On the Coverability Problem for Asynchronous Broadcast Networks

Giorgio Delzanno and Riccardo Traverso

Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Italy

Abstract. We study verification problems for networks in which nodes communicate via asynchronous broadcast messages. This type of communication is achieved by using a distributed model in which nodes have a local buffer. We consider here safety properties expressed as a coverability problem with an arbitrary initial configurations. This formulation naturally models the search of initial topology that may lead to an error state in the protocol. We consider here different policies for handling local buffers such as unordered, FIFO and lossy FIFO queues. Coverability turns out to be decidable for unordered and for lossy FIFO buffers. Undecidability for FIFO buffers follows from a reduction from the halting problem of counter machines obtained via a non-trivial protocol that controls the interferences due to the use of broadcast communication.

1 Introduction

Background: Models of Broadcast Communication Broadcast communication is commonly adopted in the design of protocols for wireless, mobile and ad hoc networks. In this setting broadcast communication provides a synchronization pattern independent from the current configuration of the underlying topology. Formal models of ad hoc and wireless networks are often based on an abstract representation of broadcast communication, namely an operation that atomically synchronizes a sender node with the set of all connected receivers ready to receive the message [7,8,20,21,16,18]. In order to model delays and interferences between different communications, the start and end time of a transmission may be kept distinct, thus modeling its boundaries as in [15,12], or an explicit asynchronous communication mechanism may be defined, e.g., implemented via local message buffers as in AWN [17,10].

The study of the relative expressiveness of the different types of communication becomes particularly intriguing when the network configurations are not fixed a priori. This type of analysis can be formalized, e.g., by studying decidability and undecidability of problems like reachability (of a fixed target configuration) and coverability (reachability of a configuration that is greater or equal than a target configuration w.r.t. some ordering). In particular, the coverability problem is well-suited to reason on parameterized systems, i.e., systems in which the number of processes is not fixed a priori. Coverability is also tightly related to verification of safety properties (violations of properties like mutual exclusion can be expressed as coverability of bad patterns) [3].

A New Model In this paper we present decidability and undecidability results for the coverability problem for asynchronous broadcast networks (ABN), a model that combines a topology-dependent, asynchronous communication mechanism. Our formal model of asynchronous broadcast communication combines three main features:

– a graph representation of a network configuration decoupled from the specification of individual process behavior,
– a topology-dependent semantics of synchronization,
– the use of local mailboxes to deliver messages to individual nodes.

The resulting communication layer is similar to that of languages like AWN [10]. As in other protocol models like ω [20,21] and AHN [7], our main abstraction comes from considering protocols defined via a communicating finite-state automaton replicated on each node of the network. In our setting the coverability problem is formulated here as follows. We first define an initial configuration as any graph in which nodes have labels that represents the initial state of the protocol (and no constraints on edges). Coverability consists then in checking whether there exists an initial configuration that can reach a target configuration that contains a specific pattern given in form of subgraphs or nodes in a given state. A similar definition is considered in [7] and [12] for synchronous and non-atomic semantics, respectively. Our analysis is carried out with different policies to handle buffers, namely unordered bags (an abstraction of a tuple space), and perfect or lossy FIFO channels.

Technical Results In contrast with the synchronous case, when local buffers are treated as bags of messages, the coverability problem becomes decidable. For the proof, we first give a reduction to the restricted case of fully connected topologies. Then, we resort to the theory of well-structured transition systems [1,11] and show that, for fully connected topologies, reachability of a given control state can be solved via a symbolic backward search algorithm.

When mailboxes are ordered buffers, we obtain undecidability already in the case of fully connected topologies. Indeed, by using (FIFO) ordered mailboxes, we give nodes the possibility of recognizing communication with multiple neighbours with the same role. Although we cannot use this feature to define discovery protocols as in [7], while simulating a counter machine we exploit the FIFO ordering in order to block computations in which interferences may lead to incorrect results.

The coverability problem becomes decidable when introducing non-deterministic message losses. We exploit again the theory of well structured transition systems for this positive result.

We then move to an extended model in which a node can test if a mailbox is empty. This extension leads to undecidability with unordered bags and non-selective broadcast. The emptiness test cannot be used to directly control interferences, however we can exploit it to introduce a way to distinguish good from bad computations as in the case of ordered mailboxes.

To our knowledge, these are the first results for the coverability problem for a formal model of asynchronous broadcast communication.

2 Well-structured Transition Systems

We report here the main definition underlying the theory of well-structured transition systems (wsts) [3,11]. We first need some terminology. Given a finite alphabet Σ, a bag (multiset) m is a mapping $m : \Sigma \to \mathbb{N}$ such that $m(a)$ is the number of occurrences of a in m. We often use the notation $[a_1, \ldots, a_n]$ to indicate a multiset containing (possibly repeated) occurrences $a_1, \ldots, a_n$ of elements in Σ. We use $\oplus$ for multiset union and $\ominus$ for multiset difference. Given a finite alphabet Σ, we use Σ^* to denote the set of finite words over Σ. We use $w_1 \cdot w_2$ to denote concatenation of words w_1 and w_2. We use $|w|$ to denote the length of string w, and $\overline{n}$ to denote the set $\{1, \ldots, n\}$.

A transition system is a pair $\langle D, \rightarrow \rangle$, where D is a set of configurations and $\rightarrow \subseteq D \times D$ is a transition relation. We use $\rightarrow^*$ to denote the reflexive and transitive closure of $\rightarrow$. Given $S \subseteq D$, $post(S)$ [resp. $pre(S)$] is the set of successors [resp. predecessors] of S defined as $\{c' \mid c \rightarrow c',\ c \in S\}$ [resp. $\{c' \mid c' \rightarrow c,\ c \in S\}$]. To give an example, consider a Petri net with places P and transitions T, the set of markings (multiset of symbols in P that count the number of tokens in each place) equipped with the firing relation induced by T forms a (infinite-state) transition system. Consider now a reflexive and transitive order $\langle D, \leq \rangle$ defined on the configurations. An upward closed set of configurations is a set $I \subseteq D$ s.t. for all $d, d' \in D$, if $d \in I$ and $d \leq d'$ then $d' \in I$. We say that $\leq$ is a well-quasi ordering (wqo) if for each infinite sequence $e_1 e_2 \ldots$ there exists i and j s.t. $i < j$ and $e_i \leq e_j$. If $\leq$ is a wqo, then an upward closed set I has always a finite set F of minimal elements (finite basis property). We use $F^\uparrow$ to denote the upward closed set generated by F. We first notice that if A is finite, then $(A, =)$ is a wqo. The following lemma shows other classical properties of wqo theory [13,4,11,3].

Lemma 1. *Let $(A, \leq)$ and $(B, \preccurlyeq)$ be wqo's, then the following properties hold:*

- *$(A^*, \leq^*)$ is a wqo, where A^* is the set of finite words over A, $u = u_1 \cdot \ldots \cdot u_n \leq^* v = v_1 \cdot \ldots \cdot v_m$ iff there exists a strictly increasing injection h s.t. $u_i \leq v_{h(i)}$ for $i : 1, \ldots, n$ (word embedding).*
- *$(A^b, \leq^b)$ is a wqo, where A^b is the set of bags over A and $u = [u_1, \ldots, u_n] \leq^b v = [v_1, \ldots, v_m]$ iff there exists an injection h s.t. $u_i \leq v_{h(i)}$ for $i : 1, \ldots, n$ (bag inclusion).*
- *$(A \times B, \lhd)$ is a wqo, where $\langle u, u' \rangle \lhd \langle v, v' \rangle$ iff $u \leq v$ and $u' \preccurlyeq v'$.*

For instance, in the Petri net model, verification of properties like mutual exclusion can be formulated as reachability of markings that are covered (i.e. they are greater than w.r.t. multiset inclusion) by a marking m containing only two tokens in the place that represent the critical section. In other words the ideal generated by the marking m represents all violations to mutual exclusion. Marking (multiset) inclusion is a wqo by Dickson's Lemma.

Definition 1. *For a set I of initial configurations and a finite set U, the coverability problem consists in checking if there exists an initial configuration $c_0 \in I$ and a configuration $c_f \in U^{\uparrow}$ s.t. $c_0 \rightarrow^* c_f$ (U covers c_f).*

Definition 2. *A well-structured transition system (wsts) is a transition system $T = \langle D, \rightarrow \rangle$ equipped with a quasi ordering $\leq$ on configurations, s.t.,*

- *$\leq$ is a decidable wqo (i.e. there exists an algorithm to decide $\leq$);*
- *$\rightarrow$ is monotonic (or compatible) w.r.t. $\leq$, i.e., for any c, c', d if $c \rightarrow c'$ and $c \leq d$, there exists d' s.t. $d \rightarrow d'$ and $c' \leq d'$.*
- *given a finite basis F of an upward closed set I, there exists an algorithm that takes in input B and computes a finite basis of $pre(B^{\uparrow})$, we call such an operator symbolic predecessor operator $Pre(B)$.*

For an initial set of configurations I, we call *initial test* the condition $I \cap S^{\uparrow} = \emptyset$ for any $S \subseteq D$. We say that the initial test is decidable if it can be decided for S. The following theorem then holds [11,3].

Lemma 2. *Consider a wsts $T = \langle D, \rightarrow, \leq \rangle$, a set $U \subseteq D$, an initial I with decidable initial test. Then, the coverability problem is decidable for T, I, U.*

The algorithm is based on a backward reachability analysis in which we use bases of upward closed sets to represent predecessors. We maintain intermediate results in a set of bases $I \subseteq P_f(D)$ (where $P_f(D)$ is the finite powerset over D). Intermediate sets can be compared by using the $\leq$ orderings on elements. Namely, for $B, B' \in P_f(D)$, $B \sqsubseteq_b B'$ if for each $c' \in B'$, there exists $c \in B$ s.t. $c \leq c'$ (i.e. $B_1 \sqsubseteq_b B_2$ implies $B_2^{\uparrow} \subseteq B_1^{\uparrow}$). Similarly, for $I, J \subseteq P_f(D)$, $I \sqsubseteq_s J$ iff for each $B' \in J$, there exists $B \in I$ s.t. $B \sqsubseteq_b B'$ (i.e. $I \sqsubseteq_s J$ implies $(\bigcup_{B \in J} B^{\uparrow}) \subseteq (\bigcup_{C \in I} C^{\uparrow})$).

We define then the sequence $I_0 I_1 \ldots$ as follows:

- $I_0 = \{U\}$,
- $I_{i+1} = I_i \cup \{Pre(B) | B \in I_i\}$ for $i \geq 0$.

Since $\leq$ is a wqo, then the sequence necessarily stabilizes, i.e., there exists k s.t. $I_{k+1} \sqsubseteq_s I_k$ (i.e. I_k is a least fixpoint). When a fixpoint has been detected, to decide coverability it remains to check whether I intersect $I_k^{\uparrow}$ that we assume to be a decidable problem. Again in the example of Petri nets, we can use marking inclusion to define upward closed set of markings and use a symbolic semantics for the transition relation to effectively compute predecessor.

3 Asynchronous Broadcast Networks (ABN)

Our model of computations consists of configurations – snapshots of the global state of a system – and of a global transition relation – a description of its dynamic behavior. A configuration is defined as a labelled graph. Nodes correspond to processes running a pre-defined protocol. Each node has a local message buffer used to collect messages sent by neighbours. We forbid self-loops in the communication graph to model half-duplex communication, a natural assumption for ad hoc and wireless networks. A protocol is specified via a finite-state automaton with send and receive operations that correspond to write [resp. read] on remote [resp. local] buffers. Communication is topology-dependent, anonymous and asynchronous: when a process at node n sends a message a, the process does not block, and the message is added to the local mailbox of all of its neighbors without explicit information about the sender (i.e. messages do not contain node identifiers). More formally, in the rest of the paper we consider a finite set Σ of messages, and different disciplines for handling the mailbox (message buffer), e.g., unordered mailboxes that we represent as bags over Σ, and ordered mailboxes that we represent as words over Σ.

The initial configuration is any graph in which all the nodes are in the initial control state and all local buffers are empty. Even if the set of control states is finite, there are infinitely many possible initial configurations. We next formalize the above intuition starting from the mailbox structure.

In order to deal in a uniform way with different mailbox types we define a transition system parametric on the data structures used to model mailboxes. More specifically, we consider a mailbox structure $\mathbb{M} = \langle \mathcal{M}, del?, add, del, \flat \rangle$, where $\mathcal{M}$ is a denumerable set of elements denoting possible mailbox contents; for $a \in \Sigma$ and $m \in \mathcal{M}$, $add(a, m)$ denotes the mailbox obtained by adding a to m, $del?(a, m)$ is true if a can be removed from m; $del(a, m)$ denotes the mailbox obtained by removing a from m when possible, undefined otherwise. Finally, $\flat \in \mathcal{M}$ denotes the empty mailbox. We call an element a of m *visible* when $del?(a, m) = true$. Their specific semantics and corresponding properties change with the type of mailbox considered.

Definition 3. *A protocol is defined by a process $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$, where Q is a finite set of control states, Σ is a finite message alphabet, $Act = \{\tau\} \cup \{!!a, ??a \mid a \in \Sigma\}$, $R \subseteq Q \times Act \times Q$ is the transition relation, $q_0 \in Q$ is an initial control state.*

The label τ represents the capability of performing an internal action, and the label $!!a$ $[??a]$ represents the capability of broadcasting [receiving] a message $a \in \Sigma$.

Definition 4 (Configurations). *Configurations are undirected $(Q \times \mathcal{M})$-graphs. A $(Q \times \mathcal{M})$-graph γ is a tuple $\langle V, E, L \rangle$, where V is a finite set of nodes, $E \subseteq V \times V$ is a finite set of edges (such that E is symmetric and $\forall v \in V.(v, v) \notin E$ to model undirected edges and half-duplex communication), and $L : V \to (Q \times \mathcal{M})$ is a labelling function.*

In the rest of the paper, for an edge $\langle u, v \rangle$ in E, we use the notation $u \sim_\gamma v$ and say that the vertices u and v are adjacent to one another in γ. We omit γ, and simply write $u \sim v$, when it is made clear by the context. We use $L(\gamma)$ to represent the set of labels in γ. The set of all possible configurations is denoted $\mathcal{C}$, while $\mathcal{C}_0 \subseteq \mathcal{C}$ is the set of all graphs in which every node has the same label $\langle q_0, \flat \rangle$ that denotes the initial state of individual processes.

Given the labeling L and the node v s.t. $L(v) = \langle q, m \rangle$, we define $L_s(v) = q$ (state component of $L(v)$) and $L_b(v) = m$ (buffer component of $L(v)$). Furthermore, for $\gamma = \langle V, E, L \rangle \in \mathcal{C}$, we use $L_s(\gamma)$ to denote the set $\{L_s(v) \mid v \in V\}$. $\Rightarrow_{\mathbb{M}} \subseteq \mathcal{C} \times \mathcal{C}$ is the transition relation defined next.

Definition 5 (Operational Semantics). *For $\mathbb{M} = \langle \mathcal{M}, del?, add, del, \flat \rangle$, an Asynchronous Broadcast Network (ABN) associated to $\mathcal{P}$ is defined by its associated transition system $\mathcal{T}(\mathcal{P}, \mathbb{M}) = \langle \mathcal{C}, \Rightarrow_{\mathbb{M}}, \mathcal{C}_0 \rangle$. For $\gamma = \langle V, E, L \rangle$ and $\gamma' = \langle V, E, L' \rangle$, $\gamma \Rightarrow_{\mathbb{M}} \gamma'$ holds iff one of the following conditions on L and L' holds:*

- *(local) there exists $v \in V$ such that $(L_s(v), \tau, L'_s(v)) \in R$, $L_b(v) = L'_b(v)$, and $L(u) = L'(u)$ for each $u \in V \setminus \{v\}$.*
- *(broadcast) there exists $v \in V$ and $a \in \Sigma$ such that $(L_s(v), !!a, L'_s(v)) \in R$, $L_b(v) = L'_b(v)$ and for every $u \in V \setminus \{v\}$*
  - *if $u \sim v$ then $L'_b(u) = add(a, L_b(u))$ and $L_s(u) = L'_s(u)$,*
  - *otherwise $L(u) = L'(u)$;*
- *(receive) there exists $v \in V$ and $a \in \Sigma$ such that $(L_s(v), ??a, L'_s(v)) \in R$, $del?(a, L_b(v))$ is satisfied, $L'_b(v) = del(a, L_b(v))$, and $L(u) = L'(u)$ for each $u \in V \setminus \{v\}$.*

A local transition only affects the state of the process that executes it. A broadcast message has the effect of adding the corresponding message to the mailboxes of all the neighbors of the sender. The sender then moves to the next state. Notice that broadcast is never blocking for the sender. Receivers can read the message in different instants. This models asynchronous communication. A reception of a message a is blocking for the receiver whenever the buffer is empty or the visible elements are all different from a. If a is visible in the mailbox, the message is removed and the process moves to the next state.

An *execution* is a sequence $\gamma_0 \gamma_1 \ldots$ such that γ_0 is an initial configuration, and $\gamma_i \Rightarrow_{\mathbb{M}} \gamma_{i+1}$ for $i \geq 0$. We use $\Rightarrow^*_{\mathbb{M}}$ to denote the reflexive and transitive closure of $\Rightarrow_{\mathbb{M}}$. We often use $\Rightarrow$ and $\Rightarrow^*$ when the mailbox type $\mathbb{M}$ is clear from the context.

3.1 Safety Analysis: the Coverability Decision Problem

The coverability problem parametric on the mailbox structure $\mathbb{M}$ is defined as follows.

Definition 6. *Given a protocol $\mathcal{P}$ with transition system $\mathcal{T}(\mathcal{P}, \mathbb{M}) = \langle \mathcal{C}, \Rightarrow_{\mathbb{M}}, \mathcal{C}_0 \rangle$ and a control state q, the coverability problem COVER$(\mathbb{M})$ states: is there an initial configuration $\gamma_0 \in \mathcal{C}_0$ and a configuration $\gamma_1 \in \mathcal{C}$ such that $\gamma_0 \Rightarrow^*_{\mathbb{M}} \gamma_1$ and $q \in L_s(\gamma_1)$?*

We often use the terminology γ_0 reaches state q as an abbreviation for $\gamma_0 \Rightarrow^*_{\mathbb{M}} \gamma_1$ and $q \in L_s(\gamma_1)$ for some configuration γ_1.

Besides being parametric on the mailbox structure, our decision problem is parametric on the shape of the initial configuration. As mentioned in the introduction, this feature models in a natural way verification problems for protocols with partial information about the structure of the network.

4 Unordered Mailboxes

In this section we study the coverability problems for ASBNs in which mailboxes are unordered buffers modeled as bags over the finite message alphabet Σ. The mailbox structure *Bag* is defined as follows: $\mathcal{M}$ is the denumerable set of bags over Σ, $add(a, m) = [a] \oplus m$ (where $[a]$ is the singleton bag containing a), $del?(a, m) = true$ iff $m(a) > 0$, $del(a, m) = m \ominus [a]$, and $\flat \in \mathcal{M}$ is the empty bag $[]$. The operational semantics follows from the general definitions.

4.1 Decidability of the Coverability Problem

Let us consider the instance $COVER(Bag)$ of the coverability problem. For synchronous broadcast, coverability is undecidable for arbitrary topologies [7]. We show next that coverability becomes decidable for unordered mailboxes.

We prove the results in two different steps. We first show that, for the purpose of deciding coverability, we can focus on fully connected topologies. We then show that ABN with fully connected topologies form a wsts for which an instance of the generic algorithm of Section 2 can be applied.

A configuration $\gamma = \langle V, E, L\rangle$ is fully connected if $u \sim_\gamma v$ for each pair of distinct nodes $u, v \in V$. We use $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, \mathbb{M})$ [resp. $COVER^{\mathcal{K}}(\mathbb{M})$] to denote the restriction of $\mathcal{T}(\mathcal{P}, \mathbb{M})$ [resp. $COVER(\mathbb{M})$] to fully connected configurations only. For asynchronous communication with unordered mailboxes, coverability for arbitrary topologies case can be reduced to the fully connected case. One side of the property is immediate. If there exists a fully connected initial configuration that reaches a configuration in which state q occurs, then coverability is solved. The other implication is the interesting case to consider. The intuition is that we can exploit the fact that mailboxes are unordered to ignore messages sent along links that are not present in a given topology. The formalization is given below.

Lemma 3. *If there exists an arbitrary topology from which we can reach state q, then there exists a fully connected topology from which we can also reach q.*

Proof. Let $\mathcal{P} = \langle Q, \Sigma, R, q_0\rangle$ be a protocol and let $\gamma = \langle V, E, L\rangle$ and $\gamma' = \langle V, E, L'\rangle$ be two configurations from the transition system $\mathcal{T}(\mathcal{P}, Bag)$ such that $\gamma \Rightarrow_{\mathbb{M}} \gamma'$ by applying some rule $r \in R$ to a node $v \in V$. Given a configuration $\delta = \langle V, V \times V, K\rangle$ in the transition system $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, Bag)$ such that, for all $u \in V$, $L_s(u) = K_s(u) \wedge L_b(u) \sqsubseteq^b K_b(u)$ we can apply r at node v in order to reach the configuration $\delta' = \langle V, V \times V, K'\rangle$ where $L'_s(u) = K'_s(u)$ for all $u \in V$, and, depending on the type of r:

- if $r = (L_s(v), \tau, L'_s(v))$ then $L'_b(u) = K'_b(u)$ for every $u \in V$;
- if $r = (L_s(v), !!a, L'_s(v))$ then $L'_b(v) = K'_b(v)$ and $K'_b(u) = add(a, K_b(u))$ for every $u \in V \setminus \{v\}$;
- if $r = (L_s(v), ??a, L'_s(v))$ then $del?(a, K_b(v))$ holds because $L_b(v) \sqsubseteq^b K_b(v)$ and by hypothesis $del?(a, L_b(v))$ is satisfied, $L'_b(v) = del(a, K_b(v))$, and $L'_b(u) = K'_b(u)$ for every remaining $u \in V \setminus \{v\}$.

In any case we obtain that $L'_s(u) = K'_s(u)$ and $L'_b(u) \sqsubseteq^b K'_b(u)$ for all $u \in V$. We can finally conclude that, given an execution $\gamma_0\gamma_1 \dots \gamma_k$ in $\mathcal{T}(\mathcal{P}, Bag)$ where $q \in L_s(\gamma_k)$ and $\gamma_0 = \langle V, E, L^0 \rangle$ we can build another one $\delta_0\delta_1 \dots \delta_k$ in $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, Bag)$ by starting from $\delta_0 = \langle V, V \times V, L^0 \rangle$ and replicating each rule application as shown in the previous scheme. The thesis then follows by observing that, because of the construction, the set of control states in δ_i is the same as those in γ_i for $0 \leq i \leq k$ and in particular $q \in K_s(\gamma_k)$.

A configuration with fully connected topology can be simply viewed as a multiset of pairs formed by a control state and a word over the message alphabet. The label $\langle q, m \rangle$ represents the current state q of the corresponding node and its bag of messages m.

Given a state q we would like to decide whether there exists an initial multiset γ_0 of node labels such that γ_0 can reach a configuration γ containing a label $[\langle q, m \rangle]$ for some bag m of messages. A natural way to attack this problem consists in performing a backward reachability analysis starting from a finite representation of all configurations that contain state q. For this purpose, we introduce an ordering $\preccurlyeq^b$ on configurations s.t.

$$\gamma = [\langle q_1, m_1 \rangle, \dots, \langle q_n, m_n \rangle] \preccurlyeq^b \gamma' = [\langle q'_1, m'_1 \rangle, \dots, \langle q'_k, m'_k \rangle]$$

iff there exists an injection $h : \overline{n} \to \overline{k}$ s.t. $q_i = q'_{h(i)}$ and $m_i \sqsubseteq^b m'_{h(i)}$ (m_i is contained in $m'_{h(i)}$) for $i : 1, \dots, n$. The ordering can be used to finitely represent the (upward closed) set U of configurations containing at least one occurrence of a given control state q. Indeed, U is generated by the configuration $\langle q, [] \rangle$. This is the first step for applying the theory of wsts described in section 2. The following properties are proved in appendix.

Lemma 4. *For fully connected transitions, the ordering $\preccurlyeq^b$ is a wqo.*

As a direct consequence of the wqo property we have that any upward closed set I of configurations has a finite basis, i.e., a finite set of elements that generates the whole set I, as in the above mentioned example of the set U. In addition, the transition system is compatible with respect to $\preccurlyeq^b$.

Lemma 5. *For fully connected transitions, $\Rightarrow_{\mathbb{M}}$ is monotonic w.r.t. $\preccurlyeq^b$.*

The previous property implies that the set of predecessors of an upward closed set is still upward closed. For this purpose, starting from the finite basis F of the upward closed set of configurations $F^{\uparrow}$, we need to effectively compute the finite basis F' of $pre(F^{\uparrow})$ (i.e. $Pre(F)$). This can be done algorithmically.

Lemma 6. *There exists an algorithm to compute the predecessor operator Pre.*

The following lemma then holds.

Lemma 7. *$COVER^{\mathcal{K}}(Bag)$ is decidable.*

Proof. We apply the backward algorithm defined in Section 2. The seed of the computation is the basis $U = \{[\langle q, []\rangle]\}$. Termination of the symbolic backward exploration is ensured by the wqo property of $\preccurlyeq^b$. To decide if the fixpoint I_k of the sequence $\{I_i\}_{i\geq 0}$ intersects the set of initial configurations, we simply have to check if there exists a $B \in I_k$ s.t. all elements have the form $\langle q_0, []\rangle$. Thus, the initial test is decidable.

Thanks to Lemmas 3, 7 and 2 we can conclude that the following property holds.

Theorem 1. *COVER(Bag) is decidable.*

5 FIFO Mailboxes

In this section we prove that the coverability problem for ABN with perfect FIFO buffers becomes undecidable. In this context we instantiate the mailbox structure $FIFO$ as follows: $\mathcal{M}$ is defined as Σ^*; $add(a, m) = m \cdot a$ (concatenation of a and m); $del?(a, m) = true$ iff $m = a \cdot m'$; $del(a, m)$ is the string m' whenever $m = a \cdot m'$, undefined otherwise; finally, $\flat \in \mathcal{M}$ is the empty string ϵ.

Theorem 2. *COVER(FIFO) is undecidable.*

Proof. The proof is based on a reduction of the halting problem for two-counter machines – a well known undecidable problem – to *COVER(FIFO)*. A two-counter machine is defined by a pair $\langle L, I\rangle$ where L is a finite set of control locations and $I \subseteq L \times Op \times L$ is a finite set of instructions such that $Op = \{c{+}{+}, c{-}{-}, c == 0 \mid c \in \{x_1, x_2\}\}$ is a set of operators over the counters x_1 and x_2, and $\ell_0 \in L$ is the initial location. Configurations are tuples $\langle \ell, v_1, v_2\rangle$ such that $\ell \in L$ is the current location and v_1, v_2 are natural numbers that denote the current value of x_1 and x_2, respectively. The operational semantics is defined in a standard way: the execution of increment and decrement updates the control location and the current value of the corresponding counter, a zero-test updates the control location whenever the test is satisfied in the current state of the counter.

The rationale behind the reduction of coverability to the halting problem of two-counter machines is as follows. We first use an election protocol that assigns fixed roles (controller/slave) to a pair of adjacent nodes. Since the initial configuration is not fixed a priori, our election protocol does not forbid the election of multiple pairs of controller/slave nodes. However, for the rest of the simulation to succeed we only require that at least one pair is elected. The controller/slave nodes set up their mailboxes in order to use them as overlapping circular queues. Messages represent the current value (in unary) of the counters. The simulation is guided by the controller. The slave node forwards all received messages back to the controller. As an example, to check that x_1 is zero, the controller reads all messages in the mailbox and checks that in between two successive reads of the marker for x_1 there are no units. We use interference to denote an unwanted message occurring in the mailbox of a controller/slave node. Since the network topology is not fixed a priori, a keypoint of the whole construction is the capability of controlling interferences with other nodes, e.g., avoiding the adjacency between multiple controllers and slaves. For this purpose, we use special control messages to coordinate the different phases and exploit the FIFO mailboxes in order to enforce the simulation to get into a deadlock state whenever the same control message is received more than once. A detailed description of the protocol is given in appendix. The following theorem then holds.

Corollary 1. *$COVER^{\mathcal{K}}(FIFO)$ is undecidable.*

The same construction can be used for the fully connected case.

6 Lossy FIFO Mailboxes

We now consider coverability for ABNs in which mailboxes are lossy FIFO channels, i.e., channels in which messages may non-deterministically be lost. Again we first consider the transition system $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, LFIFO)$ for fully connected networks and the corresponding coverability problem $COVER^{\mathcal{K}}(LFIFO)$. Given a protocol $\mathcal{P}$, a configuration γ of $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, LFIFO)$ is a multiset of pairs $\langle q, m \rangle$ where $q \in Q$ and $m \in \Sigma^*$. To model non-deterministic loss of messages, we modify the operational semantics by introducing lossy steps. We first need to define the following ordering between configurations:

$$\gamma = [\langle q_1, m_1 \rangle, \ldots, \langle q_k, m_k \rangle] \preccurlyeq^* \gamma' = [\langle q'_1, m'_1 \rangle, \ldots, \langle q'_n, m'_n \rangle]$$

if there exists an injective mapping $h : \overline{k} \to \overline{n}$ s.t. $q_i = q'_{h(i)}$ and $m_i \sqsubseteq^* m'_{h(i)}$ for $i : 1, \ldots, k$. We modify the transition relation $\Rightarrow_{\mathbb{M}}$ to include lossy steps before and after each transition in the original system as follows: $\gamma \longmapsto \gamma'$ iff there exists η and ν s.t. $\eta \preccurlyeq^* \gamma$, $\eta \Rightarrow_{\mathbb{M}} \nu$, and $\gamma' \preccurlyeq^* \nu$.

As for unordered mailbox we first consider the restricted case of fully connected topologies and show that ABN with lossy FIFO mailboxes form a wsts.

The following lemmas are proved in appendix.

Lemma 8. *The ordering $\preccurlyeq^*$ is a wqo.*

We consider again upward closed sets of configurations since the wqo properties ensure the existence of a finite basis and the target configuration can be generated by the minimal element $\langle q, \epsilon \rangle$. For fully connected topologies, the transition relation $\longmapsto$ is monotonic with respect to $\preccurlyeq^*$. (it directly follows from the lossy semantics).

Lemma 9. *For fully connected configurations, $\longmapsto$ is monotonic w.r.t. $\preccurlyeq^*$.*

Lemma 10. *For fully connected configurations, the symbolic operator Pre is effective.*

By Lemmas 9, 10, and 2, the following theorem then holds.

Theorem 3. *$COVER^{\mathcal{K}}(LFIFO)$ is decidable.*

The coverability problem for lossy FIFO mailboxes has a property in common with the one for bags, that is in both cases processes are able to ignore incoming messages indefinitely; this is achieved by either leaving the message in the multiset or by deleting it from the lossy fifo queue. We can therefore take advantage of this property to obtain the following theorem.

Theorem 4. *$COVER(LFIFO)$ is decidable.*

7 ABN with Emptiness Test

In this section we enrich the ABN model with a new type of transitions in order to enable processes to test whether their mailbox is empty, we call the resulting model ABN$_\epsilon$. The set *Act* of action labels is extended to include ϵ, i.e., $Act = \{\tau, \epsilon\} \cup \{!!a, ??a \mid a \in \Sigma\}$. The transition systems associated to an ABN$_\epsilon$ are changed accordingly to take ϵ into account; given two configurations $\gamma = \langle V, E, L\rangle$ and $\gamma' = \langle V, E, L'\rangle$, $\gamma \Rightarrow_{\mathbb{M}} \gamma'$ holds also if the following condition is met.

Emptiness test There exists a $v \in V$ such that $(L_s(v), \epsilon, L'_s(v)) \in R$, $L_b(v) = L'_b(v) = \flat$, and $L(u) = L'(u)$ for each $u \in V \setminus \{v\}$.

The only difference with respect to the semantics of τ-transitions consists in the $L_b(v) = \flat$ condition, that ensures that ϵ-transitions only fire when the mailbox is empty.

The introduction of ϵ-transitions affects the different instances of the coverability problem in different ways.

The simplest case is for $COVER^{\mathcal{K}}(FIFO)$ and $COVER(FIFO)$, which of course are still undecidable: the possibility to test the emptiness of the mailbox does not have any effect on the reduction from two-counter machines.

The decidability of $COVER^{\mathcal{K}}(LFIFO)$ depends on the Lemma 9 on monotonicity and on Lemma 10 on the computability of the predecessors. Monotonicity is preserved by ϵ-transitions: given two configurations $\gamma \preccurlyeq^* \eta$ if ϵ can be fired in γ then it is enabled in η too, through a preliminary lossy step that empties the relevant mailbox. The predecessors can be computed by extending the case analysis to rules of the form (q, ϵ, q'); it is very similar to computing predecessors of local transitions, except that we can replace $\langle q', m\rangle$ with $\langle q, m\rangle$ in the configuration G only if $m = \flat$. From these observations we can therefore derive that both $COVER(LFIFO)$ and $COVER^{\mathcal{K}}(LFIFO)$ are decidable even with ϵ-transitions.

We incur in a completely different case when considering bags: as it can be shown, the extended semantics traces indeed a sharp boundary between decidability and undecidability. Without the emptiness test, both reachability problems $COVER(Bag)$ and $COVER^{\mathcal{K}}(Bag)$ are decidable; we prove that the operator ϵ introduced with the extended model is sufficient to make them undecidable. The proof proceeds by building a reduction from the control state reachability problem for two-counter machines to $COVER(Bag)$. The reduction encodes a counter machine $\mathcal{M}$ with an ABN protocol $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$ where, like before, each location $l \in L$ and each instruction $i \in I$ corresponds respectively to a state $\mathcal{P}(l) \in Q$ and to a set of intermediate states and rules.

Theorem 5. *$COVER(Bag)$ [$COVER^{\mathcal{K}}(Bag)$] is undecidable in ABN_ϵ.*

Proof. The protocol is split in two phases. In the first phase processes follow a distributed election protocol to identify who takes care of which role and who is excluded from the simulation. The second phase is the simulation of $\mathcal{M}$. The alphabet is partitioned in two sets, Σ_e for the election and Σ_s for the simulation. Since we do not make any particular assumption on the connectivity graph, the proof works for both $COVER^{\mathcal{K}}(Bag)$ and $COVER(Bag)$ topologies.

Election A simulation must be carried out by three nodes: a controller and two slaves, one per counter. As Figure 1 shows, the election protocol is similar to the one from section 5: the process which chooses to be a controller must receive announcements from two different slaves, while each slave must receive the announce message from a controller.

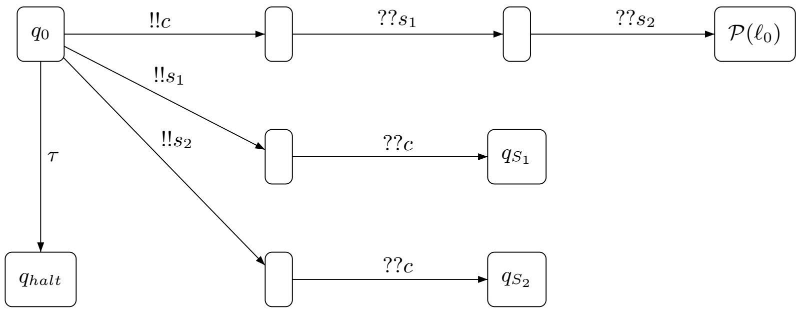


Fig. 1. $COVER(Bag)$: Election protocol

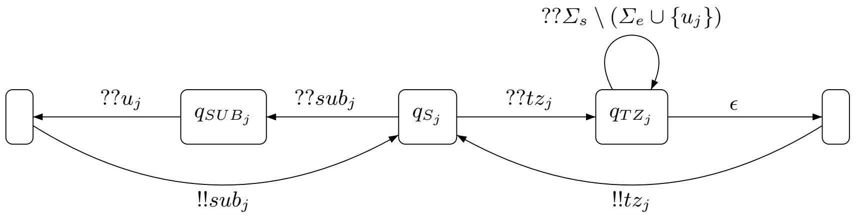


Fig. 2. $COVER(Bag)$: Slave process

Simulation Each slave S_j keeps in its mailbox a number of u_j messages equal to the current value of counter x_j. The controller sends messages sub_j or tz_j

to give orders depending on the instruction (l, op, l') that is going to be simulated by the system and waits for the interested slave to react accordingly (see Figure 2). Once the slave is done, the same control message is sent back to the controller as acknowledgment and the controller is able to proceed. When A is a set we write $??A$ to mean that for every $a \in A$ the protocol has a reception rule $??a$ with the same endpoints. Again, the increment can be done directly by the controller with a single broadcast $!!u_j$.

The election protocol only ensures the minimum connectivity requirements for the simulation, but this encoding alone is not enough to make sure that the simulation proceeds without interferences. The only instruction that may make a process of the system get stuck due to interferences is the test for zero: messages from Σ_e are removed from the mailbox only during the election, thus the only outgoing transition ϵ cannot proceed in such events. This is why the reduction does not compute reachability of the encoding $\mathcal{P}(\ell_f)$ of the target state ℓ_f, but instead it checks for the reachability of a fresh state q_{target} added according to Figure 3. It is straightforward to check that the instructions added to $\mathcal{M}$ just

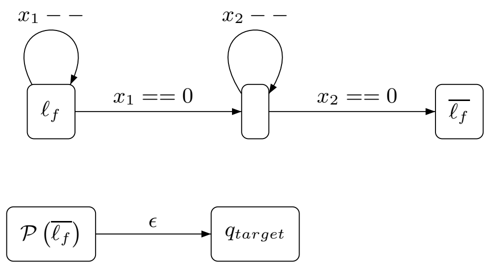


Fig. 3. *COVER*(*Bag*): Interference detection

decrement down to zero both counters before reaching the destination – thus forcing the system to deadlock in case of interferences on the slaves. The last ϵ-transition ensures the absence of interferences in the controller too.

8 Comparison with Other Semantics and Related Work

We have studied coverability problems for networks with an arbitrary number of identical finite automata equipped with asynchronous communication primitives. Nodes have local mailboxes controlled with different policies (e.g. unordered and FIFO mailbox). Coverability is defined with respect to initial configurations with an arbitrary finite number of nodes. Our results are summarized in Table 1. Our analysis completes previous work on verification and expressiveness (w.r.t. coverability) of broadcast communication. More specifically, for synchronous broadcast communication, the coverability problem is decidable for fully connected graphs [9] and undecidable for arbitrary graphs in the AHN model of [7]. Broadcast in AHN is topology-dependent. Synchronous communication is used here to

	$COVER^{\mathcal{K}}(\mathbb{M})$		$COVER(\mathbb{M})$	
	ABN	ABN$_\epsilon$	ABN	ABN$_\epsilon$
LFIFO	✓	✓	✓	✓
BAG	✓	x	✓	x
FIFO	x	x	x	x

Table 1. Decidability results for $COVER^{\mathcal{K}}(\mathbb{M})$ and $COVER(\mathbb{M})$

implement a sort of discovery protocol that, by a careful control of interferences, allows individual nodes to infer precise information about their vicinity (e.g. the existence of one and only one neighbour with a certain role). The discovery protocol is a building block for more complex computations. In this paper we use similar idea but reductions of different nature to obtain undecidability (e.g. we encode counters using mailboxes and not by using linked structures).

For variations of the synchronous semantics like those proposed in [12], intermittent nodes and non-atomic broadcast, coverability turns to be decidable. The decidability results exploit however different proof techniques. Indeed, coverability with intermittent nodes can be decided by using a weaker model than Petri nets, whereas we need to resort to the theory of wsts with nested data structures (bags of tuples containing multisets) to show decidability for the unordered case. There seems to be no direct reduction from one model to the other. Furthermore, by either introducing ϵ-transitions or moving to the case of ordered mailboxes we obtain undecidability of the resulting model. Concerning other models of broadcast communication, we would like to mention the CBS process calculi by Prasad [18,19] for fully connected networks with synchronous broadcast communication, the ω-calculus by Singh et al. [20,21] for fully connected networks with synchronous broadcast communication, and the model with topology-dependent broadcast by Ene and Muntean [8]. More recently, a process algebra for different types of communication, including asynchronous broadcast, called AWN, has been proposed in [10]. Semantics that take into consideration interferences and conflicts during a transmission have been proposed in [15,14]. Verification of unreliable communicating FIFO systems have been studied in [2,5]. In [6] the authors consider different classes of topologies with mixed lossy and perfect channels [6]. Differently from all the previous works, we consider here coverability for parametric initial configurations for a distributed model with asynchronous broadcast. Furthermore, we also consider different policies to handle the message buffers (bags/queues) and as well as unreliability of the communication media.

Concerning possible refinement of the unordered case, we are currently considering an extension with identifiers. When each node has a unique identifier that can be passed using broadcast messages and compared with equality, we conjecture that we just need a single slot of memory in each node (i.e. labels $\langle state, id, value\rangle$) and a single value in each message (i.e. labels $\langle m, value\rangle$) to obtain undecidability of coverability even for fully connected topologies. The reason is that equalities over identifiers induces an overlay network on top of the fully connected topology that can be exploited to build unbounded structures, used to simulate the memory of a Turing equivalent model. The introduction of the extended semantics with identifiers and value passing and the formal analysis of the coverability problem is left for an extended version of the work.

References

1. Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *LICS*, pages 313–321, 1996.
2. Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Inf. Comput.*, 130(1):71–90, 1996.
3. Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theor. Comput. Sci.*, 256(1-2):145–167, 2001.
4. Parosh Aziz Abdulla and Aletta Nylén. Better is better than well: On efficient verification of infinite-state systems. In *LICS*, pages 132–140, 2000.
5. Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Inf. Comput.*, 124(1):20–31, 1996.
6. Pierre Chambart and Ph. Schnoebelen. Mixing lossy and perfect fifo channels. In *CONCUR*, pages 340–355, 2008.
7. Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2010.
8. Cristian Ene and Traian Muntean. A broadcast-based calculus for communicating systems. In *IPDPS*, page 149, 2001.
9. Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *LICS*, pages 352–359, 1999.
10. Ansgar Fehnker, Rob J. van Glabbeek, Peter Höfner, Annabelle McIver, Marius Portmann, and Wee Lum Tan. A process algebra for wireless mesh networks. In *ESOP*, pages 295–315, 2012.
11. Alain Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.
12. Gianluigi Zavattaro Giorgio Delzanno, Arnaud Sangnier. Verification of ad hoc networks with node and communication failures. In *FORTE*, 2012.
13. Graham Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Mathematical Society*, 3(2):326–336, 1952.
14. Massimo Merro, Francesco Ballardin, and Eleonora Sibilio. A timed calculus for wireless systems. *Theor. Comput. Sci.*, 412(47):6585–6611, 2011.
15. Nicola Mezzetti and Davide Sangiorgi. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.
16. Sebastian Nanz and Chris Hankin. Formal security analysis for ad-hoc networks. *Electr. Notes Theor. Comput. Sci.*, 142:195–213, 2006.
17. Sebastian Nanz, Flemming Nielson, and Hanne Riis Nielson. Static analysis of topology-dependent broadcast networks. *Inf. Comput.*, 208(2):117–139, 2010.
18. K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.
19. K. V. S. Prasad. Broadcasting in time. In *COORDINATION*, pages 321–338, 1996.
20. Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. Query-based model checking of ad hoc network protocols. In *CONCUR*, volume 5710 of *Lecture Notes in Computer Science*, pages 603–619. Springer, 2009.

21. Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A process calculus for mobile ad hoc networks. *Sci. Comput. Program.*, 75(6):440–469, 2010.

A Examples

Semantics of $\Rightarrow_{Bag}$. For γ and γ', $\gamma \Rightarrow_{Bag} \gamma'$ holds iff one of the following conditions holds:

- (local) $\gamma = [\langle q, m\rangle] \oplus \eta$, $(q, \tau, q') \in R$, and $\gamma' = [\langle q', m\rangle] \oplus \eta$;
- (broadcast) $\gamma = [\langle q, m\rangle] \oplus \eta$, $\eta = [\langle q_1, m_1\rangle, \ldots, \langle q_k, m_k\rangle]$, $(q, !!a, q') \in R$, $\gamma' = [\langle q', m\rangle] \oplus \eta'$, and $\eta' = [\langle q_1, [a] \oplus m_1\rangle, \ldots, \langle q_k, [a] \oplus m_k\rangle]$ for $k \geq 0$;
- (receive) $\gamma = [\langle q, m\rangle] \oplus \eta$, $(q, ??a, q') \in R$, $m(a) > 0$, and $\gamma' = [\langle q', m \ominus [a]\rangle] \oplus \eta$;

As an example, consider a protocol with the following rules $a \xrightarrow{!!m} b$, $a \xrightarrow{??m} c$, and $c \xrightarrow{??m} d$. Starting from the initial configuration $[\langle a, []\rangle, \langle a, []\rangle, \langle a, []\rangle]$ we can first reach the configuration $[\langle b, []\rangle, \langle a, [m]\rangle, \langle a, [m]\rangle]$, from which we can obtain executions like $[\langle b, [m]\rangle, \langle b, [m]\rangle, \langle a, [m, m]\rangle]$, $[\langle b, [m]\rangle, \langle b, [m]\rangle, \langle c, [m]\rangle]$, and $[\langle b, [m]\rangle, \langle b, [m]\rangle, \langle d, []\rangle]$.

B Proofs

Proof of Lemma 4 For fully connected transitions, the ordering $\preccurlyeq^b$ is a wqo.

Proof. The ordering $\preccurlyeq^b$ consists of multiset inclusion of bags of pairs in which the first components are compared using $=$ and the second components are compared using multiset inclusion. The latter orderings are well quasi orderings. So is the point-wise ordering over pairs of such elements. By Lemma 1, we obtain then that $\preccurlyeq^b$ is a wqo.

Proof of Lemma 5

Proof. The proof is by case analysis on the type of applied rule. The interesting cases are send and receive. For the send case, assume that $q \xrightarrow{!!m} q'$ is a rule that can be fired in $\gamma = [\langle q, m\rangle, \langle q_1, m_1\rangle, \ldots, \langle q_k, m_k\rangle]$. Its firing leads to the configuration $\gamma = [\langle q', m\rangle, \langle q_1, [a] \oplus m_1\rangle, \ldots, \langle q_k, [a] \oplus m_k\rangle]$. Now assume that $\gamma \preccurlyeq^b \eta$. Then, $\eta = [\langle q, m'\rangle, \langle q_1, m'_1\rangle, \ldots, \langle q_k, m'_k\rangle] \oplus \nu$, where $m \sqsubseteq^b m'$ and $m_i \sqsubseteq^b m'_i$ for $i : 1, \ldots, k$. Clearly, the transition is still fireable in η and leads to the configuration $\eta' = [\langle q, m'\rangle, \langle q_1, [a] \oplus m'_1\rangle, \ldots, \langle q_k, [a] \oplus m'_k\rangle] \oplus \nu'$, where ν' is obtained from ν by adding a to all bags of its elements. As a result, we have that $\gamma' \preccurlyeq^b \eta'$.

For the receive case, assume that $q \xrightarrow{??m} q'$ is a rule that can be fired in $\gamma = [\langle q, [a] \oplus m\rangle] \oplus \nu$. Its firing leads to the configuration $\gamma' = [\langle q', m\rangle] \oplus \nu$. Now assume that $\gamma \preccurlyeq^b \eta$. Then, $\eta = [\langle q, [a] \oplus m'\rangle] \oplus \nu'$, where $m \sqsubseteq^b m'$ and $\nu \preccurlyeq^b \nu'$. Clearly, the transition is still fireable in η and leads to the configuration $\eta' = [\langle q', m\rangle] \oplus \nu'$. Thus, we have that $\gamma' \preccurlyeq^b \eta'$.

Proof of Lemma 6

Proof. We give an algorithm for a single generator. The definition can be extended to sets in the natural way. Let $G = [\langle q_1, m_1\rangle, \ldots, \langle q_k, m_k\rangle]$ be a configuration that generates an upward closed set I of configurations. The generators of $Pre(I)$ are defined by case analysis on the type of rules in R:

- Consider a rule $(q, \tau, q') \in R$, for every i s.t. $q' = q_i$, we have a generator obtained by replacing $\langle q_i, m_i\rangle$ with $\langle q, m_i\rangle$ in G. Another possible generator is obtained by adding $\langle q, []\rangle$ to G.
- Consider a rule $(q, !!a, q') \in R$, for each i s.t. $q' = q_i$ and $m_j = [a] \oplus m'_j$ for $j \neq i$, we have a generator $[\langle q_1, m'_1\rangle, \ldots, \langle q, m_i\rangle, \ldots, \langle q_k, m'_k\rangle]$ (the bag of the i-th process is unchanged). Furthermore, if $m_j = [a] \oplus m'_j$ for all j, another possible generator is $[\langle q, []\rangle, \langle q_1, m'_1\rangle, \ldots, \langle q_k, m'_k\rangle]$.
- Consider a rule $(q, ??a, q') \in R$. For every i s.t. $q' = q_i$, we have a generator obtained by replacing $\langle q_i, m_i\rangle$ with $\langle q, [a] \oplus m_i\rangle$ in G. Another possible generator is obtained by adding $\langle q, [a]\rangle$ to G.

It is straightforward to check that the previous gives a representation of $Pre(I)$.
□

Proof of Theorem 2 Formally, let $\mathcal{M} = \langle L, I\rangle$ be a two-counter machine. The encoding of $\mathcal{M}$ is defined via an ABN protocol $\mathcal{P}_{\mathcal{M}} = \langle Q, \Sigma, R, q_0\rangle$, where each location $l \in L$ corresponds to a state $\mathcal{P}(l) \in Q$, and each instruction $r \in I$ corresponds to a set of auxiliary states and rules. The i-th counter is represented by the FIFO mailboxes of two collaborating processes that forward each other the units u_i and a distinguished token t_i which marks the beginning of the circular queue. The protocol is split in three phases: election, initialization and simulation. The alphabet Σ is partitioned in Σ_e, the messages exchanged during the election, and Σ_s, the messages used for the simulation.

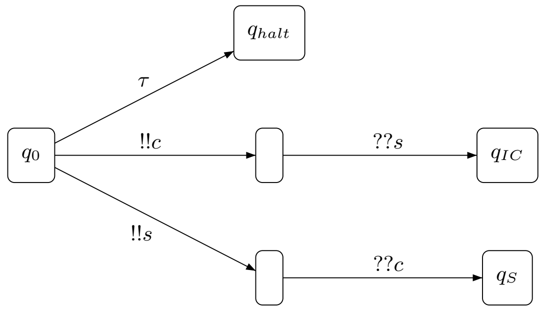
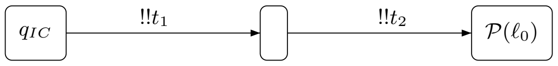


Fig. 4. Distributed election protocol

Election Since all processes start the protocol in the same initial state, the first thing is to distinguish their roles and make sure that all required communication links are present. This is the purpose of the election phase, during which

processes try to build pairs of communicating nodes. An active process may be either a controller or a slave. The duty of the controller is to orchestrate the whole simulation; the purpose of the slave is to bounce back every simulation message received from the controller. We do not care of how pairs will be able to proceed to the next phase, as long as their minimum connectivity requirements are fulfilled: each of them will try to independently carry on its own simulation, or deadlock due to interferences.
Figure 4 shows the election protocol. At first every node nondeterministically chooses an active role and announces it to its neighbours ($!!c$ or $!!s$) or shuts down itself by going to q_{halt}. After choosing a role, an handshake is needed to ensure the existence of the interconnection between controller and slave: a controller needs to receive the announce from a slave, and vice versa. The slave reaches the state q_S that starts its main loop, while the controller goes to an intermediate initialization state q_{IC}.

In order to simplify the presentation we introduce some syntactic sugar. Let $q \in Q$ be a state with an outgoing reception transition labeled by $??(m)$; then, in the actual ABN automaton, the transition is replaced by a regular $??m$ and appropriate deletion rules (q, m', q) are added to get rid of all those messages $m' \in \Sigma_s$ that cannot be consumed by some outgoing transition.



$$\begin{tikzcd} q_{IC} \arrow[r, "!!t_1"] & \arrow[r, "!!t_2"] & \mathcal{P}(\ell_0) \end{tikzcd}$$

Fig. 5. Initialization protocol

Initialization Once all nodes have been elected, it is time for the controller to initialize the representation of the counters. This is simply achieved by creating two tokens t_1 and t_2 to be used as markers for the circular queues representing the counters (Figure 5). At this point the controller is able to move to the encoding of the initial location of $\mathcal{M}$, $\mathcal{P}(\ell_0)$.

Simulation When everything is ready, the nodes proceed to the simulation phase. The slave node starts in q_{M_j}, a sort of idle state which manages the forwarding of messages of the counters (Figure 6). In order to increment the j-th counter the controller needs to put one more unit u_j in the circular queue, so a simple $!!u_j$ is enough. Subtraction is achieved by removing a u_j token from the loop (Figure 7); if x_j is already at 0, the controller will loop forever on q_{SUB_j} without being able to proceed. Testing for zero x_j can be done by checking for a sequence of two messages t_j in a row – or going to deadlock in any other case. Of course both operations must carefully keep forwarding messages for the other counter x_i. Remember that the election cannot forbid to multiple nodes to choose the same roles, nor it can avoid interferences between different simulations. The simulation protocol has to deal with this possibility by itself, and this the reason for the choice of the semantics for $??(a)$: messages in Σ_e cannot be read during the simulation

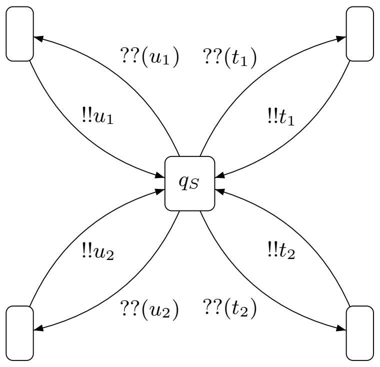


Fig. 6. Main loop of the slave node q_S

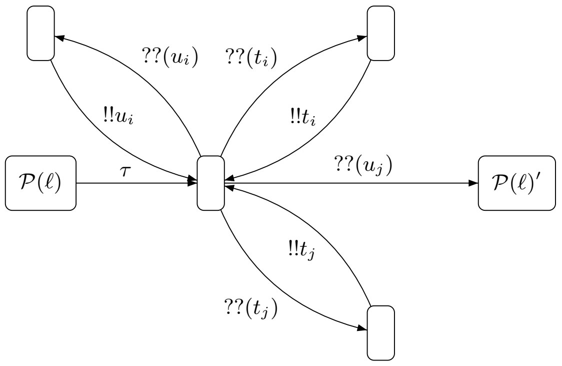


Fig. 7. Subtraction from counter j, with $i \neq j$

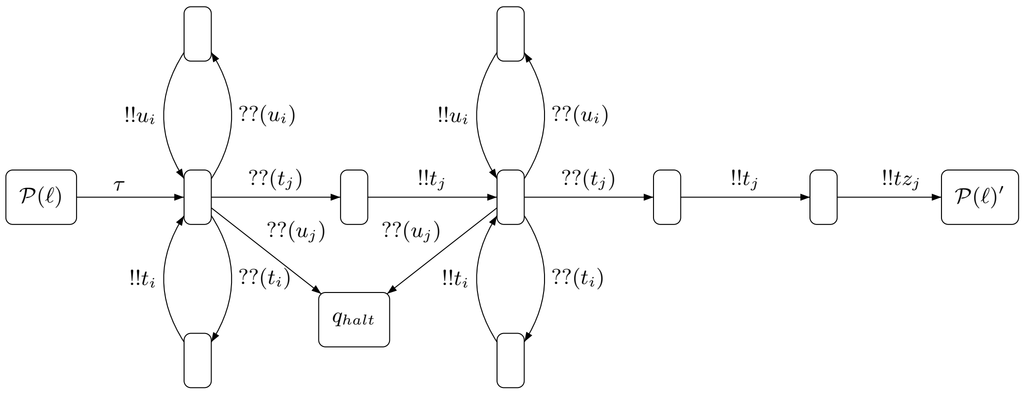


Fig. 8. Testing for zero counter j, with $i \neq j$

phases, and since they must appear before any message in Σ_s – having FIFO mailboxes – they always lead to a deadlock just before the system starts to consume untrustworthy messages. Subtraction and zero testing are not atomic, and to be sure that the slave was alive during the whole execution the reduction applies the greeting protocol shown in Figure 9 after each operation, so that for each instruction of the form $(l, op, l') \in I$ the system may reach the $\mathcal{P}(l')$ state only after exchanging hello messages; in case of interferences deadlocks are propagated to the controller before completing the simulation of the transition.

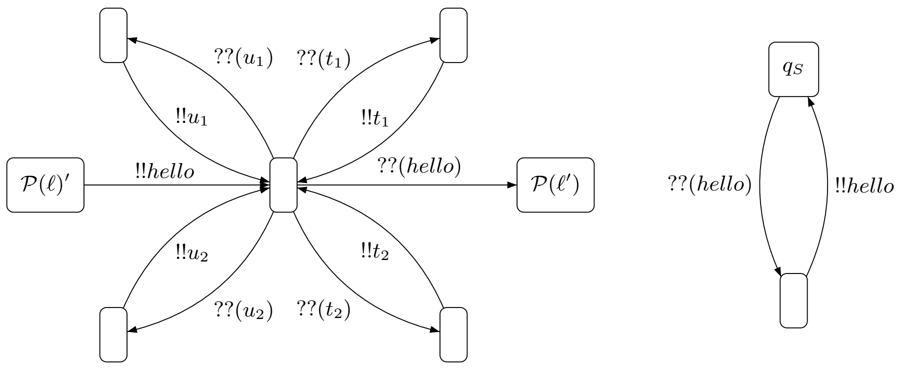


Fig. 9. Greeting protocol

Thanks to the FIFO ordering of the queues, we have the following properties. If the two-counter machine $\mathcal{M}$ halts, i.e., the target location ℓ_{target} is reachable from the initial configuration of the machine $\langle \ell_0, 0, 0 \rangle$, then there exists an initial configuration for the ABN protocol $\mathcal{P}_{\mathcal{M}}$ to reach a configuration in which the controller is labelled by $\mathcal{P}(\ell_{target})$. The property follows by selecting a configuration with only two nodes: one acting as controller and the other acting as slave. Vice versa, assume that there exists an initial configuration for the ABN protocol $\mathcal{P}_{\mathcal{M}}$ from which it is possible to reach a configuration in which ℓ_{halt} occurs. Our protocol is designed so as to ensure that the controller node and the slave node exchange acknowledgment messages in order to be informed of the presence of the link between them. This implies that at least a pair of nodes concludes the simulation (i.e. the controller reaches the state associated to ℓ_{halt}). We show that such a computation correctly mimics an execution of the machine $\mathcal{P}_{\mathcal{M}}$.

– Any increment instruction may be executed regardless of the current value of the counter; because of this, interferences before or after increments are not a problem at all: the execution is still valid.
– Zero-tests and decrements must be concluded by a run of the greeting protocol, meaning that only interference-free executions are able to proceed.

From the previous properties, it follows that if the controller node reaches the halting state, then the simulation of the counter operation is consistent with the original two counter machine. The undecidability proof is valid also for fully connected topologies.

Proof of Lemma 8

Proof. The ordering $\preccurlyeq^*$ consists of multiset inclusion of bags of pairs in which the first components are compared using $=$ and the second components are compared using string embedding. The latter orderings are well quasi orderings. So is the point-wise ordering over pairs of such elements. By Lemma 1, we obtain that $\preccurlyeq^*$ is a wqo.

Proof of Lemma 10

Proof. We give an algorithm for a singleton generator $G = [\langle q_1, m_1\rangle, \ldots, \langle q_k, m_k\rangle]$. The generators of $Pre(I)$ are defined by case analysis on the type of rules in R:

- Consider a rule $(q, \tau, q') \in R$, if there exists i s.t. $q' = q_i$, then one generator is obtained by replacing $\langle q_i, m_i\rangle$ with $\langle q, m_i\rangle$ in G. Another possible generator is obtain by adding $\langle q, \epsilon\rangle$ to G.
- Consider a rule $(q, !!a, q') \in R$. We use $S(a, m)$ to denote the singleton with m if $m = m' \cdot b$ with $a \neq b$, and the set $\{m', m\}$ (it models the loss of messages) if $m = m' \cdot a$. For each i s.t. $q' = q_i$, the corresponding generators have the form $[\langle q_1, m'_1\rangle, \ldots, \langle q, m'_i\rangle, \ldots, \langle q_k, m'_k\rangle]$ for $m'_j \in S(a, m_j)$ for every $j \neq i$. Furthermore, we have a generator $[\langle q, \epsilon\rangle, \langle q_1, m'_1\rangle, \ldots, \langle q, m'_i\rangle, \ldots, \langle q_k, m'_k\rangle]$ for $m'_j \in S(a, m_j)$ (it models the loss of messages) for every j.
- Consider a rule $(q, ??a, q') \in R$. For each i s.t. $q' = q_i$, the corresponding generators are G itself (loss of messages), and $G' = [\langle q_1, m_1\rangle, \ldots, \langle q, a \cdot m'_i\rangle, \ldots, \langle q_k, m_k\rangle]$.

It is straightforward to check that the above case analysis corresponds to all possible generators for $Pre(I)$.

Proof of Theorem 4

Proof. Let $\mathcal{P} = \langle Q, \Sigma, R, q_0\rangle$ be a protocol and $q \in Q$. There exist an execution $k_0 k_1 \cdots k_n$ in the transition system $\mathcal{T}^{\mathcal{K}}(\mathcal{P}, LFIFO)$ such that $q \in L_s(k_n)$ if and only if there exists another execution $g_0 g_1 \cdots g_n$ in $\mathcal{T}(\mathcal{P}, LFIFO)$ such that $q \in L_s(g_n)$. The theorem follows from this proposition and from theorem 3. One implication is trivial, being the first transition system a restricted version of the second. The other side can be proved by means of a monotonicity property on the individual transition steps, just like for lemma 3 but relying on the relation $\preccurlyeq^*$ instead of $\preccurlyeq^b$, and exploiting lossy steps instead of the bag semantics.