The Verification of Probabilistic Lossy Channel Systems

Ph. Schnoebelen

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

Abstract. Lossy channel systems (LCS's) are systems of finite state automata that communicate via unreliable unbounded fifo channels. Several probabilistic versions of these systems have been proposed in recent years, with the two aims of modeling more faithfully the losses of messages, and circumventing undecidabilities by some kind of randomization. We survey these proposals and the verification techniques they support.

1 Introduction

Channel systems are systems of finite state automata that communicate via asynchronous unbounded fifo channels. An example, S_{exmp}, is depicted in Fig. 1.

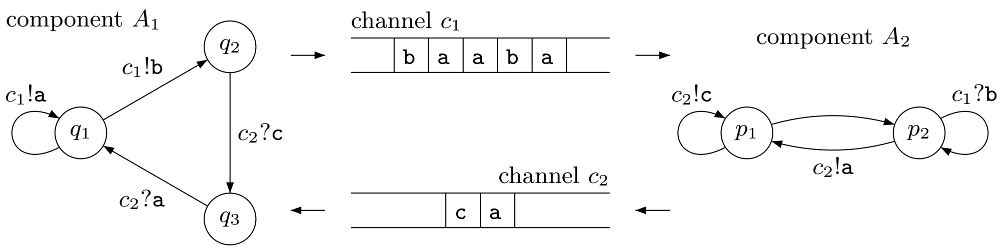

Fig. 1. S_{exmp}: a channel system with two component automata and two channels

They are a natural model for asynchronous communication protocols and, indeed, they form the semantical basis of protocol specification languages such as SDL and Estelle.

The behaviour of a system like S_{exmp} is as expected: component A_1 may move from q_1 to q_2 by sending a `b` message to channel c_1 where it will be enqueued (channels are fifo buffers). Then A_1 may move from q_2 to q_3 if it can read a `c` from channel c_2, which is only possible if the channel is not empty and the first available message is a `c`, in which case the message is dequeued (consumed). The two components, A_1 and A_2, evolve asynchronously.

That channel systems are a *bona fide* model of computation, indeed a Turing-powerful one, was pointed out by Brand and Zafiropulo [BZ83]. This is easy to see: the channels are unbounded and one channel can simulate the work-tape of a Turing machine. One does not have a real scanning and overwriting head that moves back and forth along this "work-tape", but this can be simulated by rotating the contents of the channel to position oneself on any required cell (one keeps track of where the reading head is currently sitting by means of some extra marking symbols).

As an immediate corollary, a Rice Theorem can be stated: "*all nontrivial behavioral properties are undecidable for channel systems*". Hence fully algorithmic verification (model checking) of arbitrary channel systems cannot be achieved. One is left with investigating restricted methods (that only deal with subclasses of the general model, e.g. systems with a bound on the size of the channels) or approximate methods (that admit false negatives) or semi-algorithmic methods (that may fail to terminate).

Lossy channels. A few years ago Finkel [Fin94] and, independently, Abdulla and Jonsson [AJ96b], introduced *lossy channel systems* (LCS's), a very interesting class of channel systems. In lossy systems, messages can be lost while they are in transit, without any notification. These lossy systems are the natural model for fault-tolerant protocols where the communication channels are not supposed to be reliable.

Surprisingly, several verification problems become decidable when one assumes channels are lossy: termination, reachability, safety properties over traces, inevitability properties over states, and several variant problems are decidable for lossy channel systems [Fin94,AK95,CFP96,AJ96b,MS02].

This does not mean that lossy channel systems are an artificial model where, since no communication can be fully enforced, everything becomes trivial. To begin with, many important problems are undecidable: recurrent reachability properties are undecidable, so that model checking of liveness properties is undecidable too [AJ96a]. Furthermore, boundedness is undecidable [May00], as well as all behavioral equivalences [Sch01]. Finally, none of the decidable problems listed in the previous paragraph can be solved in primitive recursive time [Sch02]!

Probabilistic lossy channel systems. It is natural to see message losses as some kind of faults having a probabilistic behaviour. This idea, due to Purushothaman Iyer and Narasimha, led to the introduction of the first Markov chain model for lossy channel systems [PN97].

There are two different benefits one can expect from investigating probabilistic versions of lossy channel systems: (1) they are a more realistic model, where quantitative information on faults is present, or (2) they are a more tractable model, where randomisation effectively rules out some malicious behaviours.

The verification of probabilistic lossy channel systems is a challenging problem because the underlying objects are countably infinite Markov chains (or Markovian decision processes). Furthermore, these infinite Markov chains are not *bounded*: probabilities of individual transitions can be arbitrarily close to zero.

Below we survey the main existing results on the verification of probabilistic lossy channel systems. The literature we review is still scarce, consisting of [PN97,BE99,ABPJ00,BS03,AR03,BS04]. We try to abstract from specific notational or definitional details and present the above works uniformly. As far as verification is concerned, we also try to abstract from specific algorithmic details [1] and extract the main ideas, in the hope that they can be applied to other infinite-state probabilistic verification problems. A recurrent theme is that decidability results rely on the existence of finite attractors.

Plan of this chapter. In section 2, we recall the main definitions and results on classical (non-probabilistic) lossy channel systems. The Markovian models for probabilistic lossy channel systems are given in section 3. We describe qualitative verification in section 4 and quantitative verification in section 5. Finally, in section 6, we present a Markovian decision process model and some results on adversarial verification.

2 Lossy channel systems

2.1 Perfect channel systems

The simplest way to present lossy channel systems is to start with *perfect*, i.e. not lossy, channel systems.

Definition 2.1 (Channel system). *A* channel system *(with m channels) is a tuple $S = \langle Q, C, \Sigma, \Delta, \sigma_0 \rangle$ where*
– $Q = \{r, s, \ldots\}$ *is a finite set of* control locations *(or* control states*),*
– $C = \{c_1, \ldots, c_m\}$ *is a finite set of m* channels*,*
– $\Sigma = \{a, b, \ldots\}$ *is a finite alphabet of* messages*,*
– $\Delta \subseteq Q \times C \times \{?, !\} \times \Sigma \times Q$ *is a finite set of* rules*.*

A rule $\delta \in \Delta$ of the form $(s, c, ?, a, r)$ or $(s, c, !, a, r)$ is written "$s \xrightarrow{c?a} r$" (resp. "$s \xrightarrow{c!a} r$") and means that S can move from control location s to r by reading a from (resp. writing a to) channel c. Reading a is only possible if c is not empty and its first available message is a.

Remark 2.2. Def. 2.1 assumes there is only one component automaton in a channel system. This is no loss of generality since several components can be combined into a single one via a classical asynchronous product of automata. □

[1] The papers we survey are mostly theoretical and the algorithms they propose have not yet been implemented and tested on actual examples.

Remark 2.3. Def. 2.1 further assumes that all rules either consume or produce one message. Again, this is no loss of generality and internal rules (no message involved), or rules consuming and producing several messages, or rules testing for emptiness of a channel, etc., could be accounted for. □

The operational semantics of S is given via a transition system $\mathcal{T}_{\rm perf}(S) = \langle Conf, \rightarrow_{\rm perf}\rangle$ where $Conf = Q \times \Sigma^{*C}$ is the set of configurations (with typical elements $\sigma, \theta, \ldots$), and $\rightarrow_{\rm perf} \subseteq Conf \times Conf$ is the unlabelled transition relation.

A *configuration* of S is a pair $\sigma = \langle r, U\rangle$ where $r \in Q$ is a control location and $U \in \Sigma^{*C}$ is a *channel contents*, i.e. a C-indexed vector of Σ-words: for any $c \in C$, $U(c) = u$ means that c contains u. The void channel contents, where every channel contains the empty word ε, is also denoted ε. The configuration of $S_{\rm exmp}$ in Fig. 1 has channel contents $U = \{c_1 \mapsto \texttt{abaab}; c_2 \mapsto \texttt{ca}\}$.

The possible transitions between configurations are given by the rules of S. Formally, for $\sigma, \sigma' \in Conf$, we have $\sigma \rightarrow_{\rm perf} \sigma'$ iff either

reads: σ is some $\langle s, U\rangle$, there is a rule $s \xrightarrow{c?a} r$ in Δ, $U(c)$ is some $a.u'$, and $\sigma' = \langle r, U\{c \mapsto u'\}\rangle$ (using the standard notation $U\{\_ \mapsto \_\}$ for variants).

writes: σ is some $\langle s, U\rangle$, there is a rule $s \xrightarrow{c!a} r$ in Δ, $U(c)$ is some $u \in \Sigma^*$, and $\sigma' = \langle r, U\{c \mapsto u.a\}\rangle$.

We write $\sigma \xrightarrow{\delta}_{\rm perf} \sigma'$ when we want to explicit that rule $\delta \in \Delta$ allows the step between σ and σ', and let $En(\sigma)$ denote the set $\{\delta \in \Delta \mid \sigma \xrightarrow{\delta}_{\rm perf}\}$ of rules that are *enabled* in configuration σ.

2.2 Lossy channel systems

Lossy channel systems are channel systems where messages can be lost while they are in transit. A first formal definition was proposed by Finkel (who named them *completely specified protocols*) [Fin94]. Finkel defined them as the subclass of channel systems where it is required that for any control state $s \in Q$, any channel $c \in C$ and any message $a \in \Sigma$, there is a rule $s \xrightarrow{c?a} s$. These rules implement the losses by always allowing the removal, without any modification of the current control state, of whatever message is at the front of any buffer.

Later Abdulla and Jonsson proposed a different definition where, rather than requiring special rules implementing the message losses, we have an altered operational semantics [AJ96b]. While this provides for essentially the same behaviour, their definition is much more tractable mathematically (see Remark 2.4) and this is the one we adopt below.

Formally, given two channel contents U and U', we write $U \sqsubseteq U'$ when U can be obtained from U' by removing an arbitrary number of messages at arbitrary places in U'. Thus $U \sqsubseteq U'$ iff for all $c \in C$, $U(c)$ is a *subword* of $U'(c)$. This extends into a relation between configurations:

$$\langle s, U\rangle \sqsubseteq \langle s', U'\rangle \overset{\rm def}{\Leftrightarrow} s = s' \wedge U \sqsubseteq U'. \tag{1}$$

Observe that $U \sqsubseteq U$ for all U (by removing no message), and that $\sqsubseteq$ is a partial ordering between channel contents and between configurations. Higman's lemma [Hig52] states it is a well-quasi-ordering (a *wqo*). Thus sets of configurations have a finite number of minimal elements.

It is now possible to define lossy steps, written $\sigma \xrightarrow{\delta}_{\text{loss}} \sigma'$, with

$$\sigma \xrightarrow{\delta}_{\text{loss}} \sigma' \stackrel{\text{def}}{\Leftrightarrow} \sigma \sqsupseteq \theta \xrightarrow{\delta}_{\text{perf}} \theta' \sqsupseteq \sigma' \text{ for some } \theta, \theta' \in \mathit{Conf}. \tag{2}$$

Thus a lossy step is a perfect step possibly preceded and followed by an arbitrary number of message losses.

The *lossy semantics* of a channel system S is the transition system $\mathcal{T}_{\text{loss}}(S) = \langle \mathit{Conf}, \rightarrow_{\text{loss}} \rangle$.

Below we omit the "loss" subscript since we consider the lossy semantics by default (we never omit the "perf" subscript). As usual, $\xrightarrow{+}$ and $\xrightarrow{*}$ will denote the transitive and (resp.) reflexive-transitive closures of the one-step $\rightarrow$ relation.

Remark 2.4. What is mathematically nice in Abdulla and Jonsson's definition is that it induces the following monotonicity property: if $\theta \rightarrow \theta'$ and $\sigma \sqsupseteq \theta$, then $\sigma \rightarrow \theta'$. Similarly, if $\theta' \sqsupseteq \sigma'$, then $\theta \rightarrow \sigma'$. Thus sets of predecessors are upward-closed and sets of successors are downward-closed w.r.t. $\sqsubseteq$. Systems exhibiting such a monotonicity property are said to be *well-structured*, and enjoy general decidability results [AČJT00,FS01]. □

2.3 Verification of lossy channel systems

Several verification problems are decidable for lossy channel systems. In this survey, we only need the decidability of *reachability* and of *control state reachability*. These problems ask:

Reachability:
Given: a LCS S and two configurations σ_0 and σ_f,
Question: does $\sigma_0 \xrightarrow{*} \sigma_f$?

Control state reachability:
Given: a LCS S, a configuration σ_0, and a control state $s \in Q$,
Question: does $\sigma_0 \xrightarrow{*} \langle s, U \rangle$ for some U?

These two problems are equivalent (inter-reducible). Their decidability is shown in [CFP96,AJ96b]. They cannot be solved in primitive recursive time and are thus highly intractable from a worst-case complexity viewpoint [Sch02]. However, there exist clever symbolic methods that can answer reachability questions in many cases [AAB99].

Some other verification problems are undecidable for lossy channel systems. Most notably *Büchi acceptance* and *control state loop*, where one asks:

Büchi acceptance:
Given: a LCS S, a configuration σ_0, and a control state $s \in Q$,
Question: is it possible, starting from σ_0, to visit s infinitely many times, i.e. does $\sigma_0 \xrightarrow{*} \langle s, U_1\rangle \xrightarrow{+} \langle s, U_2\rangle \xrightarrow{+} \ldots$ for an infinite sequence $U_1, U_2, \ldots$?

Control state loop:
Given: a LCS S, a configuration σ_0, and a control state $s \in Q$,
Question: does $\sigma_0 \xrightarrow{*} \langle s, U\rangle \xrightarrow{+} \langle s, U\rangle$ for some U?

These two problems are equivalent (inter-reducible). Their undecidability is proved in [AJ96a]. A corollary is that model checking of liveness properties is undecidable for LCS's.

3 Probabilistic lossy channel systems

Purushothaman Iyer and Narasimha were the first to consider probabilistic variants of LCS's [PN97].

Investigating such variants is rather natural since just saying that "*any message can be lost at any time*" leads to very pessimistic conclusions about the behaviour of channel systems. In reality, many protocols dealing with unreliable channels are designed with the idea that message losses are usually unlikely, so that a bounded number of retries is a sufficient solution in most situations [2]. Capturing these ideas requires models where notions such as "message losses are unlikely" and "most situations" are supported. Hence the introduction of PLCS's, i.e. LCS's where message losses follow some kind of probabilistic distribution.

Definition 3.1. [PN97] *A* probabilistic lossy channel system *(PLCS) is a tuple* $S = \langle Q, C, \Sigma, \Delta, \sigma_0, p_{\text{loss}}, D\rangle$ *where*
– $\langle Q, C, \Sigma, \Delta, \sigma_0\rangle$ *is some* underlying channel system,
– $p_{\text{loss}} \in (0, 1)$ *is a* loss probability*, and*
- $D : \Delta \mapsto (0, \infty)$ *is a* weight function *of the rules.*

3.1 The global-fault model

The semantics of a PLCS S is given in terms of a Markov chain $\mathcal{M}_g(S) = \langle \mathit{Conf}, p_g\rangle$ where *Conf* is the same set of configurations that appears in $\mathcal{T}_{\text{loss}}(S)$ and $p_g : \mathit{Conf} \times \mathit{Conf} \to [0, 1]$ is the transition probability (the g subscript stands for "global", as explained in section 3.2).

Here p_g assigns probabilities to the transitions of $\mathcal{T}_{\text{loss}}(S)$ in accordance with the following principles:

1. A step in $\mathcal{M}_g(S)$ is either a perfect step, or the loss of *one* message.

[2] The *Alternating Bit Protocol* is not so optimistic. As a result, it is not efficient enough for networks where transmission delays are long (e.g., the Internet).

2. In any given configuration, there is a fixed probability p_{loss} that the next step will be the loss of a message. If there are several messages in σ, each of these messages can be lost with equal probability, so that when $\sigma' \sqsubseteq \sigma$ can be obtained by removing one message from σ, we would expect something like

$$p_g(\sigma, \sigma') \;=\; \frac{p_{\text{loss}}}{|\sigma|}, \tag{3}$$

assuming $|\langle s, U\rangle|$ denotes $|U|$, i.e. the number $\sum_{c \in C} |U(c)|$ of messages in U.
3. In any given configuration, there is a fixed probability $1 - p_{\text{loss}}$ that the next step will be a perfect step. The probability that $\delta \in \Delta$ will account for the next step is given by its weight $D(\delta)$ after some normalisation against the other enabled rules. If $\sigma \xrightarrow{\delta}_{\text{perf}} \sigma'$ is a step in $\mathcal{T}_{\text{perf}}(S)$, we would expect something like

$$p_g(\sigma, \sigma') \;=\; (1 - p_{\text{loss}}) \times \frac{D(\delta)}{\sum\limits_{\delta' \in En(\sigma)} D(\delta')}. \tag{4}$$

Turning these principles into a rigorous definition is a tedious and boring task, that we will not repeat here, leaving it to the reader's imagination. Note that several special cases have to be taken into account [3] or simplified away. Most of what we explain below does not depend on these details. For clarity, we assume there is at least one enabled rule in any configuration σ, so that $\sum_{\delta \in En(\sigma)} D(\delta)$ is nonzero.

Remark 3.2. Strictly speaking, the transition system underlying $\mathcal{M}_g(S)$ differs from $\mathcal{T}(S)$: a step in $\mathcal{T}(S)$ combines several steps from $\mathcal{M}_g(S)$ (one per message loss plus one for the perfect step), but this is just an unimportant question of granularity, and essentially the same behaviour is exhibited in both cases. □

We call $\mathcal{M}_g(S)$ the *global-fault* model because it assumes p_{loss} is the probability that the next step is a fault (a message loss) as opposed to a perfect step. A consequence of this assumption is that the fixed fault probability has to be distributed over all messages currently in transit: the more messages are currently in transit, the less likely it is that any single message will be lost in the next step, so that message losses are not independent events.

3.2 The local-fault model

It can be argued that a more realistic model would have p_{loss} applying to any single message, independently of the other messages. This prompted Bertrand

[3] What about losses when U is empty, and perfect steps when $En(\sigma)$ is empty? What about situations where different message losses account for a same $\sigma \to \sigma'$? Or when the reading of a message has the same effect as a loss?

and Schnoebelen [BS03] and, independently, Abdulla and Rabinovich [AR03], to introduce a new model, here called the *local-fault* model, and denoted $\mathcal{M}_l(S)$.

More precisely, $\mathcal{M}_l(S) = \langle Conf, p_l \rangle$ is defined in accordance with the following principles

1. A step in $\mathcal{M}_l(S)$ is a perfect step followed by any given number of message losses (possibly zero).
2. In any given configuration, the perfect step is chosen probabilistically, by normalising the weights of the enabled rules.
3. Then, any message is lost with probability p_{loss} (and kept with probability $1 - p_{\text{loss}}$).
4. Thus, if $\sigma \xrightarrow{\delta}_{\text{perf}} \sigma' \sqsupseteq \sigma''$, we would expect something like

$$p_l(\sigma, \sigma'') \;=\; \frac{D(\delta)}{\displaystyle\sum_{\delta' \in En(\sigma)} D(\delta')} \;\times\; (p_{\text{loss}})^{|\sigma'|-|\sigma''|} \;\times\; (1 - p_{\text{loss}})^{|\sigma''|}. \tag{5}$$

Here too the actual formal definition is more complex because there usually are several ways to reach a same σ'' from a given σ.[4]

3.3 Other models

It is of course possible to define many other Markov chain models for probabilistic channel systems, e.g. with channel-dependent loss probabilities, etc. The two proposals we just discussed strive for minimality.

In the literature, two other models can be found:

1. The undecidability proof in [ABPJ00] assumes that messages can only be lost while they are being enqueued in the channel and not later. (We leave it to the reader to write the definition of the associated Markov chain.) This choice mainly aims at simplifying the technical development of the aforementioned paper. However, it underlines the fact that our two earlier models see losses as occurring inside the channels when other possibilities are of course possible.
2. Abdulla and Rabinovich [AR03] consider different kind of faulty behaviours (not just losses). They allow insertion errors, duplication errors, and corruptions of messages. Such errors were considered in [CFP96] but Abdulla and Rabinovich propose a probabilistic definition of these transmission faults and analyse when decidability is preserved.

4 Qualitative verification of PLCS's

A Markov chain like $\mathcal{M}_g(S)$ or $\mathcal{M}_l(S)$ comes with a standard probability measure on its set of runs (see e.g. [Pan01]). We let $\mathbb{P}_{\sigma_0}(\Diamond\sigma)$ denote the measure

[4] However, compared to the global-fault model, the number of special cases is reduced since losses do not clash with perfect steps and since Eq. (5) tolerates $|\sigma| = 0$.

of the set of runs that start from σ_0 and eventually visit σ. More generally, let $\mathbb{P}_{\sigma_0}(\varphi)$ denote the measure of the set of runs from σ_0 that verify some linear-time property φ.

For verification purposes, proving that $\mathcal{M}$ has $\mathbb{P}_{\sigma_0}(\varphi) = 1$ is almost as good as proving $\mathcal{T}, \sigma_0 \models \varphi$ in the classical (non probabilistic) setting, since it shows that the set of runs where φ does not hold is "negligible". However, such negligible sets can make the difference between a decidable and an undecidable problem. Indeed, qualitative verification of PLCS's was investigated as a way to circumvent the undecidability of model checking for LCS's.

Formally, the problems we are interested in here are:

Almost-sure inevitability:
Given: a PLCS S, a configuration σ_0, and some set $W \subseteq \mathit{Conf}$ of configurations,
Question: does $\mathbb{P}_{\sigma_0}(\Diamond W) = 1$?

Almost-sure model checking:
Given: a PLCS S, a configuration σ_0, and some LTL$-\mathsf{X}$ formula φ,
Question: does $\mathbb{P}_{\sigma_0}(\varphi) = 1$?

In almost-sure inevitability, we usually only consider sets W given in some finite way, e.g. a finite W, or a W given by specifying the possible control states but putting no restriction on the channel contents. Similarly, we assume that the propositions used in temporal formula φ are simply individual configurations or control states, so that no special extra labelling of $\mathcal{M}(S)$ is required.

Remark 4.1. [PN97] and [BE99] consider formulae in LTL$-\mathsf{X}$, the fragment of LTL that is insensitive to stuttering. This is because $\mathcal{M}_g(S)$ and $\mathcal{T}_{\text{loss}}(S)$ do not have the same granularity (see Remark 3.2), so that properties sensitive to stuttering may evaluate differently in the two models. □

Here we follow [Var99] and assume properties are given under the form of a deterministic Streett automaton $\mathcal{A}_\varphi$. Checking whether $\mathcal{M}_g(S)$ satisfies φ almost surely reduces to checking whether the accepting runs in the product $\mathcal{M}_g(S) \otimes \mathcal{A}_\varphi$ have measure 1. When φ is insensitive to stuttering, we have $\mathcal{M}_g(S) \otimes \mathcal{A}_\varphi \equiv \mathcal{M}_g(S \otimes \mathcal{A}_\varphi)$ [BE99], and this also holds for $\mathcal{M}_l(S)$. Finally, writing S' for $S \otimes \mathcal{A}_\varphi$, we are left with verifying that a Streett acceptance property, of the form $\alpha = \bigwedge_{i=1}^{n} (\Box\Diamond A_i \Rightarrow \Box\Diamond A'_i)$, holds almost surely on $\mathcal{M}_g(S')$.

4.1 The decidability results

Baier and Engelen [BE99] show that almost-sure model checking is decidable for $\mathcal{M}_g(S)$ when $p_{\text{loss}} \geq \frac{1}{2}$ [5]. Bertrand and Schnoebelen [BS03], and Abdulla

[5] It is believed that this threshold cannot be improved: using a slightly modified model (see section 3.3), Abdulla *et al.* were able to show that the problem is undecidable when $p_{\text{loss}} < \frac{1}{2}$ [ABPJ00].

and Rabinovich [AR03], show that almost-sure model checking is decidable for $\mathcal{M}_l(S)$ for any $p_{\text{loss}} \in (0,1)$. These results apply to almost-sure inevitability since it is a special case of almost-sure model checking.

The techniques underlying all these decidability results are similar and rely on the existence of finite attractors (see below). Furthermore, in all cases, whether $\mathcal{M}(S)$ almost surely satisfies φ does not depend on the precise value of the fault probability p_{loss} [6], of the weights D, and of the choice of a local vs. global fault model! This is one benefit of qualitative verification: one does not have to worry too much about whether the numerical constants used in the PLCS are realistic or not.

We argued in section 3 that the local-fault model is more realistic than the global-fault model. This is especially true for quantitative properties (dealt with in next section). For qualitative properties, the real superiority of this model is that the existence of finite attractors (entailing decidability) is guaranteed and does not require $p_{\text{loss}} \geq \frac{1}{2}$.

4.2 The algorithmic ideas

Verifying that a finite Markov chain almost surely satisfies a Streett property is decidable [CY95,Var99]. However, the techniques involved do not always extend to *infinite* chains, in particular to chains that are not bounded.

It turns out it is possible to adapt these techniques to countable Markov chains *where a finite attractor exists*. We now develop these ideas, basically by simply streamlining the techniques of [BE99]. As is customary, rather than answering the question whether $\mathbb{P}(\varphi) = 1$, we deal with the dual question of whether $\mathbb{P}(\neg\varphi) > 0$, which allows a simpler technical exposition. More details and full proofs are available in [BS03].

Below we assume a given Markov chain $\mathcal{M} = \langle \mathit{Conf}, p \rangle$ with underlying transition system $\mathcal{T} = \langle \mathit{Conf}, \rightarrow \rangle$.

We say a non-empty set $W_a \subseteq \mathit{Conf}$ of configurations is an *attractor* when

$$\mathbb{P}_\sigma(\Diamond W_a) = 1 \text{ for all } \sigma \in \mathit{Conf}. \tag{6}$$

Note that (6) implies $\mathbb{P}_\sigma(\Box\Diamond W_a) = 1$ for all $\sigma \in \mathit{Conf}$. The attractor is *finite* when W_a is.

Observe that an attractor is not the same thing as a set of recurrent configurations (however, an attractor must contain at least one configuration from each recurrent class).

Assume $W_a \subseteq \mathit{Conf}$ is a finite attractor. We define $G(W_a)$ as the finite directed graph $\langle W_a, \rightarrow \rangle$ where the vertices are the configurations from W_a, and where there is an edge from σ to σ' iff σ' is reachable from σ by some nonempty path in $\mathcal{T}$. Observe that the edges in $G(W_a)$ are transitive (but not reflexive in general).

[6] Assuming it is $\geq \frac{1}{2}$ in the global-fault model.

In $G(W_a)$, we have the usual graph-theoretic notion of (maximal) strongly connected components (SCC's), denoted $B, B', \ldots$ These SCC's are ordered by reachability and a minimal SCC (i.e. an SCC B that cannot reach any other SCC) is a *bottom SCC* (a BSCC). A *trivial* SCC is a singleton without the self-loop. Observe that, in $G(W_a)$, a BSCC B cannot be trivial: since W_a is an attractor, one of its configurations must be reachable from B.

Assume a given Streett property $\alpha = \bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond A'_i)$. We say a BSSC B of $G(W_a)$ is *correct* for α if, for all $i = 1, ..., n$, either A_i is not reachable from B (in $\mathcal{T}$), or A'_i is. Write $B_1, \ldots, B_k$ for the set of correct BSCC's.

Lemma 4.2. **[Rab03]** $\mathbb{P}_\sigma(\alpha) = \mathbb{P}_\sigma(\Diamond(B_1 \cup \cdots \cup B_k))$.

Proof (Idea). Since W_a is a finite attractor, almost all paths eventually visit a BSSC B of $G(W_a)$. These paths almost surely visit B infinitely often, so that they almost surely visit infinitely often any configuration reachable from B and not any other configuration. Thus these paths satisfy α almost surely iff B is correct for α. $\Box$

The key result for our purposes is a reduction of the probabilistic verification of Streett properties of $\mathcal{M}$ to reachability questions on the finite $G(W_a)$.

Corollary 4.3. *$\mathbb{P}_\sigma(\bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond A'_i)) > 0$ iff $B_1 \cup \cdots \cup B_k$ is reachable from σ.*

Remark 4.4. Corollary 4.3 reduces the question $\mathbb{P}_\sigma(\bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond A'_i)) > 0$? to graph-theoretic notions on $G(W_a)$ where the transition probability p of $\mathcal{M}$ does not appear. Similarly, p has no role in the definition of $G(W_a)$. Where p does appear is in making W_a an attractor! $\Box$

Corollary 4.3 applies if we can find finite attractors in our Markov chains: Let S be a PLCS and let $W_0 \stackrel{\text{def}}{=} \{\langle q, \varepsilon\rangle \mid q \in Q\}$ be the set of all configurations where the channels are empty.

Lemma 4.5. *W_0 is a (finite) attractor in $\mathcal{M}_l(S)$.*
Furthermore, if $p_{\text{loss}} \geq \frac{1}{2}$, W_0 is an attractor in $\mathcal{M}_g(S)$.

Proof (Idea). The computations proving this are a bit tedious but the idea is easy to understand when one has some familiarity with random walks. We consider $\mathcal{M}_l(S)$ first. Assume $|\sigma| = m$. When $m > \frac{\log(2)}{-\log(1-p_{\text{loss}})}$, and thanks to Eq. (5), it is more probable to see steps $\sigma \to \sigma'$ where σ' exhibits a decrease rather an increase of size (relative to σ). Furthermore increases are at most single increments. Thus the "small" configurations are an attractor, and since W_0 is reachable from any set (thanks to lossiness), W_0 itself is an attractor.

In $\mathcal{M}_g(S)$, the same kind of reasoning explains why the same set W_0 is an attractor when $p_{\text{loss}} \geq \frac{1}{2}$: losses become so likely that the system cannot avoid being attracted to empty configurations. $\Box$

We now have all the necessary ingredients for the proof that almost-sure reachability and almost-sure model checking are decidable for PLCS's: since reachability is decidable in the underlying transition system $\mathcal{T}$, $G(W_a)$ can be built effectively, and it can be decided if a BSSC is correct for α and if it is reachable from σ.

These techniques apply to any extension where reachability remains decidable and where finite attractors can be identified. For example, in [AR03], Abdulla and Rabinovich investigate an extension of the local-fault model where other forms of channel unreliability exist: messages in transit can be *corrupted* (randomly replaced by some other message), *duplicated* (the duplicate message appears just alongside the original message), and *spurious* messages can appear out of the blue (these are called *insertion errors* in [CFP96]). If corruptions and duplications have a per-message probability [7] of, respectively, p_{corrupt} and p_{dupl}, then W_0 is an attractor if $p_{\text{dupl}} < p_{\text{loss}}$, in which case decidability of almost-sure model-checking can be inferred from the decidability of reachability.

5 Approximate quantitative verification of PLCS's

Computing *quantitative* properties, e.g. computing the actual probability $\mathbb{P}(\varphi)$ that φ will be satisfied, is usually harder than deciding qualitative properties (like we did in Section 4). For one thing, and unless it is zero or one, $\mathbb{P}(\varphi)$ will depend on the actual values of the weights and p_{loss}.

Purushothaman Iyer and Narasimha proposed algorithms for the verification of quantitative properties of PLCS's. It turns out these algorithms are flawed (as observed by Rabinovich [Rab03]) in the global-fault model for which they were intended. However, the underlying ideas can be salvaged and made to work for the local-fault model (or the global-fault model if $p_{\text{loss}} \geq \frac{1}{2}$: the required condition is the existence of a finite attractor).

5.1 The decidability results

Under the local-fault interpretation, the following problems have effective solutions:

Quantitative probabilistic reachability:
Given: a PLCS S, two configurations σ_0 and σ_f, and a *tolerance* $\nu > 0$,
Problem: find a p such that $p - \nu \leq \mathbb{P}_{\sigma_0}(\Diamond\sigma) \leq p + \nu$.

Quantitative probabilistic model checking:
Given: a PLCS S, a configuration σ_0, some LTL$-\mathsf{X}$ formula φ, and a tolerance $\nu > 0$,
Problem: find a p such that $p - \nu \leq \mathbb{P}_{\sigma_0}(\varphi) \leq p + \nu$.

[7] Regarding insertion errors, we find it more natural to see them as having a global probability, not a per-message one. But one can model insertion as duplication+corruption going in pairs.

We speak of *approximate* quantitative verification because of the tolerance parameter ν (that can be as small as one wishes).

5.2 The algorithmic ideas

The methods for answering quantitative probabilistic reachability may be of more general interest. Assume we are given some PLCS S with two configurations σ_0 and σ_f. For $k \in \mathbb{N}$ we let $T_k(\sigma_0)$ denote the tree obtained by unfolding $\mathcal{M}_l(S)$ from σ_0 until depth k. Fig. 2 displays a schematic example for $k = 3$.

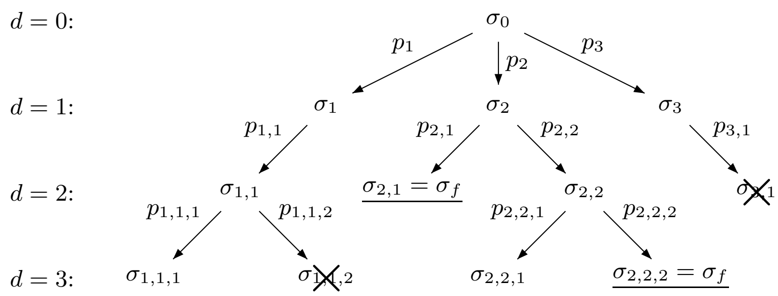

Fig. 2. A tree $T_3(\sigma_0)$ obtained by unfolding some $\mathcal{M}(S)$

Not all paths in $T_k(\sigma_0)$ have length k: paths are not developed beyond σ_f when it is encountered (these leaves are underlined in Fig. 2), or beyond any configuration σ from which σ_f is not reachable (these leaves are crossed).

Observe that it is effectively possible to build $T_k(\sigma_0)$: the crucial part is to cross out the leaves from which σ_f is not reachable, but this can be done since reachability is decidable (recall section 2.3).

A path from the root of $T_k(\sigma_0)$ to some leaf denotes a basic cylinder in the space of runs of $\mathcal{M}_l(S)$. The measure of these cylinders is given by multiplying the individual probabilities along the edges of the path, e.g. the leftmost leaf in our example denotes a set of runs with measure $p_1 \times p_{1,1} \times p_{1,1,1}$. We collect these cylinders in three classes: a path is a $\top$-*path* if it ends in σ_f, a $\perp$-*path* if it ends in a crossed-out leaf, and a ?-*path* otherwise. Thus $T_k(\sigma_0)$ partitions the runs from σ_0 in a finite number of cylinders belonging to three different types. If we now collect the measures of these cylinders according to type we end up with total measure $\mathbb{P}^k_\top$ for the $\top$-paths, $\mathbb{P}^k_\perp$ for the $\perp$-paths, and $\mathbb{P}^k_?$ for the ?-paths, ensuring $\mathbb{P}^k_\top + \mathbb{P}^k_\perp + \mathbb{P}^k_? = 1$ for all $k \in \mathbb{N}$. Obviously, T_{k+1} refines T_k in such a way that $\mathbb{P}^{k+1}_\top \geq \mathbb{P}^k_\top$ and $\mathbb{P}^{k+1}_\perp \geq \mathbb{P}^k_\perp$, entailing $\mathbb{P}^{k+1}_? \leq \mathbb{P}^k_?$.

The value $\mathbb{P}_{\sigma_0}(\Diamond \sigma_f)$ we are looking for satisfies

$$\mathbb{P}^k_\top \;\leq\; \mathbb{P}_{\sigma_0}(\Diamond \sigma_f) \;\leq\; \mathbb{P}^k_\top + \mathbb{P}^k_? \tag{7}$$

Lemma 5.1. $\lim_{k \to \infty} \mathbb{P}^k_? = 0$.

Proof. $\mathbb{P}^k_?$ is the probability that, in its first k steps, a run only visits configurations from which σ_f is reachable (called *good configurations*) without actually visiting σ_f itself. Thus $\lim_{k\to\infty}\mathbb{P}^k_?$, denoted $\mathbb{P}^\omega_?$, is the probability that an infinite path only visits good configurations but not σ_f.

Now, $\mathcal{M}_l(S)$ has a finite attractor W_0 (Lemma 4.5) and an infinite run almost surely visits W_0 infinitely often. Hence $\mathbb{P}^\omega_?$ is also the probability that an infinite run only visits good configurations, visits infinitely often some configuration $\sigma_a \in W_0$, and never visits σ_f. But if σ_a is visited infinitely often, and σ_f is reachable from σ_a, then σ_f is visited almost surely. ☐

A possible algorithm for evaluating $\mathbb{P}_{\sigma_0}(\Diamond\sigma_f)$ within ν is to build $T_k(\sigma_0)$, use it to evaluate $\mathbb{P}^k_\top$ and $\mathbb{P}^k_?$, and check if the margin provided by Eq. (7) is low enough for ν, i.e. if $\mathbb{P}^k_? \leq 2\nu$. If this is not the case, we retry with a larger k: Lemma 5.1 ensures that eventually the margin will be as small as needed.

The same method can be used for approximating the value of some $\mathbb{P}_{\sigma_0}(\bigwedge_{i=1}^n(\Box\Diamond A_i \Rightarrow \Box\Diamond A'_i))$: Lemma 4.2 equates this to some $\mathbb{P}_{\sigma_0}(\Diamond(B_1\cup\cdots\cup B_k))$.

Remark 5.2. In the global-fault model, Lemma 5.1 does not always hold and $\mathbb{P}^\omega_?$ can be strictly positive. This is what was missed in [PN97]. For example, consider a simple system with only the rule $q \xrightarrow{c!a} q$. $\mathcal{M}_g(S)$ is essentially a random walk on $\mathbb{N}$ with a reflecting barrier in 0: in configuration $\langle q, a^n\rangle$ with $n > 0$ messages, one moves to $\langle q, a^{n-1}\rangle$ with probability p_{loss} and to $\langle q, a^{n+1}\rangle$ with probability $1 - p_{\text{loss}}$. It is well-known that if $p_{\text{loss}} < \frac{1}{2}$ then there is a non-zero probability that a run from some nonempty configuration will never visit the empty configuration. This non-zero probability coincides with $\mathbb{P}^\omega_?$. ☐

The ideas underlying the above algorithm are quite general and apply to all finitely-branching countable Markov chains with a finite attractor. Effectiveness relies on the fact that $\perp$-paths can be identified: the decidability of reachability is an essential ingredient in the algorithm for quantitative probabilistic reachability.

5.3 A first assessment

The positive results of section 4 and 5 provide methods for verifying channel systems where message losses are seen as some kind of fault obeying probabilistic laws. These results can handle arbitrary LTL formulae (indeed, arbitrary ω-regular properties) and therefore they can be seen as circumventing the undecidability of LTL model checking for standard, non-probabilistic, lossy channel systems.

However there are two main limitations with these results.

1. The value of $\mathbb{P}_{\sigma_0}(\varphi)$ depends on p_{loss} and on the weights D of rules in S. Assigning a fixed fault probability is natural in many situations, and sound reasoning can be carried out even under the assumption of an overly pessimistic fault probability. However, given a distributed protocol modelled as some LCS, it is difficult to meaningfully give values for D. Thus it is hard to make sense of any quantitative evaluation of some $\mathbb{P}_{\sigma_0}(\varphi)$.

2. This difficulty disappears with qualitative verification, since the answers do not depend on the exact values of D or p_{loss}. However, it is still the case that the models we have been considering, $\mathcal{M}_l(S)$ or $\mathcal{M}_g(S)$, see the rules of S as probabilistic instead of nondeterministic. In practical verification situations, there are at least four possible reasons for nondeterminism in rules:
   - Arbitrary interleaving of deterministic but asynchronously coupled components.
   - Under-specification, at some early stage of the design process.
   - Inclusion of the unknown environment as part of the model for an open system.
   - Abstraction of some complex protocol to make it fit the finite-state control paradigm.

   The first kind of nondeterminism can perhaps accommodate randomisation of the rules, but the other kinds usually cannot: a branch at a nondeterministic choice point in the abstract model may well never be followed in the actual system, even if the choice point is visited infinitely often.

This difficulty with Markov chains is well-known and its solution usually requires using a richer family of models: the reactive (or concurrent) Markov chains, also known as Markovian decision processes.

6 PLCS's as Markovian decision processes

It is very natural to model PLCS's as Markovian decision processes (MDP's) where message losses have some probabilistic behaviour, and where the LCS rules retain their classical nondeterministic meaning. Such a model was first introduced by Bertrand and Schnoebelen [BS03].

There are several possibilities for associating a MDP with a channel system S. The proposal in [BS03] adopts the conventions of [Var99] and elaborates on the ideas underlying the definition of $\mathcal{M}_l(S)$:

1. we have two kinds of configurations: nondeterministic ones and probabilistic ones,
2. steps alternate between firing LCS rules (going from a nondeterministic configuration to probabilistic ones) and losing messages (from a probabilistic configuration to nondeterministic ones), hence the underlying graph is bipartite.

In [BS03] the losses follow the "local", per-message, interpretation of the p_{loss} parameter, so that we shall write $\mathcal{P}_l(S)$ to denote the MDP associated with S.

As usual, the behaviour of $\mathcal{P}_l(S)$ is defined by means of schedulers (denoted $u, u', \ldots$) that make the nondeterministic choices, based on what happened earlier. When such an scheduler u is provided, the system gives rise to a Markov chain (denoted $\mathcal{P}_l^u(S)$) in the usual way.

Remark 6.1. [BS03] introduces one important definitional detail that make technicalities easier: we assume that it is always possible to idle, written $\sigma \xrightarrow{0} \sigma$, instead of firing a rule of Δ. This possibility prevents deadlocks and the definitional difficulties they raise, but it also give more flexibility to schedulers: they can empty the channels by just waiting (idling) until the probabilistic losses do the emptying job, which will eventually happen almost surely. □

6.1 The decidability results

The main verification questions in this framework are

Adversarial model checking:
Given: a PLCS S, a configuration σ_0, and some LTL$-\mathsf{X}$ formula φ,
Question: does $\mathbb{P}_{\sigma_0}(\varphi) = 1$ hold in $\mathcal{P}^u(S)$ for all u?

Cooperative model checking:
Given: a PLCS S, a configuration σ_0, and some LTL$-\mathsf{X}$ formula φ,
Question: does $\mathbb{P}_{\sigma_0}(\varphi) = 1$ hold in $\mathcal{P}^u(S)$ for some u?

Note that, though the two problems are related, they cannot be reduced one to the other. It can be argued that, in verification settings, adversarial model checking is the more natural question.

Bertrand and Schnoebelen [BS03,BS04] show that adversarial and cooperative model checking are decidable when one only considers finite-memory [8] schedulers (i.e. when the quantifications over "all u" is relativized to "all finite-memory u").

They show that adversarial and cooperative model checking are undecidable when there are no restrictions on the schedulers [9].

6.2 The algorithmic ideas

When considering adversarial model checking, we follow the traditional wisdom that it is simpler to reason about questions of the form $\exists u\ \mathbb{P}(\neg \ldots) > 0$, rather than of the form $\forall u\ \mathbb{P}(\ldots) = 1$, even though the two are dual. Since Streett properties are closed by negation, we can further simplify our framework and consider questions of the form $\exists u\ \mathbb{P}(\alpha) > 0$.

To begin with, we try to explain how decidability of adversarial model checking depends on the finite-memory assumption. For this we reproduce the proof that the unrestricted problem is undecidable: this provides a lively example of the difference between unrestricted and finite-memory schedulers.

[8] A scheduler is said to be *finite-memory* when it makes its choices as a function of the current configuration and some finite-state information about the history of the computation.

[9] Or on the LTL$-\mathsf{X}$ formulae: for example, formulae of the simpler form $\Diamond W$ lead to decidable adversarial or cooperative problems with no restriction on the schedulers [BS03].

Ideas for undecidability. Let $S = \langle Q, \{c\}, \Sigma, \Delta, \sigma_0 \rangle$ be a single-channel LCS where σ_0 is $\langle r_0, \varepsilon \rangle$.

We modify S to obtain S', a new LCS. The construction is illustrated in Fig. 3, where S is copied in the dashed box.

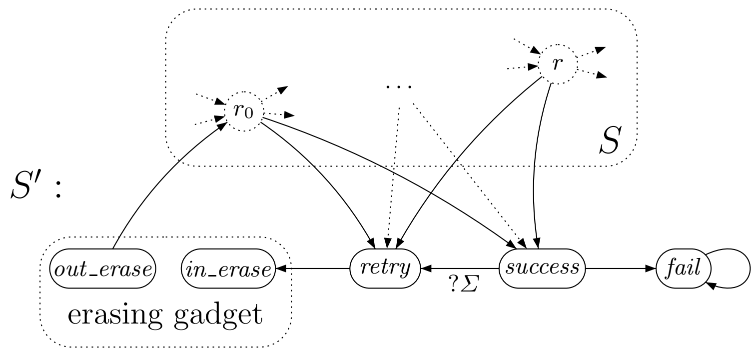

Fig. 3. The NPLCS S' associated with LCS S

S' is obtained by adding three control states (*success*, *retry* and *fail*), some fixed gadget for erasing (that is, emptying) the channel,[10] and rules allowing to jump [11] from any S-state $r \in Q$ to *success* or to *retry*. Jumping from *success* to *fail* is always possible but moving from *success* to *retry* requires that one message be read from the channel: the "$?\Sigma$" label is a shorthand for all $?a$ where $a \in \Sigma$.

Let us now ask ourselves whether we could devise a scheduler u s.t. $\mathbb{P}_{\sigma_0}(\Box\Diamond A) > 0$ in $\mathcal{P}_l^u(S')$ for $A = \uparrow\!\mathit{success} \setminus \langle \mathit{success}, \varepsilon \rangle$, i.e. whether there exists a scheduler that can make S' visit *success* infinitely often with nonzero probability (and without just idling there with an empty channel).

This seems easy: when we are in the S part of S', we can visit *success* whenever we want by jumping there, then we can erase the channels, jump to σ_0 and start again.

There is a catch however: if the channel is empty when we visit *success*, we will not be allowed to reach the erasing gadget via *retry* (this requires a read) and will end up stuck in *fail*. Now the bad news is that, whenever we decide to jump from some $\langle r, U \rangle$ configuration to *success*, there always is a nonzero probability (a *risk*) that all messages currently in the channel will be lost during this one jump, leaving us in configuration $\langle \mathit{success}, \varepsilon \rangle$ and thwarting our purposes. Precisely, if $|U| = m$, then the risk is $(p_{\text{loss}})^m$.

[10] The specification of the gadget is that, from any $\langle \mathit{in\_erase}, U \rangle$ configuration it is always possible to reach $\langle \mathit{out\_erase}, \varepsilon \rangle$ whatever happens (there are no deadlock) and it is impossible to reach $\langle \mathit{out\_erase}, V \rangle$ for $V \neq \varepsilon$. See [BS04] for a fully detailed solution.

[11] These are internal rules where no reading or writing takes place. Such rules can be simulated by writing to a dummy channel.

Therefore, visiting *success* infinitely many times requires that we make an infinite number of jumps, each of them with a nonzero risk. The product of these risks will only be nonzero if the values are diminishingly small (e.g. the risk associated with the nth jump is less than n^{-2}), which can only be the case if we jump from configurations having larger and larger channels contents. Thus, if S is bounded and such ever larger configurations do not exist, then necessarily $\mathbb{P}_{\sigma_0}(\Box\Diamond A) = 0$ for all u.

On the other hand, if S is not bounded, there exists ever larger reachable configurations in $\mathcal{T}_{\text{loss}}(S)$. It would be smart to try and reach these larger and larger configurations, and only jump to *success* when a large enough configuration is reached. Since the losses in $\mathcal{P}(S')$ are probabilistic, we may need some luck to reach these large configurations. However, when the losses do not comply, we can simply have another go (via the rules jumping from S to *retry*) so that a persistent scheduler will eventually reach any (reachable) configuration it wishes. Such a scheduler is clearly not finite-memory: it has to remember what large configuration it is currently aiming at, and there are an infinite number of them.

Finally, in $\mathcal{P}_l(S')$, $\mathbb{P}_{\sigma_0}(\Box\Diamond A) = 0$ for all u iff $\mathcal{T}_{\text{loss}}(S)$ is bounded. Since boundedness of $\mathcal{T}_{\text{loss}}(S)$ is undecidable [May00], we obtain undecidability for adversarial model checking.

Ideas for decidability. It is now possible to hint at why adversarial model checking is decidable when we restrict to finite-memory schedulers.

Assume we want to check whether there is some scheduler u that yields $\mathbb{P}(\alpha) > 0$ for some PLCS $S = \langle Q, C, \Sigma, \Delta, \sigma_0, p_{\text{loss}}, D\rangle$ and some Streett property $\alpha = \bigwedge_{i=1}^{n}(\Box\Diamond A_i \Rightarrow \Box\Diamond A'_i)$. We assume the sets $A_i, A'_i \subseteq \mathit{Conf}$ are determined by sets of control states $X_i, X'_i \subseteq Q$.

A possible strategy for such a scheduler is to try to reach a set $X \subseteq Q$ of control states s.t. when one is in X-configurations (that is, configurations with a control state from X) then one can fulfill α without stepping out of X. More formally, we want that X is reachable from σ_0, is non-trivially connected by transitions that do not visit states out of X, and satisfies α: such an X is called a *safe set*.

If a safe X exists, a scheduler exists for $\mathbb{P}(\alpha) > 0$: it tries to reach X (this succeeds with non-zero probability) and, when in X, it just has to fill some $\Box\Diamond A'_i$ obligations. Since from any $\langle x, \varepsilon\rangle$ with $x \in X$, there is a path to A'_i that does not step out of X, it is enough to try that path. When this fails because probabilistic losses did not comply with the path, the scheduler waits until the channels are empty and tries again from the $\langle x', \varepsilon\rangle$ configuration we end up in. This scheme is bound to eventually succeed, and only requires finite memory. Thus, once we are in a safe X, we have $\mathbb{P}(\alpha) = 1$ (the only risk is in whether we can reach X from σ_0, but this has a nonzero probability of success).

The remarkable thing with finite-memory schedulers is that, if there exists a finite-memory scheduler ensuring $\mathbb{P}(\alpha) > 0$, then there exists one that follows the simple "pick a safe $X \subseteq Q$" strategy.

To see this, remember that any scheduler u will make us visit the attractor W_0 infinitely often, and in particular some $\langle x, \varepsilon \rangle$ infinitely often. But a finite-memory scheduler cannot distinguish between all these infinitely many visits to $\langle x, \varepsilon \rangle$ and there is some sequence $(\langle x, \varepsilon \rangle =) \langle x_0, U_0 \rangle \xrightarrow{\delta_1} \langle x_1, U_1 \rangle \xrightarrow{\delta_2} \langle x_2, U_2 \rangle \cdots \langle x_n, U_n \rangle$ of transitions, with x_n in some obligation A'_i, that it will try infinitely many times. Trying this infinitely many times means that all the $\langle x_j, \varepsilon \rangle$ for $j = 0, \ldots, n$ are bound to happen infinitely often because of probabilistic losses. Assuming that u ensures α with a nonzero probability entails that the x_j's form a safe set X.

Now that we have equated the existence of a finite-memory u with the existence of a safe set, it remains to check that safe sets can be computed: this is easily done by enumerating the finite number of candidates X and checking each of them using the decidability of reachability in $\mathcal{T}_{\mathrm{loss}}$.

6.3 An assessment of the adversarial verification of PLCS's

Modelling PLCS's as Markovian decision processes is a recent idea, and many questions remain unanswered. However, it seems this approach may lead to satisfactory ways of circumventing the undecidability of (classical) LCS model checking: decidability is recovered by simply omitting to take into account exaggeratedly malicious nondeterministic behaviours, be they schedulers that need finite memory or losses that have a zero probability of occurring in real life. One further advantage of this approach is that it relies on reachability questions of the kind that has been shown tractable via symbolic methods.

7 Conclusions and perspectives

There are two main reasons for moving from LCS's to probabilistic LCS's:

1. obtaining quantitative information about the behaviour of the system (average times, probability that something happens, etc.), and
2. getting rid of exaggeratedly malicious nondeterministic behaviours by imposing some form of fairness.

The results we surveyed are very recent and it is not yet clear what will be the main directions for further research in this area. We just mention the problems that have been left unanswered in our survey:

In the *quantitative* approach:

– Can one compute exactly (not approximately) $\mathbb{P}_{\sigma_0}(\varphi)$ in the Markov chain models?
– Can one compute, or approximate, other numerical values like mean hitting time, etc.?
– Can one compute, or approximate, the extremal values of $\mathbb{P}_{\sigma_0}(\varphi)$ when ranging over all adversaries in the MDP models?

In the *qualitative* approach:

– Is cooperative model checking decidable?
– Do the decidability results extend to more general MDP models of LCS's (e.g. where probabilistic steps are not limited to message losses)?

From a more general perspective, it would be interesting to see how much of the ideas that have been put forward in the analysis of probabilistic lossy channel systems can be of use in other infinite-state probabilistic models.

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.

[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.

[AR03] P. A. Abdulla and A. Rabinovich. Verification of probabilistic systems with faulty communication. In *Proc. 6th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2003), Warsaw, Poland, Apr. 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2003.

[BE99] C. Baier and B. Engelen. Establishing qualitative properties for probabilistic lossy channel systems: An algorithmic approach. In *Proc. 5th Int. AMAST Workshop Formal Methods for Real-Time and Probabilistic Systems (ARTS'99), Bamberg, Germany, May 1999*, volume 1601 of *Lecture Notes in Computer Science*, pages 34–52. Springer, 1999.

[BS03] N. Bertrand and Ph. Schnoebelen. Model checking lossy channels systems is probably decidable. In *Proc. 6th Int. Conf. Foundations of Software Science and Computation Structures (FOSSACS'2003), Warsaw, Poland, Apr. 2003*, volume 2620 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2003.

[BS04] N. Bertrand and Ph. Schnoebelen. Verifying nondeterministic channel systems with probabilistic message losses. In *Proc. 3rd Int. Workshop on Automated Verification of Infinite-State Systems (AVIS'04), Barcelona, Spain, Apr. 2004*, 2004. To appear.

[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.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[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.

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

[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.

[MS02] B. Masson and Ph. Schnoebelen. On verifying fair lossy channel systems. In *Proc. 27th Int. Symp. Math. Found. Comp. Sci. (MFCS'2002), Warsaw, Poland, Aug. 2002*, volume 2420 of *Lecture Notes in Computer Science*, pages 543–555. Springer, 2002.

[Pan01] P. Panangaden. Measure and probability for concurrency theorists. *Theoretical Computer Science*, 253(2):287–309, 2001.

[PN97] S. Purushothaman Iyer and M. Narasimha. Probabilistic lossy channel systems. In *Proc. 7th Int. Joint Conf. Theory and Practice of Software Development (TAPSOFT'97), Lille, France, Apr. 1997*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 1997.

[Rab03] A. Rabinovich. Quantitative analysis of probabilistic lossy channel systems. In *Proc. 30th Int. Coll. Automata, Languages, and Programming (ICALP'2003), Eindhoven, NL, July 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 1008–1021. Springer, 2003.

[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.

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.

[Var99] M. Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *Proc. 5th Int. AMAST Workshop Formal Methods for Real-Time and Probabilistic Systems (ARTS'99), Bamberg, Germany, May 1999*, volume 1601 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 1999.