gamma_1, \gamma_2, \gamma_1' \in \mathcal{C}^K$ such that $\gamma_1 \Rightarrow_K \gamma_2$ and $\gamma_1 \preccurlyeq_{is} \gamma_1'$, there exists $\gamma_2' \in \mathcal{C}^K$ such that $\gamma_1' \Rightarrow_K \gamma_2'$ and $\gamma_2 \preccurlyeq_{is} \gamma_2'$.*

Proof. The interesting case is the application of a broadcast rule $\mathbf{b}(a)$. Assume that the rule is applied to a node n adjacent in $G(\gamma_1)$ to nodes $N = \{n_1, \ldots, n_k\}$. Assume that the subset N' of N contains nodes that are enabled by message a. By applying the operational semantics, the state of n and the states of nodes in N' are updated simultaneously Assume now that $G(\gamma_1)$ is isomorphic to an induced subgraph of $G(\gamma_1')$ via the injection h. Then, $h(n)$ is adjacent to the set of nodes $h(N)$ (there cannot be more connections since $h(G(\gamma_1))$ is an induced subgraph of $G(\gamma_1')$). Thus, the same rule is enabled in $h(n)$ and in $h(N')$ and yields the same effect on the labels. Thus, we obtain γ_2' such that $G(\gamma_2) \preccurlyeq_{is} G(\gamma_2')$. □

Monotonicity ensures that if S is an upward closed set of configurations (w.r.t. $(\mathcal{C}^K, \preccurlyeq_{is})$), then the set of predecessors of S accroding to $\Rightarrow_K$, defined as $pre_K(S) = \{\gamma \mid \gamma \Rightarrow_K \gamma' \text{ and } \gamma' \in S\}$, is still upward closed. We now show that we can effectively compute a finite representation of $S \cup pre_K(S)$.

Lemma 3. *Given a finite basis B of an upward closed set $S \subseteq \mathcal{C}^K$, there exists an algorithm to compute a finite basis B' of $S \cup pre_K(S)$ such that $S \cup pre_K(S) = B'\uparrow$.*

In Appendix E, we give an example of the symbolic computation of $S \cup pre_K(S)$. We can now state the main theorem of this section.

Theorem 5. *For $K \geq 1$, COVER is decidable for AHN restricted to K-bounded path configurations.*

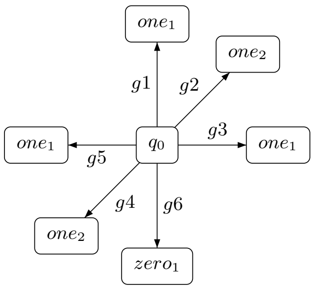

Fig. 7. Configuration $\langle q_0, c_1 = 3, c_2 = 2\rangle$ for $c_1 \in [0, 4]$ and $c_2 \in [0, 2]$.

Proof. From Lemmas 1, 2, and 3 it follows that the transition system induced by any AHN is well structured with respect to $(\mathcal{C}^K, \preccurlyeq_{is})$. The theorem then follows from the general properties of well-structured transition systems described in [1,2,10]. The decision procedure is based on a symbolic backward exploration in which we use finite sets of configurations in $\mathcal{C}^K$ to symbolically represent upward closed sets of configurations (those generated by the bases). Thus, to solve COVER for a control state q, we compute $Pre^*(\{q\} \uparrow)$. The wqo of $(\mathcal{C}^K, \preccurlyeq_{is})$ ensures that the symbolic computation of predecessors terminates after finitely many step whenever we use $\preccurlyeq_{is}$ to discard graphs that do not add new information. We can then test if the resulting fixpoint contains initial configurations by looking for graphs in which all nodes have labels in Q_0. $\square$

Undecidability of TARGET **and** REPEAT-COVER**.** In order to show that TARGET is undecidable for K-bounded path configurations, we show how to model a Minsky machine in such a way that the machine terminates if and only if the corresponding AHN has a computation (restricted to K-bounded path configurations) that reaches a configuration in which all nodes are in some specific final state. For this purpose, we design a protocol that succeeds only on star topologies in which the center node represents the current control state and the satellite nodes the units of the two counters. Such units are initially in the $zero_i$ state (with $i \in \{1, 2\}$). The number of satellite nodes needed to guess the maximal values reached by the counters during the computation is non-deterministically chosen in a preliminary part of the simulation . Only runs that initially guess a sufficient number of satellite nodes can successfully terminate the simulation. A satellite node moves from the $zero_i$ to the one_i state when the i-th counter is incremented, and a single node moves from the one_i back to the $zero_i$ state when the counter is decremented. For instance, the star in Figure 7 represents a configuration with control state q_0 and counters $c_1 = 3$ (with maximal value equals to 4), and $c_2 = 2$ (with maximal value equals to 2).

Test for zero actions are more difficult to be simulated as it is not possible to check the absence of neighbours in the state one_i. Nevertheless, it is possible to ensure that no node is in the state one_i after a test for zero is executed. It is sufficient to use broadcast communication to move all the satellite nodes in the one_i state to a special $sink$ state. If the simulation terminates exposing the final control state and no node is in the $sink$ state (i.e. a configuration is reached in which all the nodes are in the final control state, in the $zero_i$, or the one_i state), we can conclude that the simulated computation is correct, thus also the corresponding Minsky machine terminates.

Note that the number of satellite nodes is not fixed a priori. However the graph have bounded path (the construction works for paths of length 3), so we can conclude what follows:

Theorem 6. TARGET *is undecidable for AHN restricted to K-bounded path configurations (with $K \geq 3$).*

As a corollary we now prove the undecidability of REPEAT-COVER. We need to slightly modify the way we model Minsky machines. The idea is to repeatedly simulate the computation of the Minsky machine, in such a way that the final control state can be exposed infinitely often if and only if the simulated Minsky machine terminates.

Every simulation phase simulates only a finite number of steps of the Minsky machine, and if the final control state is reached then a new simulation phase is started. This is achieved by including in the initial star topology also satellite nodes in the $free$ state, and ensuring that every simulated action moves one of those nodes to the $done$ state. In this way, a simulation cannot perform more steps than the number of $free$ nodes in the initial star topology. If the final control state is reached, a new simulation is started by moving all the nodes from the $done$ to the $free$ state, all the nodes from the one_i to the $zero_i$ state, and by restarting from the initial control state. Notice that nodes reaching the $sink$ state (due to a wrong execution of a test for zero action) are no longer used in the computation. For this reason, as every time a wrong test for zero is executed some node moves in the $sink$ state, we are sure that only finitely many wrong actions can occur. Hence, if the final control state is exposed infinitely often, we have that only finitely many simulation phases could be wrong, while infinitely many are correct. As all simulation phases reaches the final control state (necessary to start the subsequent phase), we have that the corresponding Minsky machine terminates. Hence, we have the following Corollary of Theorem 6:

Corollary 3. REPEAT-COVER *is undecidable for AHN restricted to K-bounded path configurations (with $K \geq 3$).*

6 Conclusions

In this paper we have studied different types of verification problems for a formal model of Ad Hoc Networks in which communication is achieved via a selective type of broadcast. Perhaps surprisingly, a model with static topology turns out to be more difficult to analyze with respect to a model with spontaneous node movement. A similar dichotomy appears in verification of perfect and lossy channel systems. Studying the expressiveness of other variations on the semantics to model for instance conflicts, noise and lossiness, is an interesting research direction for future works.

References

1. Abdulla, P. A., Čerāns, C., Jonsson, B., and Tsay. Y.-K. General decidability theorems for infinite-state systems. LICS ’96: 313–321

2. Abdulla, P. A., Čerāns, C., Jonsson, B., and Tsay. Y.-K. Algorithmic Analysis of Programs with Well Quasi-ordered Domains. Inf. Comput. 160(1-2): 109-127 (2000)
3. Abdulla, P. A., and B. Jonsson, B., Verifying programs with unreliable channels. Inf. Comput. 127(2): 91–101 (1996)
4. Ding, G.: Subgraphs and well quasi ordering. J. of Graph Theory 16(5): 489 – 502 (1992)
5. Emerson, E. A., Namjoshi, K. S.: On Model Checking for Non-Deterministic Infinite-State Systems. LICS 1998: 70-80
6. Ene, C., Muntean, T.: A Broadcast based Calculus for Communicating Systems. IPDPS '01: 149
7. Esparza, J., Nielsen, M.: Decidability Issues for Petri Nets-a Survey. Bulletin of the EATCS **52**: 245–262 (1994)
8. Esparza J.: Some applications of Petri Nets to the Analysis of Parameterised Systems. Talk at WISP'03.
9. Fehnker, A., van Hoesel, L., Mader, A.: Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks. IFM '07: 253–272
10. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! TCS **256**(1-2): 63–92 (2001)
11. Godskesen, J.C.: A Calculus for Mobile Ad Hoc Networks. COORDINATION '07: 132Đ150
12. Merro, M.: An Observational Theory for Mobile Ad Hoc Network. Inf. Comput. 207(2): 194–208 (2009)
13. Meyer, R.: On Boundedness in Depth in the pi-Calculus. IFIP TCS '08: 477–489
14. Mezzetti, N., Sangiorgi, D.: Towards a Calculus For Wireless Systems. ENTCS 158: 331-353 (2006)
15. Milner R.: Communicating and Mobile Systems: the Pi-Calculus. Cambridge Univ. Press (1999)
16. Minsky, M.: Computation: finite and infinite machines. Prentice Hall (1967)
17. Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks. TCS, 367(1–2):203Đ227 (2006)
18. Prasad, K.V.S.: A Calculus of Broadcasting Systems. Sci. of Comp. Prog., 25(2–3): 285–327 (1995).
19. Saksena, M., Wibling, O., Jonsson, B.: Graph Grammar Modeling and Verification of Ad Hoc Routing Protocols. TACAS '08: 18–32
20. Singh, A., Ramakrishnan, C. R., Smolka, S. A.: Modeling AODV in omega-calculus. LISAT '06
21. Singh, A., Ramakrishnan, C. R., Smolka, S. A.: A Process Calculus for Mobile Ad Hoc Networks. COORDINATION '08
22. Singh, A., Ramakrishnan, C. R., Smolka, S. A.: Query-Based Model Checking of Ad Hoc Network Protocols. CONCUR '09: 603–61
23. Wies T., Zufferey D., Henzinger T. A.: Forward Analysis of Depth-Bounded Processes. FOSSACS '10: 94–108

A Proof of Theorem 1

We show how to reduce the halting problem for deterministic Minsky machine to the COVER problem. To perform this reduction we will use many times the same principle as for the Req/Ack/Ok-protocol. We build a process $P = \langle Q, E, \Sigma, Q_0 \rangle$ given by the Figures 8, 9, 10, 11, 12, 13 and 14. The initial states are $Q_0 = \{R, C_1, C_1', C_1'', C_2, C_2', C_2''\}$. For each label L of the Minsky machine there is a control state L in Q and the instructions of the Minsky machine are encoded by the Figures 13 and 14. In the sequel we call process S with $S \in \{R, C_1, C_1', C_1'', C_2, C_2', C_2''\}$ the process whose initial state corresponds to the given name.

We consider an infinite denumerable set of names $\mathcal{G}$ and we will show that the Minsky machine eventually reaches the label L_F if and only if there is an initial configuration γ associated to P and $\mathcal{G}$ and a configuration γ' such that $\gamma \Longrightarrow^* \gamma'$ and $L_F \in \sigma(\gamma')$.

We first assume that there is an initial configuration γ associated to P and $\mathcal{G}$ and a configuration γ' such that $\gamma \Longrightarrow^* \gamma'$ and $L_F \in \sigma(\gamma')$. Intuitively the AHN associated to P and $\mathcal{G}$ works in two phases, in the first phase it ensures that the topology has a certain form (using the same method as the RAO protocol) and in the second phase it simulates the behavior of the Minsky machine. Note that the RAO protocol works to isolate two nodes in the graph, since we want here to build lists of nodes, we need at least three types of process (whereas the RAO protocol is using only two kinds of process, the one labelled by A_i and the one labelled by B_i in the Figure 3).

Let us now explain the first phase more in details. Let γ'' be a configuration such that $L_0 \in \sigma(\gamma'')$ (note that before reaching a configuration containing L_F the system has to be in a configuration like γ''). Using a reasoning similar to the one developed in the proof of Proposition 1, we can deduce that the graph $G(\gamma'')$ enjoys the following properties. If n is a node of $G(\gamma'')$ labelled with L_0 then there are two sequences of nodes $n_1, n_2, \ldots, n_k$ and $n_1', n_2', \ldots, n_l'$ such that:

- n_1 and n_1' are connected to n;
- for $i \in \{1, \ldots, k-1\}$, n_i is adjacent to n_{i+1};
- for $i \in \{1, \ldots, l-1\}$, n_i' is adjeacent to n_{i+1}';
- for $i \in \{1, \ldots, k\}$, if $(i \mod 3 = 1)$ then n_i is labelled with Z_1', if $(i \mod 3 = 2)$ then n_i is labelled with Z_1'' and if $(i \mod 3 = 0)$ then n_i is labelled with Z_1;
- for $i \in \{1, \ldots, l\}$, if $(i \mod 3 = 1)$ then n_i' is labelled with Z_2', if $(i \mod 3 = 2)$ then n_i' is labelled with Z_2'' and if $(i \mod 3 = 0)$ then n_i' is labelled with Z_2.

In other words in the graph we have two such sequences of labels $L_0 - Z_1' - Z_1'' - Z_1 - Z_1' - Z_1'' - Z_1 - \ldots$ and $L_0 - Z_2' - Z_2'' - Z_2 - Z_2' - Z_2'' - Z_2 - \ldots$. Furthermore, we have that :

- all the other nodes which are adjacent to n are in state Err (and so they will not play any role in the further communication);
- for $i \in \{1, \ldots, k\}$ all the other nodes adjacent to n_i are either in state Err or in a state belonging to process C_2 or C_2' or C_2'' (which is not a problem since these processes cannot communicate directly with each other);

- for $i \in \{1, \dots, l\}$ all the other nodes adjacent to n'_i are either in state Err or in a state belonging to process C_1 or C'_1 or $C''1$ (which is not a problem since these processes cannot communicate directly with each other).

We have the following property because each node first answers to the protocol RAO to its predecessor and then either it launches a new protocol RAO with a successor node or it sends a $done_i$ which is transmitted to the node n sending it in state L_0.

Once the form of the topology is ensured, the simulation of the Minsky machine begins. In this second phase we forget about the distinction between Z_i, Z'_i and Z''_i [resp. NZ_i, NZ'_i and NZ''_i] which was useful only to explain how to obtain two lists of nodes. The simulation of the Minsky machine is performed as follows:

- For the incrementation of the counter c_1, the node n sends an inc_1 and waits for an $ackinc_1$ (see Figure 13); this inc_1 is transmitted to the first node n_j which is in state Z_1 and which then goes in state NZ_1 sending the acknowledgement (see Figure 12). So the number of nodes n_1 in state NZ_1 characterizes the value of the counter at each step of the simulation. Note that if there is no more node in state Z_1 then the node n is in a pending state;
- For the decrementation of the counter c_1 the node n sends a dec_1 and then there are two cases (see Figure 14):
  1. if the node n_1 is in state Z_1, it sends back a $zero_1$ to the node n,
  2. if the node n_1 is not in state Z_1, but in state NZ_1, the dec_1 is transmitted to first node n_j in state NZ_1, this node then sends a $zero_1$ and the node n_{j-1} receives this message and goes from NZ_1 to state Z_1 sending a $ackdec_1$ which is transmitted back to node n by its predecessors.

The incrementation and decrementation of the second counter are performed exactly the same way. It is then clear that if the AHN reaches the configuration γ' such that $L_F \in \sigma(\gamma')$ we can build a run of the Minsky machine which reaches the label L_F.

We now assume that the Minsky machine reaches the state L_F during its unique execution. We denote by k [resp. by l] the highest value taken by the first [resp. the second] counter during this execution. Then it is enough to consider the following initial configuration $n'_{l+1} - n'_l - \dots - n'_1 - n - n_1 - \dots - n_k - n_{k+1}$ where:

- n is labelled by R;
- for $i \in 1, \dots, l+1$, if $(i \mod 3 = 1)$ then n'_i is labelled by C'_2, if $(i \mod 3 = 2)$ then n'_i is labelled by C''_2 and if $(i \mod 3 = 0)$ then n'_i is labelled by C_2;
- for $i \in 1, \dots, k+1$, if $(i \mod 3 = 1)$ then n_i is labelled by C'_1, if $(i \mod 3 = 2)$ then n_i is labelled by C''_1 and if $(i \mod 3 = 0)$ then n_i is labelled by C_1.

The execution of the AHN from this initial configuration will necessarily reaches a configuration γ' such that $L_F \in \sigma(\gamma')$, the details of the execution being the same as to prove the previous implication. □

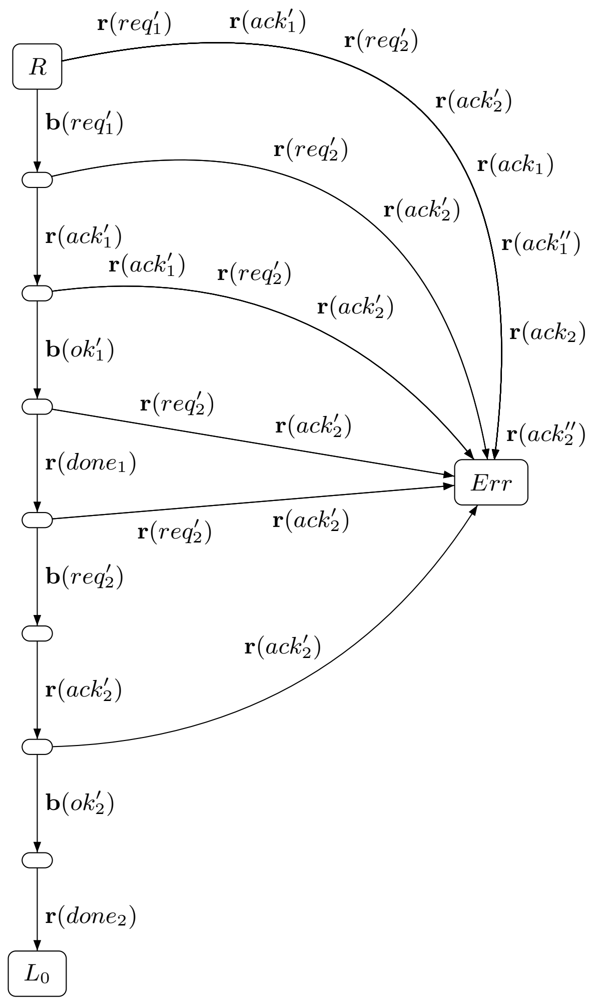

Fig. 8. Ensuring a topology to prove undecidability (I)

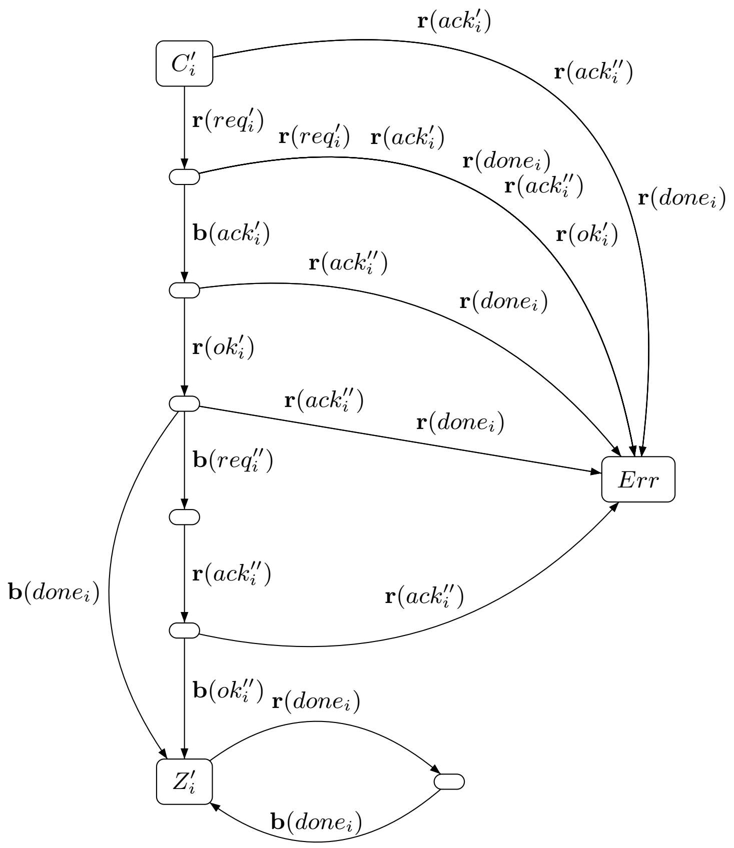

Fig. 9. Ensuring a topology to prove undecidability with $i \in \{1, 2\}$ (II)

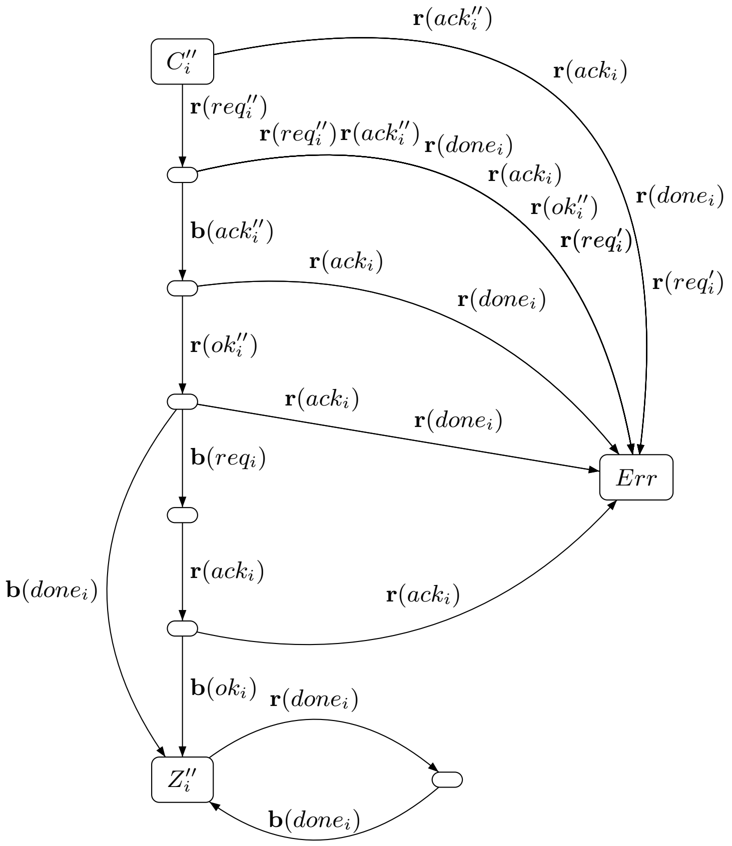

Fig. 10. Ensuring a topology to prove undecidability with $i \in \{1, 2\}$ (III)

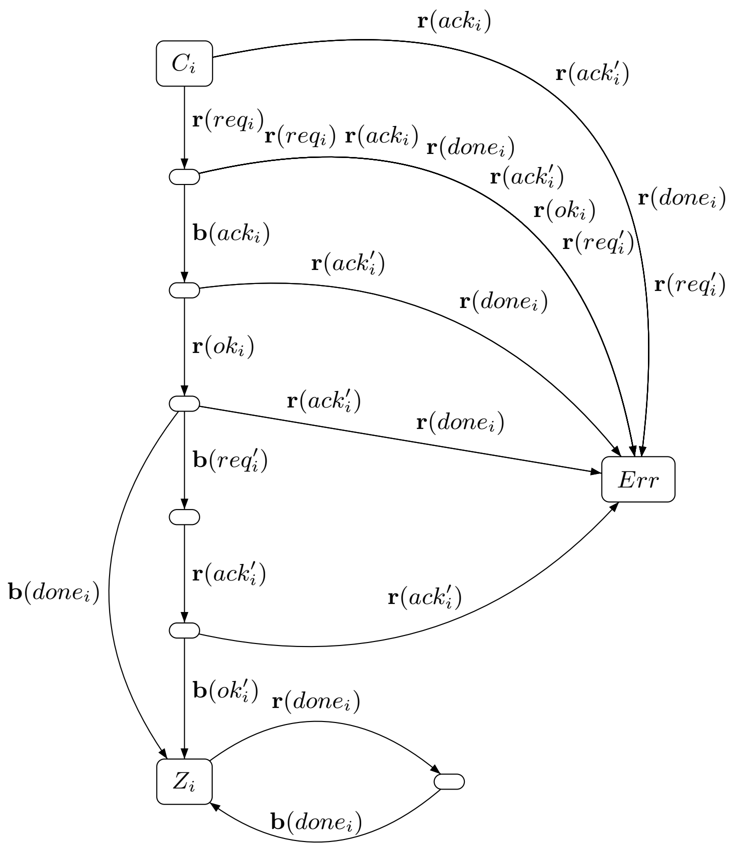

Fig. 11. Ensuring a topology to prove undecidability with $i \in \{1, 2\}$ (IV)

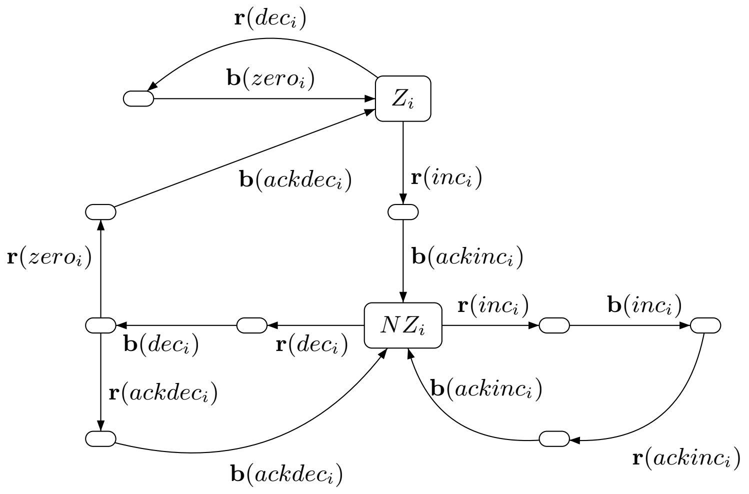

Fig. 12. Encoding the counter behavior for $i \in \{1, 2\}$ (Z_i and NZ_i can be replaced by Z'_i and NZ'_i or by Z''_i and NZ''_i)

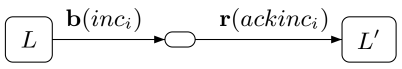

Fig. 13. Encoding $L : c_i := c_i + 1;$ `goto` L'

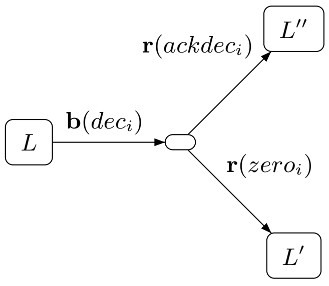

Fig. 14. Encoding $L :$ `if` $c_i = 0$ `then goto` L' `else` $c_i := c_i - 1;$ `goto` L''

B Proof of Theorem 2

Let $\langle P, \mathcal{G}\rangle$ be a AHN with $P = \langle Q, \Sigma, E, Q_0\rangle$ and $TS_M(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow_M, \mathcal{C}_0\rangle$ its associated transition system with mobility. We associate to P the following Petri net $[\![P]\!] = (S, T, m_0)$ where the set of places S is defined as follows

$$S \ = \ Q \cup \{q_b, q_r \mid q \in Q\} \cup \{a\!\uparrow \mid a \in \Sigma\} \cup \{start, ok\}$$

the set of transitions T is the minimal set containing the following classes of transitions

Prepare: if $q_0 \in Q_0$ then $\{start\} \mapsto \{start, q_0\} \in T$;
Start: if $q_0 \in Q_0$ $\{start\} \mapsto \{ok, q_0\} \in T$;
Start protocol: if $(q, \mathbf{b}(a), q') \in E$ then $\{q, ok\} \mapsto \{q'_b, a\!\uparrow\} \in T$;
Receive: if $(q, \mathbf{r}(a), q') \in E$ then $\{q, a\!\uparrow\} \mapsto \{q'_r, a\!\uparrow\} \in T$;
End protocol: if $q \in Q$ and $a \in \Sigma$ then $\{q_b, a\!\uparrow\} \mapsto \{q, ok\} \in T$;
Complete receive: if $q \in Q$ and $a \in \Sigma$ then $\{q_r, ok\} \mapsto \{q, ok\} \in T$.

and the initial marking $m_0 = \{start\} \in T$.

This Petri net includes the *Prepare* and *Start* transitions that are used to reach the markings corresponding to every possible (parameterized) initial configuration of the AHN. The other transitions are used to implement the broadcast protocol.

We now prove that the behavior of $[\![P]\!]$ models the behavior of $\langle P, \mathcal{G}\rangle$ with mobility. This proof is divided in two distinct results: a completeness result stating that all execution in $TS_M(P, \mathcal{G})$ are mimicked by computations of $[\![P]\!]$ and a soundness result showing that for every marking not including tokens in transient places (places of the form q_b or q_r) there is a corresponding configuration reachable in the AHN.

We use the following notation: given a configuration $\gamma = \langle n_1, \ldots, n_k\rangle$ in $\mathcal{C}$, the marking corresponding to γ is denoted with $dec(\gamma) = \biguplus_{i \in \{1, \cdots, k\}} \sigma(n_i) \uplus \{ok\}$.

Lemma 4 (Completeness). *Let $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$. If $\gamma \Rightarrow_M^* \gamma'$ then the marking $dec(\gamma')$ is reachable in $[\![P]\!]$.*

Proof. By induction on the length of the computation $\gamma \Rightarrow_M^* \gamma'$. □

Lemma 5 (Soundness). *Let m be a marking reachable in the Petri net $[\![P]\!]$ such that $m(ok) = 1$ and $m(p) = 0$ for every place $p \in \{q_b, q_r \mid q \in Q\}$. There exists $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}'$ such that $dec(\gamma') = m$ and $\gamma \Rightarrow_M^* \gamma'$.*

Proof. By induction on the number of broadcast protocols executed during the computation $m_0 \rightarrow^* m$ of the Petri net $[\![P]\!]$. In the inductive case we note that the computation $m_0 \rightarrow^* m$ can be divided in three parts $m_0 \rightarrow^* m' \rightarrow^* m'' \rightarrow^* m$ where $m' \rightarrow^* m''$ is the last execution of the broadcast protocol. The transitions in $m'' \rightarrow^* m$ are all *Complete receive* transitions and can be divided in two groups: (i) those that apply to tokens introduced in transient places by the last broadcast protocol, and (ii) those that apply to the tokens in transient places when the last broadcast protocol is started. It is easy to see that an alternative computation $m_0 \rightarrow^* m$ can be obtained simply by anticipating the transitions of type (ii) immediately before the execution of the last broadcast protocol. In this way, the last broadcast protocol starts from a marking in which all transient places are empty, thus it is possible to apply the inductive hypothesis to prove the thesis. □

As a corollary of these two Lemmas we have that COVER in an AHN with mobility can be reduced to marking coverability in the corresponding Petri net, thus COVER turns out to be decidable.

Lemma 6. *Let $q \in Q$. We have that there exists an $\gamma \in \mathcal{C}_0$ and $\gamma' \in \mathcal{C}$ such that $\gamma \Rightarrow_M \gamma'$ and $q \in \sigma(\gamma')$ if and only if the marking $\{q, ok\}$ is coverable in the Petri net $[P]$.*

Proof. The *only-if* part directly follows from Theorem 4. In the *if* part we observe that if in $[P]$ there exists a computation $m_0 \to^* \{q, ok\} \uplus m$, then the computation can be extended with *Complete receive* transitions in order to remove all the tokens present in transient places in the marking m. Thus the thesis directly follows from Lemma 5. □

C Proof of Theorem 4

Let $\langle P, \mathcal{G}\rangle$ be a AHN with $P = \langle Q, \Sigma, E, Q_0\rangle$ and $TS_M(P, \mathcal{G}) = \langle \mathcal{C}, \Rightarrow_M, \mathcal{C}_0\rangle$ its associated transition system with mobility. We consider a set $F \subseteq Q$. We associate to P the following Petri net $[\![P, F]\!] = (S, T, m_0)$ where the set of places S includes the places of the Petri nets built in the proof of Theorem 2 plus the place end, the initial marking m_0 is defined as in the proof of Theorem 2, and the set of transitions T includes the transitions defined in proof of Theorem 2 plus the following

End $\{ok\} \mapsto \{end\} \in T$;
Remove Final: if $q \in F$ then $\{end, q\} \mapsto \{end\} \in T$.

We have then:

Lemma 7. *There exists $\gamma \in \mathcal{C}_0$ and $\gamma \in \mathcal{C}$ such that $\gamma \Rightarrow_M^* \gamma'$ and $\sigma(\gamma') \subseteq F$ if and only if the marking $\{end\}$ is reachable in the Petri net $[\![P, F]\!]$.*

Proof. The *only-if* part follows from Lemma 4; it is sufficient to observe that given a configuration γ' such that $\sigma(\gamma') \subseteq F$, then the marking $\{end\}$ is reachable in $[\![P, F]\!]$ from the marking $dec(\gamma')$. The *if* part follows from Theorem 5; it is sufficient to observe that every computation $m_0 \to^* \{end\}$ is of the form $m_0 \to^* \{ok\} \uplus m' \to \{end\} \uplus m' \to^* \{end\}$ where m' has tokens only in places corresponding to states in F. □

D Proof of Lemma 3

The interesting case is the backward application of a broadcast rule $(q, \mathbf{b}(a), q')$ to a configuration in the upward closure of B. The computation of the set $B\uparrow \cup\, pre_K(B\uparrow)$ where $pre_K(B\uparrow) = \{\gamma \mid \gamma \Rightarrow_K \gamma' \text{ and } \gamma' \in B\uparrow\}$ is done according to the following steps :

- $B' := B$;
- For each $\gamma \in B$:
  1. For each vertex n labelled with q' in the graph $G(\gamma)$, let N be the set of nodes adjacent to q'

If There exists a node in N with state r such that $(r, \mathbf{r}(a), r')$ is a rule in the model

Then We add no predecessor to B' (because every node $n' \in S$ in state r **must** react to the broadcast);

Else For any subset $N' = \{n_1, \ldots, n_k\}$ of nodes in N such that n_i has state r'_i and $(r_i, \mathbf{r}(a), r'_i)$ is a rule in the model, we build a predecessor configuration γ' in which the label of n is updated to q and the label of n_i is updated to r_i for $i \in \{1, \ldots, k\}$ and if there is no γ'' in B' such that $\gamma'' \preccurlyeq_{is} \gamma'$, we add γ' to B' (Note that we have to select all possible subset N' of N because we must consider the cases in which nodes connected to n are already in the target state of some reception rule).

2. Let Γ' be the set of configurations γ' in $\mathcal{C}^K$ obtained by adding a node n in state q' to γ such that in $G(\gamma')$, n is adjacent to at least one node (i.e. in Γ' we have all the configurations obtained by added a connected node to γ and which are still K-bounded path configurations). We then apply the precedent rule $1.$ to each configuration in Γ' considering the added node n labelled with q'.

□

E An Example of Pre-computation

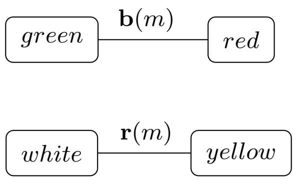

Fig. 15. Example of process.

We consider the process P represented on the Figure 15. The Figure 16 shows example of the computation of $pre_6(\{\gamma\}\uparrow)$, in other words for this example we restrict the configurations to 6-bounded path configurations. We build four configurations such that $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\} \uparrow\subseteq pre_6(\{\gamma\}\uparrow)$. Configuration γ_1 is obtained by matching the right-hand side of the broadcast rule with the topleftmost node and the righthand side of the reception rule with all of its yellow neighbors, while γ_2 is obtained by matching the righthand side of the reception rule with a subset of them. Configuration γ_3 is obtained by assuming that the broadcast node is sent by a node in the upward closure, connected to two yellow nodes. In the example the effect is that of adding a new green node and changing one of two yellow neighbors (we must consider other cases like changing all yellow neighbors etc.). Similarly, configuration γ_4 is obtained by assuming that the broadcast node is sent by a node in the upward closure, connected to a red node and two yellow nodes. The effect is that of adding a new green node, turning the yellow nodes

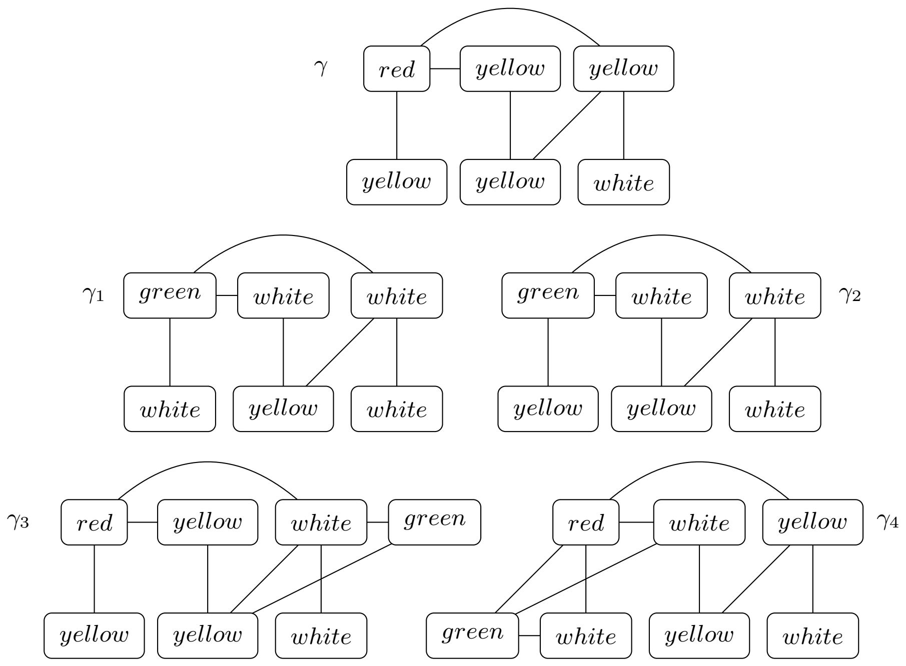

Fig. 16. Example of symbolic *pre* computation.

into white, and leaving the red node unchanged. It is important to notice that in the last two cases the new graphs have both a new node and additional edges.

F Proof of Theorem 6

Let L_0 be the initial location of the Minsky machine. To simulate a run of the Minsky machine in which the upper bound on the values of the two counters are B_1 and B_2, in a preliminary phase we non-deterministically select in some part of the network a control node and a sufficient number of unit nodes (at least $B_1 + B_2$) initially all in state $null_1$ or $null_2$. The protocol is defined in Figure 17. The broadcast message req sent by a node in state $start$ forces all other nodes $start$ in its vicinity to enter an error state. The sending node moves to an auxiliary state in which non-deterministically either accepts acknowledgment from unit nodes or stops the initialization phase. Null nodes in the vicinity send acknowledgements and move to an auxiliary state. Reception of acknowledgments from another unit node results in an error state. The control node sends an ok message to send unit nodes which did not acknowledge the req to a special error state and the others to the state $zero_1$ or $zero_2$. This protocol ensures that every node in $zero_1$ or $zero_2$ state has no other $zero_1$ or $zero_2$ neighbors (i.e. every edge connecting a satellite node must be labeled with a different group name).

In the second phase we run the simulation of the counter machine. The rule $L : c_i := c_i + 1;$ `goto` L' is simulated by the automaton in Figure 18. The control node broadcasts inc_i message forcing all $zero_i$ nodes to move to an auxiliary state. The first

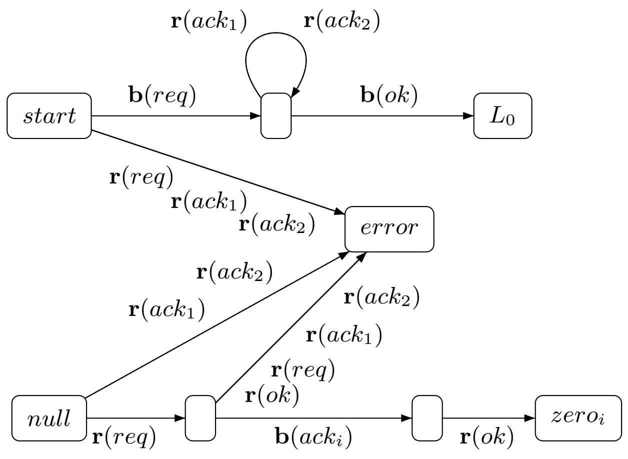

Fig. 17. Initialization of the star: Selection of the control node and initialization of satellite node associated to counter $i \in \{1, 2\}$.

node in that state that broadcasts the acknowledgment moves to state ia_i and forces the control node to move to a special state. If the control node receives another acknowledgment it moves to an error state. Otherwise, it sends an iok_i message that forces the ia_i node to state one_i and all other nodes in auxiliary states back to $zero_i$. This way, only one satellite in state $zero_i$ changes its state to state one_i. The rule $L : \texttt{if}\ c_i = 0\ \texttt{then}\ \texttt{goto}\ L'\ \texttt{else}\ c_i := c_i - 1;\ \texttt{goto}\ L''$ is simulated by the automaton in Figure 19. The simulation works as follows: the control node chooses to broadcast non deterministically either dec_i or $zero_i$. If it broadcasts dec_i, it launches a symmetric protocol to the one for incrementation which forces one satellite node to move from one_i to $zero_i$. If it broadcasts a $zero_i$ message, it forces all one_i satellite nodes to move to a special $sink$ state. Thus, a zero test is correctly executed only if the $sink$ state is never generated. Hence the Minsky machine stops in state L_F if an only if there exists an execution of the protocol that starts from an initial configuration in in which nodes have state in $\{start, null_0, null_1\}$ and, in some neighbor, it non-deterministically selects a control node and a sufficient number of satellite nodes (initially all in state $zero$) such that the protocol stops in a configuration γ_1 in which there are no nodes in $error$ and $sink$ (nowhere in the network) state and at least of the control node is in state L_F. From this property, it follows that TARGET is undecidable for AHN restricted to K-bounded path configurations with $K \geq 3$.

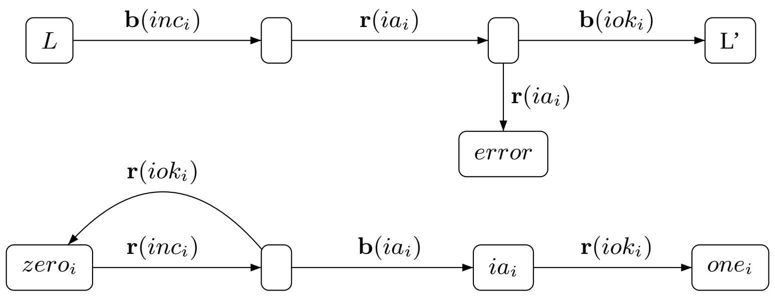

Fig. 18. Encoding $L : c_i := c_i + 1;$ `goto` L'.

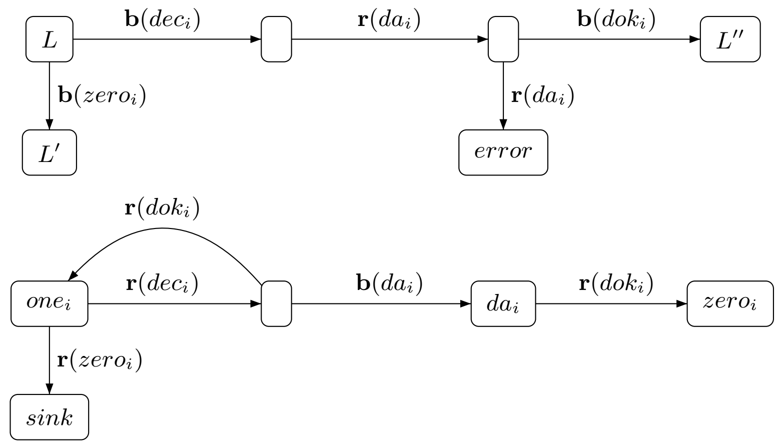

Fig. 19. Encoding $L :$ `if` $c_i = 0$ `then goto` L' `else` $c_i := c_i - 1;$ `goto` L''