Verifying Lossy Channel Systems has Nonprimitive Recursive Complexity

Ph. Schnoebelen

Lab. Spécification et Vérification
ENS de Cachan & CNRS UMR 8643
61, av. Pdt. Wilson,
94235 Cachan Cedex France
email: phs@lsv.ens-cachan.fr

December 7, 2001

Keywords: verification of infinite-state systems, communication protocols, program correctness, formal methods.

Abstract

Lossy channel systems are systems of finite state automata that communicate via unreliable unbounded fifo channels. It is known that reachability, termination and a few other verification problems are decidable for these systems. In this article we show that these problems cannot be solved in primitive recursive time.

1 Introduction

Channel systems, also called *Finite State Communicating Machines*, are systems of finite state automata that communicate via asynchronous unbounded fifo channels [Boc78, BZ83]. Figure 1 displays an example, where the labels $c!x$ and $c?x$ mean that message x (a letter) is sent to (respectively read from) channel c. Channel systems are a natural model for asynchronous

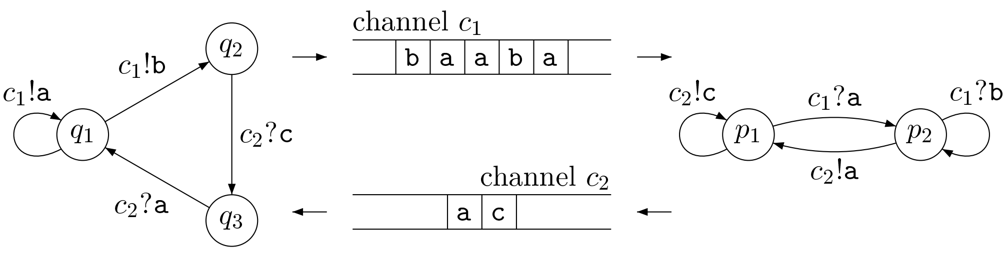

Figure 1: A channel system with two automata and two channels

communication protocols and constitute the semantical basis for ISO protocol specification languages such as SDL and Estelle. Channel systems are Turing powerful, and no verification method for them can be general and fully algorithmic.

A few years ago, Abdulla and Jonsson identified *lossy channel systems* as a very interesting model: in lossy channel systems messages can be lost while they are in transit, without any notification. These systems are very close to the *completely specified protocols* independently introduced by Finkel, and for which he showed the decidability of termination. Abdulla and Jonsson showed that reachability, safety properties over traces, and eventuality properties over states are decidable for lossy channel systems. The decidability results of [Fin94, CFP96, AJ96b] are fundamental since lossy systems are the natural model for fault-tolerant protocols where the communication channels are not supposed to be reliable (see [AKP97, ABJ98, AAB99] for applications).

For lossy channel systems, the aforementioned decidability results lead to algorithms whose termination rely on Higman's Lemma (see [AČJT00, FS01] for more examples of this phenomenon). No complexity bound is known and, e.g., Abdulla and Jonsson stated in [AJ96b] that they could not evaluate the cost of their algorithm.

In this article we show that all the above-mentioned decidable problems have nonprimitive recursive complexity, i.e., cannot be solved by algorithms with running time bounded by a primitive recursive function of their input size. This puts these problems among the hardest decidable problems.

Our proof relies on a simple construction showing how lossy channel systems can weakly compute some fast growing number-theoretic functions $A_2, A_3, \ldots$ related to Ackermann's function and their inverses $A_2^{-1}, A_3^{-1}, \ldots$ By "weakly computing f" we mean that, starting from x, all values between 0 and $f(x)$ can be obtained. This notion was used by Rabin in his proof that equality of the reachability sets of Petri nets is undecidable, a proof based on the weak computability of multivariate polynomials (see [Hac76]). Petri nets can weakly compute the A_n functions (see [MM81]) but they cannot weakly compute their inverses A_n^{-1} as lossy channel systems can do.

There exist other families of systems that can weakly compute both A_n and A_n^{-1}: the lossy counter machines of [May00], or the reset nets of [DFS98, DJS99]. Our construction can easily be adapted to show that, for these systems too, decidable problems like termination, control-state reachability, ..., are nonprimitive recursive.

Finally, let us observe that there does not exist many uncontrived problems that have been shown decidable but not primitive recursive. In the field of verification, we are only aware of one instance: the "finite equivalence problem for Petri nets" [1] introduced by Mayr and Meyer [MM81]. This problem is, given two Petri nets, to decide whether they both have the same set of reachable markings *and this set is finite* (equivalence is undecidable without the finiteness assumption). It can be argued that the verification (termination or reachability) of lossy channel systems is a less contrived problem.

2 Channel systems, from perfect to lossy

A channel system usually combines several finite-state automata that communicate through several channels. Here, and without loss of generality, we assume our systems only have one

[1] See [Jan01] for a more general proof that *all* finite equivalence problems for Petri nets are nonprimitive recursive.

automaton that uses its several channels as fifo buffers.

Formally, a *channel system* is a tuple $S = \langle Q, C, \Sigma, \Delta\rangle$ where $Q = \{q, \ldots\}$ is a finite set of *control states*, $C = \{c_1, \ldots, c_k\}$ is a finite set of *channels*, $\Sigma = \{\mathtt{a}, \mathtt{b}, \ldots\}$ is a finite alphabet of *messages*, and $\Delta \subseteq Q \times C \times \{?, !\} \times \Sigma^* \times Q$ is a finite set of transition rules (see below).

A *configuration* of S is a tuple $\gamma = \langle q, w_1, \ldots, w_k\rangle$ denoting that control is currently in state q, while channels c_1 to c_k contain words $w_1, \ldots, w_k$ (from Σ^*).

The transition rules in Δ state how S can move from a configuration to another. Formally, S has a "perfect" step $\gamma \to_{\text{perf}} \gamma'$ iff (1) γ is some $\langle q, w_1, \ldots, w_k\rangle$, (2) γ' is some $\langle q', w_1, \ldots, w_{i-1}, v, w_{i+1}, \ldots, w_k\rangle$, and (3.1) there is a rule $(q, c_i, !, u, q') \in \Delta$ such that $v = w_i u$ (*u has been written to the tail of c_i*) or (3.2) there is a rule $(q, c_i, ?, u, q') \in \Delta$ such that $w_i = uv$ (*u has been read from the head of c_i*). These steps are called perfect because no message is lost.

It is well known that, assuming perfect steps, channel systems can faithfully simulate Turing machines in quadratic time [BZ83] (a single channel is enough to replace a Turing machine work tape; reading and writing in the middle of the channel requires rotating the content of the channel for positioning reasons, hence the quadratic overhead). Thus all interesting verification problems are undecidable for systems with perfect channels, even when restricted to single-channel systems.

2.1 Lossy systems

The most elegant and convenient way to model *lossy channel systems* is to see them as channel systems with an altered notion of steps [AJ96b].

We write $u \sqsubseteq v$ when u is a subword of v, i.e. u can be obtained by deleting any number (including 0) of letters from v. E.g. $\mathtt{abba} \sqsubseteq \mathtt{\underline{ab}racada\underline{b}r\underline{a}}$ as indicated by the underlining. The subword ordering extends to configurations: $\langle q, w_1, \ldots, w_k\rangle \sqsubseteq \langle q', v_1, \ldots, v_k\rangle \overset{\text{def}}{\Leftrightarrow} q = q'$ and $w_i \sqsubseteq v_i$ for all $i = 1, \ldots, k$. By Higman's Lemma [Hig52], this gives a a well-quasi-ordering:

Lemma 2.1 *Every infinite sequence $\gamma_0, \gamma_1, \gamma_2, \ldots$ of configurations contains an infinite increasing subsequence $\gamma_{i_0} \sqsubseteq \gamma_{i_1} \sqsubseteq \gamma_{i_2} \sqsubseteq \cdots$ (with $i_0 < i_1 < i_2 < \cdots$).*

When $\gamma' \sqsubseteq \gamma$ we write $\gamma \rightsquigarrow \gamma'$ and say that S may evolve from γ to γ' by losing messages. The steps of a lossy channel system are all $\gamma \to \gamma'$ s.t. $\gamma \rightsquigarrow \delta \to_{\text{perf}} \delta' \rightsquigarrow \gamma'$ for some configurations δ, δ' (i.e. losses may occur before and after a perfect step is performed).

Note that a perfect step is a special case of a lossy step. A *run* is a sequence $\gamma_0 \to \gamma_1 \to \gamma_2 \cdots$ of chained lossy steps. A perfect run is a run that uses perfect steps only. We use $\gamma \xrightarrow{*} \gamma'$ and $\gamma \xrightarrow{*}_{\text{perf}} \gamma'$ to denote the existence of a finite run (resp. perfect run) that goes from γ to γ'.

We are interested in the following two problems:

Termination: Given a channel system S and an initial configuration γ_0, are all runs from γ_0 finite?

Reachability: Given a channel system S and two configurations γ_0 and γ_f, is there a run from γ_0 to γ_f?

Theorem 2.2 [Fin94, AJ96b]. *Termination and reachability are decidable for lossy channel systems.*

In the remaining of this note we show

Theorem 2.3 *Termination and reachability for lossy channel systems have nonprimitive recursive complexity.*

Theorem 2.3 also applies to the other verification problems that are known decidable for lossy channel systems. Indeed, termination is an instance of inevitability (shown decidable in [AJ96b]). Reachability is easily reduced to control-state reachability (shown decidable in [AJ96b]). Finally, termination can be reduced to simulation with a finite-state system [2] (shown decidable in [AK95, AČJT00]). Thus we are entitled to claim that "verifying lossy channel systems has nonprimitive recursive complexity". Note that there exist many undecidable problems for lossy channel systems [AK95, CFP96, AJ96a, May00, ABPJ00, Sch01].

3 The main construction

3.1 Ackermann's function

Let $(A_n)_{n\geq 2}$ be the following sequence of functions over the natural numbers:

$$A_2(k) \stackrel{\text{def}}{=} 2k \tag{1}$$

$$A_n(k) \stackrel{\text{def}}{=} \underbrace{A_{n-1} \circ \cdots \circ A_{n-1}}_{k \text{ times}}(1) \qquad \text{if } n > 2. \tag{2}$$

Thus $A_n(0) = 1$ for all $n > 2$, and A_2 is followed by

$$A_3(k) = 2^k \tag{3}$$

$$A_4(k) = 2^{2^{\cdot^{\cdot^{\cdot^2}}}} \qquad \text{(a tower of } k \text{ 2's)} \tag{4}$$

and so on. The A_n's are monotonic and expansive in the following sense: for any $n \geq 2$ and $k \geq 0$

$$A_n(k) \geq k, \qquad A_n(k+1) \geq A_n(k), \qquad A_{n+1}(k) \geq A_n(k). \tag{5}$$

We define inverse functions $(A_n^{-1})_{n\geq 2}$ via

$$A_n^{-1}(m) \stackrel{\text{def}}{=} \max\{k \mid A_n(k) \leq m\}. \tag{6}$$

Observe that the A_n^{-1}'s are partial functions. Another way to understand these functions is to notice that

$$A_n^{-1}(m) = k \text{ iff } \underbrace{A_{n-1}^{-1} \circ \cdots \circ A_{n-1}^{-1}}_{k \text{ times}}(m) = 1. \tag{7}$$

There exists many versions of Ackermann's function. One possible definition is $Ack(n) \stackrel{\text{def}}{=} A_n(2)$. It is well-known that $Ack(n)$ dominates any primitive recursive function of n. Thus it follows from classical complexity-theoretic results that halting problems for Turing machines running in time or space bounded by $Ack(n)$ (n being the size of the input) cannot be decided in primitive recursive time or space.

[2] S terminates iff it is not simulation-equivalent with a simple loop.

3.2 Weakly computing A_n with expanders

We construct a family E_2, E_3, ... of *expanders*, channel systems that weakly compute $A_2(k)$, $A_3(k)$, ... As illustrated in Fig. 2, E_n, the nth expander, uses n channels: c_1 is the "output

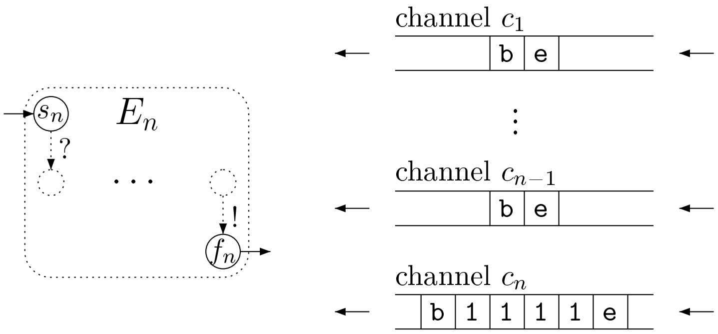

Figure 2: Interface for expander E_n

channel" (in which the system will write the result $A_n(k)$), c_n is the "input channel" (from which the argument k is read), and channels c_2 to c_{n-1} are used to store auxiliary results. E_n has one starting and one ending state, called s_n and f_n respectively.

These systems use a simple encoding for numbers: $k \in \mathbb{N}$ is encoded as a string $\lceil k \rceil$ over the alphabet $\Sigma_0 \stackrel{\text{def}}{=} \{\mathtt{1}, \mathtt{b}, \mathtt{e}\}$. Formally, $\lceil k \rceil \stackrel{\text{def}}{=} \mathtt{b1}^k\mathtt{e}$ is made of k letters "1" surrounded by one "b"egin and one "e"nd marker. For example, the channels in Fig. 2 contain respectively $\lceil 0 \rceil$, ..., $\lceil 0 \rceil$, and $\lceil 4 \rceil$.

Before explaining the construction of the expanders, we describe the simple transferring devices $T_2, T_3, \ldots$ they contain. This illustrates the way encodings $\lceil k \rceil$ of numbers are used. The T_n's, defined in Fig. 3, start in state t_n and end in state x_n after they have transfered the contents of channel c_1 into channel c_n (assuming c_1 contains at least $\lceil 1 \rceil$). In Fig. 3 (and in future constructions) we sometimes omit depicting all intermediate states when their names are not required in the proof.

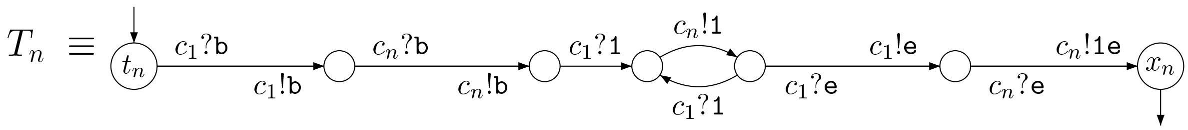

Figure 3: T_n, channel system for transferring c_1 into c_n $(n > 1)$

Formally, the lossy behaviors of T_n can be characterized by:

Proposition 3.1 *For all $n \geq 2$,*

$$\langle t_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle \xrightarrow{*} \langle x_n, \lceil b_1 \rceil, \ldots, \lceil b_n \rceil \rangle \text{ iff } \begin{cases} a_1 \geq 1,\ b_1 = 0, \\ b_i \leq a_i \text{ for } 1 < i < n, \text{ and} \\ b_n \leq a_1. \end{cases} \tag{8}$$

Proof. Omitted. □

Note that the design of T_n ensures that it blocks when c_1 contains $\lceil 0 \rceil$. This property is used in the proof of Lemma 3.4.

We now move to the expanders themselves. The internals of E_n are given in Fig. 4.

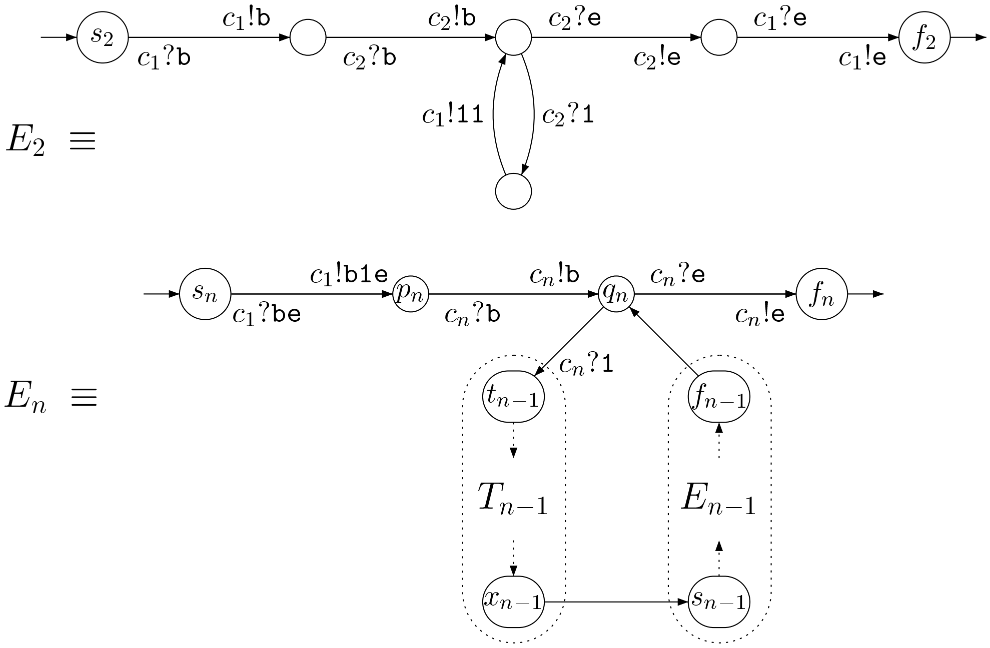

Figure 4: Expanders $E_2, E_3, \ldots$

For $n > 2$, E_n contains E_{n-1} and T_{n-1} as subsystems. This is because E_n implements equation (2): every time a 1 is consumed from c_n, E_{n-1} is run and the result is transfered (by T_{n-1}) from c_1 to c_{n-1} so that E_{n-1} can be applied again.

It is easy to convince oneself that the perfect (non-lossy) behavior of E_n is to compute A_n in the following formal sense:

$$\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil k \rceil \rangle \xrightarrow{*}_{\text{perf}} \langle f_n, w_1, \ldots, w_n \rangle \text{ iff } \begin{cases} w_1 = \lceil A_n(k) \rceil \\ \text{and} \\ w_2 = \ldots = w_n = \lceil 0 \rceil. \end{cases} \tag{9}$$

Indeed, for $n > 2$ a perfect run has the following form (where configurations are displayed in vector notation):

$$
\begin{array}{ccccccccccccc}
s_n & & p_n & & q_n & & t_{n-1} & & x_{n-1} & & s_{n-1} & & f_{n-1} \\
\lceil 0 \rceil & & \lceil 1 \rceil & & \lceil 1 \rceil & & \lceil 1 \rceil & & \lceil 0 \rceil & & \lceil 0 \rceil & & \lceil A_{n-1}(1) \rceil \\
\vdots & \rightarrow & \vdots & \rightarrow & \vdots & \rightarrow & \vdots & \xrightarrow{+} & \vdots & \rightarrow & \vdots & \xrightarrow{+} & \vdots \\
\lceil 0 \rceil & & \lceil 0 \rceil & & \lceil 0 \rceil & & \lceil 0 \rceil & & \lceil 1 \rceil & & \lceil 1 \rceil & & \lceil 0 \rceil \\
\lceil k \rceil & & \lceil k \rceil & & \mathtt{1}^k\mathtt{eb} & & \mathtt{1}^{k-1}\mathtt{eb} & & \mathtt{1}^{k-1}\mathtt{eb} & & \mathtt{1}^{k-1}\mathtt{eb} & & \mathtt{1}^{k-1}\mathtt{eb}
\end{array}
$$

$$
\begin{array}{cccccccc}
 & x_{n-1} & & f_{n-1} & & f_{n-1} & & f_n \\
 & \lceil 0 \rceil & & \lceil A_{n-1}(A_{n-1}(1)) \rceil & & \lceil A_n(k) \rceil & & \lceil A_n(k) \rceil \\
\xrightarrow{+} & \vdots & \xrightarrow{+} & \vdots & \xrightarrow{+} & \vdots & \xrightarrow{+} & \vdots \\
 & \lceil A_{n-1}(1) \rceil & & \lceil 0 \rceil & & \lceil 0 \rceil & & \lceil 0 \rceil \\
 & \mathtt{1}^{k-2}\mathtt{eb} & & \mathtt{1}^{k-2}\mathtt{eb} & & \mathtt{eb} & & \lceil 0 \rceil
\end{array}
$$

When perfect behavior is not assumed, E_n still computes A_n, this time in a weak sense, according to the following statement:

Proposition 3.2 *For all $n \geq 2$,*

$$
\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil k \rceil \rangle \xrightarrow{*} \langle f_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle \;\; \textit{iff} \;\; \begin{cases} a_1 \leq A_n(k) \\ \textit{and} \\ a_2 = \ldots = a_n = 0. \end{cases} \tag{10}
$$

Proof. The "$\Leftarrow$" direction is an easy consequence of (9). A detailed proof of the "$\Rightarrow$" direction can be found in the Appendix but the underlying idea is simple: First, if some `1`s are lost at any time during the computation, the final result will end up being smaller because of the monotonicity properties (5). Then, if a `b` or a `e` marker is lost, the system can never recover it and will fail to reach a configuration where all channels contain encodings of numbers. $\square$

Note that the difference between (9) and (10) does not only come from the replacement of a "$= A_n(k)$" by a "$\leq A_n(k)$": (9) guarantees that the w_is are encodings of numbers, while (10) assumes this.

3.3 Weakly computing A_n^{-1} with folders

Folder systems $F_2, F_3, \ldots$ are channel systems that weakly compute the A_n^{-1}'s. Here channel c_1 is the input channel and c_n is the output, while channels c_2 to c_{n-1} store auxiliary results. The definition of the F_n's, given in Fig. 5, is based on (7). It uses transferring systems T'_n: these systems are variants of the T_n's and move the contents of channel c_n into channel c_1 (instead of the other way around).

When possibly lossy behaviors are considered, F_n weakly computes A_n^{-1} in the following formal sense:

Proposition 3.3 *For all $n \geq 2$,*

$$
\langle s'_n, \lceil m \rceil, \lceil 0 \rceil, \ldots, \lceil 0 \rceil \rangle \xrightarrow{*} \langle f'_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle \;\; \textit{iff} \;\; \begin{cases} A_n(a_n) \leq m, \;\; \textit{and} \\ a_1 = \ldots = a_{n-1} = 0. \end{cases} \tag{11}
$$

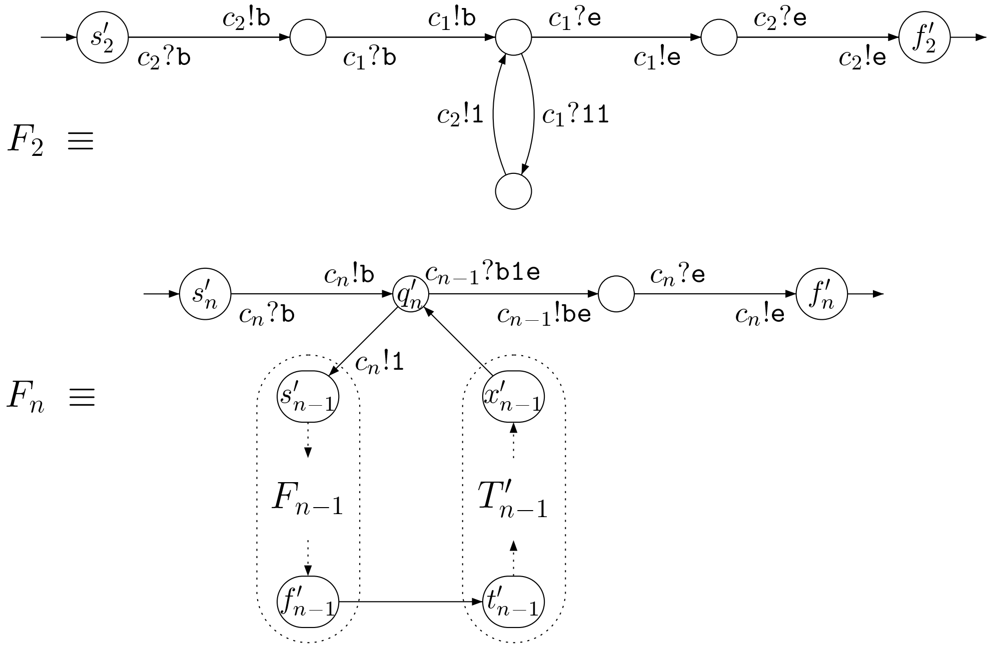

Figure 5: Folders $F_2, F_3, \ldots$

Proof. We prove the "$\Rightarrow$" direction in the Appendix and omit the easier "$\Leftarrow$" direction. $\square$

One additional property of the construction will be useful in the following:

Lemma 3.4 *E_n and F_n have no infinite run, regardless of the initial configuration.*

Proof. We first deal with the E_n's by induction over n. E_2 terminates since its loop consumes from c_2. For $n > 2$, a run of E_n cannot visit q_n infinitely often (this would consume infinitely from c_n) and, by ind. hyp., cannot contain an infinite subrun of E_{n-1} (obviously, the T_{n-1} part must terminate too).

For the F_n's, we first observe (again using induction on n) that, if c_1 contains at least one 1, then a run from s'_n to f'_n removes strictly more 1's than it writes back. Finally, traversing T_{n-1} must move some 1's to c_1 (or lose them). $\square$

4 The hardness results

Expander and folder systems can now be used to prove Theorem 2.3.

4.1 Hardness for reachability

Let's consider a Turing Machine (a TM) M that is started on a blank work tape of size m (for some m) and that never goes beyond the allocated workspace. One can build a channel system S that simulates M using as workspace channel c_1 initially filled with $\mathtt{b1}^m\mathtt{e}$. We do not describe further the construction of S since it follows the standard simulation of TM's by channel systems (from [BZ83]) [3]. Since M never goes beyond the allocated workspace, the

[3]And since TM's are not really required and could be replaced by perfect channel systems that preserve the size of the channel contents.

transition rules of S always write exactly as many messages as they read, so that *if no loss occurs* the channel always contains the same number of letters. The resulting system has two types of runs: perfect runs where M is simulated faithfully, and lossy runs that do not really simulate M but where messages have been irremediably lost.

Now, in order to know whether M accepts in space m, there only remains to provide S with enough workspace, and to look at runs where no message is lost. This is exactly what is done by S^n_M, depicted in Fig. 6, where expander E_n provides a potentially large $\lceil m \rceil$ in channel c_1 and folder F_n is used to check that no message has been lost.

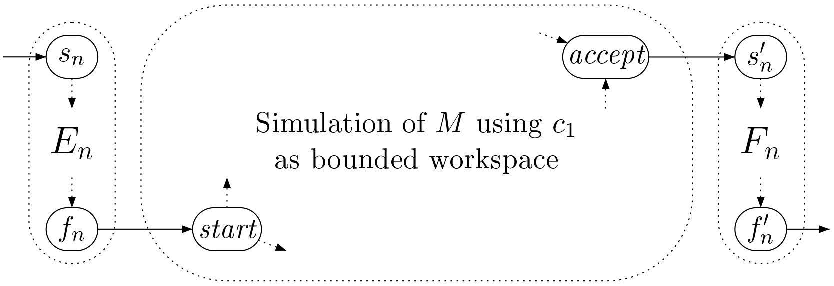

Figure 6: S^n_M: simulating Turing machine M with huge workspace

Thus any run of S^n_M of the form $\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil 2 \rceil\rangle \xrightarrow{*} \langle f'_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil 2 \rceil\rangle$ is perfect and must visit both $\langle \mathit{start}, \lceil \mathit{Ack}(n) \rceil, \lceil 0 \rceil, \ldots, \lceil 0 \rceil\rangle$ and $\langle \mathit{accept}, \lceil \mathit{Ack}(n) \rceil, \lceil 0 \rceil, \ldots, \lceil 0 \rceil\rangle$. Hence

Proposition 4.1 *$\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil 2 \rceil\rangle \xrightarrow{*} \langle f'_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil 2 \rceil\rangle$ in S^n_M iff M accepts in space $\mathit{Ack}(n)$.*

Therefore, since S^n_M has size $O(n + |M|)$, reachability for lossy channel systems is at least as hard as termination for TM's running in Ackermann space. Hence

Corollary 4.2 *Reachability for lossy channel systems has nonprimitive recursive complexity.*

4.2 Hardness for termination

The second hardness result uses a slight adaptation of our previous construction, and relies on the following simulation (Fig. 7).

Here S'^n_M fills two channels with $\lceil \mathit{Ack}(n) \rceil$: c_1 used as before as working space, and c_0 used as a countdown that ensures termination of the simulation of M. Every time one step of M is simulated, S'^n_M consumes one `1` from c_0. When the accepting state of M is reached, S'^n_M moves to s'_n where it uses F_n to check that c_1 does contain $\lceil \mathit{Ack}(n) \rceil$ (i.e. the simulation was faithful). If the check succeeds, S'^n_M can enter a loop. Therefore, a run of S'^n_M terminates when the simulation is not perfect, or when M does not accept in at most $\mathit{Ack}(n)$ steps.

Proposition 4.3 *S'^n_M has an infinite run from $\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil 2 \rceil\rangle$ iff M accepts in time $\mathit{Ack}(n)$.*

Proof. Clearly, S'^n_M can only reach the final loop if the simulation is faithful, and halts in at most $\mathit{Ack}(n)$ steps. There remains to show that the unfaithful, lossy, behaviors are terminating: Lemma 3.4 deals with the E_n and F_n part of S'^n_M, the duplication gadget (between f_n and *start*) obviously terminates, and we solve the problem for the simulation part by programming it in such a way that the rotation of the tape (necessary for simulating a

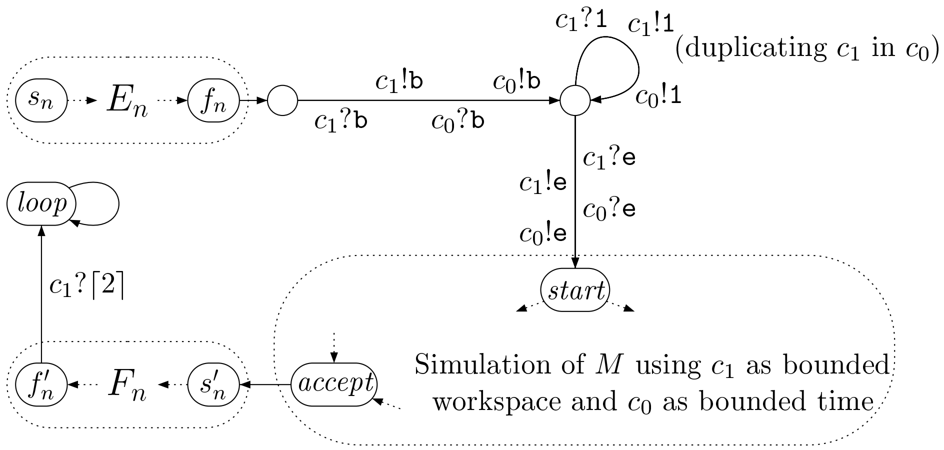

Figure 7: ${S'}^n_M$: another simulation of M with huge workspace

TM) cannot induce non-termination. One way [4] to achieve this is to use two copies (one positive and one negative) of the TM alphabet: in "+" mode, the simulation reads +-letters and writes back their −-twin. Only when an actual TM step is performed does ${S'}^n_M$ swap from "+" mode to "−" mode and *vice versa*. More details can be found in section 5 where the same trick is used. □

This shows that termination for lossy channel systems is at least as hard as termination for TM's runnings in Ackermann time. Hence

Corollary 4.4 *Termination for lossy channel systems has nonprimitive recursive complexity.*

5 Systems with only one channel

Our construction used several channels for clarity, not out of necessity, and our result still holds when we restrict ourselves to lossy channel systems with only one channel. This is one more application of the slogan "*lossy systems with k channels can be encoded into lossy systems with one channel*". The encoding given in [AJ96a, Section 4.5] preserves the existence of runs that visit a given control state infinitely often. Below we give another encoding that further preserves termination and reachability. It uses standard techniques (e.g. from the study of TM's with k tapes) and the only original aspect is the lossy behavior of our systems.

Consider a system $S = \langle Q, C, \Sigma, \Delta\rangle$ that uses channels $c_1, \ldots, c_k$. We simulate S by a system $S' = \langle Q', \{c\}, \Sigma', \Delta'\rangle$ that uses one single channel c. Without loss of generality we assume a different subalphabet is used with every channel of S (i.e. Σ is partitioned in disjoint alphabets $\Sigma_1 \cup \cdots \cup \Sigma_k$). The encoding uses a larger alphabet where k markers $\#_1, \ldots, \#_k$ have been added, and where every letter comes in two copies (a *positive* and a *negative* one). Formally $\Sigma' \stackrel{\text{def}}{=} (\Sigma \cup \{\#_1, \ldots, \#_k\}) \times \{+, -\}$ and a pair $(x, \alpha) \in \Sigma'$ is written shortly x^α. For $x \in \Sigma_i$, an occurrence of some x^α in c means "one x in c_i" (and the *polarity* α is only used for bookkeeping purposes).

[4] An alternative solution would use c_0 as a countdown *for channel system steps* rather than TM steps, but Prop. 4.3 would have to be reworded in a clumsy way.

A k-tuple $\langle w_1, \ldots, w_k \rangle \in \Sigma_1^* \times \cdots \times \Sigma_k^*$ is encoded as $(\#_1 w_1 \#_2 w_2 \ldots \#_k w_k)^\alpha$ where a same polarity α is used to label all letters. For example, $\langle \mathtt{ab}, \varepsilon, \mathtt{ddc} \rangle$ is coded as $U = \#_1^+ \mathtt{a}^+ \mathtt{b}^+ \#_2^+ \#_3^+ \mathtt{d}^+ \mathtt{d}^+ \mathtt{c}^+$ under positive polarity.

Fig. 8 shows how two example rules from Δ (left-hand side) are encoded in S' (right-hand side). On this figure, four loops (marked by $*$) provide for the rotation of the contents of c

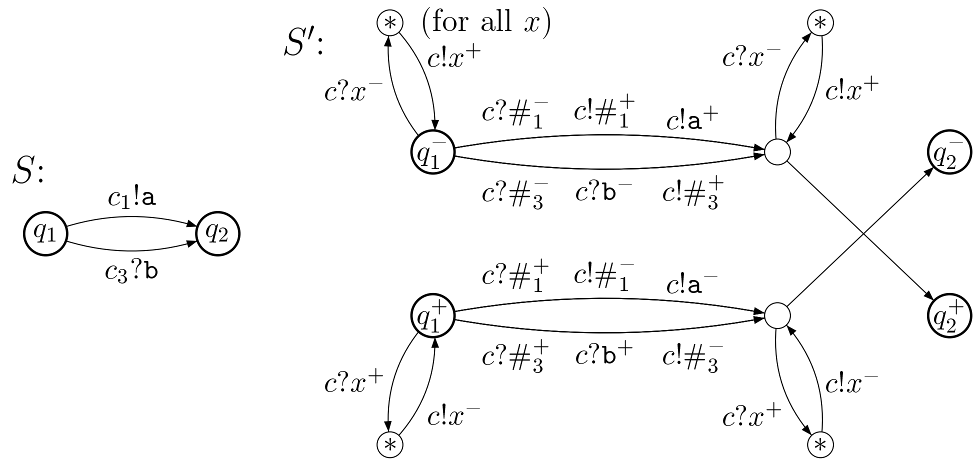

Figure 8: From S to S': encoding k channels into one

(with a change of polarity): they are shorthands for several loops as indicated by the "for all x" comment. A state q from S gives rise to two copies q^+ and q^- in S': q^+ reads positive letters and writes negative letters, thus preventing non-termination induced by the rotation loops, q^- does the converse, and S' changes polarity each time a step of S has been simulated.

Now, if W and V are encodings of $\langle w_1, \ldots, w_k \rangle$ and $\langle v_1, \ldots, v_k \rangle$, if $\alpha \in \{+, -\}$, we have $\langle q, w_1, \ldots, w_k \rangle \xrightarrow{*}_{\text{perf}} \langle q', v_1, \ldots, v_k \rangle$ (in S) iff $\langle q^\alpha, W^\alpha \rangle \xrightarrow{*}_{\text{perf}} \langle q'^\beta, V^\beta \rangle$ for some $\beta \in \{+, -\}$. This extends to lossy behaviors:

$$\langle q, w_1, \ldots, w_k \rangle \xrightarrow{*} \langle q', v_1, \ldots, v_k \rangle \text{ iff } \langle q^\alpha, W^\alpha \rangle \xrightarrow{*} \langle q'^\beta, V^\beta \rangle \text{ for some } \beta, \tag{12}$$

$$S \text{ terminates from } \langle q, w_1, \ldots, w_k \rangle \text{ iff } S' \text{ terminates from } \langle q^\alpha, W^\alpha \rangle. \tag{13}$$

Note that these equivalences only hold for W and V that are correct encodings of k-tuples, and for q, q' that are original states from Q. S' has behaviors that do not correspond to behaviors of S in the sense of (12), e.g. when it gets blocked in new states, or when it loses one of the $\#_i$ markers.

Corollary 5.1 *Reachability and termination for lossy single-channel systems have nonprimitive recursive complexity.*

6 Conclusion

There exist several constructions in the literature where a problem P is shown undecidable for lossy channel systems by simulating a Turing machine in such a way that the faithfulness of the simulation can be ensured, or checked, or rewarded in some way. Our construction uses similar tricks since it first builds a nonprimitive recursive number of messages, and later checks that none has been lost. Still, there are a number of new aspects in our construction, and this explains why it is the first complexity result for decidable problems on lossy channel systems.

Acknowledgements

We are grateful to Alain Finkel who brought this topic to our attention, to Petr Jančar who suggested major improvements on an earlier draft, and to the anonymous referee who helped us improve section 5.

A Appendix: Proof of Prop. 3.2

The difficulty with the "⇒" direction of Prop. 3.2 is that we have to consider lossy behaviors that need not respect the logic of the E_n systems we designed. However, when we restrict our attention to behaviors that do not lose the `b` and `e` markers, managing the problem becomes feasible.

We start with the following lemma:

Lemma A.1 *If a run* $\langle s_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil\rangle \xrightarrow{*} \langle f_n, w_1, \ldots, w_n\rangle$ *of* E_n *is such that every* w_i *contains one* `b` *and one* `e`*, then every* w_i *has the form* `b1*e`*, i.e. encodes a number.*
The same holds for a run $\langle t_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil\rangle \xrightarrow{*} \langle x_n, w_1, \ldots, w_n\rangle$ *of* T_n.

Proof. Since our systems always write a `b` or a `e` after they consumed one, saying that every w_i contains one `b` and one `e` means that losses did not concern the markers. Therefore, it only remains to prove that the pattern `b1*e` is respected, even when some `1`'s have been lost.

E_2 has to consume all of $\lceil a_1 \rceil$ and $\lceil a_2 \rceil$ and what it writes in c_1 and c_2 encodes a number even after losses. The same reasoning applies to the channels c_1 and c_n of T_n, while the other channels are untouched (and losses there respect the `b1*e` pattern).

For E_n with $n > 2$, we proceed by induction over n. For c_n, observe that all of a_n is consumed and replaced by $\lceil 0 \rceil$. For the other channels (c_1 to c_{n-1}), they all contain a number when the run first reaches q_n and, since the lemma holds for T_{n-1} and E_{n-1} (by ind. hyp.), this remains the case every time the run revisits q_n, until it eventually reaches f_n. Then all channels contain encodings of numbers. □

We are now ready to prove that

$$\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil k \rceil\rangle \xrightarrow{*} \langle f_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil\rangle \Rightarrow \begin{cases} a_1 \leq A_n(k), \text{ and} \\ a_2 = \ldots = a_n = 0. \end{cases} \tag{H$_n$}$$

by induction over n. The base case $n = 2$ is left to the reader: a simple inspection of E_2 shows it weakly computes $A_2(k) = 2k$.

For $n > 2$ we consider a run

$$\langle s_n, \lceil 0 \rceil, \ldots, \lceil 0 \rceil, \lceil k \rceil\rangle \xrightarrow{*} \langle f_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil\rangle \tag{14}$$

and isolate the configurations where (14) visits q_n and f_{n-1} by writing it under the form

$$\begin{aligned} \langle s_n, \lceil 0 \rceil, \ldots, \lceil k \rceil\rangle &\xrightarrow{*} \langle q_n, w_1^0, \ldots, w_n^0\rangle \xrightarrow{*} \langle f_{n-1}, v_1^1, \ldots, v_n^1\rangle \\ &\to \langle q_n, w_1^1, \ldots, w_n^1\rangle \xrightarrow{*} \langle f_{n-1}, v_1^2, \ldots, v_n^2\rangle \to \cdots \xrightarrow{*} \langle f_{n-1}, v_1^m, \ldots, v_n^m\rangle \\ &\to \langle q_n, w_1^m, \ldots, w_n^m\rangle \xrightarrow{*} \langle f_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil\rangle. \end{aligned}$$

Since the $\mathtt{b}$ and $\mathtt{e}$ markers are not lost in (14), we can state that all w_n^i and v_n^i have the form $\mathtt{1}^{k_i}\mathtt{eb}$ (resp. $\mathtt{1}^{k'_i}\mathtt{eb}$). Since the transition leaving q_n for t_{n-1} consumes one $\mathtt{1}$ from c_n, one sees that

$$k \geq k_0 > k'_1 \geq k_1 > k'_2 \geq k_2 > \cdots \geq k_m,$$

implying $m \leq k$. Finally $a_n = 0$ (since f_n can only be reached by consuming $\mathtt{e}$ from c_n) and this concludes the proof for the part that concerns c_n.

When we consider the other channels (c_1 to c_{n-1}) the proof of Lemma A.1 shows that, for $i < n$, the w_i^j and v_i^j all are encodings of numbers.

Furthermore, they satisfy

$$w_1^j \sqsubseteq \lceil \underbrace{A_{n-1} \circ \cdots A_{n-1}}_{j \text{ times}}(1) \rceil \qquad \text{and} \qquad w_i^j = \lceil 0 \rceil \text{ if } 2 \leq i \leq n-1 \tag{W$_j$}$$

$$v_{n-1}^j \sqsubseteq w_1^{j-1} \qquad \text{and} \qquad v_i^j = \lceil 0 \rceil \text{ if } 1 \leq i \leq n-2. \tag{V$_j$}$$

as we prove by induction on j. The base case, (W$_0$) is a consequence of the assumption (14). Then one shows that (W$_j$) entails (V$_{j+1}$) using Prop. 3.1. Finally, one proves that (V$_j$) and (W$_{j-1}$) entail (W$_j$) using (H$_{n-1}$).

One concludes the proof of (H$_n$) by observing that $m \leq k$ and (W$_m$) entail the right hand side of (H$_n$) because of the monotonicity and expansion properties of A_n stated in (5).

B Appendix: Proof of Prop. 3.3

For the "$\Rightarrow$" direction, we proceed as with the proof of Prop. 3.2, and start with a result mimicking Lemma A.1:

Lemma B.1 *If a run $\langle s'_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle \xrightarrow{*} \langle f'_n, w_1, \ldots, w_n \rangle$ of F_n is such that every w_i contains one* $\mathtt{b}$ *and one* $\mathtt{e}$*, then every w_i encodes a number.*

Proof. Omitted. ☐

We now prove

$$\langle s'_n, \lceil m \rceil, \lceil 0 \rceil, \ldots, \lceil 0 \rceil \rangle \xrightarrow{*} \langle f'_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle \Rightarrow \begin{cases} A_n(a_n) \leq m, \text{ and} \\ a_1 = \ldots = a_{n-1} = 0. \end{cases} \tag{H$'_n$}$$

by induction over n. The base $n = 2$ is easy to see. For $n > 2$, we consider a run $\langle s'_n, \lceil m \rceil, \lceil 0 \rceil, \ldots, \lceil 0 \rceil \rangle \xrightarrow{*} \langle f'_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle$ and isolate the configurations where it visits q'_n and f'_{n-1} by writing it under the form

$$\begin{aligned}
\langle s'_n, \lceil m \rceil, \ldots, \lceil 0 \rceil \rangle &\xrightarrow{*} \langle q'_n, w_1^0, \ldots, w_n^0 \rangle \xrightarrow{*} \langle f'_{n-1}, v_1^1, \ldots, v_n^1 \rangle \\
&\xrightarrow{*} \langle q'_n, w_1^1, \ldots, w_n^1 \rangle \xrightarrow{*} \langle f'_{n-1}, v_1^2, \ldots, v_n^2 \rangle \xrightarrow{*} \cdots \xrightarrow{*} \langle f'_{n-1}, v_1^l, \ldots, v_n^l \rangle \\
&\xrightarrow{*} \langle q'_n, w_1^l, \ldots, w_n^l \rangle \xrightarrow{*} \langle f'_n, \lceil a_1 \rceil, \ldots, \lceil a_n \rceil \rangle.
\end{aligned}$$

We start with the contents of c_n: the w_n^i and v_n^i have the form $\mathtt{eb1}^{k_i}$ and, resp., $\mathtt{eb1}^{k'_i}$. The transition from q'_n to s'_{n-1} writes one $\mathtt{1}$ to c_n, so that, for $i = 1, \ldots, l$, $k_i \leq k'_i \leq 1 + k_{i-1}$. With $k_0 = 0$ and $k_l \geq a_n$ we deduce $l \geq a_n$.

When we consider the contents of the other channels, the w_i^j's and v_i^j's encode numbers for $j = 1, \ldots, n-1$ (by Lemma B.1). Using (H'_{n-1}) one shows, by induction on i, that $w_i^j = \lceil 0 \rceil$ for $j = 2, \ldots, n-1$, and $v_i^j = \lceil 0 \rceil$ for $j = 1, \ldots, n-2$, so that $a_1 = \ldots = a_{n-1} = 0$. Then, writing $w_i^1 = \lceil m_i \rceil$ and $v_i^{n-1} = \lceil x_i \rceil$, ($\mathrm{H}'_{n-1}$) further entails

$$\begin{array}{ccc} A_{n-1}(x_1) \leq m_0, & A_{n-1}(x_2) \leq m_1 \leq x_1, & \ldots \\ A_{n-1}(x_l) \leq m_{l-1} \leq x_{l-1}, & m_l \leq x_l. & \end{array}$$

so that, by monotonicity of A_{n-1},

$$\underbrace{A_{n-1} \circ \cdots A_{n-1}}_{l \text{ times}}(m_l) \leq m_0.$$

Since $1 \leq m_l$ (because finally leaving q'_n consumes $\lceil 1 \rceil$ from c_{n-1}), $m_0 \leq m$, $a_n \leq l$ and (5) entail that $A_n(a_n) \leq m$, completing the proof.

References

[AAB99] P. A. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proc. 5th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99), Amsterdam, The Netherlands, Mar. 1999*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 1999.

[ABJ98] P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98), Vancouver, BC, Canada, June-July 1998*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer, 1998.

[ABPJ00] P. A. Abdulla, C. Baier, S. Purushothaman Iyer, and B. Jonsson. Reasoning about probabilistic lossy channel systems. In *Proc. 11th Int. Conf. Concurrency Theory (CONCUR'2000), University Park, PA, USA, Aug. 2000*, volume 1877 of *Lecture Notes in Computer Science*, pages 320–333. Springer, 2000.

[AČJT00] P. A. Abdulla, K. Čerāns, B. Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1/2):109–127, 2000.

[AJ96a] P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.

[AJ96b] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[AK95] P. A. Abdulla and M. Kindahl. Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In *Proc. 6th Int. Conf. Theory of Concurrency (CONCUR'95), Philadelphia, PA, USA, Aug. 1995*, volume 962 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 1995.

[AKP97] P. A. Abdulla, M. Kindahl, and D. Peled. An improved search strategy for lossy channel systems. In *Proc. Joint Int. Conf. Formal Description Techniques and Protocol Specification, Testing, and Verification (FORTE/PSTV'97), Osaka, Japan, Nov. 1997*, pages 251–264. Chapman & Hall, 1997.

[Boc78] G. von Bochmann. Finite state description of communication protocols. *Computer Networks and ISDN Systems*, 2:361–372, 1978.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.

[CFP96] G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[DFS98] C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98), Aalborg, Denmark, July 1998*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer, 1998.

[DJS99] C. Dufourd, P. Jančar, and Ph. Schnoebelen. Boundedness of Reset P/T nets. In *Proc. 26th Int. Coll. Automata, Languages, and Programming (ICALP'99), Prague, Czech Republic, July 1999*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1999.

[Fin94] A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135, 1994.

[FS01] A. Finkel and Ph. Schnoebelen. Well structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.

[Hac76] M. Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2(1):77–95, 1976.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.

[Jan01] P. Jančar. Nonprimitive recursive complexity and undecidability for Petri net equivalences. *Theoretical Computer Science*, 256(1–2):23–30, 2001.

[May00] R. Mayr. Undecidable problems in unreliable computations. In *Proc. 4th Latin American Symposium on Theoretical Informatics (LATIN'2000), Punta del Este, Uruguay, Apr. 2000*, volume 1776 of *Lecture Notes in Computer Science*, pages 377–386. Springer, 2000.

[MM81] E. W. Mayr and A. R. Meyer. The complexity of the finite containment problem for Petri nets. *Journal of the ACM*, 28(3):561–576, 1981.

[Sch01] Ph. Schnoebelen. Bisimulation and other undecidable equivalences for lossy channel systems. In *Proc. 4th Int. Symp. Theoretical Aspects of Computer Software (TACS'2001), Sendai, Japan, Oct. 2001*, volume 2215 of *Lecture Notes in Computer Science*, pages 385–399. Springer, 2001.