Parameterized Verification of Broadcast Networks of Register Automata

Giorgio Delzanno[1], Arnaud Sangnier[2], and Riccardo Traverso[1]

[1] DIBRIS, University of Genova, Italy
[2] LIAFA, Univ Paris Diderot, Paris Cité Sorbonne, CNRS, France

Abstract. We study parameterized verification problems for networks of interacting register automata. We consider safety properties expressed in terms of reachability, from arbitrarily large initial configurations, of a configuration exposing some given control states and patterns.

1 Introduction

We introduce a formal model of data-sensitive distributed protocols, called Broadcast Networks of Register Automata (BNRA), aimed at modelling both the local knowledge of distributed nodes as well as their interaction via broadcast communication. A network is modelled via a finite graph where each node runs an instance of a common protocol. A protocol is specified via a register automaton, an automaton equipped with a finite set of registers [20]. Each register assumes values taken from the set of natural numbers. Node interaction is specified via broadcast communication, well-suited to model scenarios in which individual nodes have partial information about the network topology. Messages are allowed to carry data, that can be assigned to or tested against the local registers of receivers. Dynamic updates of the current configuration are modelled via non-deterministic reconfigurations of the underlying connectivity graph. A node may disconnect from its neighbours and connect to other ones at any time of the execution. This behaviour models in a natural way unexpected power-off and dynamic movement of devices. The resulting model can be used to reason about core parts of client-server protocols as well as of routing protocols, e.g. route maintainance as in Link Reversal Routing.

In the paper we focus our attention on the decidability and complexity of parameterized verificatiom, i.e., the problem of finding a sufficient number of nodes and an initial topology that may lead to a configuration exposing a bad pattern (e.g. a loop in the information contained in the routing tables). The considered class of verification problems is parametric in four dimensions, namely, the number of nodes, the topology of the initial configuration to be discovered, and the amount of data contained in local registers and exchanged messages.

Related Works. Our formal model of topology-sensitive broadcast communication with data naturally extends those obtained in [11,12,10]. Formal models of broadcast networks date back to CBS [22], extended in several ways (time,

P.A. Abdulla and I. Potapov (Eds.): RP 2013, LNCS 8169, pp. 109–121, 2013.

asynchrony, etc) in successive works. Automated verification methods have been tested on protocols for Ad Hoc Networks with a fixed number of processes in [16,25,15]. Verification of broadcast protocols in fully connected networks in which nodes and messages range over a finite set of states has been considered, e.g., in [13,18,5]. Via an adequate counting abstraction, the problem can be reformulated in terms of Petri nets with transfer arcs [14,7]. The non-elementary complexity of coverability in this class of nets is proved in [24]. Symbolic backward exploration procedures for network protocols specified in graph rewriting have been presented in [19] (termination guaranteed for ring topologies) and [23] (approximations without termination guarantees). Decidability issues for broadcast communication in fully connected networks have been studied in [14]. Verification of unreliable communicating FIFO systems has been studied in [3]. Coverability problems for broadcast communication in fully connected networks with data is investigated in [2,21,8].

2 Broadcast Networks of Register Automata

2.1 Syntax and Semantics

We model a distributed network using a graph in which the behaviour of each node is described via an automaton with operations over a finite set of registers. A node can transmit part of its current data to adjacent nodes using broadcast messages. A message carries both a type and a finite tuple of data. Receivers can test/store/ignore the data contained inside a message. We assume that broadcasts and receptions are executed without delays (i.e. we simultaneously update the state of sender and receiver nodes).

Actions. Let us first describe the set of actions. We use $r \geq 0$ to denote the number of registers in each node. We use $f \geq 0$ to denote the number of data fields available in each message and we consider a finite alphabet Σ of message types. We often use $[i..j]$ to denote the set $\{k \in \mathbb{N} \mid i \leq k \leq j\}$. We also assume that if $r = 0$ then $f = 0$ (no registers, no information to transmit). The set of broadcast actions parameterized by r, f and Σ is defined follows:

$$Send_{\Sigma}^{r,f} = \{\mathbf{b}(m, p_1, \ldots, p_f) \mid m \in \Sigma \text{ and } p_i \in [1..r] \text{ for } i \in [1..f]\}$$

The action $\mathbf{b}(a, p_1, \ldots, p_f)$ corresponds to a broadcast message of type a whose i-th field contains the value of the register p_i of the sending node. For instance, for $r = 2$ and $f = 4$, $\mathbf{b}(req, 1, 1, 2, 1)$ corresponds to a message of type req in which the current value of the register 1 of the sender is copied in the first two fields and in the last field, and the current value of register 2 of the sender is copied into the third field.

A receiver node can then either compare the value of a message field against the current value of a register, store the value of a message field in a register, or simply ignore a message field. Reception actions parameterized by r, f and Σ

are defined as follows:

$$Rec_{\Sigma}^{r,f} = \left\{ \mathbf{r}(m, \alpha_1, \ldots, \alpha_f) \;\middle|\; \begin{array}{l} m \in \Sigma,\ \alpha_i \in Act^r \text{ for } i \in [1..f] \\ \text{and if } \alpha_i = \alpha_k = \downarrow j \text{ then } i = k \end{array} \right\}$$

where the set of field actions Act^r is: $\{?k, ?\overline{k}, \downarrow k, * \mid k \in [1..r]\}$. When used in a given position of a reception action, $?k$ [resp. $?\overline{k}$] tests whether the content of the k-th register is equal [resp. different] to the corresponding value of the message, $\downarrow k$ is used to store the corresponding value of the message into the k-th register, and $*$ is used to denote that the corresponding value is ignored.

As an example, for $r = 2$ and $f = 4$, $\mathbf{r}(req, ?\overline{2}, ?1, *, \downarrow 1)$ specifies the reception of a message of type req in which the first field is tested for inequality against the current value of the second register, the second field is tested for equality against the first register, the third field is ignored, and the fourth field is assigned to the first register. We now provide the definition of a protocol that models the behaviour of an individual node.

Definition 1. *A (r, f)-*protocol *over Σ is a tuple $\mathcal{P} = \langle Q, R, q_0 \rangle$ where: Q is a finite set of control states, $q_0 \in Q$ is an initial control state, and $R \subseteq Q \times (Send_{\Sigma}^{r,f} \cup Rec_{\Sigma}^{r,f}) \times Q$ is a set of broadcasting and reception rules.*

In the rest of the paper we call a (r, f)-protocol over Σ simply a (r, f)-protocol when the alphabet is clear from the context.

A configuration is a graph in which nodes represent the current state of the corresponding protocol instance running on it (control state and current value of registers) and edges denote communication links. In this paper we assume that the value of registers are naturals. Therefore, a valuation of registers is defined as a map from register positions to naturals. More formally, a configuration γ of a (r, f)-protocol $\mathcal{P} = \langle Q, R, q_0 \rangle$ is an undirected graph $\langle V, E, L \rangle$ such that V is a finite set of nodes, $E \subseteq V \times V \setminus \{(v, v) \mid v \in V\}$ is a set of edges, and $L : V \to Q \times \mathbb{N}^r$ is a labelling function (current valuation of registers).

Before we give the semantics of our model, we introduce some auxiliary notations. Let $\gamma = \langle V, E, L \rangle$ be a configuration. For a node $v \in V$, we denote by $L_Q(v)$ and $L_M(v)$ the first and second projection of $L(v)$. For $u, v \in V$, we write $u \sim_\gamma v$ – or simply $u \sim v$ when γ is clear from the context – the fact that $(u, v) \in E$, i.e. the two nodes are neighbours. Finally, the configuration γ is said to be initial if $L_Q(v) = q_0$ for all $v \in V$ and, for all $u, v \in V$ and all $i, j \in [1..r]$, if $u \neq v$ or $i \neq j$ then $L_M(v)[i] \neq L_M(v)[j]$. In an initial configuration, all the registers of the nodes contain different values. We write Γ [resp. Γ_0] for the set of all [resp. initial] configurations, and Γ^{fc} [resp. Γ_0^{fc}] for the set of configurations [resp. initial configurations] $\langle V, E, L \rangle$ that are fully connected, i.e. such that $E = V \times V \setminus \{(v, v) \mid v \in V\}$. Note that for a given (r, f)-protocol the sets Γ, Γ_0, Γ^{fc}, and Γ_0^{fc} are infinite since we do not impose any restriction on the number of processes present in the graph.

Furthermore, from two nodes u and v of a configuration $\gamma = \langle V, E, L \rangle$ and a broadcast action of the form $\mathbf{b}(m, p_1, \ldots, p_f)$, let $\mathcal{R}(v, u, \mathbf{b}(m, p_1, \ldots, p_f)) \subseteq Q \times \mathbb{N}^r$ be the set of the possible labels that can take u on reception of the corresponding message sent by v, i.e. we have $(q'_r, M) \in \mathcal{R}(v, u, \mathbf{b}(m, p_1, \ldots, p_f))$ if and only

if there exists a receive action of the form $\langle L_Q(u), \mathbf{r}(m, \alpha_1, \ldots, \alpha_f), q'_r \rangle \in R$ verifying the two following conditions:

(1) For all $i \in [1..f]$, if there exists $j \in [1..r]$ s.t. $\alpha_i = ?j$ [resp. $\alpha_i = ?\overline{j}$], then $L_M(u)[j] = L_M(v)[p_i]$ [resp. $L_M(u)[j] \neq L_M(v)[p_i]$];
(2) For all $j \in [1..r]$, if there exists $i \in [1..f]$ such that $\alpha_i = \downarrow j$ then $M[j] = L_M(v)[p_i]$ otherwise $M[j] = L_M(u)[j]$.

Given a (r, f)-protocol $\mathcal{P} = \langle Q, R, q_0 \rangle$, we define a Broadcast Network of Register Automata (BNRA) as the transition system $BNRA(\mathcal{P}) = \langle \Gamma, \Rightarrow, \Gamma_0 \rangle$ where Γ [resp. Γ_0] is the set of all [resp. initial] configurations and $\Rightarrow \subseteq \Gamma \times \Gamma$ is the transition relation. Specifically, for $\gamma = \langle V, E, L \rangle$ and $\gamma' = \langle V', E', L' \rangle \in \Gamma$, we have $\gamma \Rightarrow \gamma'$ if and only if $V = V'$ and one of the following conditions holds:

(Broadcast) $E = E'$ and there exist $v \in V$ and $\langle q, \mathbf{b}(m, p_1, \ldots, p_f), q' \rangle \in R$ such that $L_Q(v) = q$, $L'_Q(v) = q'$ and for all $u \in V \setminus \{v\}$:
- if $u \sim v$ then $L'(u) \in \mathcal{R}(v, u, \mathbf{b}(m, p_1, \ldots, p_f))$, or, $\mathcal{R}(v, u, \mathbf{b}(m, p_1, \ldots, p_f)) = \emptyset$ and $L(u) = L'(u)$;
- if $u \nsim v$, then $L(u) = L'(u)$.

(Reconfiguration) $L = L'$ (no constraint on new edges E').

Reconfiguration steps model dynamic changes of the connection topology, e.g., loss of links and messages or node movement. An internal transition τ can be defined using a broadcast of a special message such that there are no reception rules associated to it. A register $j \in [1..r]$ is said to be read-only if and only if there is no $\langle q, \mathbf{r}(m, \alpha_1, \ldots, \alpha_f), q' \rangle \in R$ and $i \in [1..f]$ such that $\alpha_i = \downarrow j$. Read-only registers can be used as identifiers of the associated nodes.

Given $BNRA(\mathcal{P}) = \langle \Gamma, \Rightarrow, \Gamma_0 \rangle$, we use $\Rightarrow_b$ to denote the restriction of $\Rightarrow$ to broadcast steps only, and $\Rightarrow^*$ [resp. $\Rightarrow_b^*$] to denote the reflexive and transitive closure of $\Rightarrow$ [resp. $\Rightarrow_b$]. Now we define the set of reachable configurations as: $Reach(\mathcal{P}) = \{\gamma' \in \Gamma \mid \exists \gamma \in \Gamma_0 \text{ s.t. } \gamma \Rightarrow^* \gamma'\}$, $Reach^b(\mathcal{P}) = \{\gamma' \in \Gamma \mid \exists \gamma \in \Gamma_0 \text{ s.t. } \gamma \Rightarrow_b^* \gamma'\}$, and $Reach^{fc}(\mathcal{P}) = Reach^b(\mathcal{P}) \cap \Gamma^{fc}$.

2.2 Coverability Problem

Our goal is to decide whether there exists an initial configuration (of any size and topology) from which it is possible to reach a configuration exposing (covered by w.r.t. graph inclusion) a bad pattern. We express bad patterns using reachability queries defined as follows. Let $\mathcal{P} = \langle Q, R, q_0 \rangle$ be a (r, f)-protocol and Z a denumerable set of variables. A reachability query φ for $\mathcal{P}$ is a formula generated by the following grammar:

$$\varphi ::= q(\mathbf{z}) \mid M_i(\mathbf{z}) = M_j(\mathbf{z}') \mid M_i(\mathbf{z}) \neq M_j(\mathbf{z}') \mid \varphi \wedge \varphi$$

where $\mathbf{z}, \mathbf{z}' \in Z$, $q \in Q$ and $i, j \in [1..r]$. We now define the satisfiability relation for such queries. Given a configuration $\gamma = \langle V, E, L \rangle \in \Gamma$, a valuation is a function $f : Z \mapsto V$. The satisfaction relation $\models$ is parameterized by a valuation and is defined inductively as follows:

- $\gamma \models_f q(\mathbf{z})$ if and only if $L_Q(f(\mathbf{z})) = q$,
- $\gamma \models_f M_i(\mathbf{z}) = M_j(\mathbf{z}')$ if and only if $L_M(f(\mathbf{z}))[i] = L_M(f(\mathbf{z}'))[j]$,
- $\gamma \models_f M_i(\mathbf{z}) \neq M_j(\mathbf{z}')$ if and only if $L_M(f(\mathbf{z}))[i] \neq L_M(f(\mathbf{z}'))[j]$,
- $\gamma \models_f \varphi \wedge \varphi'$ if and only if $\gamma \models_f \varphi$ and $\gamma \models_f \varphi'$.

We say that a configuration γ satisfies a reachability query φ, denoted by $\gamma \models \varphi$ if and only if there exists a valuation f such that $\gamma \models_f \varphi$. Furthermore we assume that our queries do not contain contradictions w.r.t. $=$ and $\neq$. We now define the parameterized verification problem, i.e., finding an initial configuration that leads to a configuration containing a sub-configuration that matches the query.

Definition 2. *The problem $Cov(r, f)$ is defined as follows: given a (r, f)-protocol $\mathcal{P}$ and a reachability query φ, does there exist $\gamma \in Reach(\mathcal{P})$ such that $\gamma \models \varphi$?*

The problem $Cov^b(r, f)$ [resp. $Cov^{fc}(r, f)$] is obtained by replacing the reachability set with $Reach^b(\mathcal{P})$ [resp. $Reach^{fc}(\mathcal{P})$]. Finally, $Cov(*, f)$ denotes the disjunction of the problems $Cov(r, f)$ varying on $r \geq 0$ (i.e. for any (finite) number of registers).

3 An Example: Route Discovery Protocol

Consider the problem of building a route from nodes of type *sender* to nodes of type *dest*. We assume that nodes have two registers, called *id* and *next*, used to store a pointer to the next node in the route to *dest*. The protocol that collects such information is defined in Figure 1. Initially nodes have type *sender*, *idle*, and *dest*. Request messages like *rreq* are used to query adjacent nodes in search

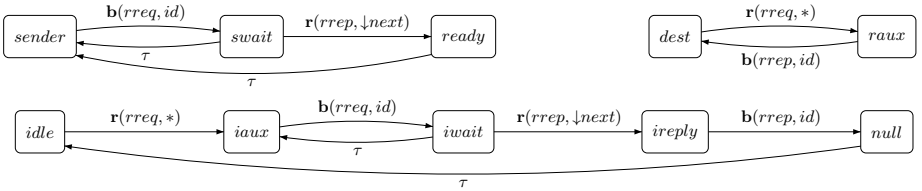

Fig. 1. Route discovery example

for a valid neighbour. Back edges are used to restart the protocol in case of loss of intermediate messages or no reply at all.

In this example an undesired state is, e.g., any configuration in which two adjacent nodes n and n' point to each other. Bad patterns like this one can be specified using a query like $ready(z_1) \wedge ready(z_2) \wedge M_{id}(z_1) = M_{next}(z_2) \wedge M_{next}(z_1) = M_{id}(z_2)$.

4 Reconfiguration in Arbitrary Graphs

4.1 Undecidability of $Cov(2, 2)$

Our first result is the undecidability of coverability for nodes with two registers (one read-only) and messages with two data fields. The proof is based on a

reduction from reachability in two counter machines. The reduction builds upon an election protocol that can be applied to select a linked list (of arbitrary length) of nodes in the network. The existence of such a list-builder protocol is at the core of the proof. The simulation of a two counter machine becomes easy once a list has been constructed. We assume that protocols have at least one read-only register $id \in [1..r]$. We formalize next the notion of list and list-builder that we use in the undecidability proofs presented across the paper. We first say that a node v points to a node v' via x if the register x of v contains the same value as register id of v'. For $q_a, q_b, q_c \in Q$, a list (linked via x) is a set of nodes $\{v_1, \cdots, v_k\}$ such that v_1 has label q_a, v_k has label q_c, v_i has label q_b for $i \in [2..k-1]$, and v_j is the unique node in V that points to v_{j+1} via x and has label in $\{q_a, q_b\}$ for $j \in [0..k-1]$. In other words q_a and q_c are sentinels for a list made of q_b elements. A backward list is defined as before but with reversed pointers, i.e., v_{j+1} points to v_j.

Definition 3. *A protocol $\mathcal{P} = \langle Q, R, q_0 \rangle$ with $\{q_a, q_b, q_c\} \subseteq Q$ is a list-builder for q_a, q_b, and q_c on $x \in [1..r]$ if, for any γ such that $\gamma \in Reach(\mathcal{P})$, if a node v in γ has label q_a, then v is the first node of a list linked via x.*

A backward list-builder is defined in a similar way for backward lists.

Lemma 1. *For $r \geq 2$ and $f \geq 1$, $Cov(r, f)$ is undecidable if there exists a list-builder (r, f)-protocol on $x \in [1..r]$ that can generate lists of any finite length.*

The proof exploits the list (of arbitrary length) generated by a list-builder protocol to build a simulation of a two counter machine. Indeed, notice that if node v is the only one pointing to node v' then the pair of actions $\mathbf{b}(m, x)$ and $\mathbf{r}(m, ?id)$ can be used to send a message from v to v' (v' is the only node that can receive m from v). Furthermore, the pair of actions $\mathbf{b}(m, id)$ and $\mathbf{r}(m, ?x)$ can be used to send a message from v' to v (v is the only node that can receive m from v'). This property can be exploited to simulate counters by using intermediate nodes as single units (the value of the counter is the sum of unit nodes in the list). One of the sentinels is used as program location, and the links in the list are used to send messages (in two directions) to adjacent nodes to increment or decrement (update of labels) the counters. Test for zero is encoded by a double traversal of the list in order to check that each intermediate node represents zero units. The details of the protocol that extends a list-builder are given in [9]. A similar result can be stated for backward list-builders.

The previous lemma tells us that to prove undecidability of coverability we just have to exhibit a list-builder protocol. In the case of $Cov(2, 2)$, we apply Lemma 1, by showing that protocol $\mathcal{P}_{lb}$ of Figure 2 is a backward list-builder for q_h, q_z, and q_t on $x \in [1..r]$. The rationale is as follows. Lists $\{v_1, \cdots, v_k\}$ are built one node at a time, starting from the tail v_k, in state q_t. The links point from each node to the previous one, up to the head v_1, in state q_h. Any node in the initial state q_0 (e.g., v_1) may decide to become a tail by starting to build its own list. Every such construction activity, however, is guaranteed not to interfere in any way with the others, thanks to point to point communication between nodes simulated on top of network reconfigurations and broadcast by exploiting

the two payload fields. This is achieved via a three-way handshake where the first and second fields respectively identify the sender and the recipient. When the sub-protocol is done, v_1 moves to state q_t, v_2 moves to the intermediate state q_i, and one points to the other. Node v_2 decides whether to stop building the list by becoming the head q_h, or to continue by executing another handshake to elect node v_3. The process continues until some v_k finally ends the construction by moving to state q_h. The following theorem then holds (the proof can be found in [9]).

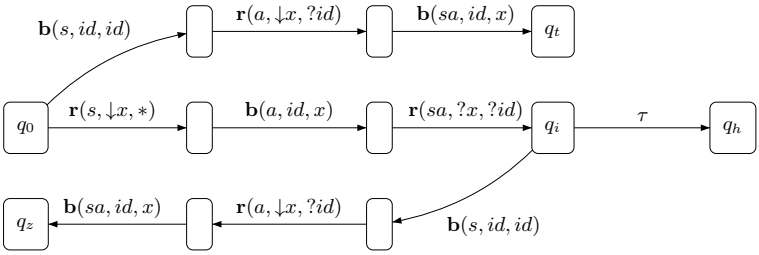

Fig. 2. $\mathcal{P}_{lb}$: backward list-builder for q_h, q_z, q_t, and Γ_0 on x

Theorem 1. *$Cov(2, 2)$ is undecidable even when restricting one register to be read-only.*

4.2 Decidability of $Cov(*, 1)$

In this section, we will prove that $Cov(*, 1)$, i.e. the restriction of our coverability problem to processes with only one field in the message, is PSPACE-complete.

We obtain PSPACE-hardness through a reduction from the reachability problem for 1-safe Petri nets, which is PSPACE-complete [6]. The detail of this construction is provided in [9].

Proposition 1. *$Cov(*, 1)$ is* PSPACE*-hard.*

We now provide a PSPACE algorithm for solving $Cov(*, 1)$. The algorithm is based on a saturation procedure that computes a symbolic representation of reachable configurations. The representation is built using graphs that keep track of control states that may appear during a protocol execution and of relations between values in their registers. The set of symbolic configurations we consider is finite and each symbolic configuration can be encoded in polynomial space.

Assume a $(r, 1)$-protocol $\mathcal{P} = \langle Q, R, q_0 \rangle$ over Σ. A symbolic configuration θ for $\mathcal{P}$ is a labelled graph $\langle W, \delta, \lambda \rangle$ where W is a set of nodes, $\delta \subseteq W \times [1..r] \times [1..r] \times W$ is the set of labelled edges and $\lambda : W \mapsto Q \times \{0, 1\}^r$ is a labelling function (as for configurations, we will denote λ_Q [resp. λ_M] the projection of λ to its first [resp. second] component) such that the following rules are respected:

- For $w, w' \in W$, $w \neq w'$ implies $\lambda_Q(w) \neq \lambda_Q(w')$, i.e. there cannot be two nodes with the same control state;

- If $(w, a, b, w') \in \delta$ then $\lambda_M(w)[a] = 1$ or $\lambda_M(w')[b] = 1$ (or both);
- For $w \in W$ and $j \in [1..r]$, if $\lambda_M(w)[j] = 1$ then $(w, j, j, w) \in \delta$.

The labels $\{0,1\}^r$ are redundant (they can be derived from edges) but simplify some of the constructions needed in the algorithm. We denote by Θ the set of symbolic configurations for $\mathcal{P}$. Let $\theta = \langle W, \delta, \lambda \rangle$ be a symbolic configuration for $\mathcal{P}$. Then, $\langle V, E, L \rangle \in [\![\theta]\!]$ iff the following conditions are satisfied:

1. For each $v \in V$, there is a node $w \in W$ such that $L_Q(v) = \lambda_Q(w)$, i.e. v and w have the same control state;
2. For each $v \neq v' \in V$, if there exist registers $j, j' \in [1..r]$ s.t. $L_M(v)[j] = L_M(v')[j']$, i.e., two distinct nodes with the same value in a pair of registers, then there exists an edge $(w, j, j', w') \in \delta$ with $\lambda_Q(w) = L_Q(v)$ and $\lambda_Q(w') = L_Q(v')$, i.e. we store possible relations on data in registers using edges in θ;
3. For each $v \in V$, if there exist $j \neq j' \in [1..r]$ s.t. $\lambda_M(v)[j] = \lambda_M(v)[j']$, i.e. a node with the same value in two distinct registers, then there exists a self loop $(w, j, j', w) \in \delta$.

We remark that we do not include any information on the communication links of γ, indeed reconfiguration steps can change the topology in an arbitrary way. We define the initial symbolic configuration $\theta_0 = \langle \{w_0\}, \emptyset, \lambda_0 \rangle$ with $\lambda_0(w_0) = (q_0, \mathbf{0})$. Clearly, we have $[\![\theta_0]\!] = \Gamma_0$, i.e. the set of concrete configurations represented by θ_0 is the set of initial configurations of the protocol $\mathcal{P}$. In order to perform a symbolic reachability on symbolic configurations, we use a operator $\texttt{POST}_{\mathcal{P}}$ that, by working on a graph θ simulates the effect of the application of a broadcast rule on its instances $[\![\theta]\!]$. The formal definition of the $\texttt{POST}_{\mathcal{P}}$ operation is given in [9]. We illustrate the key points underlying its definition with the help of an example. Consider the symbolic configurations θ_1 and θ_2 in Figure 3, where we represent edges $(w, a, b, w') \in \delta$ with arrows from w to w' labelled by a, b. Please note that, even though we use directed edges for the graphical representation, the relation between nodes in W symmetrical as $(w, a, b, w') \in \delta$ is equivalent to (w', b, a, w). θ_1 denotes configurations with any number of nodes with label q_0 or q_1. Nodes in state q_0 must have registers containing distinct data (label $0, 0$). Nodes in state q_1 may have the same value in their second register (label $0, 1$ is equivalent to edge $\langle q_1, 2, 2, q_1 \rangle$), that in turn may be equal to the value

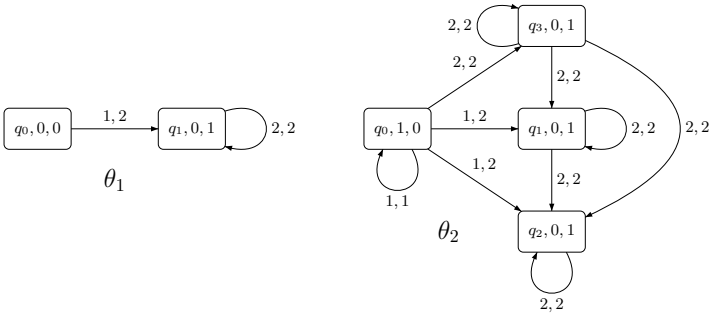

Fig. 3. Example of computations of symbolic post

of the first register in a node labelled q_0 (edge $\langle q_0, 1, 2, q_1\rangle$). θ_1 can be obtained from the initial symbolic configuration by applying rules like $\langle q_0, \mathbf{b}(\alpha, 1), q_0\rangle$ and $\langle q_0, \mathbf{r}(\alpha, \downarrow 2), q_1\rangle$. Indeed, in q_0 we can send the value of the first register to other nodes in q_0 that can then move to q_1 and store the data in the second register (i.e. we create a potential data relation between the first and second register).

We now give examples of rules that can generate the symbolic configuration θ_2 starting from θ_1. The pair $\langle q_0, \mathbf{b}(\beta, 1), q_0\rangle$ and $\langle q_0, \mathbf{r}(\beta, \downarrow 1), q_0\rangle$ generates a new data relation between nodes in state q_0 modelled by changing from 0 to 1 the value of $\lambda_M(q_0)[1]$. We remark that a label 1 only says that registers in distinct nodes may be (but not necessarily) equal.

Consider now the reception rule $\langle q_1, \mathbf{r}(\beta, ?2), q_2\rangle$ for the same message β. The data relation between nodes in state q_0 and q_1 in θ_1 tells us that the rule is fireable. To model its effect we need to create a new node with label q_2 with data relations between registers expressed by the edges between labels q_0, q_1 and q_2 in the figure. Due to possible reconfigurations, not all nodes in q_1 necessarily react, i.e. θ_2 contains the denotations of θ_1.

A rule like $\langle q_1, \mathbf{r}(\beta, ?\overline{2}), q_3\rangle$ can also be fireable from instances of θ_1. Indeed, the message β can be sent by a node in state q_0 that does not satisfy the data relation specified by the edge $(1, 2)$ in θ_1, i.e., the sending node is not the one having the same value in its first register as the node q_1 reacting to the message, hence the guard $?\overline{2}$ could also be satisfied. This leads to a new node with state q_3 which inherits from q_1 the constraints on the first register, but whose second register can have the same value as the second register of nodes in any state.

We now define how to evaluate a reachability query over a symbolic configuration . Let $\theta = \langle W, \delta, \lambda\rangle$ be a symbolic configuration and φ be a reachability query. We denote by $\mathit{Vars}(\varphi)$ the subset of variables used in the query φ and we assume that $\varphi = \bigwedge_{k\in[1..m]} \varphi_k$ where for each $k \in [1..m]$, φ_k is of the form $q(\mathbf{z})$ or $M_i(\mathbf{z}) = M_j(\mathbf{z}')$ or $M_i(\mathbf{z}) \neq M_j(\mathbf{z}')$. We will then say that $\theta \models \varphi$ if there exists a function $g : \mathit{Vars}(\varphi) \mapsto W$ such that for all $k \in [1..m]$ we have the following properties: if $\varphi_k = q(\mathbf{z})$, then $\lambda_Q(g(\mathbf{z})) = q$; if $\varphi_k = (M_i(\mathbf{z}) = M_j(\mathbf{z}'))$ with $\mathbf{z} \neq \mathbf{z}'$ or $i \neq j$, then $(g(\mathbf{z}), i, j, g(\mathbf{z}')) \in \delta$. We have then the following lemma.

Lemma 2. *Given a symbolic configuration θ and a reachability query φ, we have $\theta \models \varphi$ if and only there exists $\gamma \in [\![\theta]\!]$ such that $\gamma \models \varphi$.*

Before giving the properties of the $\mathtt{POST}_{\mathcal{P}}$ operator, we introduce some notations. First we introduce an order on symbolic configurations. Given two symbolic configurations $\theta = \langle W, \delta, \lambda\rangle$ and $\theta' = \langle W', \delta', \lambda'\rangle$, we say that $\theta \sqsubseteq \theta'$ if and only if there exists an injective function $h : W \mapsto W'$ such that for all $w, w' \in W$:

- $\lambda_Q(w) = \lambda'_Q(h(w))$;
- for all $j \in [1..r]$, if $\lambda_M(w)[j] = 1$ then $\lambda'_M(h(w))[j] = 1$;
- if $(w, a, b, w') \in \delta$ then $(h(w), a, b, h(w')) \in \delta'$.

In other words, we have $\theta \sqsubseteq \theta'$ if there are more nodes in θ' than in θ and all the labels of θ appears in θ' as well, and for what concerns the symbolic register valuation, the one of θ' should "cover" the one of θ. One can easily prove the following result.

Lemma 3. *(1) If $\theta \sqsubseteq \theta'$ then $[\![\theta]\!] \subseteq [\![\theta']\!]$. (2) If there exists an infinite increasing sequence $\theta_0 \sqsubseteq \theta_1 \sqsubseteq \theta_2 \ldots$ then there exists $i \in \mathbb{N}$ s.t. for all $j \geq i$, $\theta_j = \theta_i$.*

Furthermore, given a set of configurations $S \subseteq \Gamma$ of the $(r,1)$-protocol $\mathcal{P} = \langle Q, R, q_0 \rangle$ (with $BNRA(\mathcal{P}) = \langle \Gamma, \Rightarrow, \Gamma_0 \rangle$), we define $\mathtt{post}_{\mathcal{P}}(S) = \{\gamma' \in \Gamma \mid \exists \gamma \in \Gamma \text{ s.t. } \gamma \Rightarrow \gamma'\}$ and $\mathtt{post}^*_{\mathcal{P}}$ is the reflexive and transitive closure of $\mathtt{post}_{\mathcal{P}}$ (and $\mathtt{POST}^*_{\mathcal{P}}$ the reflexive and transitive closure of $\mathtt{POST}_{\mathcal{P}}$). Note that since symbolic configuration generate a single node for each label, repeated application of $\mathtt{POST}_{\mathcal{P}}$ are ensured to terminate. We can now give the properties of the $\mathtt{POST}_{\mathcal{P}}$ operator.

Lemma 4. *Let θ be a symbolic configuration of the protocol $\mathcal{P}$. Then we have $\theta \sqsubseteq \mathtt{POST}_{\mathcal{P}}(\theta)$ and for all reachability query φ, there exists $\gamma \in \mathtt{post}^*_{\mathcal{P}}([\![\theta]\!])$ such that $\gamma \models \varphi$ iff $\mathtt{POST}^*_{\mathcal{P}}(\theta) \models \varphi$.*

We have consequently an algorithm to solve whether there exists $\gamma \in \mathit{Reach}(\mathcal{P}) = \mathtt{post}_{\mathcal{P}}(\Gamma_0)$. In fact it is enough to compute $\mathtt{POST}^*_{\mathcal{P}}(\theta_0)$ and to check whether $\mathtt{POST}^*_{\mathcal{P}}(\theta) \models \varphi$. This computation is feasible thanks to Lemma 3 and thanks to the first point of the previous lemma. Note that each each symbolic configuration of the $(r,1)$-protocol $\mathcal{P}$ is a graph with at most $|Q|$ nodes and at most $|Q|^2 * |r|^2$ edges and hence we need only polynomial space in the size of the protocol $\mathcal{P}$ to compute $\mathtt{POST}^*_{\mathcal{P}}(\theta_0)$. Finally we can check in non-deterministic linear time whether $\mathtt{POST}^*_{\mathcal{P}}(\theta_0) \models \varphi$ (it is enough to guess the function g from $\mathit{Vars}(\varphi)$ to the nodes of $\mathtt{POST}^*_{\mathcal{P}}(\theta_0)$). Using Lemma 2, this gives us a polynomial space procedure to check whether there exists $\gamma \in \mathit{Reach}\mathcal{P}$ such that $\gamma \models \varphi$. Furthermore, thanks to the lower bound given by Proposition 1, we can deduce the exact complexity of coverability for protocols using a single field in their messages.

Theorem 2. *$\mathit{Cov}(*,1)$ is* PSPACE*-complete.*

5 Fully Connected Topologies and No Reconfiguration

5.1 Undecidability of $\mathit{Cov}^{fc}(2,1)$

We now move to coverability in fully connected topologies. In contrast with the results obtained without identifiers in [11] it turns out that, without reconfiguration, coverability is undecidable already in the case of nodes with two registers and one payload field. Following the same line as in Lemma 1, to prove the result it is enough to define a (forward) list-builder protocol. We refer to Lemma 1* as the variation of Lemma 1 obtained considering the relation $\Rightarrow_b$ (see [9]). The protocol builds lists backwards from the tail q_t. At each step, a node v among the ones which are not part of the list broadcasts its identifier to the others (which store the value, thus pointing to v), and moves to q_z (or q_t, if it is the first step) electing itself as the next node in the list. The construction ends when such a node will instead move to q_h and force everyone else to stop. By applying Lemma 1*, the following theorem then holds (a complete proof is in [9]).

Theorem 3. *$\mathit{Cov}^{fc}(2,1)$ is undecidable even when one register is read-only.*

5.2 Decidability of $Cov^{fc}(1,1)$

We now consider the problem $Cov^{fc}(1,1)$, where configurations are fully connected and do not change dynamically, processes have a single register, and each message has a single data field. To show decidability, we employ the theory of well-structured transition systems [1,17] to define an algorithm for backward reachability based on a symbolic representation of infinite set of configurations, namely multisets of multisets of states in Q. In the following we use $[a_1, \ldots, a_k]$ to denote a multiset containing (possibly repeated) occurrences $a_1, \ldots, a_k$ of elements from some fixed domain. For a multiset m, we use $m(q)$ to denote the number of occurrences of q in m.

Let $\mathcal{P} = \langle Q, R, q_0 \rangle$ be a $(1,1)$-protocol. The set Ξ of symbolic configurations contains, for every $k \in \mathbb{N}$, all multisets of the form $\xi = [m_1, \ldots, m_k]$, where m_i for $i \in [1..k]$ is in turn a multiset over Q. Given $\xi = [m_1, \ldots, m_k] \in \Xi$, $\langle V, E, L \rangle \in [\![\xi]\!]$ iff there is a function $f : V \to [1..k]$ such that (1) for every $v, v' \in V$, if $L_M(v) = L_M(v')$ then $f(v) = f(v')$ and (2) for all $i \in [1..k]$ and $q \in Q$, $m_i(q)$ is equal to the number of nodes $v \in V$ s.t. $f(v) = i$ and $L_Q(v) = q$ Intuitively, each m_i is associated to one of the k distinct values of the register (the actual values do not matter), and $m_i(q)$ counts how many nodes in state q have the corresponding value. We now define an ordering over Ξ.

Definition 4. *Given $\xi = [m_1, \ldots, m_k] \in \Xi$ and $\xi' = [m'_1, \ldots, m'_p] \in \Xi$, $\xi \prec \xi'$ iff $k \leq p$ and there exists an injection $h : [1..k] \to [1..p]$ such that for all $i \in [1..k]$ and all $q \in Q$, $m_i(q) \leq m_{h(i)}(q)$, i.e. m_i is included in $m_{h(i)}$.*

The following properties then hold.

Proposition 2. *The ordering $(\xi, \prec)$ over symbolic configurations is a well-quasi ordering (wqo), i.e. for any infinite sequence $\xi_1 \xi_2 \ldots$ there exist $i < j$ s.t. $\xi_i \prec \xi_j$.*

Proposition 3. *Let $pre_{\mathcal{P}}(S) = \{\gamma \mid \gamma \Rightarrow_b \gamma', \gamma' \in S\}$. There exists an algorithm $\texttt{PRE}_{\mathcal{P}}$ taking in input $I \subseteq \Xi$ and returning a set $I' \subseteq \Xi$ s.t. $[\![I']\!] = pre_{\mathcal{P}}([\![I]\!])$.*

The formal definition of the predecessor operator is given in [9], together with an example. Following [4], the algorithm for $\texttt{PRE}_{\mathcal{P}}$ can be used to effectively compute a finite representation of the set of predecessors $pre^*_{\mathcal{P}}([\![Bad]\!])$ for a set of symbolic configurations Bad. The computation iteratively applies PRE until a fixpoint is reached. The termination test is defined using $\prec$. The wqo $\prec$ ensures termination of the computation [1]. The following theorem then holds.

Theorem 4. *$Cov^{fc}(1,1)$ is decidable.*

An alternative proof can be given by resorting to an encoding into coverability in data nets [21]. We present such an encoding in [9].

We consider now the complexity. We observe that, without registers and fields our model boils down to the AHNs of [11]. For fully connected topologies, AHN can simulate reset nets as shown in [12]. Following from the complexity of coverability in reset nets [24], we have the the following theoretical lower bound.

Corollary 1. *$Cov^{fc}(0,0)$ and $Cov^{fc}(1,1)$ are non elementary.*

Transitions/Topology	r	f	Status	Complexity
B+R,G	0	0	DEC	PTIME
	$k \geq 1$	1	DEC	PSPACE
	2	2	UNDEC	–
B,FC	0	0	DEC	NON EL
	1	1	DEC	NON EL
	2	1	UNDEC	–
B,G	0	0	UNDEC	–

Fig. 4. Decidability and complexity boundaries: B=broadcast transitions, R=reconfiguration, FC=fully connected topologies, and G=arbitrary graphs.

6 Conclusions

We have investigated decidability and complexity for coverability in a data-sensitive model of broadcast communication (Figure 4). From a technical point of view, our results can be viewed as a fine grained refinement of those obtained for the case without data. For instance, undecidability follows from constructions similar to those adopted in [11]. They are based on special use of data for building synchronization patterns that can be applied even in fully connected networks. Concerning possible applications, the symbolic algorithm for messages with a single data field can be applied to abstract models of routing protocols like the protocol of Section 3. Finally, as future extensions it would be interesting to study ordered data fields and time-sensitive communication.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems. In: LICS 1996, pp. 313–321. IEEE Computer Society (1996)
2. Abdulla, P.A., Delzanno, G., Van Begin, L.: A classification of the expressive power of well-structured transition systems. Inf. Comput. 209(3), 248–279 (2011)
3. Abdulla, P.A., Jonsson, B.: Undecidable verification problems for programs with unreliable channels. Inf. Comput. 130(1), 71–90 (1996)
4. Abdulla, P.A., Jonsson, B.: Ensuring completeness of symbolic verification methods for infinite-state systems. Theor. Comput. Sci. 256(1-2), 145–167 (2001)
5. Arons, T., Pnueli, A., Ruah, S., Xu, J., Zuck, L.D.: Parameterized verification with automatically computed inductive assertions. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 221–234. Springer, Heidelberg (2001)
6. Cheng, A., Esparza, J., Palsberg, J.: Complexity results for 1-safe nets. TCS 147(1&2), 117–136 (1995)
7. Delzanno, G.: Constraint-based verification of parameterized cache coherence protocols. FMSD 23(3), 257–301 (2003)
8. Delzanno, G., Rosa-Velardo, F.: On the coverability and reachability languages of monotonic extensions of petri nets. Theor. Comput. Sci. 467, 12–29 (2013)

9. Delzanno, G., Sangnier, A., Traverso, R.: Parameterized verification of broadcat networks of register automata (technical report) (2013), http://verify.disi.unige.it/publications/
10. Delzanno, G., Sangnier, A., Traverso, R., Zavattaro, G.: On the complexity of parameterized reachability in reconfigurable broadcast networks. In: FSTTCS 2012. LIPIcs, vol. 18, pp. 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)
11. Delzanno, G., Sangnier, A., Zavattaro, G.: Parameterized verification of ad hoc networks. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 313–327. Springer, Heidelberg (2010)
12. Delzanno, G., Sangnier, A., Zavattaro, G.: On the power of cliques in the parameterized verification of ad hoc networks. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 441–455. Springer, Heidelberg (2011)
13. Emerson, E.A., Namjoshi, K.S.: On model checking for non-deterministic infinite-state systems. In: LICS 1998, pp. 70–80. IEEE Computer Society (1998)
14. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: LICS 1999, pp. 352–359. IEEE Computer Society (1999)
15. Fehnker, A., van Glabbeek, R., Höfner, P., McIver, A., Portmann, M., Tan, W.L.: Automated analysis of AODV using UPPAAL. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 173–187. Springer, Heidelberg (2012)
16. Fehnker, A., van Hoesel, L., Mader, A.: Modelling and verification of the LMAC protocol for wireless sensor networks. In: Davies, J., Gibbons, J. (eds.) IFM 2007. LNCS, vol. 4591, pp. 253–272. Springer, Heidelberg (2007)
17. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1-2), 63–92 (2001)
18. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39(3), 675–735 (1992)
19. Joshi, S., König, B.: Applying the graph minor theorem to the verification of graph transformation systems. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 214–226. Springer, Heidelberg (2008)
20. Kaminski, M., Francez, N.: Finite-memory automata. Theor. Comput. Sci. 134(2), 329–363 (1994)
21. Lazic, R., Newcomb, T., Ouaknine, J., Roscoe, A.W., Worrell, J.: Nets with tokens which carry data. Fundam. Inform. 88(3), 251–274 (2008)
22. Prasad, K.V.S.: A calculus of broadcasting systems. Sci. Comput. Program. 25(2-3), 285–327 (1995)
23. Saksena, M., Wibling, O., Jonsson, B.: Graph grammar modeling and verification of ad hoc routing protocols. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 18–32. Springer, Heidelberg (2008)
24. Schnoebelen, P.: Revisiting ackermann-hardness for lossy counter machines and reset petri nets. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 616–628. Springer, Heidelberg (2010)
25. Singh, A., Ramakrishnan, C.R., Smolka, S.A.: Query-based model checking of ad hoc network protocols. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 603–619. Springer, Heidelberg (2009)