A Finite Covering Tree for Analysing Entropic Broadcast Protocols (Abstract)

Alain Finkel* Jerome Leroux

Abstract

For general broadcast protocols, the natural generalization of the Karp and Miller procedure does not terminate; even the problem of deciding if a broadcast protocol may exhibit an infinite behavior is undecidable [4]. We define a subclass of broadcast protocols, called entropic broadcast protocols, which "contains" the MESI protocol, and we prove that the Generalized Karp and Miller procedure terminates and it gives a finite covering tree. We also prove that liveness is undecidable.

1. Introduction

A generalization of the Karp and Miller procedure in the framework of well-structured transition systems has been presented in [7] (a similar construction can be found in [6]). It generalizes the Karp-Miller construction of a covering tree for Petri nets [10]. This abstract reachability procedure —called the GKM-procedure in the sequel– 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 [12], Emerson and Namjoshi show that it can be used to analyze safety and liveness properties [6].

The termination of the GKM-procedure depends on the class of systems being considered. In [6] termination is proved for the class of parameterized systems of [5, 9]; termination for Petri nets and vector addition systems was already proved in [10]. Termination for some classes of FIFO automata was studied in a survey [2].

In the case of parameterized systems, the GKM-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.

One of the most interesting points of [6] is the application of the GKM-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 [9, 5], the extension to broadcasts is considered in [6] for the first time. The addition of broadcasts allows to model simplified versions of cache coherence protocols like MESI-protocols (figure 1).

In [6] it is shown that broadcast protocols satisfy the abstract conditions necessary for the applicability of the GKM-procedure. However, neither the termination issue nor the decidability of the model-checking problems for safety and liveness properties are examined. In [4], it is shown that the GKM-procedure may not terminate for broadcast protocols but the model-checking for safety properties can be decided by the means of a backward procedure of [8] which operates on upward closed sets. Another recent paper about broadcast protocols, by Delzanno, Esparza and Podelski, studies the good constraints to optimize the backward procedure [1].

There are many advantages of obtaining a finite covering graph compared to the use of the backward procedure which computes $Pre^*(\uparrow s)$ where $\uparrow s$ is the upward closure of the state s.

- We may automatically find some invariants while we need to divine them if we only use the backward procedure.
- We may find some infinite loops in the system, hence we can solve some liveness properties.
- We are able to find bounded places. They correspond to control states; hence we can construct the counter

*Lab. Spécification et Vérification, CNRS UMR 8643, ENS de Cachan, 61, av. du Président Wilson, 94235 Cachan Cedex, France. E-mail: {finkel,leroux}@lsv.ens-cachan.fr

automaton associated with an entropic broadcast protocol, and sometimes, by using [3], we can exactly compute the semilinear reachability set.

In this paper we obtain the main following results:

- The GKM-procedure terminates for all entropic broadcast protocols.
- The problem of deciding whether an entropic broadcast protocol may exhibit an infinite behavior is decidable.
- Model checking liveness properties is undecidable.

The paper is organized as follows: Section 2 introduces broadcast protocols and defines liveness problem, Section 3 gives a proof of the termination of the GKM-procedure for entropic broadcast protocols, Section 4 shows that the model-checking of liveness properties is undecidable.

2. Broadcast Protocols

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 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 constraint : for each state s and each broadcast label $a??$ there is exactly one state s' such that $s \xrightarrow{a??} s'$ (determinism).

Remark 2.1 *To make figure easy reading, we don't always show the transitions of the form $s \xrightarrow{a??} s$ (If from a state s, we don't show a reception broadcast $a??$, then the reception $s \xrightarrow{a??} s$ is implicit).*

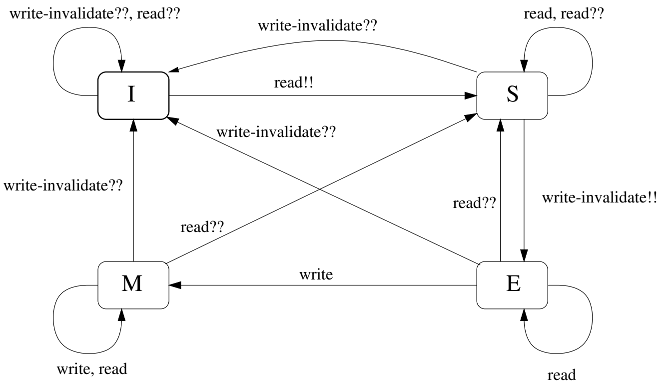

Figure 1. The MESI protocol

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:

$$\mathbf{c}_1 = \mathbf{c} - \mathbf{u}_i \quad (1)$$

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

$$\mathbf{c}' = \mathbf{c}_2 + \mathbf{u}_j \quad (3)$$

  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 by ordering the state by (M,E,S,I).

$$(3,1,2,5) \xrightarrow{read} (0,0,7,4)$$
$$(3,1,2,5) \xrightarrow{write-inv} (0,1,0,10)$$

Given a broadcast protocol with n states, we call the $n \times n$ matrices having unit vectors as columns *broadcast matrices* [6]. 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 $a = read$ in the example of Figure 1 we have

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

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 $\mathbf{c}$. 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 [12]), 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 [6] 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 [6] 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. A Finite Covering Tree

There exists a broadcast protocol for which the GKM does not terminate [4]. Here, we present a class of broadcast protocols for which the procedure terminates and which contains the MESI Protocol (Figure 1). The GKM-procedure for the construction of the covering tree is described below.

3.1 The GKM-procedure

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 $n \in \mathbb{N}$, $\omega + \omega = \omega$. 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 point-wise 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 GKM-procedure examines pairs $(\mathbf{e}, a)$, where $\mathbf{e}$ is an ω-configuration and $a \in \Sigma$. It is initialized with the empty tree 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 GKM-procedure goes as follows:

0. Add the node $\mathbf{e}_0$ to the tree.
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 tree 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 $\mathbf{d} \preceq T_{\sigma a}(\mathbf{d}) \preceq T^2_{\sigma a}(\mathbf{d}) \preceq \ldots$, and make $\mathbf{l}$ the a-successor of $\mathbf{e}$;
   - 3.2 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 sequence effectively computable? and, (b) does the procedure terminate, i.e., is the covering tree finite? In [6], 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). In[4] it is proved that the GKM-procedure may not terminate for broadcast protocols.

3.2 Analyzing Entropic Broadcast Protocols

We present a class, inspired by the MESI-protocol, showing that the procedure terminates for it.

Formally, a broadcast protocol $B = (S, L, R)$ is in the class if it satisfies the two following conditions :

(1.) There is no rendezvous ($\Sigma_r = \emptyset$), and

(2.) every broadcast label $a \in \Sigma_b$ satisfies (2.1) or (2.2)

- (2.1) for each transition $s_i \xrightarrow{a!!} s_j$, the two following transitions exist : $s_i \xrightarrow{a??} s_i$ and $s_j \xrightarrow{a??} s_j$, and,
  for each path $s_i \xrightarrow{a??} s_j \xrightarrow{a??} s_k$, one have $s_j = s_k$.
- (2.2) there exists a state $s_a \in S$ such that for all $p \xrightarrow{a??} q$, we have $q = s_a$.

Condition 2.1 says "Every process which has previously receive or send a broadcast a remains in the same state if it receives an other a" and condition 2.2 says "The broadcast a reboot the broadcast protocol".

Definition 3.1 (Entropic Broadcast Protocol) *We name a broadcast protocol satisfying the previous conditions (1, 2.1, 2.2) an entropic broadcast protocols (The word "entropic" will take it sense later in this text).*

Remark 3.1 *The MESI protocol (Figure 1) is an entropic broadcast protocol. The broadcast action $write - inv$ is of type (2.2) and $read$ is of type (2.1).*

Remark 3.2 *The following broadcast protocol is not entropic (it doesn't satisfy (2.1) because $s_2 \xrightarrow{a??} s_2$ doesn't exists and doesn't satisfy (2.2) because $s_1 \xrightarrow{a??} s_3$ doesn't exist) :*

$$s_1 \xrightarrow{a!!} s_2 \xrightarrow{a??} s_3$$

Remark 3.3 *We can easily simulate a counter with a state and simulate a transfer from a counter s_i to a counter s_j with an entropic broadcast $s_i \xrightarrow{a??} s_j$.*

We now prove that the construction of a finite covering tree is possible for entropic broadcast protocols. We start modifying the line 3.1 to enforce the GKM-procedure just to compute *lub* for *one transition* and not for a path of transitions.

3.1' else, if the tree contains *an arc* from some node $\mathbf{d}$ to $\mathbf{e}$ such that $\mathbf{d} \preceq \mathbf{e}'$, then let $\mathbf{l}$ be the *lub* of the chain $\mathbf{d} \preceq T_a(\mathbf{d}) \preceq T^2_a(\mathbf{d}) \preceq \ldots$, and make $\mathbf{l}$ the a-successor of $\mathbf{e}$;

As for the GKM-procedure, we may prove that this new algorithm also computes a finite covering tree when it terminates [11]. Moreover, we remark that this new algorithm may finish less times than the GKM-procedure. But this algorithm takes advantage of the following result which is more simple to demonstrate. Let's (S,L,R) be an entropic broadcast protocol with $S = \{s_1, ..., s_n\}$.

We introduce a new definition that enables us to prove the following theorem.

Definition 3.2 *Let's $E : \mathbb{N}^n_\omega \longrightarrow \mathbb{N}$ be the function defined for a state $\mathbf{e} \in \mathbb{N}^n_\omega$ by*

$$E(\mathbf{e}) = \sum_{\substack{i = 1...n \\ \mathbf{e}(s_i) \neq \omega}} \mathbf{e}(s_i)$$

We call this function the entropic function because it measure the entropy of a state $\mathbf{e} \in \mathbb{N}^n_\omega$.

Theorem 3.1 *The new algorithm terminates for any entropic broadcast protocol and it computes a finite covering graph.*

The following result will allow us to prove that the nodes of the tree built by the previous algorithm have a decreasing entropy. So, because a sequence of states with decreasing entropy is finite, we will be able to demonstrate that the covering tree is finite.

Proposition 3.1 *Let $\mathbf{e} \in \mathbb{N}_\omega^n$ be an ω-configuration of $B = (S, L, R)$. Let $a \in \Sigma$ be a local action or a broadcast of type (2.1).*

- *If $T_a(\mathbf{e}) > \mathbf{e}$ then $E(lim\, T_a^k(\mathbf{e})) \leq E(\mathbf{e})$.*
- *If $T_a(\mathbf{e})$ and $\mathbf{e}$ are not comparable then $E(T_a(\mathbf{e})) \leq E(\mathbf{e})$.*

Proof:

(1) $a \in \Sigma_l$ is a local action.

In this case, there exists two states $s_i \in S$ and $s_j \in S$ such that $s_i \xrightarrow{a} s_j$ is in R and $\mathbf{e} \xrightarrow{a} \mathbf{e}'$ in B. We have $\mathbf{e}' = T_a(\mathbf{e})$. If $s_i = s_j$ then $\mathbf{e} = \mathbf{e}'$. So we have neither $T_a(\mathbf{e}) > \mathbf{e}$, nor $T_a(\mathbf{e})$ not comparable with $\mathbf{e}$. Therefore, let's suppose $s_i \neq s_j$.
Besides, If $\mathbf{e}(s_j) = \omega$ then we have neither $T_a(\mathbf{e}) > \mathbf{e}$ nor $T_a(\mathbf{e})$ not comparable with $\mathbf{e}$. Therefore, let's suppose $\mathbf{e}(s_j) \in \mathbb{N}$. It remains two cases :

- If $\mathbf{e}(s_i) \in \mathbb{N}$, then $\mathbf{e}$ and $\mathbf{e}'$ are incomparable and the entropy of $\mathbf{e}$ and $\mathbf{e}'$ are the same because from $\mathbf{e}$ to $\mathbf{e}'$, one process move from the state s_i to the state s_j. We have $E(T_a(\mathbf{e})) = E(\mathbf{e})$.
- If $\mathbf{e}(s_i) = \omega$, then $T_a(\mathbf{e}) > \mathbf{e}$. Let $\mathbf{p} = lim\, T_a^n(\mathbf{e})$. We have $\mathbf{p}(s_i) = \omega$ and $\mathbf{p}(s_j) = \omega$. So $E(\mathbf{p}) < E(\mathbf{e})$.

(2) $a \in \Sigma_b$ is a broadcast of type (2.1)

In this case, there exists two states $s_i \in S$ and $s_j \in S$ such that $s_i \xrightarrow{a!!} s_j$ is in R. The only case when entropy may strictly increase is represented by the following (In fact the sum of the entropy of the component other than i and j decrease by the same argument as previous):

$$\begin{cases} s_i \xrightarrow{a!!} s_j \\ \mathbf{e} \xrightarrow{a} \mathbf{e}' \\ \mathbf{e}(s_i) = \omega \\ \mathbf{e}'(s_j) \in \mathbb{N} \end{cases}$$

Remark that only component j of $\mathbf{e}'$ increases the entropy by 1.

Suppose $s_i \neq s_j$, because the case $s_i = s_j$ is much simpler. For simplifying the notations, we fix $i = 1$ and $j = 2$. Let's write an ω-configuration $\mathbf{e}$ as $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ where $\mathbf{e}_1 \in \mathbb{N}_\omega^2$ and $\mathbf{e}_2 \in \mathbb{N}_\omega^{n-2}$ (So e_1 contains the two first components of $\mathbf{e}$ corresponding to s_i and s_j). The previous remark shows that $E(\mathbf{e}_2) \leq E(\mathbf{e}_2')$.

Besides, for all $k \geq 1$, the definition of a broadcast action of type (2.1) (such a broadcast may not consecutively move a process two times) give us $[T_{a^k}(\mathbf{e})]_2 = [T_a(\mathbf{e})]_2$. So, the variation of entropy from $\mathbf{e}_1$ to $\mathbf{e}'_1$ is not modified if we replace the broadcast action a by a local action. Therefore, the result comes from the case local actions.

■

Let us suppose that the new algorithm does not terminate, then the tree built by the algorithm would be infinite. So, as the tree is finite branching, it contains an infinite path. Let us write P such a path.

We remark that after a broadcast action of type (2.2), we can obtain at most two ω-configurations $\mathbf{e}_1$ and e_2 such that:

$\mathbf{e}_1$: all the components are equal to zero, but one ω.

$\mathbf{e}_2$: all the component are equal to zero, but two : one is equal to one and the other is equal to ω.

So as the path P may not have two nodes with the same ω-configuration, we can extract from P an infinite sub-path P' which contains no broadcast of type (2.2). But, the previous proposition shows us that the sequence of the entropy of the nodes of P' is decreasing. So there exists a constant M such that the entropy of all nodes of P' is less than M.

We remark that the number of ω-configurations with entropy less than M is finite. So, there is two distinct nodes in P' which have the same ω-configuration. It's absurd by construction of the algorithm.

So we obtain :

Theorem 3.2 *The new algorithm terminates for any entropic broadcast protocol and it computes a finite covering graph.*

4. Undecidability of Liveness

In fact, we prove that the model checking of liveness properties is undecidable for entropic broadcast protocols. The key idea is to simulate any broadcast by two entropic broadcast actions.

Then, by using [4], we conclude

Theorem 4.1 ([11]) *The model checking of liveness properties for entropic broadcast protocols is undecidable*

Conclusion

Without modifying the above results, we can extend the class of the entropic broadcast protocols by adding some weak rendezvous and some weak broadcast [[11]].

- (weak rendezvous a)

  if there exits a transition $s_i \xrightarrow{a!} s_j$ then $i = j$.

- (weak broadcast a)

  if there exits a transition $s_i \xrightarrow{a!!} s_j$ then $i = j$.

To do this, we can prove that the proposition 3.1 is true for the weak rendezvous and the weak broadcasts.

References

[1] G. Delzanno, J. Esparza, and A. Podelski. Constraint-based analysis of broadcast protocols. In Proceedings of CSL '99, Lecture Notes in Computer Science.

[2] A.Finkel, L.Rosier. A survey on the decidability questions for classes of fifo nets. Advances in Petri Nets, LNCS 340, 1988.

[3] A. Finkel and G. Sutre. Decidability of reachability problems for classes of two counters automata. In *Proc. 17th Ann. Symp. Theoretical Aspects of Computer Science (STACS'2000), Lille, France, Feb. 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 346–357. Springer, 2000.

[4] J. Esparza, A. Finkel, R. Mayr. On the Verification of Broadcast Protocols. LICS, 1999.

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

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

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

[8] 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.

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

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

[11] J. Leroux Vérification des systèmes paramétrés. Mémoire de DEA, LSV 1999.

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