Symbolic Verification of Lossy Channel Systems: Application to the Bounded Retransmission Protocol

Parosh Abdulla[1] Aurore Annichini[2] Ahmed Bouajjani[2]

[1] Dept. of Computer Systems, P.O. Box 325, S-75105 Uppsala, Sweden.
parosh@docs.uu.se
[2] VERIMAG, Centre Equation, 2 av. de Vignate, 38610 Gières, France.
Aurore.Annichini@imag.fr, Ahmed.Bouajjani@imag.fr

Abstract. We consider the problem of verifying automatically infinite-state systems that are systems of finite machines that communicate by exchanging messages through *unbounded lossy fifo channels*. In a previous work [1], we proposed an algorithmic approach based on constructing a symbolic representation of the set of reachable configurations of a system by means of a class of regular expressions (SREs). The construction of such a representation consists of an iterative computation with an *acceleration technique* which enhances the chance of convergence. This technique is based on the analysis of the effect of iterating control loops. In the work we present here, we experiment our approach and show how it can be effectively applied. For that, we developed a tool prototype based on the results in [1]. Using this tool, we provide an automatic verification of (the parameterized version of) the Bounded Retransmission Protocol.

1 Introduction

Communication protocols are naturally modeled as an asynchronous parallel composition of finite-state machines that exchange messages through unbounded fifo channels. Moreover, in a large class of communication protocols, e.g., link protocols, channels are assumed to be *lossy* in the sense that they can at any time lose messages. Then, an important issue is to develop automatic analysis techniques for *lossy channel systems*.

Many verification problems, e.g., verification of safety properties, reduce to computing the set of reachable configurations. However, since lossy channel systems are infinite-state systems, this set cannot be constructed by enumerative search procedures, and naturally a symbolic approach must be adopted allowing finite representations of infinite sets of configurations. Moreover, it has been shown that there is no algorithm for computing reachability sets of lossy channel systems [8]. Then, the approach we adopt is to develop semi-algorithms based on a forward iterative computation with a mechanism allowing to enhance the chances of convergence. This mechanism is based on *accelerating* the calculation [19, 9] by considering *meta-transitions* [6] corresponding to an arbitrary number of executions of control loops: in one step of the iterative computation, we add successors by the transitions of the system as well as all the reachable configurations by iterating control loops. So, to realize this approach, we need a *good* symbolic representation which should be expressive, and allow efficient performance of certain operations that are used in the computation of reachability sets, e.g., inclusion testing, computing successors by transitions of the system, as well as the effect of iterating control loops. In [1], we proposed a new symbolic representation formalism based on a class of regular expressions called SREs (simple regular expressions) for use in the reachability analysis of lossy channel systems. We showed in that work that SREs are good symbolic respresentations: we showed that SREs can define the reachability set of any lossy channel system (but not effectively in general), and that all the needed operations on SREs are rather simple and can be carried out in polynomial time.

The aim of this paper is to show the power of the approach we adopt and how our results in [1] can be effectively applied. Based on these results, we developed a tool prototype, called Lcs. Given a lossy channel system, this tool generates automatically its set of reachable configurations by means of SREs, and produces a *symbolic graph* which constitutes a finite-state abstract model of the system. Furthermore, the tool allows on-the-fly verification of safety properties given by finite-state labelled transition systems. The Lcs tool is connected to the Cadp toolbox [11] which provides a variety of procedures on finite-states labelled transition systems, e.g., comparison and minimization w.r.t. behavioural equivalences, model-checking for temporal logics. For instance, it is possible to generate automatically a finite abstract model of a system using the Lcs tool, and then apply standard finite-state verification techniques on this abstract model.

We show an interesting experimentation we have done with our tool, which consists of an automatic verification of the Bounded Retransmission Protocol (BRP) of Philips. The BRP is a data link protocol which can be seen as an extended version of the well known alternating bit protocol. It consists of a sender and a receiver that communicate through two lossy channels. The service the protocol delivers is the transmission of large files seen as sequences of data of arbitrary length. In addition, both the sender and receiver must indicate to their clients whether the whole file has been delivered successfully or not. The sender reads a sequence of data and transmit successively each datum in a separate frame following an alternating bit protocol-like procedure. However, the sender can resend a non-acknowledged frame up to a fixed number of retransmission MAX, which is a parameter of the protocol. Our modeling of the BRP assumes that the sizes of the transmitted sequences and the value MAX can be arbitrary positive integers. The assumption concerning MAX leads to a model with unbounded channels which represents the whole family of BRPs with any value of MAX. This shows an example where the model of unbounded channels allows a *parametric reasoning* about a family of systems.

We use our Lcs tool to generate automatically the set of reachable configurations of the BRP and the corresponding finite symbolic graph (0.56 seconds on UltraSparc). After projecting this graph on the set of external actions of the protocol and minimization w.r.t. observational trace equivalence, we get an abstract model with 5 states and 10 transitions which corresponds exactly to the expected external behaviour of the protocol.

Related Work: There are several works on symbolic verification of *perfect* fifo-channel systems [21, 13, 4, 5, 7]. Pachl proposed to represent the set of reachable configurations of a protocol as a recognizable set (carthesian product of regular sets), but he gave no procedures for computing such a representation. Finkel and Marcé proposed a symbolic analysis procedure using a class of regular expressions (not comparable with SREs), and which is based on an analysis of the unbounded iterability of a control loop [13]. The set of configurations computed by this procedure is, however, an upper approximation of the reachability set. Boigelot et al. use finite automata (under the name of QDDs) to represent recognizable sets of configurations [4, 5]. However, QDDs cannot characterize the effect of any control loop of a perfect fifo-channel system (restrictions on the type of loops are considered in order to preserve recognizability). To compute and represent the effect of any control loop, stuctures called CQDDs combining finite automata with linear arithmetical constraints must be used [7]. Our work ([1] and this paper) takes advantage from the fact that we are analysing specifically *lossy* channel systems. For these systems, we propose a symbolic representation (SREs) which captures exactly the class of reachability sets of such systems. Then, while the operations on QDDs and CQDDs are of exponential complexity and are performed by quite non-trivial algorithms, all needed operations on SREs can be performed by much simpler algorithms and in polynomial time. Moreover, although QDDs and CQDDs are more expressive than SREs, the algorithms in [4, 5, 7] cannot simulate the ones we use on SREs. The reason is that lossy transitions are implict in our model, whereas all transitions are explicitly represented in the algorithms in [4, 5, 7]. Thus to simulate in [4, 5, 7] the effect of iteration of a loop in the lossy channel model, we have to add transitions explicitly to model the losses. These transitions add in general new loops to the system, implying that a loop in the lossy channel system is simulated by a *nested* loop in the perfect channel system. However analysis of nested loops is not feasible in the approaches of [4, 5, 7].

Several works addressed the specification and verification of the BRP. To tackle the problem of unboundedness of the size of the transmitted files and the parameter MAX, these works propose proof-based approaches using theorem provers, combined with abstraction techniques and model checking. In [15] the system and its external specification are described in μCRL and are proved to be (branching) bisimilar. The proof is carried out by hand and checked using Coq. An approach based on proving trace inclusion (instead of bisimulation) on I/O automata is developed in [18]. In [17] the theorem prover PVS is used to prove that the verification of the BRP can be reduced by means of abstraction to a finite-state problem that can be solved by model checking. In [14, 3] a more automated approach is applied based on constructing automatically a *finite* abstract model using PVS, for an *explicitely given* abstraction function.

It is possible to see the unbounded lossy channel system we use to model the BRP as an *abstraction* of the whole family of the BRPs for all possible values of its parameters. But this model is *infinite-state*: the unboundedness of the parameters is in some sense transformed into an unboundedness of the channels. Then, starting from this infinite-state system, our verification technique is fully automatic. It is based on an automatic generation of a finite abstract model, without giving explicitly the abstraction relation. So, our work provides a fully automatic, and efficient, verification of the (untimed) *parameterized* version of the BRP.

Finally, we mention two works where the BRP has been verified automatically but *only for some fixed instances of its parameters*: In [20], an untimed version of the BRP is verified using both a bisimulation-based approach and a model checking approach using CADP. In [10] a timed version of the BRP is verified using the tools SPIN and UPPAAL. These two works avoid the issue of parameter unboundedness and use standard finite-state techniques. However, the work in [10] consider timing aspects that we have abstracted since our model is untimed.

Outline: In Section 2 we define the model of lossy channel systems. In Section 3 we present the verification approach we adopt. In Section 3.3 we present the class of SREs and we overview our results concerning this symbolic representation. In Section 4 we describe our tool prototype. In Section 5 we present our modeling and verification of the BRP. Concluding remarks are given in Section 6.

2 Lossy Channel Systems

We consider system models consiting of asynchronous parallel compositions of finite-state machines that communicate through sending and receiving messages via a finite set of unbounded *lossy* fifo channels (in the sense that they can nondeterministically lose messages).

A *Lossy Channel System* (LCS) $\mathcal{L}$ is a tuple $(S, s_{init}, C, M, \Sigma, \delta)$, where

- S is a finite set of *(control) states*, The control states of a system with n finite-state machines is formed as the Cartesian product $S = S_1 \times \cdots \times S_n$ of the control states of each finite-state machine.
- $s_{init} \in S$ is an *initial state*, The initial state of a system with n finite-state machines is a tuple $\langle s_{init_1}, \ldots, s_{init_n} \rangle$ of initial states of the components.
- C is a finite set of *channels*,
- M is a finite set of *messages*,
- Σ is a finite set of *transition (or action) labels*,
- δ is a finite set of *transitions*, each of which is of the form (s_1, ℓ, Op, s_2), where s_1 and s_2 are states, $\ell \in \Sigma$, and Op is a mapping from C to *(channel) operations*. An operation is either a *send* operation $!a$, a *receive* operation $?a$, or an empty operation *nop*, where $a \in M$.

A *configuration* γ of $\mathcal{L}$ is a pair $\langle s, w \rangle$ where $s \in S$ is a control state, and w is a mapping from C to M^* giving the contents of each channel. The *initial configuration* γ_{init} of $\mathcal{L}$ is the pair $\langle s_{init}, \epsilon \rangle$ where ϵ denotes the mapping where each channel is assigned the empty sequence ϵ.

We define a labelled transition relation on configurations in the following manner: $\langle s_1, w_1 \rangle \xrightarrow{\ell} \langle s_2, w_2 \rangle$ if and only if there exists a transition $(s_1, \ell, Op, s_2) \in \delta$

such that, for each $c \in C$, we have: if $Op(c) = !a$, then $w_2(c) = w_1(c) \cdot a$, and if $Op(c) = ?a$, then $a \cdot w_2(c) = w_1(c)$, and if $Op(c) = nop$, then $w_2(c) = w_1(c)$.

Let $\preceq$ denote the subsequence relation on M^*. For two mappings w and w' from C to M^*, we use $w \preceq w'$ to denote that $w(c) \preceq w'(c)$ for each $c \in C$. Then, we introduce a *weak* transition relation on configurations in the following manner: $\langle s_1, w_1 \rangle \overset{\ell}{\Longrightarrow} \langle s_2, w_2 \rangle$ if and only if there are w_1' and w_2' such that $w_1' \preceq w_1$, $w_2 \preceq w_2'$, and $\langle s_1, w_1' \rangle \xrightarrow{\ell} \langle s_2, w_2' \rangle$. Intuitively, $\langle s_1, w_1 \rangle \overset{\ell}{\Longrightarrow} \langle s_2, w_2 \rangle$ means that $\langle s_2, w_2 \rangle$ can be reached from $\langle s_1, w_1 \rangle$ by first losing messages from the channels and reaching $\langle s_1, w_1' \rangle$, then performing the transition $\langle s_1, w_1' \rangle \xrightarrow{\ell} \langle s_2, w_2' \rangle$, and thereafter losing messages from channels. Given a configuration γ, we let $post(\gamma)$ denote the set of immediate successors of γ, i.e., $post(\gamma) = \{\gamma' \,:\, \exists \ell \in \Sigma.\ \gamma \overset{\ell}{\Longrightarrow} \gamma'\}$. The function *post* is generalized to sets of configurations in the obvious manner. Then, we let $post^*$ denote the reflexive transitive closure of *post*, i.e., given a set of configurations Γ, $post^*(\Gamma)$ is the set of all reachable configurations starting from Γ. Let $Reach(\mathcal{L})$ be the set $post^*(\gamma_{init})$. For every control location $s \in S$, we define $\mathcal{R}(s) = \{w \,:\, \langle s, w \rangle \in Reach(\mathcal{L})\}$.

A *run* of $\mathcal{L}$ starting from a configuration γ is a finite or infinite sequence $\rho = \gamma_0 \ell_0 \gamma_1 \ell_1 \gamma_2 \ldots$ such that $\gamma_0 = \gamma$ and $\forall i \geq 0.\ \gamma_i \overset{\ell_i}{\Longrightarrow} \gamma_{i+1}$. The *trace* of the run ρ is the sequence of action labels $\tau = \ell_0 \ell_1 \ell_2 \ldots$. We denote by $Traces(\mathcal{L})$ (resp. $Traces_f(\mathcal{L})$) the set of all traces (resp. finite traces) of $\mathcal{L}$ starting from the initial configuration γ_{init}.

We introduce two extensions of the basic model given above: the first one consists in introducing *channel emptiness testing*: we use enabling conditions on transitions involving a predicate *empty* on channels telling whether a channel is empty. The second extension consists in allowing the components of a system to test and set *boolean shared variables* (remember that we consider here asynchronous parallel composition following the interleaving semantics). The formal semantics of the extended model is an obvious adaptation of the one given above.

3 Symbolic Reachability Analysis

We adopt an algorithmic verification approach based on the computation of the set of reachable configurations. We explain hereafter the general principle we consider in order to compute reachability sets, and how it can be applied to solve verification problems.

3.1 Computing reachability sets

The basic question is how to construct the set $Reach(\mathcal{L})$ for any given system $\mathcal{L}$, or more generally, how to construct the set $post^*(\Gamma)$ for any given set of configurations Γ of the system. Clearly, $post^*(\Gamma)$ is the least solution of the equation $X = \Gamma \cup post(X)$, and thus, it is the limit of the increasing sequence of sets $(X_i)_{i \geq 0}$ where $X_0 = \Gamma$ and $X_{i+1} = X_i \cup post(X_i)$. From this fact, one can derive an iterative procedure computing the set $post^*(\Gamma)$ which consists in computing the elements of the sequence of the X_i's until the inclusion $X_{i+1} \subseteq X_i$ holds for some index i, which means that the limit is reached. However, since the systems we are interested in have an infinite number of reachable configurations, this naive procedure does not terminate in general. Moreover, in the case of lossy channel systems, it has been shown that the set $Reach(\mathcal{L})$ cannot be effectively constructed although it is recognizable (finite-state automata definable) [8].

Hence, since an algorithm to construct the reachability sets does not exist in general, we adopt the approach of using semi-algorithms with a mechanism allowing to enhance their chance to terminate. This mechanism is based on the idea of *accelerating fixpoint computations* [19, 9]. For instance, consider a control loop of a lossy channel system that sends a symbol a on a channel, initially empty (we mean by control loop a circuit in the graph (S, δ)). The set of all reachable contents of the channel by iterating this loop is the regular language a^*. However, the naive procedure given above will compute successively: $X_0 = \{\epsilon\}, X_1 = \{\epsilon, a\}, X_2 = \{\epsilon, a, a^2\}, \ldots$, and never reach the limit. This example shows that if we are able to compute the effect of a loop on a set of configurations, we can use it to *jump to the limit* in one step, and help the fixpoint computation to converge: Given a control loop θ and a set of configurations Γ, let $post^*_\theta(\Gamma)$ be the set of reachable configurations by iterating θ an arbitrary number of times starting from Γ. Then, if the $post^*_\theta$ image of any set of configurations is effectively constructible, we can consider the loop θ as a *meta-transition* of the system [6]. This means that at each step of the iterative computation of the reachability set, we add immediate successors by original transitions of the system as well as successors by meta-transitions.

To realize this procedure, we need *representation structures* of sets of configurations. A *good* representation structure must allow a *finite* representation of the infinite sets of configurations we are interested in, it should be at least effectively closed under union and *post*, and it must have a decidable inclusion problem. Furthermore, this representation structure must allow the computation of the effects of control loops. Finally, any reasonable representation structure should be "normalizable", i.e., for every representable set, there is a unique normal (or canonical) representation which can be derived from any alternative representation (there is a normalization procedure). Indeed, all operations (e.g., entailement testing) are often easier to perform on normal forms. Furthermore, in many cases normality (canonicity) corresponds to a notion of minimality (e.g. for deterministic automata), which is crucial for practical reachability analysis procedures.

3.2 Use in verification

Verification of invariance properties It consists in checking whether *starting from the initial configuration of the system, a state property φ is always satisfied.* Clearly, this statement holds if $Reach(\mathcal{L}) \subseteq [\![\varphi]\!]$, where $[\![\varphi]\!]$ is the set of configurations satisfying φ. Thus, if $Reach(\mathcal{L})$ can be computed using a class $\mathcal{C}$ of good representation structures, and if $[\![\varphi]\!]$ is also effectively representable in $\mathcal{C}$, then our problem is solvable (inclusion is decidable for good representations).

Automata-based verification of safety properties A *regular safety property* is a set of finite traces over Σ. Then, the system $\mathcal{L}$ satisfies a property Π iff

$$Traces_f(\mathcal{L}) \subseteq \Pi \tag{1}$$

Naturally, a regular safety property Π is represented by a deterministic finite-state labelled transition system A_Π. This system is *completed* by adding a special state *bad* to the set of states Q, and adding transitions (q, ℓ, bad) for every $q \in Q$ and $\ell \in \Sigma$ such that there is no transitions in A_Π starting from q which are labelled by ℓ. Let A_Π^{bad} be the so obtained transition system and let $\mathcal{L} \times A_\Pi^{bad}$ be the synchronous product of $\mathcal{L}$ and A_Π^{bad}. The system $\mathcal{L} \times A_\Pi^{bad}$ is a lossy channel system (with the n channels of $\mathcal{L}$) whose control states are elements of $S \times (Q \cup \{bad\})$. Then, the problem (1) reduces to checking if $Reach(\mathcal{L} \times A_\Pi^{bad}) \subseteq S \times Q \times (M^*)^n$ (i.e., bad configurations are never reached).

It is convenient to consider a safety property Π as a set of traces over a set of *observable actions* $\Omega \subseteq \Sigma$. Then its verification problem consists in checking if $Traces_f(\mathcal{L})|_\Omega \subseteq \Pi$, where $|_\Omega$ denotes projection on Ω (i.e., hiding all symbols except those in Ω). Given A_Π^{bad} defined as previousely, this problem is equivalent to $Reach(\mathcal{L} \times_\Omega A_\Pi^{bad}) \subseteq S \times Q \times (M^*)^n$, where $\times_\Omega$ is the product of labelled transition systems with synchronisation on actions in Ω.

Generation of finite abstractions A C-indexed language W over M is a mapping from C to 2^{M^*} representing a set of C-indexed sequences such that $w \in W$ iff $\forall c \in C.\ w(c) \in W(c)$.

A *symbolic state* of $\mathcal{L}$ is a pair $\phi = \langle s, W \rangle$ where $s \in S$ is a control state and W is a C-indexed language over M. The symbolic state ϕ represents the set of configurations $[\![\phi]\!] = \{\langle s, w \rangle \ : \ w \in W\}$.

Let Φ be a finite set of symbolic states. Then, the *symbolic graph* associated with Φ is the finite-state labelled transition system $\mathcal{G}_\Phi$ such that its set of states is Φ and, $\forall \phi_1, \phi_2 \in \Phi.\ \forall \ell \in \Sigma.\ \phi_1 \xrightarrow{\ell} \phi_2$ iff $\exists \gamma_1 \in \phi_1, \gamma_2 \in \phi_2.\ \gamma_1 \xrightarrow{\ell} \gamma_2$. We consider as initial state in $\mathcal{G}_\Phi$ any configuration which contains the initial configuration γ_{init}.

In particular, we consider the partition of $Reach(\mathcal{L})$ according to the control states, i.e., $\Phi_\mathcal{L} = \{\langle s, W \rangle \ : \ s \in S \text{ and } [\![W]\!] = \mathcal{R}(s)\}$. The labelled transition system $\mathcal{G}_{\Phi_\mathcal{L}}$ is called the *canonical symbolic graph* of $\mathcal{L}$.

Lemma 1. *For every finite set of symbolic states Φ, if $Reach(\mathcal{L}) \subseteq \bigcup_{\phi \in \Phi} [\![\phi]\!]$, then $\mathcal{G}_\Phi$ simulates $\mathcal{L}$.*

Indeed, it is easy to see that the membership relation, i.e., the relation R such that $\gamma R \phi$ iff $\gamma \in [\![\phi]\!]$, is a simulation relation (using the fact that every reachable configuration of $\mathcal{L}$ belongs to at least one symbolic state in Φ).

Clearly, Lemma 1 holds for the canonical symbolic graph of $\mathcal{L}$. This means that if $Reach(\mathcal{L})$ can be constructed, we obtain directly a *finite-state abstraction* of the system $\mathcal{L}$. This abstract model can be used to check linear-time properties and, if the result is positive, to deduce that the same result holds for the *concrete* system $\mathcal{L}$[3]. More precisely, given an ∞-regular linear-time property Π, i.e., a set of finite or infinite traces over Σ, a system $\mathcal{L}$ satisfies Π if $Traces(\mathcal{L}) \subseteq \Pi$. By Lemma 1, we have $Traces(\mathcal{L}) \subseteq Traces(\mathcal{G}_{\Phi_\mathcal{L}})$. Hence, for every ∞-regular property Π, if $\mathcal{G}_{\Phi_\mathcal{L}}$ satisfies Π, then $\mathcal{L}$ satisfies Π too.

Notice that if $\mathcal{G}_{\Phi_\mathcal{L}}$ does not satisfy Π, this could be due to the fact that the abstraction corresponding to the partition of $Reach(\mathcal{L})$ according to control state is too coarse. Then, one could try to check Π on refinements of this partition.

3.3 Computing Reachability Sets of LCSs

We introduced in [1] a new symbolic representation formalism, based on a class of regular expressions called SREs (simple regular expressions), for use in the calculation of reachability sets of lossy channel systems. We showed in that previous work that SREs are "*good*" representation structures in the sense introduced in Section 3.1. We give hereafter the definition of SREs and a brief overview of the results of [1] concerning these representations.

Definition 2 (SREs). An *atomic simple expression* over M is a regular expression of one of two following forms: $(a+\epsilon)$, where $a \in M$, or $(a_1 + \cdots + a_m)^*$, where $a_1, \ldots, a_m \in M$. A *simple product* p over M is either ϵ (denoting the language $\{\epsilon\}$) or a concatenation $e_1 \cdot e_2 \cdots e_n$ of atomic simple expressions over M. A *simple regular expression* (SRE) r over M is either $\emptyset$ (denoting the empty language) or a sum $p_1 + \cdots + p_n$ of simple products over M. Given an SRE r, we denote by $[\![r]\!]$ the language it defines. A language is said to be *simply regular* if it is definable by an SRE.

A C-indexed SRE R over M is a mapping from C to the set of SREs. The expression R defines the C-indexed language L (denoted $[\![R]\!]$) such that, for every $c \in C$, $L(c) = [\![R(c)]\!]$. A C-indexed language is said to be *simply recognizable* if it is a finite union of languages definable by C-indexed SREs.

Any set of configurations Γ is a union of the form $\bigcup_{s \in S} \{s\} \times W_s$ where the W_s's are C-indexed languages over M. We say that Γ is SRE definable if W_s is simply recognizable for each $s \in S$.

For a lossy channel system $\mathcal{L}$, the set $Reach(\mathcal{L})$ is SRE definable (the set $\mathcal{R}(s)$ is simply recognizable for each control state s) [1]. This means that SREs are expressive enough to represent the reachability set of any lossy channel system. However, as we mentionned before, there is, in general, no algorithm for computing a representation of $Reach(\mathcal{L})$ for a lossy channel system $\mathcal{L}$ [8].

An *entailment relation* can be defined on SREs: For SREs r_1 and r_2, we say that r_1 *entails* r_2 (we write $r_1 \sqsubseteq r_2$), if $[\![r_1]\!] \subseteq [\![r_2]\!]$. This relation is extended to indexed SREs in the obvious manner. It can be shown that entailment among indexed SREs can be checked in quadratic time [1].

Definition 3 (Normal form). A simple product $e_1 \cdots e_n$ is said to be *normal* if $\forall i \in \{1, \ldots, n\}.\ e_i \cdot e_{i+1} \not\sqsubseteq e_{i+1}$ and $e_i \cdot e_{i+1} \not\sqsubseteq e_i$. An SRE $r = p_1 + \cdots + p_n$ is said to be *normal* if $\forall i \in \{1, \ldots, n\}.\ p_i$ is normal, and $\forall i, j \in \{1, \ldots, n\}.\ i \neq j.\ p_i \not\sqsubseteq p_j$.

[3] This approach can also be applied for branching-time properties expressed in universal positive fragments of temporal logics or μ-calculi like $\forall$CTL* [16] or $\Box L_\mu$ [2].

It can be shown that for each SRE r, there is a unique (up to commutativity of products) normal SRE, denoted $\overline{r}$, such that $[\![\overline{r}]\!] = [\![r]\!]$, and which can be derived from r in quadratic time [1].

Finally, we can show that, for a lossy channel system $\mathcal{L}$ and an SRE representable set of configurations Γ, the set $post(\Gamma)$ is SRE definable and effectively constuctible in linear time, and that for any control loop θ in $\mathcal{L}$, the set $post^*_\theta(\Gamma)$ is also SRE definable and effectively constuctible in quadratic time [1].

4 Implementation

We implemented our techniques in a tool prototype called Lcs. The input of the Lcs is a finite set of communicating automata, given seperately. Then, the tool allows the following options:

Generation of the reachability set: The tool allows calling a procedure which computes a representation of the reachability set of the system by means of (normal) SREs. The computation is done according to a depth-first-search strategy, and uses the acceleration principle (see Sections 3 and 3.3): Starting from the initial configuration, the procedure explores a graph where nodes are symbolic states. When visiting a node, the procedure computes its immediate successors using the *post* function. Whenever a control loop is detected, i.e., the current symbolic state has an ancestor with the same control state, the effect of iterating this loop is computed, leading to a new symbolic state. Notice that the loops used for acceleration are found on-the-fly and are not explicitly given by the user. The set of reachable configurations is memorized progressively. If a visited node (symbolic state) is included in the set of reachable configurations computed so far, the successors of the node are not generated. Otherwise, its set of configurations is added to the current set of reachable configurations, and the search continues.

Generation of the canonical symbolic graph: During the computation the reachability set, the Lcs tool can construct the corresponding canonical symbolic graph (transitions between symbolic states).

The symbolic graph is produced in the input format of the Cadp toolbox (Caesar/Aldebaran Development Package) [11] which contains several tools on finite-state labelled transition systems, e.g., graphical visualisation, comparison with respect to various behavioural equivalences and preorders like observational bisimulation and simulation, minimization, on-the-fly automata-based verification, model-checking for an ACTL-like temporal logic (action-based variant of CTL) and the alternation-free modal μ-calculus.

On-the-fly checking of safety properties: Given a safety property described as a deterministic labelled transition system over a set observable actions $\Omega \subseteq \Sigma$, the tool checks whether the projection of the system on Ω satisfies Π. This verification (based on a reachability set generation, see Section 3.2) is done on-the-fly: the procedure stops as soon as a bad configuration is encountered.

Example: Altenating Bit Protocol Let us illustrate the use of our tool on the alternating bit protocol (ABP for short). We model the ABP by two finite-state machines, a sender and a receiver, communicating through two lossy channels K and L (see Figure 1). Then, the procedure implemented in the LCS tool

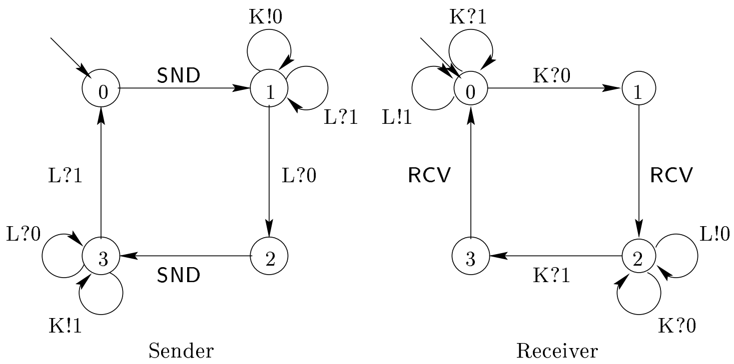


Fig. 1. Alternating Bit Protocol

Sender	Receiver	Channel K	Channel L
0	0	1*	1*
1	0	1*0*	1*
1	1	0*	1*
1	2	0*	1*0*
2	2	0*	0*
3	2	0*1*	0*
3	3	1*	0*
3	0	1*	0*1*

Table 1. Reachability set of the ABP

terminates and generates automatically the reachability set of the ABP (see Table 1), as well as the corresponding canonical symbolic graph (see Figure 2). The execution time is 0.07 seconds (UltraSparc). Then, using the ALDEBARAN tool [12], we minimize this graph according to observational trace equivalence by considering that SND and RCV are the only observable actions. The resulting minimal transition system is shown in Figure 2. It is clear from this transition system that the external behaviour of the ABP is equivalent to the behaviour of a one-place buffer.

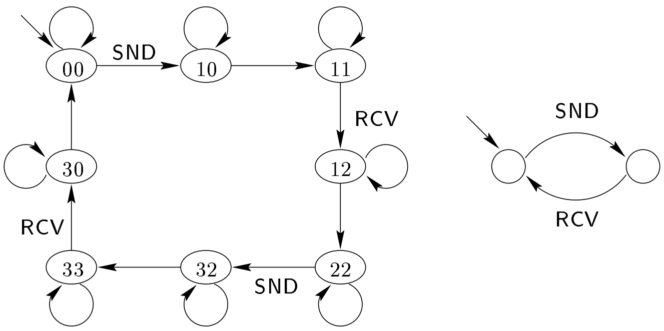


Fig. 2. Canonical symbolic graph of the ABP and its minimal version

Remark. The minimization we considered removes all silent (non-observable) transitions, including silent loops. This can be done since we are only interested in safety properties on observable actions. This is, however, not sound for checking liveness properties (silent loops leading to divergence must be taken into account in this case).

5 The Bounded Retransmission Protocol

5.1 Specification of the service

The Bounded Retransmission Protocol (BRP for short) is a data link protocol. The service it delivers is to transmit large files (sequences of data of arbitrary lengths) from one client to another one. Each datum is transferred in a separate frame. Both clients, the sender and the receiver, obtain an indication whether the whole file has been delivered successfully or not.

More precisely, at the sender side, the protocol requests a sequence of data $s = d_1, \ldots, d_n$ (action REQ) and communicates a *confirmation* which can be SOK, SNOK, or SDNK. The confirmation SOK means that the file has been transferred successfully, SNOK means that the file has not been transferred completely, and SDNK means that the file may not have been transferred completely. This occurs when the last datum d_n is sent but not acknowledged. Now, at the receiver side, the protocol delivers each correctly received datum with an indication which can be RFST, RINC, or ROK. The indication RFST means that the delivered datum is the first one and more data will follow, RINC means that the datum is an intermediate one, and ROK means that this was the last datum and the file is completed. However, when the connection with the sender is broken, an indication RNOK is delivered (without datum). Properties the service must satisfy are:

1. a request REQ must be followed by a confirmation (SOK, SNOK, or SDNK) before the next request,
2. a RFST indication (delivery of the first datum) must be followed by one of the two indications ROK or RNOK before the beginning of a new transmission (next request of the sender),

3. a SOK confirmation must be preceded by a ROK indication,
4. a ROK indication can be followed by either a SOK or a SDNK confirmation, but never by a SNOK (before next request),
5. a RNOK indication must be preceded by SNOK or SDNK (abortion),
6. if the first datum has been received (with the RFST indication), then a SNOK confirmation is followed by a RNOK indication before the next request.

5.2 Description of the protocol

The BRP consists of two processes, the sender S and the receiver R, that communicate through two lossy fifo channels K and L: messages can either be lost or arrive in the same order in which they are sent. The BRP can be seen as an extended version of the alternating bit protocol. Messages sent from the sender S to the receiver R through the channel K are frames of the form $(first, last, toggle, datum)$ where a datum is accompanied by three bits: $first$ and $last$ indicate whether the datum is the first or the last one of the considered file, $toggle$ is the alternating bit allowing to detect duplications of intermediate frames. As for the acknowledgments (sent from R to S through L), they are frames of the form $(first, last, toggle)$. Notice that in the description we consider of the BRP, the value of $toggle$ is relevant only for intermediary frames. Indeed, the first and last frames can be distinguished from the intermediary ones using the booleans $first$ and $last$.

The behaviours of S and R are the following: The sender S starts by reading (action REQ) a sequence $s = d_1, \ldots, d_n$. We consider here that $n \geq 2$, the case $n = 1$ does not introduce any difficulty. Then, S sends to R through K the first data frame $(1, 0, 0, d_1)$, and waits for the acknowledgement. Let us consider first the ideal case where frames are never lost. When R receives the frame from K, it delivers to its client the datum d_1 with the indication RFST, and sends to S an acknowledgement frame $(1, 0, 0)$ through the channel L. When S receives this acknowledgement, it transmits to R the second frame $(0, 0, 0, d_2)$ ($toggle$ is still equal to 0 since its value is relevant for intermediate frames). Then, after reception, R delivers d_2 with the indication RINC and sends the acknowledgement $(0, 0, 0)$ to S. Then, the next frame sent by S is $(0, 0, 1, d_3)$ (now $toggle$ has flipped), and the same procedure is repeated until the last frame $(0, 1, -, d_n)$ is sent (here again, like in the case of the first frame, the value of $toggle$ is not relevant). When R receives the last frame, it delivers d_n with the indication ROK, and acknowledges receipt. Then, the sender S communicates to its client the confirmation SOK meaning that the whole sequence s has been successfully transmitted.

Now, let us consider the case where frames are lost. When S send a data and realizes that it may be lost (a timer T_s expires and it did not receive a corresponding acknowledgement from R), it retransmits the same frame and waits again for the acknowledgement. However, it can try only up to a fixed maximal number of retransmissions MAX which is a parameter of the protocol. So, the sender maintains a counter of retransmissions CR, and when CR reaches the value MAX, it gives up and concludes that the connection with the receiver is broken. Then, it informs its client that a failure occured by communicating one of the two confirmations: SNOK if the frame in consideration is not the last frame of the sequence, or SDNK if it is the last one (the sender cannot know if the frame was lost or if its acknowledgement was lost). On the other side, the receiver R uses also a timer T_r to measure the time elapsed between the arrival of two different frames. When R receives a new frame, it resets T_r and, it delivers the transmitted datum with the corresponding indication, otherwise it resends the last acknowledgement. If the timer expires, it concludes that the connection with the sender is broken and delivers an indication RNOK meaning that the transmission failed. Notice that if the first frame is continuously lost, the receiver has no way to detect that the sender is trying to start a new file transmission. In addition, two assumptions are made on the behaviour of S and R:

A1 R must not conclude prematurely that the connection with S is broken.
A2 In case of abortion, S cannot start transmitting frames of another file until R has reacted to abortion and informed its client.

Assumption **A1** means that T_r must be large enough to allow MAX retransmissions of a frame. Assumption **A2** can be implemented for instance by imposing to S to wait enough time after abortion to be sure that T_r has expired.

5.3 Modeling the BRP as a Lossy Channel System

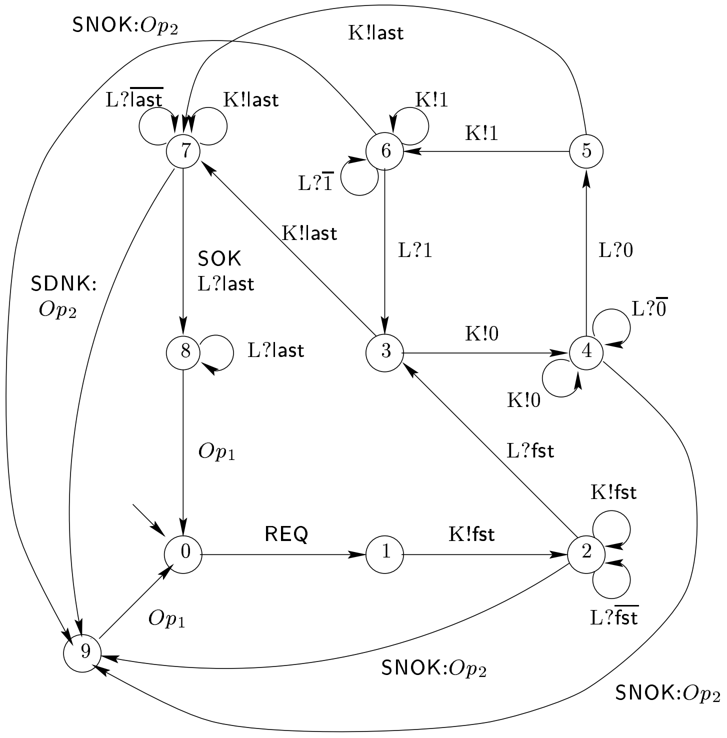
We model the BRP as a lossy channel system which consists of two communicating finite-state machines, the sender S and the receiver R represented in Figures 3 and 4 (with obvious notational conventions). For that, we proceed as follows:

Frames: Since the control of the BRP does not depend on the transmitted data, we hide their values and consider only the informations $(first, last, toggle)$. The set of relevant informations of such form corresponds to a finite alphabet $M = \{\mathsf{fst}, \mathsf{last}, 0, 1\}$, where fst (resp. last) represents the first (resp. last) frame, and 0 and 1 represents the intermediate frames since only *toggle* is relevant in this case.

The number of transmitted frames: Only is relevant whether a frame is the first one, the last one, or an intermediate one, we abstract from the actual value n corresponding to the size of the transmitted sequence of frames, and consider that it can be any positive integer, chosen nondeterministically (by the sender).

Time-outs: Since our model is *untimed*, we cannot express time-outs explicitly. Then, we consider that the sender and the receiver decide nondeterministically when time-outs occur, provided that their corresponding input channels are empty (we use channel emptiness testing).

The counter CR and the value MAX: Only is relevant whether $\mathsf{CR} < \mathsf{MAX}$ or $\mathsf{CR} \geq \mathsf{MAX}$. Then, we consider that the sender can resend frames an arbitrary number of times before considering that MAX is reached and deciding the abortion of the transmission. This makes the size of the channels K and L unbounded. Our model is an abstraction of the whole family of BRPs for arbitrary values of MAX.



$Op_1 = \neg\mathsf{rtrans} \wedge empty(K) \wedge empty(L) \mapsto \mathsf{abort} := \text{false}$
$Op_2 = empty(L) \mapsto \mathsf{abort} := \text{true}$

Fig. 3. The sender S

Assumptions A1 and A2: Again, since our model is untimed, we cannot impose real-time constraints to implement the assumptions **A1** and **A2**. Then, we use boolean shared variables to synchronise the sender and the receiver. We consider the two following variables: abort which tells whether the sender has decided abortion, and rtrans which tells whether the receiver considers that the transmission of a sequence of frames has started and is not finished yet, i.e., from the moment it receives the first frame until it informs its client that the transmission is terminated, either successfully or not.

5.4 Verifying the Bounded Retransmission Protocol

To verify the BRP, we proceed as follows: First, we use our Lcs tool to generate automatically the set of reachable configurations of the BRP and the corresponding canonical symbolic graph. The obtained graph has 24 symbolic states and 61 transitions. The execution time is 0.56 seconds (UltraSparc).

Then, we use the tool Aldebaran to minimize this graph according to the observational trace equivalence where the set of observable actions is {REQ, SOK,

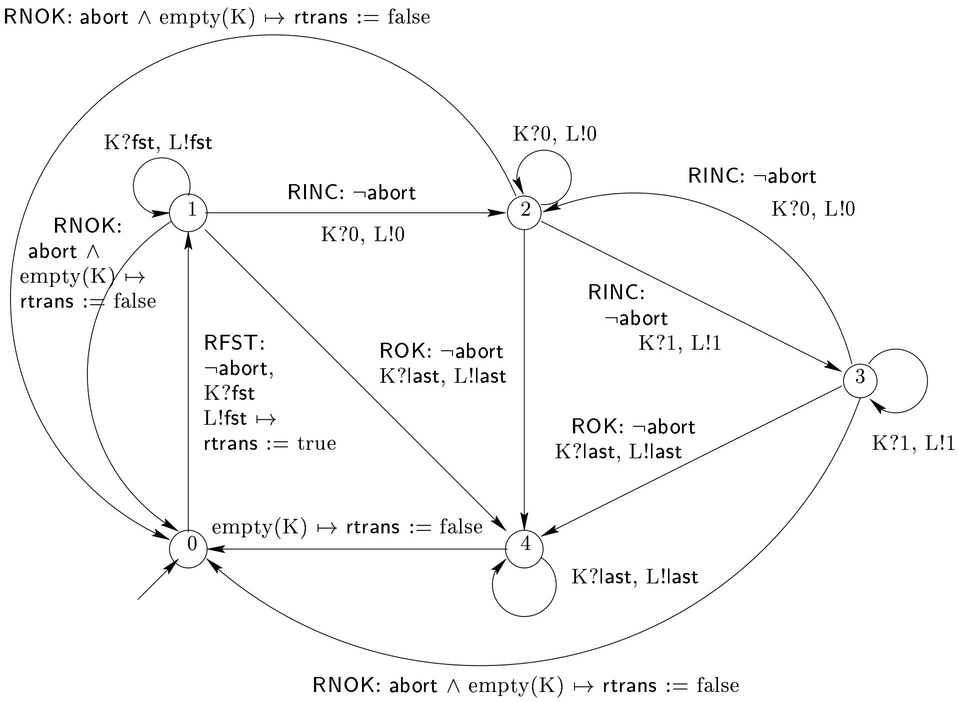


Fig. 4. The receiver R

SNOK, SDNK, RFST, RINC, ROK, RNOK}. We obtain the finite-state labelled transition system with 5 states and 10 transitions given in Figure 5. Properties such as those given in Section 5.1 are expressible in ACTL (the action-based variant of CTL) and can be automatically model checked on the obtained finite-state abstract model of the BRP.

6 Conclusion

We have presented a symbolic approach for verifying automatically a class of infinite-state systems: the class of *unbounded lossy channel systems*. This approach is based on a procedure of constructing the set of reachable configurations of the system by means of a symbolic representation (SREs), and acceleration techniques based on the analysis of the effect of control loops. In addition to the generation of the reachability set of a system, we showed that this approach allows the automatic generation of a finite abstract model of the system which can be used for checking various properties by means of standard finite-state verification methods.

We applied this approach to the non-trivial example of the BRP. We showed that considering unbounded channels allows parametric reasoning: unboundedness of the channels models the fact that the number of retransmissions can be

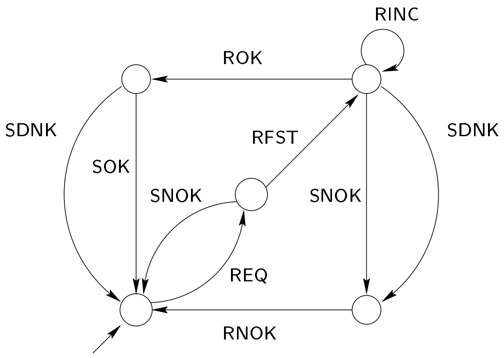


Fig. 5. The minimized symbolic graph of the BRP

any arbitrary positive integer. Our experimentation with the LCS tool show that the algorithmic approach we adopt is quite effective. For a first prototype, we obtained quite satisfactory performances.

References

1. P. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly Analysis of Systems with Unbounded, Lossy Fifo Channels. In *CAV'98*. LNCS 1427, 1998.
2. S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property-preserving simulations. In *CAV'92*. LNCS 663, 1992.
3. S. Bensalem, Y. Lakhnech, and S. Owre. Computing Abstractions of Infinite State Systems Compositionally and Automatically. In *CAV'98*. LNCS 1427, 1998.
4. B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. In *CAV'96*. LNCS 1102, 1996.
5. B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *SAS'97*. LNCS 1302, 1997.
6. B. Boigelot and P. Wolper. Symbolic Verification with Periodic Sets. In *CAV'94*. LNCS 818, 1994.
7. A. Bouajjani and P. Habermehl. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. In *ICALP'97*. LNCS 1256, 1997.
8. Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable Channels Are Easier to Verify Than Perfect Channels. *Inf. and Comp.*, 124(1):20–31, 1996.
9. P. Cousot and R. Cousot. Static Determination of Dynamic Properties of Recursive Procedures. In *IFIP Conf. on Formal Desc. of Prog. Concepts*. NH Pub., 1977.
10. P. D'Argenio, J-P. Katoen, T. Ruys, and G.J. Tretmans. The Bounded Retransmission Protocol must be on Time. In *TACAS'97*. LNCS 1217, 1997.
11. J-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A Protocol Validation and Verification Toolbox. In *CAV'96*. LNCS 1102, 1996.
12. J.Cl. Fernandez and L. Mounier. A tool set for deciding behavioural equivalences. In *CONCUR'91*. LNCS 527, August 1991.
13. A. Finkel and O. Marcé. Verification of Infinite Regular Communicating Automata. Technical report, LIFAC, ENS de Cachan, 1996.

14. S. Graf and H. Saidi. Construction of Abstract State Graphs with PVS. In *CAV'97*, volume 1254 of *LNCS*, 1997.
15. J-F. Groote and J. Van de Pol. A Bounded Retransmission Protocol for Large Data Packets. In *AMAST'96*. LNCS 1101, 1996.
16. O. Grumberg and D. Long. Model Checking and Modular Verification. *ACM TOPLAS*, 16:843–871, 1994.
17. K. Havelund and N. Shankar. Experiments in Theorem Proving and Model Checking for Protocol Verification. In *FME'96*. LNCS 1051, 1996.
18. L. Helmink, M.P.A. Sellink, and F. Vaandrager. Proof checking a Data Link Protocol. In *Types for Proofs and Programs*. LNCS 806, 1994.
19. R.M. Karp and R.E. Miller. Parallel Program Schemata: A Mathematical Model for Parallel Computation. In *8th ann. Switch. and Aut. Theo. Symp.* IEEE, 1967.
20. R. Mateescu. Formal Description and Analysis of a Bounded Retransmission Protocol. Technical report no. 2965, INRIA, 1996.
21. J.K. Pachl. Protocol Description and Analysis Based on a State Transition Model with Channel Expressions. In *Protocol Specification, Testing, and Verification VII*, 1987.