On the Verification of Broadcast Protocols

Javier Esparza * Alain Finkel† Richard Mayr‡

Abstract

We analyze the model-checking problems for safety and liveness properties in parameterized broadcast protocols, a model introduced in [5]. We show that the procedure suggested in [5] for safety properties may not terminate, whereas termination is guaranteed for the procedure of [1] based on upward closed sets. We show that the model-checking problem for liveness properties is undecidable. In fact, even the problem of deciding if a broadcast protocol may exhibit an infinite behavior is undecidable.

1. Introduction

In [5], Emerson and Namjoshi present an abstract reachability procedure —called the EN-procedure in the sequel— for the construction of a “covering graph”. It generalizes the Karp-Miller construction of a covering graph for Petri nets [9]. The EN-procedure can be applied to classes of systems satisfying some abstract conditions (essentially, computability of the least upper bounds of certain chains). By combining it with the automata-theoretic approach to model-checking [11], Emerson and Namjoshi show that it can be used to verify safety and liveness properties. Similar constructions have been studied in the framework of well-structured transition systems [6].

The termination of the EN-procedure depends on the class of systems being considered. In [5] termination is proved for the parameterized systems of [4, 8]; termination for Petri nets and vector addition systems was already proved in [9].

In the case of parameterized systems, the EN-procedure can be used to prove that a property holds *independently* of the number of processes participating in the protocol. In other words, it can show that *all* the elements of an infinite family of finite-state systems satisfy a certain property.

*Institut für Informatik, Technische Universität München, D-80290 München, Germany. E-mail: esparza@in.tum.de

†Lab. Spécification et Vérification, ENS de Cachan, 61, av. du Président Wilson, 94235 Cachan Cedex, France. E-mail: finkel@lsv.ens-cachan.fr

‡LFCS, Dept. of Computer Science, Univ. of Edinburgh, Edinburgh EH9 3JZ, UK. E-mail: mayrri@dcs.ed.ac.uk

One of the most interesting points of [5] is the application of the EN-procedure to a new parameterized model called parameterized broadcast protocols—shortened to *broadcast protocols* in the sequel. Broadcast protocols are systems composed of a finite but arbitrarily large number of indistinguishable processes that communicate by rendezvous (two processes exchange a message) or by broadcasts (a process sends a message to all other processes). It is also possible to incorporate a distinguished control process. While the case in which processes communicate only by rendezvous had already been studied in [8, 4], the extension to broadcasts is considered in [5] for the first time. The addition of broadcasts allows to model simplified versions of cache coherence protocols like MESI-protocols.

In [5] it is shown that broadcast protocols satisfy the abstract conditions necessary for the applicability of the EN-procedure. However, neither the termination issue nor the decidability of the model-checking problems for safety and liveness properties are examined. In this paper we address these points and obtain the following results:

- The EN-procedure may not terminate for broadcast protocols.
- The model-checking problem for safety properties is decidable. The decision procedure—which obviously cannot be the EN-procedure—is the result of instantiating for broadcast protocols an abstract backwards reachability algorithm introduced in [1].
- The model-checking problem for liveness properties is undecidable.

The paper is organized as follows: Section 2 introduces broadcast protocols and formalizes the model-checking problems for safety and liveness properties. Sections 3, 4, 5 present the results above, respectively.

2. Broadcast Protocols: Basic Definitions

2.1. Syntax

A *broadcast protocol* is a triple (S, L, R) where S is a finite set of *states*. L is a finite set of *labels* composed of: a set Σ_l

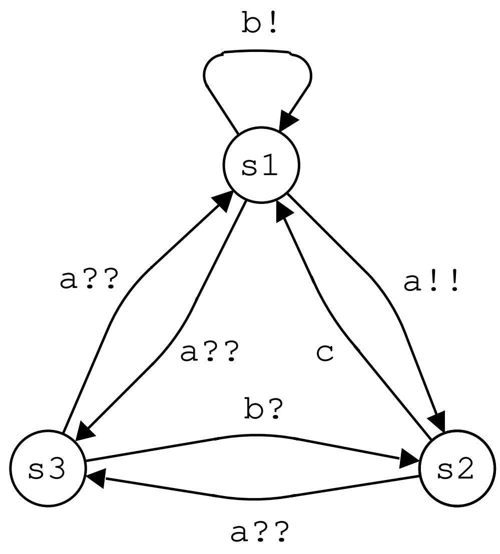

Figure 1. A broadcast protocol

of *local* labels, two sets $\Sigma_r \times \{?\}$ and $\Sigma_r \times \{!\}$ of *input* and *output rendez-vous* labels, and two sets $\Sigma_b \times \{??\}$ and $\Sigma_b \times \{!!\}$ of *input* and *output broadcast* labels, where $\Sigma_l, \Sigma_r, \Sigma_b$ are disjoint finite sets.

Along the paper $a, b, c, \ldots$ denote elements of $\Sigma = \Sigma_l \cup \Sigma_r \cup \Sigma_b$. Rendezvous and broadcast labels like $(a, ?)$ or $(b, !!)$ are shortened to $a?$ and $b!!$. Elements of Σ are called *actions*. $R \subseteq S \times L \times S$ is a set of *transitions* satisfying the following property: for every $a \in \Sigma_b$ and every state $s \in S$, there exists a state $s' \in S$ such that $s \xrightarrow{a??} s'$. Intuitively, this condition guarantees that a process is always willing to receive a broadcasted message. We represent broadcast protocols graphically as shown in Figure 1.

In this paper we consider broadcast protocols satisfying the following additional constraints. (a) For each state s and each broadcast label $a??$ there is exactly one state s' such that $s \xrightarrow{a??} s'$ (determinism). (b) Each label of the form a, $a!$, $a?$ and $a!!$ appears in exactly one transition.

These constraints are only used to simplify the presentation. All our decidability/undecidability results are valid for general broadcast protocols.

2.2. Semantics

Let $B = (S, L, R)$ be a broadcast protocol where $S = \{s_1, \ldots, s_n\}$. A *configuration* of B is a function $\mathbf{c} \colon S \to \mathbb{N}$. Intuitively, $\mathbf{c}(s_i)$ indicates how many processes are in the state s_i. We identify $\mathbf{c}$ with the vector $(\mathbf{c}(s_1), \ldots, \mathbf{c}(s_n)) \in \mathbb{N}^n$. We denote by $\mathbf{u}_i$ the configuration given by $\mathbf{u}_i(s_j) = 1$ if $i = j$ and $\mathbf{u}_i(s_j) = 0$ otherwise. Moves between configurations are either rendezvous (two processes exchange a message and move to new states) or broadcasts (a process sends a message to all other processes; all processes move to new states). The semantics of B is the smallest subset of $\mathbb{N}^n \times \Sigma \times \mathbb{N}^n$ satisfying the three conditions below, where a triple $(\mathbf{c}, a, \mathbf{c}') \in \mathbb{N}^n \times \Sigma \times \mathbb{N}^n$ is denoted by $\mathbf{c} \xrightarrow{a} \mathbf{c}'$.

- If $s_i \xrightarrow{a} s_j$ then $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ for every $\mathbf{c}$, $\mathbf{c}'$ such that $\mathbf{c}(s_i) > 0$ and $\mathbf{c}' = \mathbf{c} - \mathbf{u}_i + \mathbf{u}_j$.
  I.e. one process is removed from s_i, and one process is added to s_j.

- If $s_i \xrightarrow{a!} s_j$ and $s_k \xrightarrow{a?} s_l$ then $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ for every $\mathbf{c}$, $\mathbf{c}'$ such that $\mathbf{c}(s_i) > 0$, $\mathbf{c}(s_k) > 0$ and $\mathbf{c}' = \mathbf{c} - \mathbf{u}_i - \mathbf{u}_k + \mathbf{u}_j + \mathbf{u}_l$.
  I.e. one process is removed from s_i and s_k, and one process is added to s_j and s_l.

- If $s_i \xrightarrow{a!!} s_j$ then $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ for every $\mathbf{c}$, $\mathbf{c}'$ such that $\mathbf{c}(s_i) > 0$ and $\mathbf{c}'$ can be computed from $\mathbf{c}$ in the following three steps:

$$\mathbf{c}_1 = \mathbf{c} - \mathbf{u}_i \tag{1}$$

$$\mathbf{c}_2(s_k) = \sum_{\{s_l \mid s_l \xrightarrow{a??} s_k\}} \mathbf{c}_1(s_l) \tag{2}$$

$$\mathbf{c}' = \mathbf{c}_2 + \mathbf{u}_j \tag{3}$$

  I.e. the sending process leaves s_i (1), all other processes receive the broadcast and move to their destinations (2), and the sending process reaches s_j (3).

Thanks to our constraints (a) and (b) above, the configuration $\mathbf{c}'$ is completely determined by $\mathbf{c}$ and the action a. In the example of Figure 1 we have

$$(3,1,2) \xrightarrow{c} (4,0,2)$$
$$(3,1,2) \xrightarrow{b} (3,2,1)$$
$$(3,1,2) \xrightarrow{a} (2,1,3)$$

Given a broadcast protocol with n states, we call the $n \times n$ matrices having unit vectors as columns *broadcast matrices* [5]. Given an action $a \in \Sigma$, it is easy to see that there exists a broadcast matrix M_a and a vector $\mathbf{v}_a$ such that $\mathbf{c}' = M_a \cdot \mathbf{c} + \mathbf{v}_a$ holds whenever $\mathbf{c} \xrightarrow{a} \mathbf{c}'$. For example, for the action a in the example of Figure 1 we have

$$M_a = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad \mathbf{v}_a = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$$

Since broadcast matrices are closed under product, this observation can be generalized to arbitrary sequences $\sigma \in \Sigma^*$: If $\mathbf{c} \xrightarrow{\sigma} \mathbf{c}'$ then $\mathbf{c}' = M_\sigma \cdot \mathbf{c} + \mathbf{v}_\sigma$ for some broadcast matrix M_σ and vector $\mathbf{v}_\sigma$.

The *language* of B from an initial configuration $\mathbf{c}_0$, denoted by $L(B, \mathbf{c}_0)$, is the set of sequences $\sigma \in \Sigma^*$ such that $\mathbf{c}_0 \xrightarrow{\sigma} \mathbf{c}$ for some configuration $\mathbf{c}_0$. The ω-language of B from $\mathbf{c}_0$, denoted by $L_\omega(B, \mathbf{c}_0)$, is defined accordingly.

A *parameterized configuration* is a *partial* function $\mathbf{p}\colon S \to \mathbb{N}$. We identify it with a set of configurations, namely those extending $\mathbf{p}$ to a total function. So we identify the parameterized configuration of the broadcast protocol of Figure 1 given by $\mathbf{p}(s_1) = \mathbf{p}(s_2) = \perp$ (undefined) and $\mathbf{p}(s_3) = 3$ with the set of configurations $\{(n_1, n_2, 3) \mid n_1, n_2 \in \mathbb{N}\}$.

The *language* of B from an initial parameterized configuration $\mathbf{p}_0$, denoted by $L(B, \mathbf{p}_0)$, is defined as

$$L(B, \mathbf{p}_0) = \bigcup_{\mathbf{c} \in \mathbf{p}_0} L(B, \mathbf{c})$$

So $L(B, \mathbf{p}_0)$ contains all sequences of actions that the protocol can execute from all initial configurations that belong to the initial parameterized configuration $\mathbf{p}_0$. $L_\omega(B, \mathbf{p}_0)$ is defined analogously.

2.3. Model-Checking Problems

Following the automata-theoretic approach to model-checking (see for instance [11]), we formalize a linear safety property as a regular set of dangerous sequences of actions the protocol should *not* engage in. Similarly, a liveness property is formalized as an ω-regular language over Σ.

Notice that we consider languages over Σ, corresponding to properties on the *actions* of the system. In [5] properties on the *configurations* satisfying certain conditions are considered instead; for that, configurations are labeled with atomic properties. All the results of this paper hold for the languages of [5] as well.

We study the decidability of the following two model-checking problems:

Safety properties
Given: a broadcast protocol B, a parameterized configuration $\mathbf{p}_0$, a regular language L.
To decide: if $L(B, \mathbf{p}_0) \cap L = \emptyset$.

Liveness properties
Given: a broadcast protocol B, a parameterized configuration $\mathbf{p}_0$, an ω-regular language L.
To decide: if $L(B, \mathbf{p}_0) \cap L = \emptyset$.

These two problems can be approached using well-known automata-theoretic techniques. For the safety problem, we take a finite automaton $A = (Q, \Sigma, \delta, q_0, F)$ accepting the language L. The *combined system* of a protocol B with n states and an automaton A is a subset of $(\mathbb{N}^n \times Q) \times \Sigma \times (\mathbb{N}^n \times Q)$ defined by: $(\mathbf{c}, q) \xrightarrow{a} (\mathbf{c}', q')$ if and only if $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ in B and $q \xrightarrow{a} q'$ in A. Clearly, $L(B, \mathbf{p}_0) \cap L = \emptyset$ if and only if no path of the combined system starting at any $(\mathbf{c}, q_0)$, where $\mathbf{c} \in \mathbf{p}_0$, ever visits a combined state of the form $(\mathbf{c}', q)$ where $q \in F$. For the liveness problem we replace A by a Büchi automaton, and 'visits a state' by 'visits a state infinitely often'.

3. The EN-Procedure may not Terminate

The EN-procedure for the construction of the covering graph is described below. We exhibit a broadcast protocol for which it does not terminate. It is then straightforward to show that the procedure may not terminate either for combined systems.

Fix for the rest of this section a broadcast protocol $B = (S, L, R)$, where $S = \{s_1, \ldots, s_n\}$, and a parameterized initial configuration $\mathbf{p}_0$.

Let $(\mathbb{N} \cup \{\omega\})^n$ be the set of *ω-configurations* of B. The semantics of broadcast protocols is generalized to ω-configurations by letting $\omega + n = \omega - n = \omega$ for all $n \in \mathbb{N}$, $\omega + \omega = \omega$ and $\omega - \omega = 0$. Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be ω-configurations. We say $\mathbf{e}_1 \preceq \mathbf{e}_2$ if $\mathbf{e}_1$ is pointwise smaller than or equal to $\mathbf{e}_2$, where $n \leq \omega$ for every $n \in \mathbb{N} \cup \{\omega\}$. Clearly, $\preceq$ is a complete partial order on ω-configurations The least upper bound (*lub*) of a chain is the vector of *lubs* of the component chains. For a sequence of actions σ, define T_σ as the affine operator given by $T_\sigma(\mathbf{e}) = M_\sigma(\mathbf{e}) + \mathbf{v}_\sigma$.

The EN-procedure examines pairs $(\mathbf{e}, a)$, where $\mathbf{e}$ is an ω-configuration and $a \in \Sigma$. It is initialized with the empty graph and the set of unexamined pairs $\{(\mathbf{e}_0, a) \mid a \in \Sigma\}$, where $\mathbf{e}_0$ is defined by

$$\mathbf{e}_0(s_i) = \begin{cases} \mathbf{p}_0(s_i) & \text{if } \mathbf{p}_0(s_i) \text{ defined} \\ \omega & \text{otherwise.} \end{cases}$$

The procedure goes as follows:

0. Add the node $\mathbf{e}_0$ to the graph.
1. Choose an unexamined pair $(\mathbf{e}, a)$;
   if there are none, stop.
2. If there is no $\mathbf{e}'$ such that $\mathbf{e} \xrightarrow{a} \mathbf{e}'$,
   then mark $(\mathbf{e}, a)$ as examined and go to 1.
3. If $\mathbf{e} \xrightarrow{a} \mathbf{e}'$ for some $\mathbf{e}'$ then
   3.1 If the graph contains an
       ω-configuration $\mathbf{d} \succeq \mathbf{e}'$
       then make $\mathbf{d}$ the a-successor of $\mathbf{e}$;
   3.2 else, if the graph contains a path from
       some node $\mathbf{d}$ to $\mathbf{e}$ such that $\mathbf{d} \preceq \mathbf{e}'$,
       then let σ be the sequence of actions
       of this path, let $\mathbf{l}$ be the *lub* of the chain

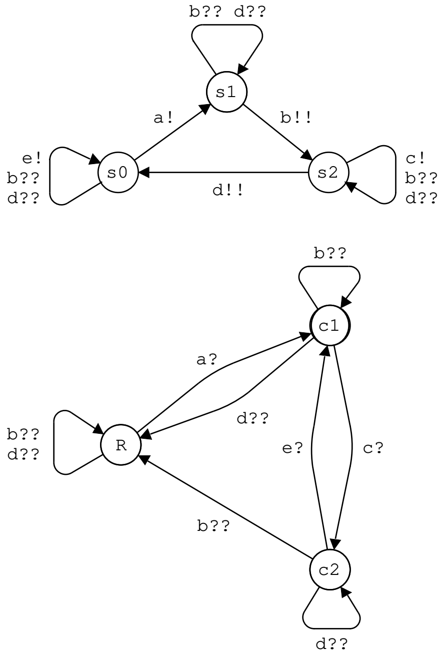

Figure 2. A protocol with an infinite covering graph

$$\mathbf{d} \preceq T_{\sigma a}(\mathbf{d}) \preceq T^2_{\sigma a}(\mathbf{d}) \preceq \ldots,$$

and make $\mathbf{l}$ the a-successor of $\mathbf{e}$;

3.3 else, create $\mathbf{e}'$ as the a-successor of $\mathbf{e}$.

4. Mark $(\mathbf{e}, a)$ as examined and go to 1.

Two questions arise: (a) is the *lub* of a chain effectively computable? and, (b) does the procedure terminate, i.e., is the covering graph finite? In [5], Emerson and Namjoshi answer (a) positively (this is essentially a consequence of the fact that there are only finitely many broadcast matrices for a given n), but they do not study (b). We present an example, inspired by [3], showing that the procedure may not terminate.

Consider the broadcast protocol B of Figure 2. Initially there is a process in state s_0 and arbitrarily many processes in state R. Following the terminology of [4, 8], the example consists of a *control process*, which is always in one of the states s_0, s_1, s_2, and an arbitrary number of identical *user processes*, initially in state R. The protocol simulates a machine operating on two counters modeled by the states c_1 and c_2, which draw their items from a repository, modeled by state R. The meaning of the different actions is:

- a: add 1 to c_1;
- b: reset c_2 to 0;
- c: transfer one item from c_1 to c_2;
- d: reset c_1 to 0;
- e: transfer one item from c_2 to c_1.

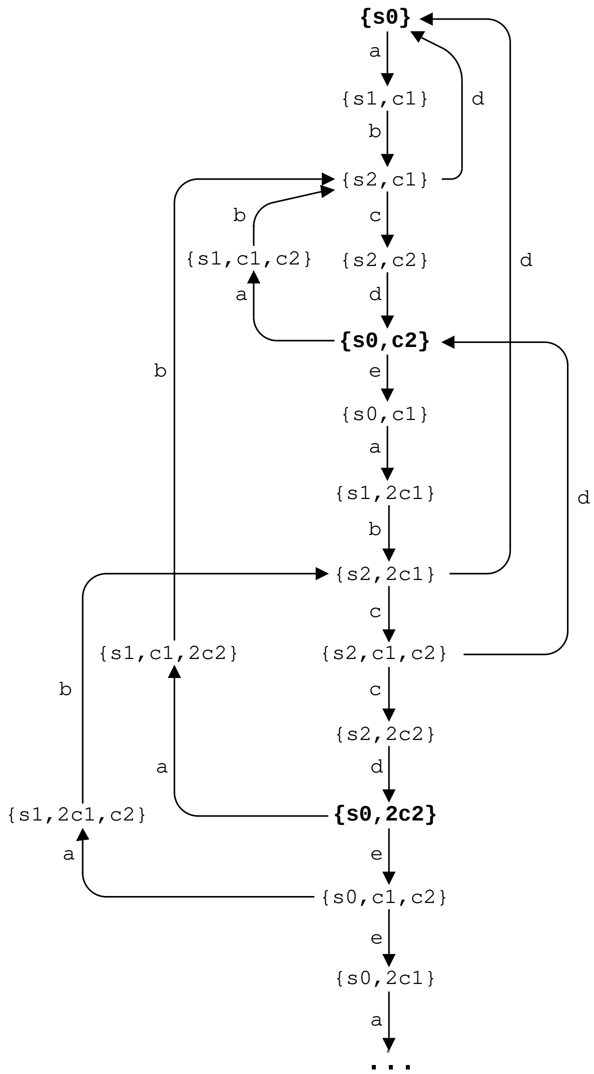

Figure 3. Semantics of the protocol of Figure 2

We construct the covering graph from $\mathbf{e}_0$, the ω-configuration putting 1 process in s_0, ω processes in R, and 0 processes elsewhere. We use a multiset notation for ω-configurations; for example, $\{s_2, 3c_1\}$ denotes the ω-configuration putting one process in s_2, 3 processes in c_1, and ω processes in R. Notice that every ω-configuration reachable from $\mathbf{e}_0$ puts ω processes in R, and so we omit this part. With this notation we have $\mathbf{e}_0 = \{s_0\}$. An initial part of the configurations of B reachable from $\mathbf{e}_0$ is shown in Figure 3. Notice that the sequence of actions $abcdeabc^2de^2abc^3de^3 \cdots abc^nde^n \cdots$ can be executed from $\mathbf{e}_0$, and that all the ω-configurations reached along this sequence are different. So, in particular, there are infinitely many reachable configurations from e_0.

Proposition 3.1 *The covering graph for the broadcast protocol of Figure 2 and the ω-configuration $\mathbf{e}_0 = \{s_0\}$ is infinite.*

Proof: Let $\sigma\tau$ be an arbitrary sequence of actions such that $\mathbf{e}_0 \xrightarrow{\sigma} \mathbf{e}_1 \xrightarrow{\tau} \mathbf{e}_2$, $\mathbf{e}_1 \preceq \mathbf{e}_2$, and $\mathbf{e}_1 \neq \mathbf{e}_2$. Since in every configuration reachable from $\mathbf{e}_0$ the total number of processes in the states s_1, s_2, s_3 is 1, both $\mathbf{e}_1$ and $\mathbf{e}_2$ coincide on these states. Since $\mathbf{e}_1 \neq \mathbf{e}_2$, τ contains at least one occurrence of b and d. Assume that the last occurrence of b precedes the last occurrence of d (the other case is similar). Then, τ has the form $\tau_1 b \tau_2 d \tau_3$, where $\tau_2\tau_3$ contains no b's and τ_3 contains no d's.

For the construction of the covering graph we can replace $\mathbf{e}_2$ by the *lub* of the chain $\mathbf{e}_1 \preceq \mathbf{e}_2 \preceq \mathbf{e}_3 \cdots$ where $\mathbf{e}_i = T_\tau^{i-1}(\mathbf{e}_1)$. We prove that $\mathbf{e}_i = \mathbf{e}_2$ for every $i \geq 2$, which implies that the *lub* is $\mathbf{e}_2$. This shows that for the protocol of Figure 2 the EN-procedure and the EN-procedure without step 3.2 compute the same graph. Since the latter computes an infinite graph, the covering graph is infinite.

To show $\mathbf{e}_i = \mathbf{e}_2$, we observe that, since $\mathbf{e}_2$ and $\mathbf{e}_i$ coincide on the states s_1, s_2, s_3 and R, it suffices to prove $\mathbf{e}_2(c_1) = \mathbf{e}_i(c_1)$ and $\mathbf{e}_2(c_2) = \mathbf{e}_i(c_2)$. We prove $\mathbf{e}_2(c_1) = \mathbf{e}_i(c_1)$, the other case being similar.

By the definition of T_τ, we have $\mathbf{e}_1 \xrightarrow{\tau} \mathbf{e}_2 \xrightarrow{\tau^{i-2}} \mathbf{e}_i$ for every $i \geq 2$. Since the occurrence of d removes all processes from c_1, $\mathbf{e}_2(c_1)$ and $\mathbf{e}_i(c_1)$ are determined by the suffix of τ and τ^{i-1} starting right after the last occurrence of d. This suffix is τ_3 in the two cases, and so we have $\mathbf{e}_2(c_1) = \#(\tau_3, a) + \#(\tau_3, e) - \#(\tau_3, c) = \mathbf{e}_i(c_1)$, where $\#$ denotes the number of occurrences of an action in a sequence. ∎

Since the protocol of Figure 2 contains both broadcast and rendezvous actions, the EN-procedure might still terminate for broadcast protocols with only broadcast moves. Unfortunately, this is not the case. To prove it, given a broadcast protocol $B = (S, L, R)$, we define the broadcast protocol $Exp(B)$ (*expansion* of B) as the result of performing the following two operations:

- each transition $s \xrightarrow{a} s'$ where a is a local action is replaced by the transition $s \xrightarrow{a!!} s'$, and a transition $t \xrightarrow{a??} t$ is added for each state t;
- each pair of transitions $s \xrightarrow{a!} s'$ and $t \xrightarrow{a?} t'$, where a is a rendezvous action, is replaced by the construction shown in Figure 4.[1] Moreover, in order to make sure that for each state s there is a state s' such that $s \xrightarrow{a_1??} s'$ and $s \xrightarrow{a_2??} s'$, transitions $s \xrightarrow{a_1??} s$ and $s \xrightarrow{a_2??} s$ are added where needed.

[1] The construction introduces two new states per rendezvous.

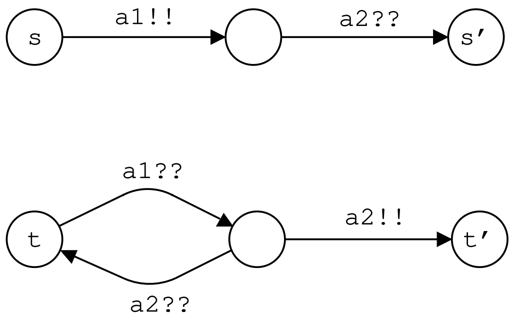

Figure 4. Simulation of a rendezvous by broadcasts

Observe that $Exp(B)$ only contains broadcast actions. We also define a morphism φ between the action sequences of B and $Exp(B)$ as follows: $\varphi(a) = a_1 a_2$ if a is a rendezvous action, and $\varphi(a) = a$ otherwise.

It is immediate to see that if $\mathbf{c} \xrightarrow{\sigma} \mathbf{c}'$ in B, then $\mathbf{c} \xrightarrow{\varphi(\sigma)} \mathbf{c}'$ in $Exp(B)$, and vice versa. Here we interpret $\mathbf{c}$ as the configuration of $Exp(B)$ that coincides with $\mathbf{c}$ on the states of B and puts no process in the new states. We now have:

Proposition 3.2 *Let B the broadcast protocol shown in Figure 2. The covering graph for the protocol $Exp(B)$ and the ω-configuration $\{s_0\}$ is infinite.*

Proof: The sequence

$$\varphi(abcdeabc^2de^2abc^3de^3 \cdots abc^nde^n \cdots)$$

can be executed from $\mathbf{e}_0$ in $Exp(B)$, and all the ω-configurations reached along this sequence are different. So there are infinitely many reachable configurations from $\mathbf{e}_0$ in $Exp(B)$. The argument used in the proof of Proposition 3.1, namely that every sequence τ must contain occurrences of b and d, is still valid, and in fact the proof can be carried out in the same way. ∎

The Exp construction also leads to the following result:

Proposition 3.3 *The safety and liveness problems for arbitrary protocols can be reduced to the same problems for broadcast protocols with only broadcast actions.*

Proof: Given an arbitrary protocol B and a regular or ω-regular language L, we have $L(B, \mathbf{p}_0) \cap L = \emptyset$ if and only if $L(Exp(B), \mathbf{p}_0) \cap \varphi(L) = \emptyset$. ∎

We finish this section with a small remark. It was shown in [8] that non-broadcast protocols with a control process and arbitrarily many user processes are more complicated to analyze than those in which all processes are identical. So one could ask if this is also the case for broadcast protocols. The answer is no. We can easily simulate the protocol of Figure 2 by another one in which all processes are identical: It suffices to add a new state *Init* and two new transitions $\mathit{Init} \xrightarrow{\mathit{init!!}} s_0$ and $\mathit{Init} \xrightarrow{\mathit{init??}} R$, and put all processes initially in the *Init* state. The new protocol must first do an *init*, by which essentially a process tells the others that it becomes the control process and the others become user processes.

4. A Model-Checking Algorithm for Safety Properties

Let B be a broadcast protocol with states $S = \{s_1, \ldots, s_n\}$ and a parameterized initial configuration $\mathbf{p}_0$, and let $A = (Q, \Sigma, \delta, q_0, F)$ be an automaton. The model-checking problem for safety properties can be reformulated as follows: Can some combined state $\mathbf{n} \in \mathbb{N}^n \times F$ be reached from a combined state $(\mathbf{c}_0, q_0)$ such that $\mathbf{c}_0 \in \mathbf{p}_0$?

We can use this observation to apply a general backwards reachability algorithm presented in [1] (see also [7]), which we "instantiate" for broadcast protocols in the rest of this section. The algorithm constructs the set of predecessors of $\mathbb{N}^n \times F$, and checks whether it has an empty intersection with $(\mathbf{p}_0, q_0)$.

We need some preliminaries. A set C of combined states is *upwards-closed* if $(\mathbf{c}, q) \in C$ implies $(\mathbf{c}', q') \in C$ for every $(\mathbf{c}, q) \trianglelefteq (\mathbf{c}', q')$, where $(\mathbf{c}, q) \trianglelefteq (\mathbf{c}', q')$ if $\mathbf{c} \preceq \mathbf{c}'$ and $q = q'$. Denote by $pred(C)$ the set of *immediate predecessors* of C (i.e. the combined states from which C can be reached in one step). We have the following result:

Proposition 4.1 *Let C be an upwards-closed set of combined states. Then:*

1. *The set of minimal elements of C is finite.*
2. *The set $pred(C)$ is upwards-closed.*
3. *The minimal elements of $pred(C)$ are effectively computable from the minimal elements of C.*

Proof: 1. Follows immediately from the fact that $\trianglelefteq$ is a well-ordering.
2. It suffices to prove that for each action a the set of immediate predecessors of C through the action a is upwards-closed. We do it for the case in which a is a broadcast action, the other cases being simpler. Assume we have $s_1 \xrightarrow{a!!} s_2$. The immediate prodecessors of C through a is the set of combined states $(\mathbf{c}, q)$ such that the following conditions hold for some minimal element $(\mathbf{c}', q')$ of C: (1) $M_a \cdot \mathbf{c} + \mathbf{v}_a \geq \mathbf{c}'$, (2) $\mathbf{c}(s_1) \geq 1$, and (3) $q \xrightarrow{a} q'$. Since

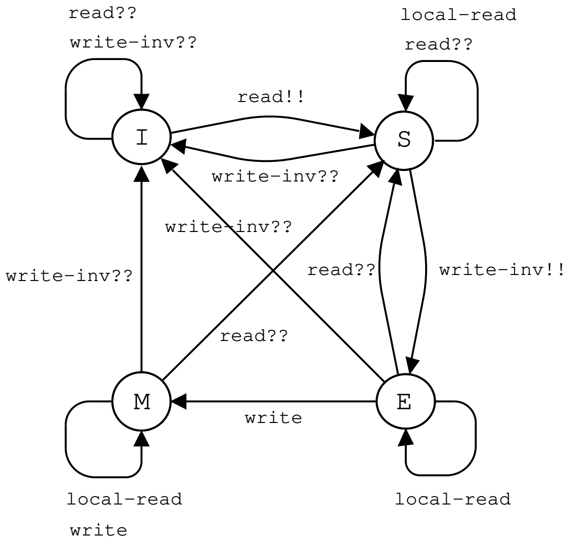

Figure 5. A MESI-protocol

M_a is a broadcast matrix, this set is upward-closed.
3. Again, it suffices to prove the result for the set of immediate predecessors of C through the action a. A little algebra shows that the minimal elements of this set are the combined states satisfying (2) and (3) above, plus a new condition (1′) of the form $M_a \cdot \mathbf{c} = \mathbf{d}$, where $\mathbf{d}$ is defined as follows. Since M_a is a broadcast matrix, there is exactly one state s such that $M_a(s, s_1) = 1$. We take $\mathbf{d}(s') = \mathbf{c}'(s') - \mathbf{v}_a(s')$ for every $s' \neq s$, and $\mathbf{d}(s) = \max(1, \mathbf{c}'(s) - \mathbf{v}_a(s))$. The set of solutions of (1′), (2), and (3) is clearly computable. ■

Since $\mathbb{N}^n \times F$ is an upwards-closed set, we can apply Proposition 4.1 and iteratively compute the minimal elements of $C_0 = \mathbb{N}^n \times F$, $C_1 = C_0 \cup pred(\mathbb{N}^n \times F)$, $C_2 = C_1 \cup pred^2(\mathbb{N}^n \times F)$, etc. But we know that in any infinite set of combined states there exist two elements $\mathbf{n}, \mathbf{n}'$ such that $\mathbf{n} \trianglelefteq \mathbf{n}'$ (i.e. $\trianglelefteq$ is a so-called *well-quasi-ordering*). Therefore, there is an n such that the minimal elements of C_n and $pred^*(\mathbb{N}^n \times F) = \bigcup_{i \geq 0} C_i$ coincide, and so the algorithm terminates.

In [5] the EN-procedure is applied to the protocol shown in Figure 5, a simplified version of a MESI-protocol for cache coherence. The initial configuration puts arbitrarily many processes in state I, and none in the other three states. For this particular protocol the EN-procedure terminates and yields a covering graph with four nodes [5]. The invariants $\#M = 0 \vee \#S = 0$ and $\#M + \#E \leq 1$, where $\#s$ denotes the number of processes in the state s, are proved to hold by observing that no node covers a configuration violating the invariants.

We can prove the same two invariants using our algorithm.

For these simple properties we can do without an automaton[2]: It suffices to compute the set of predecessors of the upwards-closed sets $\#M \geq 1 \wedge \#S \geq 1$ and $\#M + \#E \geq 2$, respectively, which we call U and V in the sequel. The reader can easily check that $pred(V) = V$, and so the procedure terminates after one step with $pred^*(V) = V$. For U we have

$$\begin{array}{rl} U\text{:} & \#M \geq 1 \wedge \#S \geq 1 \\ pred(U)\text{:} & (\#M \geq 1 \wedge \#S \geq 1) \vee \\ & (\#M = 0 \wedge \#E = 1 \wedge \#S \geq 1) \\ pred^2(U)\text{:} & pred(U) \end{array}$$

i.e. the procedure terminates after 2 steps. Since the predecessors of U and V do not contain any initial configuration, the invariants hold.

5. The Model-Checking Problem for Liveness Properties is Undecidable

We prove that it is undecidable if $L_\omega(B, \mathbf{p}_0) = \emptyset$, i.e. it is undecidable if the broadcast protocol B with initial parameterized configuration $\mathbf{p}_0$ can execute an infinite sequence. The undecidability of the model-checking problem follows.

The proof is by reduction from a problem on counter machines. It is closely related to the undecidability of a similar problem for lossy counter machines proved in [10] (in fact, it follows as a corollary from the results in [10]), and has been inspired by the undecidability proofs of [2].

We start by introducing some notations and definitions. A *counter machine* is a tuple $M = (Q, C, \Delta, q_0, H)$ where Q is a set of states, C is a set of counters, q_0 is an initial state, H is a set of halting states, and Δ is a set of transitions. Transitions are of three types:

- $q \xrightarrow{c:=c+1} q'$, which increase counter c,
- $q \xrightarrow{c:=c-1} q'$, which decrease counter c; these transitions can only be taken if the counter has a positive value;
- $q \xrightarrow{c=0} q'$, zero-tests that can only occur if the value of the counter is 0.

A *configuration* of $\mathcal{M}$ is a tuple $(q, j_1, \ldots, j_m)$, where q is a state, and $j_1, \ldots, j_m$ are natural numbers indicating the contents of the counters. The semantics of a counter machine is a relation $\rightarrow$ between configurations, defined as expected. A *run* is either an infinite sequence $c_1 \rightarrow c_2 \rightarrow \ldots$ or a finite sequence $c_1 \rightarrow \ldots \rightarrow c_n$ where c_n is halting. A configuration $(q, j_1, \ldots, j_m)$ is *initial* if $q = q_0$, and *n-bounded* if $\sum_{1 \leq i \leq m} j_i \leq n$. A run is *initial* if its first configuration is initial, *n-bounded* if all its configurations contain only n-bounded configurations, and *bounded* if it is n-bounded for some number n.

Theorem 5.1 *The following problem is undecidable:*

> *Given: a broadcast protocol B, a parameterized configuration $\mathbf{p}_0$.*
> *To decide: if $L_\omega(B, \mathbf{p}_0) = \emptyset$.*

Proof: We proceed by reduction from the following undecidable problem:

> Given: a 2-counter machine M.
> To decide: Does M halt on the input $(0, 0)$?

Let M' be a counter machine with 3 counters, behaving as follows. Initially, M' sets all counters to 0; then it simulates M on the counters c_1 and c_2, but after each step in the simulation it increases c_3 by 1. If M halts, then M' goes back to its initial state.

We make the following two observations about M':

- M' has an infinite bounded initial run if and only if M halts for $(0, 0)$.

  The only bounded initial run of M', if any, corresponds to the infinite iteration of the accepting run of M on $(0, 0)$ (all other infinite runs continuously increase c_3).

- Every infinite bounded run of M' (not necessarily initial!) contains infinitely many initial configurations.

  Such a run must set c_3 to 0 infinitely often, and this can only be done after visiting an initial configuration.

We simulate in a weak sense the machine M' by a broadcast protocol B. In B we have a state for each state and each counter of M', and two special states D and I. D is a special 'dead' state and I is introduced to keep an invariant (see below). The total number of processes in the counters of B plus the number of processes in I never increases. The following table describes the simulation:

Counter machine	Broadcast protocol
$q \xrightarrow{c:=c+1} q'$	$q \xrightarrow{inc_c!} q'$ $I \xrightarrow{inc_c?} c$
$q \xrightarrow{c:=c-1} q'$	$q \xrightarrow{dec_c!} q'$ $c \xrightarrow{dec_c?} I$
$q \xrightarrow{c=0} q'$	$q \xrightarrow{reset_c!!} q'$ $c \xrightarrow{reset_c??} D$

The parameterized configuration $\mathbf{p}_0$ puts 1 process in the initial state q_0, arbitrarily many in I, and 0 processes elsewhere.

[2]No automaton is used in [5] either.

The only situation in which the broadcast protocol does not faithfully simulate a step of the counter machine occurs when a $reset_c$ broadcast is executed at a configuration having at least one process in the counter c. We call such a broadcast a *cheat*.

Take an arbitrary run of the broadcast protocol and compute for all configurations $\mathbf{c}$ the sum $S(\mathbf{c}) = \mathbf{c}(c_1) + \mathbf{c}(c_2) + \mathbf{c}(c_3) + \mathbf{c}(I)$. The sums form a non-increasing sequence. Moreover, the sequence decreases only when the protocol cheats. We prove:

(1) If M halts for $(0,0)$, then $L_\omega(B, \mathbf{p}_0) \neq \emptyset$.
If M halts for $(0,0)$, then M' has a bounded infinite initial run, which iterates infinitely often the accepting run of M on $(0,0)$. Let b be the bound of this run. We consider the configuration $\mathbf{c} \in \mathbf{p}_0$ that puts b processes in I. We claim that B has an infinite run from $\mathbf{c}$. This run exactly mimics the infinite run of M'; since the infinite run is b-bounded, the total number of processes in the counters of B never exceeds b, and so B can mimic it even though there are only b processes in I. Since in this run the protocol only executes $q \xrightarrow{reset_c!!} q'$ when there are no processes in c, there are no cheats. So this run of B faithfully simulates the run of M', and so it is infinite.

(2) If $L_\omega(B, \mathbf{p}_0) \neq \emptyset$, then M halts for $(0,0)$.
Let $\mathbf{c} \in \mathbf{p}_0$ be a configuration such that B has an infinite run from $\mathbf{c}$. Since each cheat strictly decreases the sum $S(\mathbf{c})$, the run contains only finitely many cheats. Take a suffix of the run containing no cheats. Since the suffix is infinite, it corresponds to an infinite run r of M'. Moreover, r is bounded, because no counter can ever be larger than $\mathbf{c}(I)$. Now recall that every infinite bounded run of M' contains infinitely many initial configurations. So some suffix r' of r is an initial run of M'. Clearly M halts for the input $(0,0)$. ■

6. Conclusions

In this paper we have studied (parameterized) broadcast protocols, a model introduced by Emerson and Namjoshi in [5]. We have shown that the covering graph procedure proposed there for the verification of safety properties may not terminate, whereas termination is guaranteed for the procedure of [1] based on upward closed sets. So, while the covering graph technique is certainly adequate for several classes of systems, it is not the most suitable for broadcast protocols. Finally, we have shown that the model-checking problem for liveness properties is undecidable. In fact, even the problem of deciding if a broadcast protocol may exhibit an infinite behaviour is undecidable.

Acknowledgements Many thanks to Kedar Namjoshi for helpful discussions, to Giorgio Delzanno for implementing the backwards reachability algorithm using constraint programming, and to three ananymous referees, whose comments helped us to improve the presentation and correct a minor mistake.

References

[1] P. Abdulla, K. Cerans, B. Jonnson, Y.K. Tsay. General Decidability Theorems for Infinite State Systems. LICS, 1996.

[2] P. Abdulla, B. Jonnson. Undecidable Verification problems for Programs with Unreliable Channels. ICALP, LNCS 820, 1994.

[3] C. Dufourd, A. Finkel, P. Schnoebelen. Reset Nets Between Decidability and Undecidability. ICALP, LNCS 1443, 1998.

[4] E.A. Emerson, K.S. Namjoshi. Automatic Verification of Parameterized Synchronous Systems. CAV, LNCS 1102, 1996.

[5] E.A. Emerson, K.S. Namjoshi. On Model Checking for Non-Deterministic Infinite-State Systems. LICS, 1998.

[6] A. Finkel. Reduction and covering of infinite reachability trees. Information and Computation, 89(2):144-179, 1990.

[7] A. Finkel, P. Schnoebelen. Well-structured Transition Systems Everywhere! Research Report LSV-98-4, Lab. Specification and Verification, ENS de Cachan, France, 1998. To appear in TCS.

[8] S.M. German, A.P. Sistla. Reasoning about Systems with Many Processes. JACM 39(3), 1992.

[9] R. Karp, R. Miller. Parallel Program Schemata. JCSS 3, 1969.

[10] R. Mayr. Lossy Counter Machines. Technical Report TUM-I9827, Technische Universität München, 1998.

[11] M. Vardi, P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. LICS, 1986.